

1. Klausur zur Vorlesung
 Informatik III
 Wintersemester 2004/2005

Lösung!

Beachten Sie:

- Bringen Sie Ihren *Aufkleber* auf diesem Deckblatt an, und beschriften Sie *jedes weitere Blatt* mit Ihrem Namen und Ihrer Matrikelnummer.
- Schreiben Sie Ihre Lösungen auf die Aufgabenblätter und Rückseiten. Zusätzliches Papier erhalten Sie bei Bedarf von der Aufsicht.
- Zum Bestehen der Klausur sind **20** der möglichen **60** Punkte hinreichend.
- Es sind keine Hilfsmittel zugelassen.

| Aufgabe | Mögliche Punkte | | | | | Erreichte Punkte | | | | |
|----------|-----------------|---|---|---|----------|------------------|---|---|---|----------|
| | a | b | c | d | Σ | a | b | c | d | Σ |
| 1 | 3 | 3 | 3 | 3 | 12 | | | | | |
| 2 | 2 | 3 | 3 | 4 | 12 | | | | | |
| 3 | 3 | 3 | 3 | 3 | 12 | | | | | |
| 4 | 3 | 3 | 3 | 3 | 12 | | | | | |
| 5 | 12x1 | | | | 12 | | | | | |
| Σ | | | | | 60 | | | | | |

Aufgabe 1:

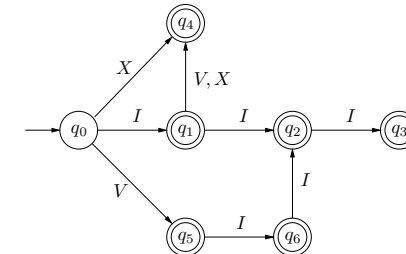
(3+3+3+3=12 Punkte)

- (a) Geben Sie einen endlichen Automaten über dem Alphabet $\{I, V, X\}$ mit *nicht mehr als sieben Zuständen* an (Übergangsdiagramm genügt), der genau die römischen Zahlen zwischen 1 und 10 erkennt, also die Menge

$$\{I, II, III, IV, V, VI, VII, VIII, IX, X\}.$$

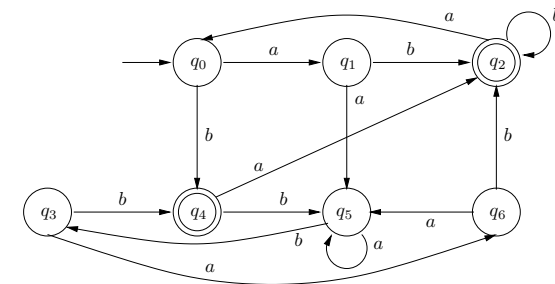
Hinweis: Für jeden weiteren benötigten Zustand verringert sich die Anzahl der für diese Teilaufgabe erreichbaren Punkte um 1.

Lösung:



Bemerkung: Zustände q_3 und q_4 lassen sich sogar noch ‚zusammenlegen‘.

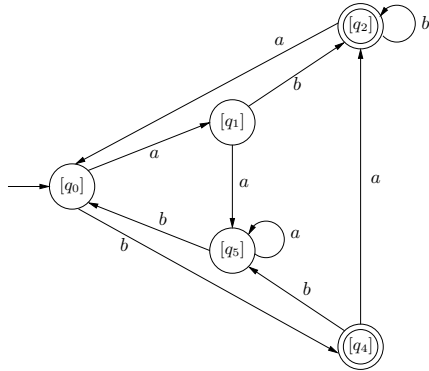
- (b) Minimieren Sie den durch nachfolgendes Diagramm gegebenen Automaten durch eine systematische Konstruktion, und geben Sie den daraus resultierenden minimalen DEA (Übergangsdiagramm genügt) an.



