Name:	Klausur-ID:
Vorname:	
Matrikelnummer:	

Lösungsvorschlag

Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. P. Sanders 19.8.2016

Nachklausur Theoretische Grundlagen der Informatik

Aufgabe 1.	Automatentheorie	5 Punkte
Aufgabe 2.	Kontextfreie Grammatiken	9 Punkte
Aufgabe 3.	Reguläre und kontextfreie Sprachen	5 Punkte
Aufgabe 4.	Turingmächtigkeit	7 Punkte
Aufgabe 5.	Berechenbarkeitstheorie	9 Punkte
Aufgabe 6.	Komplexitätstheorie	11 Punkte
Aufgabe 7.	Informationstheorie	6 Punkte
Aufgabe 8.	Kleinaufgaben	8 Punkte

Bitte beachten Sie:

- Als Hilfsmittel ist nur **ein** DIN-A4-Blatt mit Ihren **handschriftlichen** Notizen zugelassen.
- Merken Sie sich Ihre **Klausur-ID** auf dem Aufkleber für den Notenaushang.
- **Schreiben** Sie auf **alle Blätter** der Klausur und Zusatzblätter Ihre **Klausur-ID**.
- Die Klausur enthält 17 Blätter.
- Die durch Übungsblätter gewonnenen Bonuspunkte werden erst nach Erreichen der Bestehensgrenze hinzugezählt. Die Anzahl Bonuspunkte entscheidet nicht über das Bestehen.

Aufgabe		1	2	3	4	5	6	7	8	$\sum$
max. Punkte		5	9	5	7	9	11	6	8	60
Punkte	EK									
	ZK									

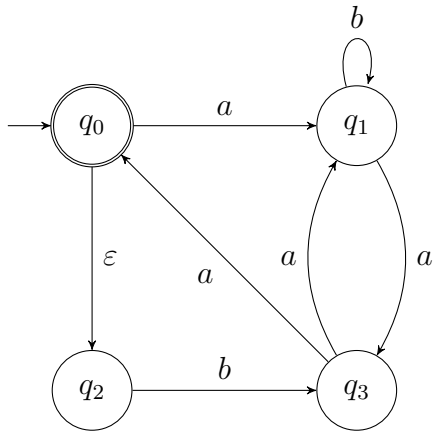
Bonuspunkte:	Summe:	**Note:**

Lösungsvorschlag

Aufgabe 1. Automatentheorie [5 Punkte]

Potenzmengenkonstruktion

Gegeben sei der endliche Automat $\mathcal{A} = (\mathcal{Q} = \{q_0, \ldots, q_3\}, \Sigma = \{a, b\}, \delta, q_0, \{q_0\})$, wobei δ durch das Zustandsdiagramm unten gegeben ist.



Geben Sie einen zu $\mathcal{A}$ äquivalenten vollständigen deterministischen endlichen Automaten tabellarisch an. Füllen Sie hierfür die gegebene Tabelle der Potenzmengenkonstruktion aus. Geben Sie Start- und Endzustände des konstruierten Automaten an.

Hinweis: Wenn Sie möchten, können Sie zuerst den ε-Abschluss von $\mathcal{A}$ bilden, dieser ist aber nicht Teil der Lösung. Eine graphische Darstellung des Automaten ist nicht gefordert.

Zustand	Übergänge	
	a	b
$\{q_0, q_2\}$	$\{q_1\}$	$\{q_3\}$
$\{q_1\}$	$\{q_3\}$	$\{q_1\}$
$\{q_3\}$	$\{q_0, q_1, q_2\}$	$\emptyset$
$\{q_0, q_1, q_2\}$	$\{q_1, q_3\}$	$\{q_1, q_3\}$
$\{q_1, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_1\}$
$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_1, q_3\}$
$\emptyset$	$\emptyset$	$\emptyset$

Der Startzustand ist $\{q_0, q_2\}$.
Die akzeptierenden Zustände sind $\{q_0, q_2\}$, $\{q_0, q_1, q_2\}$ und $\{q_0, q_1, q_2, q_3\}$.

Lösungsvorschlag

Aufgabe 2. Kontextfreie Grammatiken [9 Punkte]

a. Bestimmen Sie die Chomsky-Normalform der folgenden kontextfreien Grammatik $\mathcal{G}_a$. Geben Sie die vollständigen Produktionsregeln in jedem Schritt an. [5 Punkte]

$$\mathcal{G}_a = (\mathcal{N} = \{S, E, F, J\}, \mathcal{T} = \{a, b\}, S, \mathcal{P}_a) \quad \text{mit}$$
$$\begin{aligned}\mathcal{P}_a = \{&S \to FJ,\\ &E \to F \mid ab,\\ &F \to E \mid aFa,\\ &J \to bJE \mid bJ \mid \varepsilon\}\end{aligned}$$

Entfernen von ε-Produktionen:

Lösung

$$\begin{aligned}\mathcal{P}'_a = \{&S \to F \mid FJ,\\ &E \to F \mid ab,\\ &F \to E \mid aFa,\\ &J \to bJE \mid bE \mid bJ \mid b\}\end{aligned}$$

Lösungsende

Entfernen zyklischer Einheitsproduktionen:

Lösung

Die einzige zyklische Einheitsproduktion ist $E \leftrightarrow F$.

$$\begin{aligned}\mathcal{P}''_a = \{&S \to E \mid EJ,\\ &E \to aEa \mid ab,\\ &J \to bJE \mid bE \mid bJ \mid b\}\end{aligned}$$

Lösungsende

Entfernen nicht-zyklischer Einheitsproduktionen:

Lösung

Hier muss nur $S \to E$ geinlined werden.

$$\begin{aligned}\mathcal{P}'''_a = \{&S \to aEa \mid ab \mid EJ,\\ &E \to aEa \mid ab,\\ &J \to bJE \mid bE \mid bJ \mid b\}\end{aligned}$$

Lösungsende

(Fortsetzung und weitere Teilaufgaben auf den nächsten Blättern)

Lösungsvorschlag

Fortsetzung von Aufgabe 2

Umwandlung zu langer und gemischter rechter Seiten:

Lösung

$$\begin{aligned}\mathcal{P}_a'''' = \{\, & S \to AE_A \mid AB \mid EJ,\\ & E \to AE_A \mid AB,\\ & J \to BJ_E \mid BE \mid BJ \mid b,\\ & E_A \to EA,\\ & J_E \to JE,\\ & A \to a,\\ & B \to b\}\end{aligned}$$

Lösungsende

(weitere Teilaufgabe auf dem nächsten Blatt)

Lösungsvorschlag

Fortsetzung von Aufgabe 2

b. Gegeben sei die untenstehende Grammatik $\mathcal{G}_b$. Liegt das Wort `011101` in der von ihr erzeugten Sprache? Verwenden Sie den in der Vorlesung vorgestellten Cocke-Younger-Kasami-Algorithmus (CYK-Algorithmus) und füllen Sie die untenstehende Tabelle vollständig aus. [4 Punkte]

$$\mathcal{G}_b = (\mathcal{N} = \{S, T, U, W\}, \mathcal{T} = \{\texttt{0}, \texttt{1}\}, S, \mathcal{P}) \quad \text{mit}$$
$$\mathcal{P} = \{S \to TT,$$
$$T \to UU \mid TU \mid \texttt{0},$$
$$U \to WT \mid UT \mid WW,$$
$$W \to \texttt{1}\}$$

0	1	1	1	0	1
T	W	W	W	T	W
-	U	U	U	-	
T	-	U	-		
-	T	-			
T, S	-				
-					

Lösung

Das untere linke Kästchen enthält kein S. Dies zeigt, dass `011101` sich nicht aus dem Startsymbol ableiten lässt. Damit liegt es auch nicht in der von $\mathcal{G}_b$ erzeugten Sprache.

Lösungsende

Lösungsvorschlag

Aufgabe 3. Reguläre und kontextfreie Sprachen [5 Punkte]

a. Sei $n_x(w)$ die Anzahl Vorkommen von Zeichen x in Wort w. Zeigen oder widerlegen Sie: Die Sprache $\mathcal{L}_1 = \{w \in \{a,b\}^* \mid \frac{n_a(w)}{n_b(w)} \leq 2\}$ ist regulär. [3 Punkte]

Lösung

$\mathcal{L}_1$ ist nicht regulär. Beweis durch Widerspruch mit dem Pumpinglemma: Sei n die Zahl ab der das Pumpinglemma gilt. Wir wählen das Wort $w = a^{2n}b^n \in \mathcal{L}_1$ mit $|w| = 3n > n$. Sei $w = uvx$ eine beliebige Zerlegung gemäß des Pumpinglemmas, d.h. $|uv| \leq n$ und $|v| > 0$. Dann ist $v = a^k$ für ein $k \leq n$. Damit liegt aber $w' = uv^2x = a^k w$ nicht in $\mathcal{L}_1$, denn $n_a(w') = 2n + k > 2n$ und $n_b(w') = n$, und daher $\frac{n_a(w')}{n_b(w')} > 2$. Dies ist gemäß des Pumpinglemmas ein Widerspruch zur Regularität von $\mathcal{L}_1$.

Lösungsende

b. Zeigen oder widerlegen Sie: Die Sprache $\mathcal{L}_2$ über dem Alphabet $\Sigma = \{a,b,c\}$ ist kontextfrei. $\mathcal{L}_2 = \{a^i b^j c^k \mid i,j,k \in \mathbb{N} \text{ und } (i = j \text{ oder } j = k \text{ oder } i = k)\}$ [2 Punkte]

Lösung

$\mathcal{L}_2$ ist kontextfrei: Es ist die Vereinigung dreier kontextfreier Sprachen ($\{a^i b^i c^k\}$, $\{a^i b^j c^i\}$ und $\{a^i b^j c^j\}$, jeweils mit $i,j,k \in \mathbb{N}$). Diese sind trivial kontextfrei: $a^n b^n$ mit etwas Unrat davor, danach oder dazwischen. Da die Vereinigung kontextfreier Sprachen selbst kontextfrei ist, folgt die Behauptung.