Lösung:

ε : $\{q_0, q_1, q_3, q_5, q_6\}, \{q_2, q_4\}$
 a : $\{q_0, q_1, q_3, q_5, q_6\}, \{q_2\}, \{q_4\}$
 b : $\{q_0, q_1, q_3, q_6\}, \{q_5\}, \{q_2\}, \{q_4\}$
 aa : $\{q_0, q_1, q_3, q_6\}, \{q_5\}, \{q_2\}, \{q_4\}$
 ab : $\{q_0, q_3\}, \{q_1, q_6\}, \{q_5\}, \{q_2\}, \{q_4\}$
 ba : $\{q_0, q_3\}, \{q_1, q_6\}, \{q_5\}, \{q_2\}, \{q_4\}$
 bb : $\{q_0, q_3\}, \{q_1, q_6\}, \{q_5\}, \{q_2\}, \{q_4\}$

Wörter der Länge 3 trennen keine Zustände mehr.



- (c) Zeigen Sie für die Sprache $L_1 = \{01^k0^l1 : k, l \geq 0\}$ *explizit*, dass die im Pumping-Lemma formulierte notwendige Bedingung für Regularität erfüllt ist. (Geben Sie also eine konkrete ‚Belegung‘ für die existenzquantifizierten Ausdrücke an, und begründen Sie.)

Lösung: Sei $n := 3$. Alle Wörter $w = 01^k0^l1 \in L_1$ der Länge größer oder gleich n (sei o. B. d. A. $k \geq 0$) können in uvx mit $u = 0$, $v = 1$ und $x = 1^{k-1}0^l1$ zerlegt werden ($|uv| \leq n$ und $v \neq \varepsilon$ sind so erfüllt), womit für alle $i \in \mathbb{N}_0$ gilt, dass $uv^i x \in L_1$.

- (d) Sei L_2 die reguläre Sprache $(0 \cup 1)^+00$. Geben Sie für L_2 die Äquivalenzklassen bezüglich der Neroderelation (in ‚Mengenschreibweise‘) an sowie für jede Klasse einen Repräsentanten.

Lösung: Σ^* zerfällt über R_{L_2} in folgende Äquivalenzklassen:

- $[\varepsilon] = \{\varepsilon\}$
- $[1] = \{w \in \Sigma^* : w = 0 \text{ oder } w \text{ endet auf } 1\}$
- $[10] = \{w \in \Sigma^* : w = 00 \text{ oder } w \text{ endet auf } 10\}$
- $[100] = \{w \in \Sigma^* : |w| > 2 \text{ und } w \text{ endet auf } 00\}$

Aufgabe 2:

(2+3+3+4=12 Punkte)

Betrachten Sie die Turingmaschine \mathcal{M} mit Zustandsmenge $Q = \{s, f, p, q, r\}$, Eingabealphabet $\Sigma = \{a, b\}$, Bandalphabet $\Gamma = \Sigma \cup \{\sqcup\}$, Startzustand s , Endzustandsmenge $F = \{f\}$ und folgender Übergangsfunktion δ :

| | | |
|--|--|---|
| $\delta(s, a) \rightarrow (p, b, R)$ | $\delta(p, a) \rightarrow (r, \sqcup, R)$ | $\delta(q, a) \rightarrow (p, b, R)$ |
| $\delta(s, b) \rightarrow (r, \sqcup, R)$ | $\delta(p, b) \rightarrow (q, a, R)$ | $\delta(q, b) \rightarrow (r, \sqcup, R)$ |
| $\delta(s, \sqcup) \rightarrow (f, a, N)$ | $\delta(p, \sqcup) \rightarrow (r, \sqcup, R)$ | $\delta(q, \sqcup) \rightarrow (f, b, R)$ |
| $\delta(r, a) \rightarrow (r, \sqcup, R)$ | $\delta(f, a) \rightarrow (f, a, N)$ | |
| $\delta(r, b) \rightarrow (r, \sqcup, R)$ | $\delta(f, b) \rightarrow (f, b, N)$ | |
| $\delta(r, \sqcup) \rightarrow (r, \sqcup, N)$ | $\delta(f, \sqcup) \rightarrow (f, \sqcup, N)$ | |

- (a) Geben Sie die von \mathcal{M} durchlaufenen Konfigurationen bei Abarbeitung der Wörter ba beziehungsweise $ababab$ an.

Lösung: $\sqcup(s)ba, \sqcup(r)a, \sqcup(r)\sqcup$

$\sqcup(s)ababab, b(p)babab, ba(q)abab, bab(p)bab, baba(q)ab, babab(p)b, bababa(q)\sqcup, bababab(f)\sqcup$

- (b) Geben Sie die von \mathcal{M} akzeptierte Sprache $L(\mathcal{M})$ an.

Sei g die durch \mathcal{M} realisierte Funktion. Geben Sie g , eingeschränkt auf $L(\mathcal{M})$, explizit an.

Wie sieht $g(aab)$ aus?

Lösung: Per Induktion läßt sich zeigen, dass $L(\mathcal{M}) = \{(ab)^n : n \geq 0\}$. Bei der Abarbeitung des leeren Wortes (nur \sqcup auf dem Band), wird ein a geschrieben und in den Endzustand f übergegangen, da $\delta(s, \sqcup) = (f, a, N)$. Bei einem nicht leeren Wort der Form $(ab)^n$ (mit $n > 0$) folgt per Induktion, dass die Buchstaben a und b jeweils vertauscht werden ($\delta(p, b) = (q, a, R)$ und $\delta(q, a) = (p, b, R)$). Durch das Lesen des letzten bs wird in den Zustand q gewechselt und anschließend ein \sqcup gelesen. Da $\delta(q, \sqcup) = (f, b, R)$, wird noch ein b geschrieben und in den akzeptierenden Endzustand gewechselt. Bei anderen Wörtern wird nie ein akzeptierender Endzustand erreicht. Damit gilt:

$$g: (ab)^n \mapsto \begin{cases} (ba)^n b & , \text{ falls } n > 0 \\ a & , \text{ sonst} \end{cases}$$

und $g(aab) = b$

- (c) Sei L eine nichtleere Sprache über einem Alphabet Σ . Zeigen Sie: Falls es eine total berechenbare Funktion $h: \mathbb{N} \rightarrow \Sigma^*$ mit $h(\mathbb{N}) = L$ gibt, dann ist L semientscheidbar.

Lösung: Um zu zeigen, dass L semientscheidbar ist, konstruieren wir eine Turingmaschine \mathcal{M} , die bei einer Eingabe $w \in L$ in einem akzeptierenden Endzustand hält und alle anderen Worte nicht akzeptiert. Die TM \mathcal{M} erhält als Eingabe ein Wort w und wird folgendes abarbeiten:

- (i) counter $\leftarrow 1$
- (ii) **while** ($h(\text{counter}) = w$) **do**
 - counter \leftarrow counter + 1

(iii) akzeptiere das Eingabewort w

Falls das Eingabewort w in L enthalten ist, dann gibt es ein i mit $h(i) = w$ und somit wird w akzeptiert. Anderenfalls ist der Test $h(\text{counter})! = w$ stets wahr und somit terminiert die TM nie, insbesondere wird w nicht akzeptiert.

(d) Sei \mathcal{G} die Menge aller kontextfreien Grammatiken, die mindestens ein Palindrom erzeugen, und

$$L_P = \{\text{code}(G) \mid G \in \mathcal{G}\},$$

wobei $\text{code}(G)$ eine geeignete Kodierung von G über einem festen Alphabet bezeichne.

Zeigen Sie, dass L_P *nicht* entscheidbar ist.

Hinweis: Zu einer PKP-Instanz I kann eine kontextfreie Grammatik konstruiert werden, deren Sprache genau dann ein Palindrom enthält, wenn I eine Lösung hat.