Lösungsende

Lösungsvorschlag

Aufgabe 4. Turingmächtigkeit [7 Punkte]

Im folgenden betrachten wir das Maschinenmodell einer *Jump Machine*. Es handelt sich dabei um eine deterministische Turingmaschine, die zusätzlich zu den normalen Bewegungsmöglichkeiten (`L`, `R`, `H`) auch einen Jump-Befehl `J` besitzt. Dieser Befehl springt zu einer Zelle, welche vorher markiert wurde.

In jedem Schritt kann die gerade besuchte Stelle markiert werden (`mark`). Der Jump-Befehl `J` bewegt den Lesekopf zur zuletzt markierten Stelle. Wenn in einem Schritt sowohl markiert als auch gesprungen wird ($\texttt{J} \times \texttt{mark}$), so wird für den Sprung die alte Markierung verwendet. In diesem Fall werden also Kopfposition und Markierung vertauscht (siehe $\texttt{J} \times \texttt{mark}$-Szenario in der Graphik unten). Es wird also erst das Sprungziel eingelesen, dann die neue Markierung gesetzt, und zuletzt der Sprung ausgeführt.

Da es zu jedem Zeitpunkt genau eine Markierung geben soll ist initial die erste Zelle der Eingabe markiert. Außerdem löscht das Setzen der Markierung alle vorhergehenden Markierungen.

Formell sei

$$\mathit{JTM} = (\mathcal{Q}, \Sigma, \Gamma, \delta \ : \ \mathcal{Q} \times \Gamma \to \mathcal{Q} \times \Gamma \times \{\texttt{R}, \texttt{L}, \texttt{H}, \texttt{J}\} \times \{\texttt{mark}, -\}, q_0, \mathcal{F}), \text{ wobei}$$

- $\mathcal{Q}$ die Zustandsmenge,
- Σ das Eingabealphabet,
- Γ das Bandalphabet mit $\Sigma \cup \{\sqcup\} \subseteq \Gamma$,
- $\delta : \mathcal{Q} \times \Gamma \to \mathcal{Q} \times \Gamma \times \{\texttt{R}, \texttt{L}, \texttt{H}, \texttt{J}\} \times \{\texttt{mark}, -\}$ die Zustandsübergangsfunktion,
- $q_0 \in \mathcal{Q}$ der Startzustand und
- $\mathcal{F}$ die Menge der Endzustände ist.

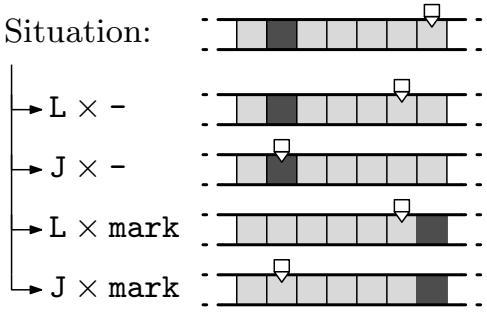


Abb.: Ausgangssituation mit verschiedenen möglichen Übergängen

a. Beschreiben Sie ein Problem, welches mit einer *JTM* asymptotisch echt schneller gelöst werden kann als mit einer normalen deterministischen Einband-Turingmaschine. Begründen Sie Ihre Antwort kurz (ohne Beweis). [2 Punkte]

Lösung

Ein typisches Problem, welches mit einer *JTM* schneller gelöst werden kann, beinhaltet normalerweise viel hin und her laufen in der normalen Lösung. Beispiel: Der Vergleich zweier gegebener Worte, oder das Verrechnen zweier Zahlen.
Um zwischen zwei Wörtern hin und her zu wechseln genügt ein `J+mark` Schritt (Markierung in Wort a, Lesekopf in Wort b), womit man sich $\mathrm{O}(n)$ Schritte sparen kann $\Rightarrow$ $\mathrm{O}(n)$ statt $\mathrm{O}(n^2)$ für den Vergleich zweier Worte.