Lösung: Sei PKP-Instanz $I = ((x_1, y_1), \dots, (x_n, y_n))$ über einem Alphabet Σ gegeben. Konstruieren dazu Grammatik $G_I = (\Sigma \cup \{z\}, \{S\}, S, R)$ mit

$$R = \{S \rightarrow x_i S y_i^R \mid x_i z y_i^R : 1 \leq i \leq n\},$$

wobei w^R das Spiegelwort des Wortes w bezeichne.

Dann existiert eine Lösung für I genau dann, wenn sich durch G_I ein Palindrom erzeugen lässt. (Die Gültigkeit der Rückrichtung wird durch die Benutzung des ‚Trennsymbols‘ z gesichert; dies verhindert, dass ein Palindrom durch Konkatenation unterschiedlich langer Teilwörter ‚links bzw. rechts von S ‘ zustande kommt.)

Das Problem der Entscheidbarkeit von L_P kann also auf die Unentscheidbarkeit des PKP reduziert werden, d. h. falls L_P entschieden werden könnte, so wäre auch das PKP entscheidbar:

Annahme L_P ist entscheidbar, d.h. es gibt eine Turingmaschine \mathcal{M} , die die Eingabe $w \in L_P$ akzeptiert und bei allen anderen Eingabe hält (aber nicht akzeptiert). Damit konstruieren wir eine Turingmaschine \mathcal{M}' , die PKP entscheidet. Die TM \mathcal{M}' berechnet zuerst $\text{code}(G_I)$, wobei G_I wie oben aufgebaut ist, und simuliert anschließend das Verhalten von \mathcal{M} bei Eingabe $\text{code}(G_I)$. Falls \mathcal{M} die Eingabe akzeptiert, akzeptiert auch \mathcal{M}' ihre Eingabe (und hält), ansonsten hält \mathcal{M}' in einem nicht-akzeptierenden Zustand. Da \mathcal{M} nach Voraussetzung nur endlich viele Operationen durchführt, kann \mathcal{M}' zur Entscheidung des PKP-Problems benutzt werden (da JA-Instanzen von PKP genau auf Wörter in L_P und NEIN-Instanzen von PKP genau auf Wörter nicht in L_P abgebildet werden). Dies ist ein Widerspruch, da PKP nicht entscheidbar ist.

Aufgabe 3:

(3+3+3+3=12 Punkte)

(a) **Problem 4TA-SAT**

Gegeben: Variablenmenge U , Menge C von Klauseln über U .

Frage: Gibt es vier verschiedene erfüllende Belegungen von U ?

Zeigen Sie die \mathcal{NP} -Vollständigkeit von 4TA-SAT.

Lösung:

• *4TA-SAT* $\in \mathcal{NP}$:

Für vier Belegungen der Variablen aus U mit Wahrheitswerten kann mit Aufwand $\mathcal{O}(6 \cdot |U| + 4 \cdot T_{\text{SAT}})$ überprüft werden, ob alle vier Belegungen voneinander verschieden sind und die Klauseln aus C erfüllen, wobei T_{SAT} den entsprechenden Aufwand für das Überprüfen einer SAT-Instanz bezeichne. Der Aufwand ist somit polynomiell in der Eingabelänge.

• *4TA-SAT* ist \mathcal{NP} -schwer:

Transformieren eine SAT-Instanz $I = (U, C)$ in eine Instanz $I' = (U', C')$ von 4TA-SAT wie folgt: Seien y_1, y_2 zwei Variablen nicht aus U . Definieren dann $U' := U \cup \{y_1, y_2\}$ und $C' := C \cup \{y_1 \vee \overline{y_1}, y_2 \vee \overline{y_2}\}$.

Diese Transformation ist offensichtlich in polynomieller Zeit (gemäß der Eingabelänge von I) durchführbar, da lediglich zwei in U nicht vorkommende Variablen sowie zwei sich direkt daraus ergebende Klauseln zu I hinzugefügt werden müssen.

bleibt zu zeigen, dass I genau dann eine Lösung hat, wenn I' eine besitzt:

Da die Klauseln $y_i \vee \overline{y_i}$ ($i \in \{1, 2\}$) für die Belegung von y_i sowohl mit **wahr** als auch **falsch** erfüllt sind, können zu einer Lösung von I vier Lösungen von I' angegeben werden, indem die Belegung von U nacheinander mit allen Kombinationen von Belegungen für y_1 und y_2 verbunden werden. Für die Rückrichtung genügt *eine* Belegung aus der Lösung für I' ; diese induziert unmittelbar eine Lösung für I .

(b) **Problem CLIQUE COLORING**

Gegeben: Graph G mit $2n$ Knoten ($n \in \mathbb{N}$), wobei G aus isolierten (untereinander nicht verbundenen) Cliques besteht.

Frage: Ist für alle möglichen Färbungen der Cliques mit den Farben rot und grün (so, dass alle Knoten einer Clique dieselbe Farbe besitzen) die Anzahl der roten Knoten in G ungleich n ?

Zeigen Sie, dass CLIQUE COLORING in $\text{co-}\mathcal{NP}$ liegt.

Lösung: CLIQUE COLORING ist das Kopproblem zu PARTITION:

Die Frage bei PARTITION besteht darin, ob es zu gegebenen Objekten mit spezifischen Gewichten eine Aufteilung der Objekte in zwei Mengen derart gibt, dass das Gesamtgewicht in beiden Mengen übereinstimmt.

Bei CLIQUE COLORING entspricht die Anzahl der Knoten einer Clique jeweils einem Objektgewicht. Die Fragestellung bei diesem Problem, ob die Gesamtzahlen in der durch die Knotenfärbung

induzierten beiden Mengen von Cliques *nicht* übereinstimmen, ist offensichtlich komplementär zu der bei PARTITION.

(c) **Integer Linear Program (ILP)**

Ein ILP besteht allgemein aus einem System von m Ungleichungen in n Variablen der Form

$$\sum_{j=1}^n a_{ij}x_j \leq b_i \quad (1 \leq i \leq m),$$

wobei alle a_{ij}, b_i, x_j ganzzahlig sind.

Problem INDEPENDENT SET

Gegeben: Graph $G = (V, E)$, Parameter $K \leq |V|$.

Frage: Gibt es eine Teilmenge $V' \subseteq V$ mit $|V'| \geq K$ so, dass für alle $u, v \in V'$ mit $u \neq v$ gilt, dass $\{u, v\} \notin E$.

Geben Sie ein ILP für INDEPENDENT SET an.

Lösung: Sei $I = ((V = \{v_1, \dots, v_n\}, E = \{e_1, \dots, e_l\}), K)$ eine Instanz von INDEPENDENT SET.

Die Variablen x_1, \dots, x_n mögen anzeigen, ob die zugehörigen Knoten v_1, \dots, v_n jeweils in V' enthalten sind, also

$$x_j := \begin{cases} 1 & \text{falls } v_j \in V' \\ 0 & \text{sonst} \end{cases}.$$

Definieren weiterhin für jede Kante e_i ($1 \leq i \leq l$) einen n -dimensionalen Vektor a_i , der in einer Komponente genau dann eine 1 besitzt, falls der entsprechende Knoten inzident zu e_i ist:

$$a_i := (a_{i1}, \dots, a_{in}) \text{ mit } a_{ij} := \begin{cases} 1 & \text{falls } v_j \in e_i \\ 0 & \text{sonst} \end{cases}.$$

Damit kann zu I folgendes ILP formuliert werden:

$$\begin{aligned} -\sum_{j=1}^n x_j &\leq -K \\ \sum_{j=1}^n a_{ij}x_j &\leq 1 \quad (1 \leq i \leq l) \\ x_j &\leq 1 \quad (1 \leq j \leq n) \\ -x_j &\leq 0 \quad (1 \leq j \leq n) \end{aligned}$$

(d) Zeigen Sie: $\text{SAT} \in \mathcal{DTAPE}(n)$, wobei n die Eingabelänge einer SAT-Instanz bezeichne.