Lösungsende

(weitere Teilaufgabe auf dem nächsten Blatt)

Lösungsvorschlag

Fortsetzung von Aufgabe 4

b. Beschreiben Sie, wie eine solche *JTM* von einer normalen deterministischen Einband-Turingmaschine simuliert werden kann. Konstruieren Sie hierzu zu einer gegebenen *JTM* $\mathcal{J}$ eine zugehörige *DTM* $\mathcal{M}_J$, welche dieselbe Funktion berechnet (gleicher Bandinhalt nach der Ausführung). [5 Punkte]

Lösung

Zunächst beschreiben wir die neue Zustandsmenge und das neue Bandalphabet:

$$\mathcal{Q}'_J = \mathcal{Q}_J \times \{\mathtt{r}, \mathtt{l}\} \times \{\mathtt{del\_r}, \mathtt{ret}, \mathtt{del}\} \cup \{q_S\}$$

$$\Gamma'_J = \Gamma_J \times \{\mathtt{m}, \mathtt{c}, -\}$$

Mit $(\cdot, \mathtt{m})$ wird die in $\mathcal{J}$ markierte Zelle markiert. Zu Beginn der Berechnung muss die erste Markierung gesetzt werden (q_S). In den Teilzuständen $(\cdot, \mathtt{r}, \cdot)$ und $(\cdot, \mathtt{l}, \cdot)$ speichern wir die Richtung, in welcher der momentan markierte Zustand liegt (vom Lesekopf aus gesehen).

Die Bewegungen des Lesekopfes können einfach umgesetzt werden. Bewegungen nach links/rechts (`L`, `R`) werden direkt umgesetzt. Jump-Befehle werden umgesetzt indem der Kopf sich zu dem letzten Marker bewegt (Die Richtung wird im Zustand gespeichert). Bei dieser Bewegung muss der aktuelle Zustand erhalten werden (Zustände $(\cdot, \cdot, \mathtt{ret})$).

Wenn eine neue Zelle markiert werden soll, so muss die alte Markierung gelöscht werden. Hierzu wird die aktuelle Position mit $(\cdot, \mathtt{c})$ markiert, danach wird der Lesekopf zum vorherigen Marker bewegt (Zustände $(\cdot, \cdot, \mathtt{del\_r})$). Dieser wird gelöscht und der Lesekopf wird zum letzten Zustand zurückbewegt (Zustände $(\cdot, \cdot, \mathtt{ret})$). Dieses Vorgehen funktioniert nicht, wenn gleichzeitig gesprungen und markiert wird. In diesem Fall wird sofort der neue Marker gesetzt, und der Lesekopf bleibt über der Zelle mit dem alten Marker, nachdem dieser gelöscht wurde $(\cdot, \cdot, \mathtt{del})$.

Da diese Konstruktion die beschriebene *JTM* Schritt für Schritt simuliert ist klar, dass die berechnete Funktion sich nicht verändert.

Um genau das selbe Ergebnis zu erzielen, muss nach der Ausführung jedes Zeichen des Bandalphabets durch das äquivalente Zeichen des ursprünglichen Bandalphabets ersetzt werden. Hierdurch wird auch die Markierung vom Band entfernt (dieser Schritt ist für die Bepunktung nicht relevant).

Lösungsende

Lösungsvorschlag

Aufgabe 5. Berechenbarkeitstheorie [9 Punkte]

Teil 1: PKP. In der Vorlesung wurde das Postsche Korrespondenzproblem (PKP) vorgestellt. Eine PKP-Instanz besteht aus einer Menge von Wortpaaren $M \subset \Sigma^* \times \Sigma^*$. Sie ist genau dann lösbar, wenn Paare $t_0, \ldots, t_m$ mit $t_i = (x_i, y_i) \in M$ gewählt werden können, sodass die beiden Wörter $x_0 \cdot \ldots \cdot x_m$ und $y_0 \cdot \ldots \cdot y_m$ gleich sind. Dabei dürfen Paare auch mehrfach verwendet werden.

a. Ist die folgende Instanz des Postschen Korrespondenzproblems lösbar? Begründen Sie. [1 Punkt]

$$M = \left\{ \begin{pmatrix} \texttt{raa} \\ \texttt{aa} \end{pmatrix}, \begin{pmatrix} \texttt{pu} \\ \texttt{mp} \end{pmatrix}, \begin{pmatrix} \texttt{i} \\ \texttt{ui} \end{pmatrix}, \begin{pmatrix} \texttt{m} \\ \texttt{raa} \end{pmatrix}, \begin{pmatrix} \texttt{pa} \\ \texttt{p} \end{pmatrix}, \begin{pmatrix} \texttt{paa} \\ \texttt{ap} \end{pmatrix} \right\}$$

Lösung

Es lässt sich das Wort `papaaraampui` bilden. Hierzu ist die Tupelfolge:

$$\begin{pmatrix} \texttt{pa} \\ \texttt{p} \end{pmatrix} \begin{pmatrix} \texttt{paa} \\ \texttt{ap} \end{pmatrix} \begin{pmatrix} \texttt{raa} \\ \texttt{aa} \end{pmatrix} \begin{pmatrix} \texttt{m} \\ \texttt{raa} \end{pmatrix} \begin{pmatrix} \texttt{pu} \\ \texttt{mp} \end{pmatrix} \begin{pmatrix} \texttt{i} \\ \texttt{ui} \end{pmatrix}$$