Lösung: Idee: Aufzählen aller Wahrheitsbelegungen.

Sei $I = (U, C)$ ein beliebige Instanz von SAT mit $U = \{u_1, \dots, u_k\}$ und

$$C = \left\{ y_1^{(i)} \vee \dots \vee y_{s_i}^{(i)} : \begin{array}{l} y_j^{(i)} \in U \cup \bar{U} \cup \{\text{wahr}, \text{falsch}\}, \\ 1 \leq i \leq \ell \text{ und } s_i \geq 1 \text{ für } 1 \leq i \leq \ell \end{array} \right\}$$

Auf linearem Platz kann ein beliebiger k -Vektor w über $\{0, 1\}$ dargestellt werden, wobei $w \in \{0, 1\}^k$ einer Wahrheitsbelegung entsprechend soll:

$$w(i) = 1 \iff u_i = \text{wahr}$$

Die Verifikation, ob w für I eine erfüllende Wahrheitsbelegung ist, kann ohne Band-Speicherplatz durchgeführt werden (Hilfsvariablen und deren Zustände können direkt in die TM encodiert werden):

- initialisiere w mit $(0, 0, \dots, 0)$
- überprüfe alle Klauseln, ob sie wahr sind, d.h. jede Klausel scannen, ob mindestens ein **wahr** enthalten ist
- falls ja, dann ist w eine erfüllende Wahrheitsbelegung für I (und I hat eine Lösung)
- falls nein, erhöhe w um eins (als Bitvektor) und teste die Klauseln erneut
- falls alle Bitvektoren keine erfüllenden Wahrheitsbelegungen sind, dann hat I keine Lösung

Es nun leicht einzusehen, dass diese Verfahren I richtig entscheidet. Falls es eine erfüllende Wahrheitsbelegung gibt, dann wird sie irgendwann durch w dargestellt. Anderenfalls kann keine der Belegungen, die durch w darstellbar sind, erfüllend sein.

Aufgabe 4:

(3+3+3+3=12 Punkte)

Die Grammatiken G bzw. G' seien gegeben durch das Alphabet $\{a, b, c\}$, die Variablenmenge $\{S, T\}$, das Startsymbol S und die folgenden Regeln R bzw. R' :

$$R = \{S \rightarrow aSc \mid T \mid ac, \quad T \rightarrow aTb \mid ab\}, \quad R' = R \cup \{T \rightarrow aTc\}.$$

- (a) Geben Sie einen Kellerautomaten an, der genau die Sprache
- $L(G)$
- akzeptiert (ohne Begründung).

Lösung: $Q = \{q_a, q_b, q_c, q_J\}$, Startzustand q_a , Stack-Alphabet $\{Z_0, A\}$, Stack-Init Z_0

$$\begin{aligned} \delta(q_a, a, X) &\rightarrow (q_a, AX) && \text{für } X \in \{Z_0, A\} \\ \delta(q_a, b, A) &\rightarrow (q_b, \varepsilon) \\ \delta(q_a, c, A) &\rightarrow (q_c, \varepsilon) \\ \delta(q_b, b, A) &\rightarrow (q_b, \varepsilon) \\ \delta(q_b, c, A) &\rightarrow (q_c, \varepsilon) \\ \delta(q_b, \varepsilon, Z_0) &\rightarrow (q_J, \varepsilon) \\ \delta(q_c, c, A) &\rightarrow (q_c, \varepsilon) \\ \delta(q_c, \varepsilon, Z_0) &\rightarrow (q_J, \varepsilon) \end{aligned}$$

- (b) Bringen Sie
- G'
- durch eine systematische Konstruktion auf Chomsky-Normalform.

Lösung:

$$(i) R' = \{S \rightarrow aSc \mid T \mid ac, \quad T \rightarrow aTb \mid ab \mid aTc\}$$

- (ii) Terminalersetzung:

$$R'_1 = \{S \rightarrow Y_aSY_c \mid T \mid Y_aY_c, \quad T \rightarrow Y_aTY_b \mid Y_aY_b \mid Y_aTY_c, \quad Y_x \rightarrow x \text{ für } x \in \{a, b, c\}\}$$

- (iii) Längenbegrenzung:

$$\begin{aligned} R'_2 = \{ & S \rightarrow Y_aC_1 \mid T \mid Y_aY_c, \\ & T \rightarrow Y_aC_2 \mid Y_aY_b \mid Y_aC_3, \\ & C_1 \rightarrow SY_c, \\ & C_2 \rightarrow TY_b, \\ & C_3 \rightarrow TY_c, \\ & Y_x \rightarrow x \text{ für } x \in \{a, b, c\}\} \end{aligned}$$

- (iv) transitive Reduktion:

$$\begin{aligned} R'_3 = \{ & S \rightarrow Y_aC_1 \mid Y_aY_c \mid Y_aC_2 \mid Y_aY_b \mid Y_aC_3, \\ & T \rightarrow Y_aC_2 \mid Y_aY_b \mid Y_aC_3, \\ & C_1 \rightarrow SY_c, \\ & C_2 \rightarrow TY_b, \\ & C_3 \rightarrow TY_c, \\ & Y_x \rightarrow x \text{ für } x \in \{a, b, c\}\} \end{aligned}$$

- (c) Zeigen Sie mit Hilfe des CYK-Algorithmus, dass das Wort
- $aabc$
- in
- $L(G')$
- enthalten ist.

Lösung:

$$\begin{array}{cccc} & a & a & b & c \\ \hline Y_a & Y_a & Y_a & Y_b & Y_c \\ & \emptyset & \{S, T\} & \emptyset & \\ & \emptyset & \{C_1, C_3\} & & \\ & \{S, T\} & & & \end{array}$$

Das Wort $aabc$ ist damit in der Sprache enthalten, da S in der letzten Menge enthalten ist.

- (d) Zeigen Sie, dass die Sprache
- $L = \{a^p b^q c^r d^s : p = q + r = s; p, q, r, s \in \mathbb{N}_0\}$
- nicht kontextfrei ist.

Lösung: Annahme L ist kontextfrei, dann existiert ein $n \in \mathbb{N}$, so dass es für alle $z \in L$ mit $|z| > n$ eine Zerlegung $z = uvwx$ gibt mit:

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $\forall i \in \mathbb{N}_0: uv^i wx^i y \in L$.

Betrachte dann das Wort $z = a^{n+1} b^{n+1} d^{n+1}$. Für jede Zerlegung $z = uvwx$ mit $|vx| \geq 1$ und $|vwx| \leq n$ gilt dann genau einer der folgende Bedingungen:

- (i) $|vx|_a > 0$ und $|vx|_b = |vx|_d = 0$
- (ii) $|vx|_a > 0$, $|vx|_b > 0$ und $|vx|_d = 0$
- (iii) $|vx|_b > 0$ und $|vx|_a = |vx|_d = 0$
- (iv) $|vx|_b > 0$, $|vx|_d > 0$ und $|vx|_a = 0$
- (v) $|vx|_d > 0$ und $|vx|_a = |vx|_b = 0$

In jedem der Fälle kommt mindestens ein Buchstabe nicht vor. Da allerdings mindestens ein Buchstabe vorkommen muß, gilt $z' := uv^2 wx^2 y \neq z$ und z' kann nicht gleich viele as , bs und ds enthalten. Da z' auch keine cs enthält (da z keine cs enthält), folgt $z' \notin L$, was ein Widerspruch zur Annahme ist, dass L kontextfrei ist.