Lösungsende

b. Beweisen Sie: Jede Instanz des Postschen Korrespondenzproblems (PKP), in der *je zwei* zusammengehörende Teilworte x_i und y_i gleich lang sind, lässt sich in polynomieller Zeit entscheiden. Formell: $\forall_i : |x_i| = |y_i| \Rightarrow$ Instanz ist polynomiell entscheidbar. [3 Punkte]

Lösung

Jede mögliche Lösung eines PKP beginnt mit einem Worttupel $t_i = (x_i, y_i)$ in dem entweder x_i Präfix von y_i ist oder y_i Präfix von x_i ist. Wenn in jedem Tupel x_i und y_i gleich lang sind folgt daraus $x_i = y_i$. D.h. es kann nur eine Lösung geben, wenn ein Tupel existiert, dessen Teilwörter gleich sind. Die Existenz eines solchen Tupels kann in linearer Zeit (im RAM-Modell) festgestellt werden.
Wenn ein solches Tupel gefunden wird folgt daraus immer eine korrekte Lösung, da das Tupel alleine eine korrekte Lösung darstellt.

Lösungsende

(weitere Teilaufgaben auf dem nächsten Blatt)

Lösungsvorschlag

Fortsetzung von Aufgabe 5

Teil 2: Entscheidbarkeit.

c. Zeigen oder widerlegen Sie: Sprachen über einem einelementigen Alphabet ($L \subset \{1\}^*$) sind immer entscheidbar. [2 Punkte]

Lösung

Dies ist im Allgemeinen nicht der Fall. Dies lässt sich relativ einfach an der folgenden Sprache erkennen:

$$\mathcal{L}_{UHalt} = \{(\langle \mathcal{M} \rangle)_1 \mid \mathcal{M} \textit{ hält auf der leeren Eingabe}\}$$

Die Idee ist, jede Turingmaschine durch ihre Gödelnummer zu beschreiben, wofür normalerweise eine Binärcodierung verwendet wird. Wir können aber genauso gut auch eine unäre Codierung verwenden. Egal unter welcher Codierung, das Halteproblem bleibt unentscheidbar. Deshalb muss auch $\mathcal{L}_{UHalt}$ unentscheidbar sein.

Lösungsende

d. Kann eine linear beschränkte nichtdeterministische Turingmaschine (LBA) das Halteproblem für allgemeine Turingmaschinen semientscheiden? Begründen Sie Ihre Antwort! [3 Punkte]
Hinweis: Das Halteproblem für LBA ist entscheidbar

Lösung

Nein, denn das Halteproblem für LBA ist entscheidbar. Angenommen, es gäbe eine LBA $\mathcal{M}$, die das allgemeine Halteproblem semientscheiden kann. Wir geben $\langle \mathcal{M} \rangle w$ an eine NTM, die das Halteproblem für LBA löst, wobei $w = \langle \mathcal{T} \rangle$ für eine beliebige NTM $\mathcal{T}$. Wenn $\mathcal{M}$ auf w nicht hält, dann hält $\mathcal{T}$ nicht. Wir könnten $\mathcal{M}$ also dazu nutzen, das Halteproblem für allgemeine Turingmaschinen zu entscheiden — Widerspruch!

Alternative, nicht ganz korrekte Lösung (max 2 Punkte): Nein, denn die simulierte Turingmaschine darf beliebig viel Platz verwenden, was mit linearer Platzbeschränkung nicht simuliert werden kann. Beispiel: Busy Beaver-Turingmaschine. *Diese Lösung ist nicht korrekt, denn es könnte einen Algorithmus geben, der das Halteproblem semientscheidet, ohne die Eingabemaschine zu simulieren und so weniger Platz benötigt.*

Lösungsende

Lösungsvorschlag

Aufgabe 6. Komplexitätstheorie [11 Punkte]

Gegeben seien die Probleme 3SAT und 3OCC-3SAT ("Three-Occurrence 3SAT").

3SAT: Gegeben einer Menge von aussagenlogischen Variablen $x_1, \ldots, x_n$ und einer Menge von disjunktiven Klauseln $k_1, \ldots, k_m$ mit je *genau drei* Literalen, gibt es eine Belegung der Variablen die alle Klauseln erfüllt?

3OCC-3SAT: Gegeben einer Menge von aussagenlogischen Variablen $x_1, \ldots, x_n$ und einer Menge von disjunktiven Klauseln $k_1, \ldots, k_m$ mit je **zwei oder drei** Literalen, *sodass jede Variable maximal dreimal in der Klauselmenge vorkommt*, gibt es eine Belegung der Variablen die alle Klauseln erfüllt?

Beachten Sie, dass eine 3OCC-3SAT-Klausel *nicht* genau drei Literale haben muss!

Hinweis: Sie können die folgenden logischen Äquivalenzen als bekannt voraussetzen.

$$(x \Rightarrow y) \Leftrightarrow (\neg x \vee y) \qquad \neg(x \wedge y) \Leftrightarrow (\neg x \vee \neg y) \qquad \neg(x \vee y) \Leftrightarrow (\neg x \wedge \neg y)$$

$$(a \Leftrightarrow b \Leftrightarrow c \Leftrightarrow \ldots \Leftrightarrow z) \quad \Leftrightarrow \quad (a \Rightarrow b \Rightarrow c \Rightarrow \ldots \Rightarrow z \Rightarrow a)$$

a. Gegeben k Variablen $y_1, \ldots, y_k$, konstruieren Sie eine Menge von disjunktiven Klauseln für 3OCC-3SAT, die alle Variablen auf die selbe Belegung zwingt. Jede Variable darf dabei nur **zweimal** vorkommen. [3 Punkte]

Lösung

Wir konstruieren eine Implikationskette $y_1 \Rightarrow y_2 \Rightarrow \ldots \Rightarrow y_k \Rightarrow y_1$. Diese lässt sich in Teilimplikationen $y_1 \Rightarrow y_2$, $y_2 \Rightarrow y_3$, etc zerlegen und daher wie folgt als Klauselmenge ausdrücken:

$$(\neg y_1, y_2) \qquad (\neg y_2, y_3) \qquad \ldots \qquad (\neg y_{k-1}, y_k) \qquad (\neg y_k, y_1)$$

Wir erhalten also k Klauseln mit je zwei Literalen, und jede Variable kommt genau einmal negiert und einmal nichtnegiert darin vor.

Lösungsende

(weitere Teilaufgaben auf den nächsten Blättern)

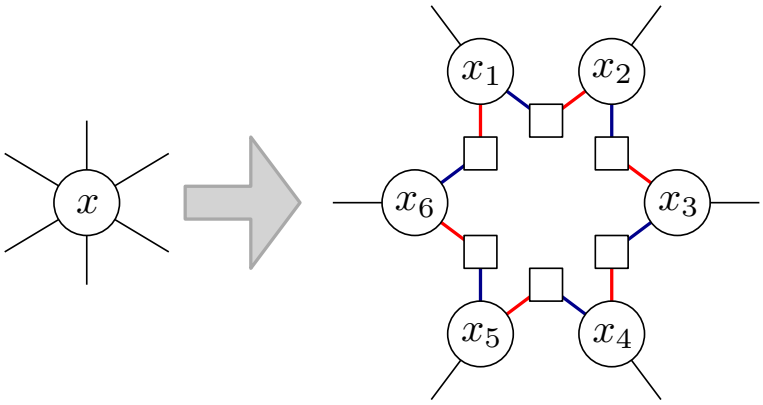
Lösungsvorschlag

Fortsetzung von Aufgabe 6

b. Wenn in einer 3SAT-Instanz eine Variable z existiert, die $k > 3$ mal vorkommt, müssen wir einen Umbau vornehmen, um eine 3OCC-3SAT-Instanz zu konstruieren. Wie können Sie Ihre Konstruktion aus Teilaufgabe **a.** verwenden, um die 3SAT-Instanz so umzubauen, dass z maximal dreimal vorkommt? [2 Punkte]

Lösung

Wir ersetzen jedes der k Vorkommen von z durch eine neue Variable z_i ($i = 1, \ldots, k$). Dann zwingen wir alle z_i mit der Konstruktion aus Teilaufgabe **a.** auf den selben Wert. Graphisch sieht das unter Verwendung der PLANAR 3SAT-Malkonvention (siehe Übung 9) für eine Variable x mit sechs Vorkommen dann so aus (nicht gefordert):



Lösungsende

(weitere Teilaufgabe auf dem nächsten Blatt)

Lösungsvorschlag

Fortsetzung von Aufgabe 6

c. Zeigen Sie, dass 3OCC-3SAT NP-vollständig ist. Verwenden Sie dazu, dass 3SAT ein NP-vollständiges Problem ist. [6 Punkte]

Hinweis: Wenn Sie die Konstruktionen der vorherigen Teilaufgaben nicht gefunden haben, können Sie deren Existenz annehmen und immernoch Teilpunkte erreichen!

Lösung

Zuerst zeigen wir, dass 3OCC-3SAT in NP liegt. Dies ist einfach zu sehen, denn gegeben einer potentiellen Lösung der 3OCC-3SAT-Instanz (diese können wir mit einer nichtdeterministischen Turingmaschine „erraten“) können wir den selben Algorithmus wie für 3SAT verwenden, um zu prüfen, ob die Instanz erfüllt ist. Da 3SAT in NP liegt, liegt also auch 3OCC-3SAT in NP.

Reduktion von 3SAT auf 3OCC-3SAT: Gegeben sei eine 3SAT-Instanz $(\mathcal{X}, \mathcal{C})$ mit Variablenmenge $\mathcal{X}$ und Klauselmenge $\mathcal{C}$. Wir konstruieren eine 3OCC-3SAT-Instanz $(\mathcal{X}', \mathcal{C}')$, sodass die beiden Instanzen lösbarkeitsäquivalent sind.

Für jede Variable $y \in \mathcal{X}$, die $k > 3$ Vorkommen in $\mathcal{C}$ hat, verwenden wir die Konstruktion aus Teilaufgabe **b.** und ersetzen y durch k neue Variablen $y_1, \ldots, y_k$ und fügen die k neuen Klauseln ein, die wir in Teilaufgabe **a.** gefunden haben. Dies erhält die Lösbarkeit der Formel: Die Klauselkette aus **a.** erzwingt, dass alle y_i den gleichen Wahrheitswert erhalten. Zudem kommt jedes y_i nun genau dreimal vor: Zweimal in der Kette (einmal als y_i und einmal als $\neg y_i$) und einmal in einer „Nutzklausel“, in der y durch y_i ersetzt wurde.

Sobald diese Ersetzung für alle Variablen vorgenommen wurde, haben wir eine 3OCC-3SAT-Instanz mit maximal $|\mathcal{X}| \cdot 3|\mathcal{C}|$ Variablen (keine Variable kann mehr als $3|C|$ Vorkommen gehabt haben) und höchstens $|\mathcal{C}| + |\mathcal{X}| \cdot 3|\mathcal{C}|$ Klauseln. Daher ist die Reduktion polynomiell. Da wir oben bereits gezeigt haben, dass die Instanzen lösbarkeitsäquivalent sind, folgt, dass 3OCC-3SAT NP-schwer ist. Da es außerdem in NP liegt, folgt, dass das Problem 3OCC-3SAT NP-vollständig ist. □

Zum Abschluss noch eine interessante Beobachtung. Die Formulierung „Klauseln mit je zwei oder drei Literalen“ ist tatsächlich wichtig! Sie dient nicht dazu, die Aufgabe einfacher zu machen—wenn wir genau drei Literale pro Klausel fordern, ist das Problem trivial in P*, denn es lässt sich zeigen, dass jede Instanz des Problems erfüllbar ist (Craig Tovey, 1984)!*

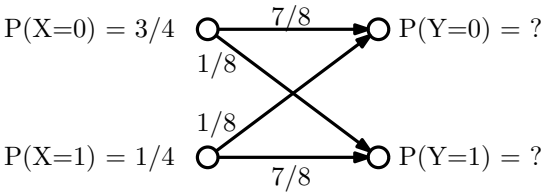
Lösungsende

Lösungsvorschlag

Aufgabe 7. Informationstheorie [6 Punkte]

a. Gegeben sei der untenstehende symmetrische binäre Kanal.

Berechnen Sie zunächst die Zeichenwahrscheinlichkeiten an der Senke. Verwenden Sie diese Wahrscheinlichkeiten um die Entropie $H(X)$ an der Quelle und $H(Y)$ an der Senke zu bestimmen. Verwenden Sie zur Vereinfachung die untenstehende approximierte Logarithmentabelle.



[5 Punkte]

Lösung

Vorbereitend:

$$P(X=0,Y=0) = \frac{3}{4}\cdot\frac{7}{8} = \frac{21}{32}$$
$$P(X=0,Y=1) = \frac{3}{4}\cdot\frac{1}{8} = \frac{1}{32}$$
$$P(X=1,Y=0) = \frac{1}{4}\cdot\frac{1}{8} = \frac{3}{32}$$
$$P(X=1,Y=1) = \frac{1}{4}\cdot\frac{7}{8} = \frac{7}{32}$$

Hauptaufgabe:

$$P(Y=0) = \frac{21}{32} + \frac{1}{32} = \frac{22}{32} = \frac{11}{16}$$
$$P(Y=1) = \frac{3}{32} + \frac{7}{32} = \frac{10}{32} = \frac{5}{16}$$

$$\begin{aligned} H(X) &= -\sum_x P(X=x)\log_2 P(X=x) \\ &= -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = -\frac{3}{4}(\log_2 3 - 2) - \frac{1}{4}(0-2) \\ &= 2 - \frac{3}{4}\log_2 3 \approx 2 - \frac{3}{4}\cdot\frac{8}{5} = \frac{4}{5} = 0.8 (\approx 0.811) \end{aligned}$$

$\log_2 3$	$\log_2 5$	$\log_2 6$	$\log_2 7$	$\log_2 10$	$\log_2 11$
$\frac{8}{5}$	$\frac{56}{25}$	$\frac{13}{5}$	$\frac{14}{5}$	$\frac{83}{25}$	$\frac{192}{55}$

Tipp: Brüche in Logarithmen lassen sich vereinfachen: $\log_2 \frac{x}{y} = \log_2 x - \log_2 y$

(Fortsetzung und weitere Teilaufgaben auf dem nächsten Blatt)

$$
\begin{aligned}
H(Y) &= -\sum_y P(Y=y)\log_2 P(Y=y) \\
&= -\frac{11}{16}\log_2\frac{11}{16} - \frac{5}{16}\log_2\frac{5}{16} = -\frac{11}{16}(\log_2 11 - 4) - \frac{5}{16}(\log_2 5 - 4) \\
&= 4 - \frac{11}{16}\log_2 11 - \frac{5}{16}\log_2 5 \approx 4 - \frac{11}{16}\cdot\frac{192}{55} - \frac{5}{16}\cdot\frac{56}{25} = \frac{9}{10} = 0.9 (\approx 0.896)
\end{aligned}
$$

Lösungsende

b. Bei welchen Ausgangs-Wahrscheinlichkeiten (an der Quelle) wird die übermittelte Information (Transinformation) bei diesem Kanal maximal? Begründen Sie kurz. [1 Punkt]

Lösung

Der gegebene Kanal ist symmetrisch, dementsprechend wird die maximale Transinformation bei einer uniformen Ausgangsverteilung maximal. Es muss dementsprechend gelten: $P(X=0) = P(X=1) = 1/2$.

Lösungsende

Lösungsvorschlag

Aufgabe 8. Kleinaufgaben [8 Punkte]

a. Ist es in polynomieller Zeit möglich zu entscheiden, ob zwei deterministische endliche Automaten die selbe Sprache entscheiden? Begründen Sie Ihre Antwort kurz. [1 Punkt]

Lösung

Minimiere beide Automaten. Der resultierende Automat ist eindeutig. Die Gleichheit der Automaten kann in linearer Zeit geprüft werden, wenn man vom Startzustand ausgeht.

Lösungsende

b. Ist das Wortproblem für Sprachen vom Chomsky-Typ 1 entscheidbar? Begründen Sie Ihre Antwort kurz. [1 Punkt]

Lösung

Ja, denn wir haben in der Vorlesung einen Algorithmus dafür gesehen, der den Ableitbarkeitsgraphen aufbaut. Das Problem ist entscheidbar, weil ein Wort durch Anwendung einer Produktionsregel nie kürzer werden kann, also ist der Ableitbarkeitsgraph endlich.

Lösungsende

c. Sind deterministische und nichtdeterministische Kellerautomaten gleich mächtig? Begründen Sie Ihre Antwort kurz! [1 Punkt]

Lösung

Nein. Beispiel: $L = \{ww^R \mid w \in \Sigma^*\}$, ein deterministischer Kellerautomat weiß nicht, wann die Wortmitte erreicht ist und er wieder Symbole vom Stack nehmen muss. Der nichtdeterministische Kellerautomat kann die Mitte des Worts „erraten“.

Lösungsende

d. Ist 3COLOUR in NP? Kann TRAVELLING SALESMAN (TSP) in Polynomialzeit darauf reduziert werden? Begründen Sie kurz. [1 Punkt]

Lösung

Beide Aussagen sind wahr. 3COLOUR $\in$ NP da eine Farbbelegung in polynomieller Zeit geprüft werden kann (prüfe für jede Kante ob die Endpunkte unterschiedliche Farben haben). Außerdem wissen wir aus der Vorlesung, dass sowohl 3COLOUR als auch TSP NP-vollständig sind, daher existiert die geforderte polynomielle Reduktion.

Lösungsende

(weitere Teilaufgaben auf dem nächsten Blatt)

Lösungsvorschlag

Fortsetzung von Aufgabe 8

e. Gilt die folgende Behauptung? Begründen Sie kurz. Wenn für einen Code über n Zeichen, der Zeichen i mit ℓ_i Binärzeichen codiert, die Kraftsche Ungleichung $\sum_{i=1}^{n} 2^{-\ell_i} \leq 1$ erfüllt ist, dann ist der Code eindeutig decodierbar. [1 Punkt]

Lösung

Nein, die Implikation in der Kraftschen Ungleichung geht in die andere Richtung. Beispielsweise erfüllt der Code aus der folgenden Teilaufgabe die Ungleichung, ist aber nicht eindeutig decodierbar.

Lösungsende

f. Ist der durch die (Zeichen, Code)-Paare $(a, 001)$, $(b, 10)$, $(c, 010)$, $(d, 0011)$ gegebene Code eindeutig decodierbar? Begründen Sie kurz. [1 Punkt]

Lösung

Nein, denn abb und dc werden beide mit 0011010 codiert.

Lösungsende

g. Wenn bei einem Huffmancode die Buchstabenhäufigkeiten einem Präfix der Fibonaccifolge entsprechen, hat der Huffmanbaum lineare Höhe. Begründen Sie.

Hinweis: Die Fibonaccifolge ist definiert als $F_0 = 0$, $F_1 = 1$, $F_i = F_{i-1} + F_{i-2}$. Zudem gilt $\sum_{i=0}^{n-2} F_i = F_n - 1$. [2 Punkte]

Lösung

Die Summe der Häufigkeiten der k seltensten Zeichen ist immer um eins geringer als die Häufigkeit des übernächsten Zeichens (gegebene Gleichung). Daher werden diese immer mit dem nächstselteneren Zeichen kombiniert und wir erhalten einen Baum linearer Höhe.

Lösungsende