Aufgabe 5:

(12x1=12 Punkte)

Kreuzen Sie für folgende Aussagen an, ob diese wahr oder falsch sind.

Hinweis: Für jede richtige Antwort gibt es einen Punkt, für jede falsche Antwort wird ein Punkt abgezogen. Es wird keine negative Gesamtpunktzahl für diese Aufgabe gegeben.

Jede Sprache, die von einem endlichen Automaten, dessen Startzustand eine Schleife (Übergang auf sich selbst) besitzt, erkannt wird, ist unendlich.

| | |
|--------------------------|-------------------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| wahr | falsch |

Lösung:

Sei $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ ein endlicher Automat. Falls es $p, q \in Q$ gibt, für die für alle $w \in \Sigma^*$ gilt, dass wenn $\delta(p, w) \in F$, dann auch $\delta(q, w) \in F$ ist, so ist \mathcal{A} nicht zustandsminimal.

| | |
|--------------------------|-------------------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| wahr | falsch |

Lösung:

Sei L eine reguläre Sprache. Dann existiert ein $n \in \mathbb{N}$ so, dass falls es ein Wort $w \in L$ mit $|w| \geq n$ gibt, w so in uvx mit $|uv| \leq n$ und $v \neq \varepsilon$ zerlegt werden kann, dass $uv^i x \in L$ ist für mindestens ein $i \in \mathbb{N}_0$.

| | |
|-------------------------------------|--------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |
| <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |

Lösung:

Sei L eine semientscheidbare Sprache, deren Komplement entscheidbar ist. Dann ist L entscheidbar.

| | |
|-------------------------------------|--------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |
| <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |

Lösung:

Sei $H_\varepsilon := \{\langle \mathcal{M} \rangle : \text{TM } \mathcal{M} \text{ hält auf Eingabe } \varepsilon\}$, dann ist die charakteristische Funktion von H_ε berechenbar.

| | |
|--------------------------|-------------------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| wahr | falsch |

Lösung:

Die Sprache $\{wv : v \in L(\overline{w})\}$ ist entscheidbar, wobei \overline{w} das bitweise Komplement des Wortes w (jede 0 wird durch 1 ersetzt und umgekehrt) bezeichne.

| | |
|--------------------------|-------------------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| wahr | falsch |

Lösung:

Es existiert eine Transformation 2SAT α 3SAT.

| | |
|-------------------------------------|--------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |
| <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |

Lösung:

Falls $\text{CLIQUE} \in \text{co-}\mathcal{NP}$, dann gilt $\mathcal{NP} = \text{co-}\mathcal{NP}$.

| | |
|--------------------------|--------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |

Lösung:

| | |
|-------------------------------------|--------------------------|
| <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |

Für KNAPSACK existiert ein Polynomialzeitalgorithmus, falls n^4 eine obere Schranke für das Gesamtgewicht W ist (wobei n die Anzahl der Objekte sei).

| | |
|-------------------------------------|--------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |
| <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |

Lösung:

Sei G eine Typ- k -Grammatik. Dann ist der maximale Chomsky-Typ von $L(G)$ ebenfalls k .

| | |
|--------------------------|-------------------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| wahr | falsch |

Lösung:

Die Sprache

$$\left[\left(\{a^n : n \in \mathbb{N}_0\} \cdot \{c\} \cup \{a^n b^n : n \in \mathbb{N}\} \right) \cdot \{b^n : n \in \mathbb{N}_0\} \right] \cap \left[\{a^n c b^n : n \in \mathbb{N}_0\} \right]$$

| | |
|--------------------------|--------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |

ist kontextfrei.

Lösung:

| | |
|-------------------------------------|--------------------------|
| <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |

Mit Grammatiken vom Typ 1 bzw. 2 lassen sich in n Ableitungsschritten nur Wörter der Länge höchstens n erzeugen.

| | |
|--------------------------|-------------------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> |
| wahr | falsch |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| wahr | falsch |

Lösung: