Name:
Vorname:
Matrikelnummer:

Klausur-ID:

Lösungsvorschlag

Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. P. Sanders 4.4.2016

Klausur Theoretische Grundlagen der Informatik

Aufgabe 1.	Automatentheorie	10 Punkte
Aufgabe 2.	Kontextfreie Grammatiken	5 Punkte
Aufgabe 3.	Reguläre und kontextfreie Sprachen	6 Punkte
Aufgabe 4.	Turingmächtigkeit	6 Punkte
Aufgabe 5.	Berechenbarkeitstheorie	7 Punkte
Aufgabe 6.	Komplexitätstheorie	13 Punkte
Aufgabe 7.	Informationstheorie	6 Punkte
Aufgabe 8.	Kleinaufgaben	7 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 18 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		10	5	6	6	7	13	6	7	60
Punkte	EK									
	ZK									

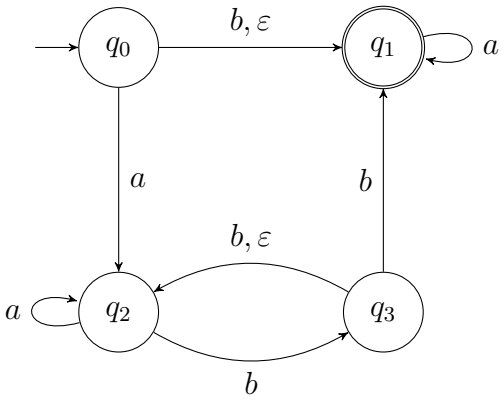
Bonuspunkte:	Summe:	**Note:**

Lösungsvorschlag

Aufgabe 1. Automatentheorie [10 Punkte]

a. Potenzmengenkonstruktion

Gegeben sei der endliche Automat $\mathcal{A} = (\mathcal{Q} = \{q_0, q_1, q_2, q_3\}, \Sigma = \{a, b\}, \delta, q_0, \mathcal{F} = \{q_1\})$, wobei δ durch das Zustandsdiagramm unten gegeben ist.



Geben Sie einen zu $\mathcal{A}$ äquivalenten vollständigen deterministischen endlichen Automaten an. Füllen Sie hierfür die gegebene Tabelle der Potenzmengenkonstruktion aus. [5 Punkte]

Hinweis: Wenn Sie möchten, können Sie zuerst den ε-Abschluss von $\mathcal{A}$ bilden. Dieser ist aber genau wie eine graphische Darstellung des deterministischen Automaten ***nicht*** *gefordert.*

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

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

Häufige Fehler: ε-Übergänge vergessen, bspw. beim Startzustand

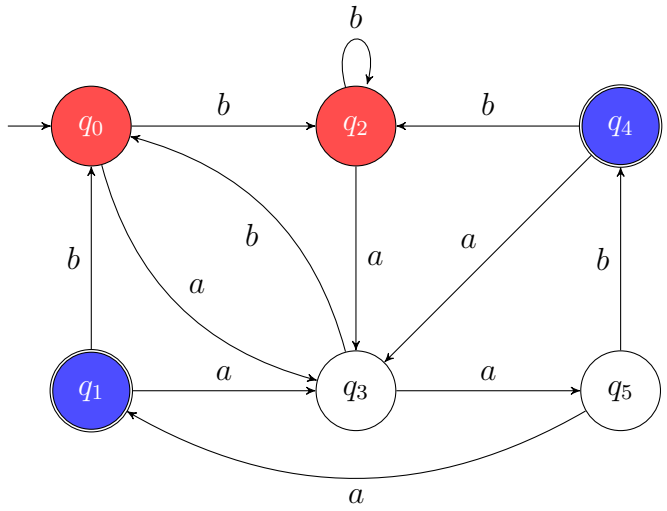
(weitere Teilaufgabe auf dem nächsten Blatt)

Lösungsvorschlag

Fortsetzung von Aufgabe 1

b. Automatenminimierung

Minimieren Sie den Automaten $\mathcal{B} = (\mathcal{Q} = \{q_0, \ldots, q_5\}, \Sigma = \{a, b\}, \delta, q_0, \{q_1, q_4\})$, dessen Zustandsübergangsfunktion durch das folgende Zustandsdiagramm gegeben ist. Verwenden Sie hierzu das Verfahren aus der Vorlesung und füllen Sie die gegebene Tabelle aus. Markieren Sie außerdem äquivalente Zustände im dargestellten Zustandsdiagramm.



[5 Punkte]

	q_0	q_1	q_2	q_3	q_4	q_5
q_0						
q_1	ε					
q_2		ε				
q_3	aa, ab	ε	aa, ab			
q_4	ε		ε	ε		
q_5	a, b	ε	a, b	a, b	ε	

Lösung

Zustände q_0 und q_2 sowie q_1 und q_4 sind äquivalent.

Lösungsende

Lösungsvorschlag

Aufgabe 2. Kontextfreie Grammatiken [5 Punkte]

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

$$\begin{aligned}\mathcal{P} = \{\, & S \to AT \mid UB \mid TU, \\ & T \to TA \mid b, \\ & U \to TU \mid BT, \\ & A \to a, \\ & B \to b \,\}\end{aligned}$$

a. Lässt sich mit der gegebenen Grammatik der in der Vorlesung vorgestellte CYK-Algorithmus durchführen? Begründen Sie kurz oder überführen Sie wenn nötig die Grammatik in die benötigte Form. [1 Punkt]

Lösung

Die Grammatik befindet sich bereits in Chomsky-Normalform, der CYK-Algorithmus kann also direkt angewendet werden.

Lösungsende

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

b	b	a	a	b
B, T	B, T	A	A	B, T
U	T	-	S	
U	T	-		
U	-			
S				

Lösung

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

Lösungsende

Lösungsvorschlag

Aufgabe 3. Reguläre und kontextfreie Sprachen [6 Punkte]

a. Zeigen oder widerlegen Sie:
Die Sprache $\mathcal{L}_1 = \{\{a,b\}^n \cdot \{a,b\}^n \mid n \in \mathbb{N}\}$ ist regulär. [1 Punkt]

Lösung

$\mathcal{L}_1$ enthält genau die Worte gerader Länge über $\{a,b\}^*$ (außer ε):
$\mathcal{L}_1 = \{w \in \{a,b\}^* \mid |w| \geq 2 \wedge |w| \equiv 0 \mod 2\}$. Damit ist $\mathcal{L}_1$ trivialerweise regulär.

Lösungsende

b. Zeigen oder widerlegen Sie: Die Sprache $\mathcal{L}_2$ ist kontextfrei.
$\mathcal{L}_2 = \left\{w_1 \cdot w_2 \cdot \ldots \cdot w_n \mid n \in \mathbb{N} \wedge \forall_{k \in \{1,\ldots,n\}} : w_k \in \{a,b\}^k\right\}$ (also $|w_k| = k$) [5 Punkte]

Lösung

Pumpinglemma für kontextfreie Sprachen:

$$\exists\, n \in \mathbb{N} \quad \forall\, w \in L_2 \mid |w| \geq n$$
$$\exists\, u,v,x,y,z \in \Sigma^* \mid w = uvxyz \wedge |vxy| \leq n \wedge |vy| > 0$$
$$\text{so dass } \forall\, i \in \mathbb{N}_0 \quad uv^ixy^iz \in L_2$$

$\mathcal{L}_2$ ist nicht kontextfrei. Beweis durch Widerspruch mit dem Pumpinglemma:

Angenommen, $\mathcal{L}_2$ sei kontextfrei. Sei n der Wert ab dem das Pumpinglemma gilt. Wähle ein beliebiges Wort $w = w_1 \cdot w_2 \cdot \ldots \cdot w_n \in \mathcal{L}_2$ sodass $|w_i| = i$ für alle $i = 1,\ldots,n$. Wir beobachten, dass dann $|w| = \sum_{i=1}^{n} i = \frac{n^2+n}{2} \geq n$ gilt.

Betrachte nun eine beliebige Zerlegung $w = uvxyz$ gemäß des Pumpinglemmas, d.h. $|vxy| \leq n$ und $|vy| > 0$. Für $w' = uv^2xy^2z$ gilt dann, dass w' länger ist als w. Sollte w' in $\mathcal{L}_2$ liegen, muss es also aus mindestens $n+1$ Teilworten $w'_1,\ldots,w'_{n+1}$ bestehen. Da die Länge von w' aber maximal $|w| + |vy| \leq |w| + |vxy| \leq |w| + n$ ist, ist dies nicht möglich: Ein Wort mit $n+1$ Bestandteilen müsste Länge $|w| + n + 1$ haben. Das Wort w' kann also nicht in $\mathcal{L}_2$ liegen. Dies ist ein Widerspruch zur Annahme. Daher kann $\mathcal{L}_2$ nicht kontextfrei sein.

Häufige Fehler:

- Pumpinglemma falsch angewendet
- Nicht erkannt, dass nur die Länge des Wortes relevant ist, nicht aber die Zeichen, aus denen es besteht
- Länge falsch berechnet, beispielsweise $n!$ statt $\frac{n^2+n}{2}$

Lösungsende

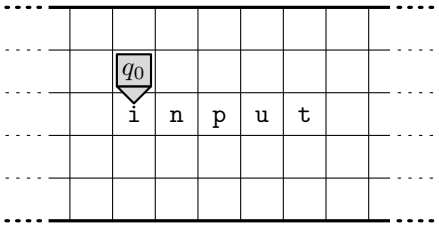
Lösungsvorschlag

Aufgabe 4. Turingmächtigkeit [6 Punkte]

Im Folgenden betrachten wir das Maschinenmodell einer deterministischen $5\times\infty$-Turingmaschine. Es handelt sich dabei um eine zweidimensionale Turingmaschine, deren Band in horizontale Richtung unendlich, in vertikale Richtung aber auf 5 Felder beschränkt ist. In jedem Schritt kann der Lesekopf nach oben, unten, links, rechts oder gar nicht bewegt werden (O, U, L, R, H). Die Eingabe steht zu Beginn der Berechnung auf der mittleren Reihe des zweidimensionalen Bandes (siehe Abbildung). Formell sei

$$\mathit{5ITM} = (\mathcal{Q}, \Sigma, \Gamma, \delta : \mathcal{Q} \times \Gamma \to \mathcal{Q} \times \Gamma \times \{\mathtt{O}, \mathtt{U}, \mathtt{L}, \mathtt{R}, \mathtt{H}\}, q_0, \mathcal{F}), \text{ wobei}$$

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



a. Begründen Sie kurz, wieso jede Sprache, die von einer normalen Turingmaschine erkannt werden kann, auch von einer $5\times\infty$-Turingmaschine erkannt werden kann. [1 Punkt]

Lösung

Jede normale Turingmaschine kann als eine $5\times\infty$-Turingmaschine betrachtet werden, die sich nur innerhalb einer Reihe bewegt. Da die Eingabe bereits in einer Reihe gegeben ist, erkennt die durch eine Turingmaschine induzierte $5\times\infty$-Turingmaschine die selbe Sprache wie die ursprüngliche Turingmaschine.

Lösungsende

b. Zeigen Sie: Jede $5\times\infty$-Turingmaschine kann von einer normalen Turingmaschine mit einem Band simuliert werden. Geben Sie hierzu eine Konstruktion an, die für jede $5\times\infty$-Turingmaschine eine Einband-Turingmaschine erstellt, welche die selbe Sprache akzeptiert (die selbe Berechnung durchführt). Sie können davon ausgehen, dass die gegebene $5\times\infty$-Turingmaschine sich nie außerhalb des gegebenen Bandes bewegt (nach oben oder unten über die Bandgrenzen). [5 Punkte]

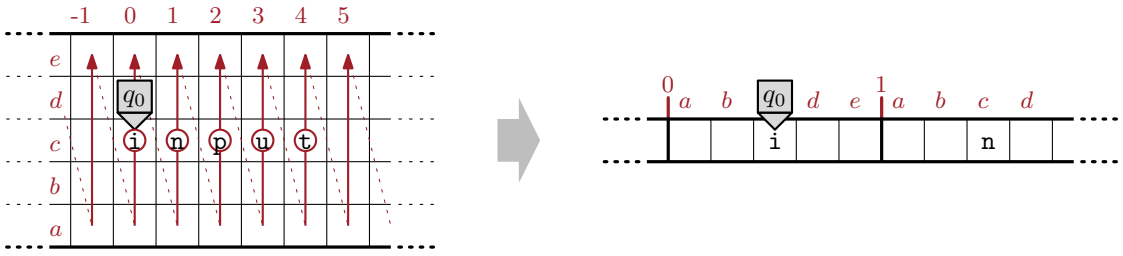
(mehr Platz auf dem nächsten Blatt)

Lösungsvorschlag

Fortsetzung von Aufgabe 4

Lösung

Es gibt drei grundlegende Lösungsansätze. Die erste Variante basiert darauf, das zweidimensionale Band (spaltenweise) auf dem eindimensionalen Band zu verteilen. Die zweite Variante basiert darauf, das Bandalphabet so zu vergrößern, dass eine gesamte Spalte der $5\times\infty$-Turingmaschine innerhalb einer Zelle gespeichert werden kann ($\mathcal{Q}' = \mathcal{Q}^5$). Die dritte Variante bildet den ausgefüllten Teil des Bandes zeilenweise auf das eindimensionale Band ab. In unserer Erklärung konzentrieren wir uns zunächst auf Varianten 1 und 2, Variante 3 erfordert eine signifikant andere Konstruktion und wird deshalb gegen Ende der Lösung abgehandelt.
Es ist klar, dass zunächst das 2D-Band auf ein 1D-Band abgebildet werden muss. In Variante 1 speichern wir hierzu die Spalten des 2D-Bandes in aufeinander folgenden Zellen des 1D-Bandes. Jeweils 5 Zellen des 1D-Bandes speichern eine Spalte des 2D-Bandes von unten nach oben (siehe Abbildung).



Um die Startkonfiguration einer 1D-Turingmaschine in die korrespondierende Startkonfiguration der simulierten 2D-Turingmaschine zu überführen, muss die Eingabe entsprechend verteilt werden. D.h. die gegebene Eingabe muss auf Zellen verteilt werden, die der mittleren Reihe entsprechen. Dies ist offensichtlich mit einer Turingmaschine (mit konstanter Zustandszahl und Laufzeit $\mathrm{O}(n^2)$) lösbar. Dieser Schritt kann bei der Variante 2 entfallen, da hier einzelne (1D-) Zellen ganze (2D-)Spalten enthalten, hierdurch ergibt sich eine sehr einfache eins zu eins Abbildung.
Nach der ursprünglichen Abbildung kann die $5\times\infty$-Turingmaschine Schritt für Schritt simuliert werden. Jeder Schritt beginnt mit dem Kopf auf der korrekten Zelle (korrespondierend zu der Zelle, auf welcher die 2D-Turingmaschine sich befinden würde). Diese Zelle wird verändert, genau wie durch die 2D-Turingmaschine es täte. Danach muss der Kopf der 1D-Turingmaschine über die neue Zelle bewegt werden. Um den Kopf auf dem simulierten 2D-Band eine Zelle nach oben (unten) zu bewegen, bewegen wir den Kopf auf dem 1D-Band nach rechts (links). Da der Kopf sich (nach Aufgabenstellung) nicht außerhalb des Bandes bewegt, benötigen wir keine speziellen Abfragen. Um den simulierten Kopf nach rechts (links) zu bewegen muss der Kopf der 1D-Turingmaschine um 5 Zellen nach rechts (links) bewegt werden. Dies lässt sich relativ einfach lösen, jedoch muss hierzu die Zustandsmenge vergrößert werden. Für jeden ursprünglichen Zustand q werden (8) zusätzliche Zustände eingefügt welche die Bewegung um 5 Zellen in beide Richtungen kontrollieren ($q_{-4}, \ldots q_4$).
In der alternativen Lösung (Variante 2) wird die Y-Position des Kopfes in dessen Zustand gespeichert. Hierzu wird die Zustandsmenge um einen Faktor 5 erhöht ($q \Rightarrow q_{Y=1}, \ldots, q_{Y=5}$). Eine Bewegung nach oben (unten) wird dementsprechend durch eine Veränderung des Zustands abgebildet (der Lesekopf bleibt an der selben Stelle). Bewegungen nach rechts (links) werden durch entsprechende Bewegungen auf dem eindimensionalen Band abgebildet.
Nach der Bewegung kann der nächste Schritt der 2D-Turingmaschine simuliert werden. Am

Ende der Berechnung befindet sich die Turingmaschine in einer Konfiguration, die zu der Endposition der entsprechenden $5 \times \infty$-Turingmaschine korrespondiert. Damit ist klar, dass sie die selbe Sprache entscheidet bzw die selbe Funktion berechnet.

Variante 3: Für Variante 3 wird zunächst die folgende Beobachtung benötigt: Obwohl das 2D-Band eine unendliche (x-) Ausdehnung besitzt, ist zu jedem Zeitpunkt der Berechnung nur ein endlicher Bereich davon ausgefüllt. Dieser Bereich kann zeilenweise auf das 1D-Band übertragen werden, die einzelnen Zeilen werden dabei durch Trennsymbole getrennt ($\# \notin \Gamma$). Auch diese Konfiguration muss zunächst aus der Eingabe der 1D Turingmaschine erzeugt werden (Bsp. `#...#...#input#...#...#`).
In der Simulation verbergen sich nun verschiedene Tücken, die mit den vorherigen Varianten einfacher umgangen werden. Zunächst muss die `O`/`U`-Bewegung umgesetzt werden; dabei ist vor allem wichtig, dass die x-Position des Kopfes erhalten wird. Damit dies überhaupt möglich ist, muss zunächst die relative Position der Bandabschnitte (#...) zueinander berücksichtigt werden. Hierzu gibt es verschiedene Möglichkeiten, z.B. eine markierte Zelle in jedem Band die sich mit dem Kopf in x-Richtung mitbewegt (regelmäßig aktualisieren), oder man stellt sicher, dass alle Abschnitte gleich lang sind (gleiche x-Position entspricht gleicher Position innerhalb des Abschnitts). Die eigentliche Bewegung nach oben und unten ist abhängig von der gewählten Methode.
Eine weitere Tücke ist, dass jeder einzelne Abschnitt immer nur eine begrenzte Länge besitzt. Dementsprechend muss der Abschnitt vergrößert werden, sobald die simulierte Turingmaschine an diese Grenzen stößt. Dabei müssen alle weiteren Bandinhalte auf dem 1D-Band verschoben werden.

Wichtige Schritte der Konstruktion, welche für volle Punkte benötigt werden:

- Wie wird das zweidimensionale Band gespeichert?
  - Werden hierfür neue Bandsymbole benötigt?
  - Wie wird das 2D-Band auf ein 1D-Band abgebildet (Spalte auf Zelle oder Zelle auf Zelle)?
  - Wie wird die Kopfposition auf dem Band dargestellt?
- Wird beschrieben, wie die Startkonfiguration verändert wird (falls nötig)?
- Schritt für Schritt Simulation der $5 \times \infty$-Turingmaschine: Wie wird der Kopf bewegt?
  - besonders wichtig: wird die Nach oben/unten Bewegung beschrieben. Wird sichergestellt, dass die x-Position des Kopfes erhalten bleibt.

Lösungsende

Lösungsvorschlag

Aufgabe 5. Berechenbarkeitstheorie [7 Punkte]

a. Begründen Sie kurz: Eine linear beschränkte nichtdeterministische Turingmaschine (LBA) kann nicht unendlich lange laufen, ohne eine Konfiguration zu wiederholen. Was bedeutet das für das Halteproblem für LBAs? [3 Punkte]

Lösung

Da sie nur linear viel Platz, ein endliches Alphabet und endlich viele Zustände hat, hat eine LBA nur endlich viele verschiedene Konfigurationen $\Rightarrow$ wenn die Maschine unendlich lange läuft, muss sie zwingend eine Konfiguration wiederholen.

Das Halteproblem für LBAs ist entscheidbar: Eine LBA läuft genau dann unendlich lange, wenn sich auf *jedem* Ausführungspfad eine Konfiguration wiederholt. Betrachte die Konfigurationen und Übergänge als Graph. Da die Wiederholung nach endlich vielen Schritten eintritt (s.o.), und der Konfigurationsgraph endlich ist, lässt er sich durch eine Tiefensuche explorieren.

Lösung für deterministische linear beschränkte Turingmaschinen (-1 Punkt): Nur wenn sich eine Konfiguration wiederholt, hält die Maschine nicht. Diese Wiederholung muss nach endlich vielen Schritten eintreten. Simuliere also die LBA und merke bisher gesehene Konfigurationen. Bei Konfigurationswiederholung hält die Maschine sicher nicht.

Lösungsende

b. Gegeben sei die Sprache $\mathcal{L}$:

$$\mathcal{L} = \{\langle \mathcal{M} \rangle \mid \mathcal{M} \text{ ist eine Turingmaschine die eine gegebene Zahl } x \text{ inkrementiert.}\}$$

Beweisen Sie, dass diese Sprache nicht entscheidbar ist. [4 Punkte]

Lösung

Wir werden zeigen, dass eine Turingmaschine welche $\mathcal{L}$ entscheidet auch das Halteproblem lösen kann.
Hierzu konstruieren wir für jede Instanz $I = (\mathcal{M}_{Halt}, w)$ des Halteproblems eine Turingmaschine $\mathcal{M}_I$ welche genau dann die Eingabe inkrementiert, wenn die Instanz I hält. Es ist klar, dass falls eine Turingmaschine $\mathcal{M}_{Test}$ existiert, welche $\mathcal{L}$ entscheidet, so könnte $\mathcal{M}_{Test}$ auch das Halteproblem entscheiden.
Konstruktion von $\mathcal{M}_I$ (Vorgehen von $\mathcal{M}$):

1. Schreibe I auf das Band (sichere dabei die ursprüngliche Eingabe z.B. durch vergrößertes Bandalphabet)
2. Simuliere die Turingmaschine $\mathcal{M}_{Halt}$ auf der Eingabe w
3. Setze das Band zurück auf die ursprüngliche Eingabe
4. Inkrementiere die Eingabe

Da das Halteproblem unentscheidbar ist, und eine Turingmaschine welche $\mathcal{L}$ entscheidet das Halteproblem lösen könnte, kann keine Turingmaschine existieren, welche $\mathcal{L}$ entscheidet.

Häufige Fehler:

- Satz von Rice als Beweis angeführt. Wie in der Übung besprochen ist das kein ausreichender Beweis.
- Einzelne Mengen haltender Maschinen können einfacher zu entscheiden sein als das allgemeine Halteproblem.
- Selbst einen Entscheider zu konstruieren und dann diesen zum Widerspruch zu führen zeigt nicht, dass das Problem unentscheidbar ist, sondern ist nur eine Aussage über diesen Entscheider. Der Entscheider muss als Black Box betrachtet werden.

Lösungsende

Lösungsvorschlag

Aufgabe 6. Komplexitätstheorie [13 Punkte]

Gegeben seien die Probleme ILP, VERTEX COVER und DOMINATING SET:

ILP: Gegeben seien eine Menge von ganzzahligen Variablen $\mathcal{X} = \{x_1, \ldots, x_n\}$ und eine Menge von Constraints $\mathcal{C} = \{c_1, \ldots, c_m\}$. Constraints haben die Form

$$c_i : \quad t_{i,1}x_1 + \ldots + t_{i,n}x_n \ \langle op \rangle \ t_i$$

mit Operator $\langle op \rangle \in \{\leq, =, \geq\}$ und $t_i,\ t_{i,j} \in \mathbb{Z}$ für $j \in \{1, \ldots, n\}$. Existiert eine ganzzahlige Belegung der $x_1, \ldots, x_n$, die alle Constraints erfüllt?

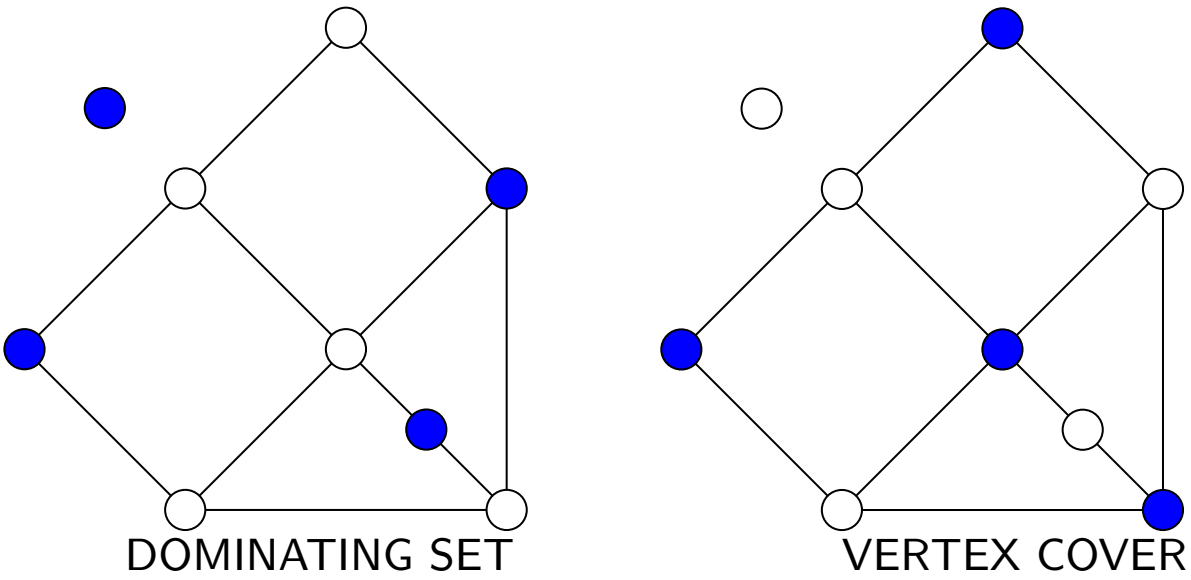
VERTEX COVER: Gegeben seien ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$. Existiert eine Teilmenge X von V der Größe höchstens k, sodass jede *Kante* des Graphen zu mindestens einem Knoten aus X inzident ist? Formell:

$$\exists_{X \subseteq V,\ |X| \leq k} : \forall_{e=\{u,v\} \in E} : (u \in X \vee v \in X)$$

DOMINATING SET: Gegeben seien ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$. Existiert eine Teilmenge Y von V der Größe höchstens k, sodass jeder *Knoten* des Graphen zu mindestens einem Knoten aus Y adjazent ist? Formell:

$$\exists_{Y \subseteq V,\ |Y| \leq k} : \forall_{v \in V} : (v \in Y \quad \vee \quad \exists_{u \in V} : (\{u, v\} \in E \wedge u \in Y))$$

a. Zeichnen Sie eine Lösung in die DOMINATING SET- und VERTEX COVER-Instanzen ein, die durch $k = 4$ und die untenstehenden Graphen gegeben sind. [2 Punkte]



Häufige Fehler: Isolierten Knoten in der DOMINATING SET-Instanz nicht beachtet

(weitere Teilaufgaben auf den nächsten Blättern)

Lösungsvorschlag

Fortsetzung von Aufgabe 6

b. Stellen Sie eine gegebene DOMINATING SET-Instanz $(G=(V,E),k)$ mit $V=\{u_1,\ldots,u_n\}$ als ILP-Instanz $(\mathcal{X},\mathcal{C})$ dar. Die ILP-Instanz soll genau dann erfüllbar sein, wenn die DOMINATING SET-Instanz lösbar ist. [3 Punkte]

Lösung

Für Variablen $\mathcal{X}=\{x_1,\ldots,x_n\}$ benötigen wir folgende Constraints:

$$\sum_{u_i\in V} x_i \leq k \quad \text{erzwingt die Kardinalität des Dominating Set}$$

$$\bigcup_{u_i\in V}\left(x_i+\sum_{\substack{u_j\in V,\\ \{u_i,u_j\}\in E}} x_j\right)\geq 1 \quad \text{damit } u_i \text{ oder einer seiner Nachbarn ausgewählt werden}$$

$$\bigcup_{u_i\in V} x_i \geq 0 \quad \text{und}$$

$$\bigcup_{u_i\in V} x_i \leq 1 \quad \text{um die Variablen auf } \{0,1\} \text{ zu zwingen}$$

Weniger formelle, aber ebenfalls akzeptierte Formulierung der Constraints:

$$x_1+\ldots+x_n \leq k$$

$$x_u+x_{v_1}+\ldots+x_{v_j}\geq 1 \quad \text{für jeden Knoten } u\in V \text{ und seine Nachbarschaft } v_1,\ldots,v_j$$

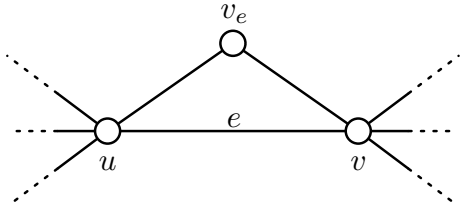
$$x_v\geq 0 \quad \text{für alle Knoten } v\in V$$

$$x_v\leq 1 \quad \text{für alle Konten } v\in V$$

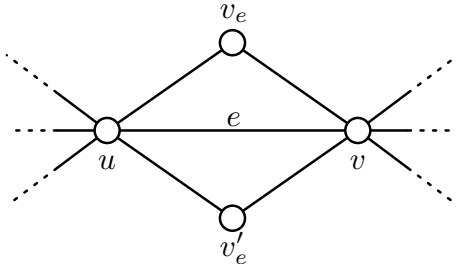
Lösungsende

c. Fügen Sie Knoten und Kanten so in das Bild unten ein, dass jedes valide Dominating Set des neuen Graphen entweder mindestens einen der Knoten u und v enthält oder so umgebaut werden kann, dass es einen der beiden Knoten enthält, ohne seine Größe zu ändern. Dies muss unabhängig von den anderen Verbindungen der Knoten (im Bild gestrichelt angedeutet) funktionieren. [2 Punkte]

Lösung



alternativ



Beide Lösungen funktionieren, das erste Gadget ist einfacher braucht aber etwas Sonderbehandlung in der Reduktion. Verwendung siehe nächste Teilaufgabe.

(weitere Teilaufgabe auf dem nächsten Blatt)

Lösungsende

Lösungsvorschlag

Fortsetzung von Aufgabe 6

d. Zeigen Sie, dass DOMINATING SET NP-vollständig ist. Verwenden Sie dazu, dass VERTEX COVER ein NP-vollständiges Problem ist. [6 Punkte]

Hinweis: Wenn Sie das Gadget in der vorherigen Teilaufgabe nicht gefunden haben, können Sie die Existenz eines solchen annehmen und immer noch Teilpunkte erreichen!

Lösung

Zuerst zeigen wir, dass DOMINATING SET in NP liegt. Dies ist recht einfach zu sehen, denn gegeben eine potentielle Lösung X der DOMINATING SET-Instanz (G, k) (wir können diese mit einer nichtdeterministischen Turingmaschine „raten“) müssen wir nur über die Knoten iterieren. Für jeden Knoten $v \in V$ prüfen wir dann, ob $v \in X$ oder ob ein Nachbar u von v existiert, sodass $u \in X$. Wenn dies für alle Knoten erfüllt ist und außerdem $|X| \leq k$ gilt, dann ist X eine Lösung der DOMINATING SET-Instanz. Dies ist offensichtlich in polynomieller Zeit möglich, und daher liegt DOMINATING SET in NP.

Reduktion von VERTEX COVER auf DOMINATING SET: Gegeben sei eine VERTEX COVER-Instanz $(G = (V, E), k)$. Wir konstruieren eine DOMINATING SET-Instanz $(G' = (V', E'), k)$, sodass die beiden Instanzen lösbarkeitsäquivalent sind.

Wir konstruieren G' wie folgt: V' enthalte alle Knoten aus V *außer isolierten Knoten, also solchen mit Grad 0* und E' alle Kanten aus E. Wir fügen für jede Kante von G zusätzliche Knoten und Kanten ein (Gadget aus der vorherigen Teilaufgabe). Für jede Kante $e = \{u, v\} \in E$ fügen wir also einen zusätzlichen Knoten v_e in V' ein, und zwei Kanten $\{u, v_e\}$ und $\{v_e, v\}$. Damit bilden u, v und v_e ein Dreieck (das Gadget, das wir eben konstruiert haben).

Jetzt müssen wir noch zeigen, dass G genau dann ein Vertex Cover der Größe k hat, wenn H ein Dominating Set der Größe k hat.

$\Longrightarrow$: Jedes Vertex Cover X von G ist auch ein Dominating Set von H. Da jede Kante von G zu einem Knoten aus X inzident ist, ist für jedes Gadget entweder u oder v in X. Nach Konstruktion des Gadgets sind damit alle seiner Knoten mit einem Knoten aus X benachbart. Es ist hier wichtig, dass H die isolierten Knoten aus G nicht enthält, denn diese sind in einem Vertex Cover egal, in einem Dominating Set aber nicht (sie liegen in keinem Dreieck).

$\Longleftarrow$: Sei Y das Dominating Set von H. Beobachtung: Wenn einer der Hilfsknoten v_e in Y liegt, können wir ihn durch einen der beiden „echten“ Knoten ersetzen. Dies liegt daran, dass v_e nur „sein“ Dreieck dominiert und es egal ist, welcher Knoten aus dem Dreieck im Dominating Set liegt.

Wenn wir alle Hilfsknoten durch Knoten aus $V' \cap V$ ersetzt haben, ist das Ergebnis Y' ein Vertex Cover von G der Größe k, denn für jede Kante von G muss einer ihrer Endpunkte in Y' liegen damit Y' ein Dominating Set von H sein kann.

Die Reduktion ist offensichtlich polynomiell, also ist DOMINATING SET NP-schwer. Da es außerdem in NP liegt, ist es also NP-vollständig. □

Häufige Fehler:

- Funktionsweise einer Reduktion offensichtlich nicht verstanden
- Beweis der Lösbarkeitsäquivalenz nicht oder nur teilweise geführt
- Kein oder ein falsches Gadget verwendet, oder Gadgets falsch angewendet
- Nicht gezeigt, dass DOMINATING SET in NP liegt

Lösungsende

Lösungsvorschlag

Aufgabe 7. Informationstheorie [6 Punkte]

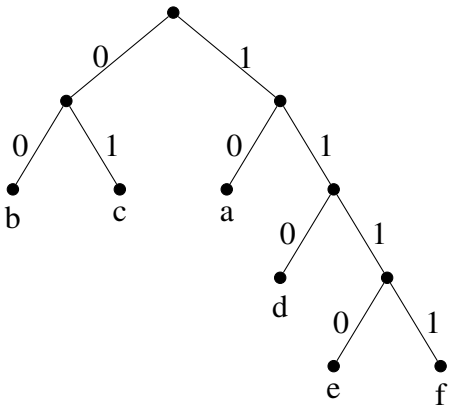
a. Konstruieren Sie einen Huffmancode für das Alphabet $\Sigma = \{a, b, c, d, e, f\}$ und zeichnen Sie den Codebaum. Die Häufigkeiten der einzelnen Buchstaben sind durch folgende Tabelle gegeben:

Zeichen	Wahrscheinlichkeit
a	0.3
b	0.2
c	0.2
d	0.15
e	0.1
f	0.05

[2 Punkte]

Lösung

Die Nullen und Einsen können auch andersrum verteilt sein, bzw die Blätter beliebig hinrotiert. Die Topologie des Baums ist aber eindeutig.



Lösungsende

b. Wir haben das Wort $w = 0010\,0100\,0000\,1011$ empfangen, von dem wir wissen, dass es mit einem Huffmancode für die Buchstabenverteilung aus der vorherigen Teilaufgabe codiert wurde. Wir wissen außerdem, dass das decodierte Wort mit e beginnt auf b endet. Die Gruppierung der Zeichen von w dient nur der Lesbarkeit und stellt *nicht* Codewortgrenzen dar.

Mit welchem Code wird jedes Zeichen codiert? Wie lautet das vollständig decodierte Wort?
[2 Punkte]

(mehr Platz und weitere Teilaufgabe auf dem nächsten Blatt)

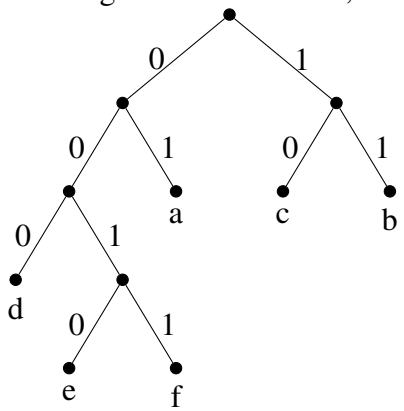
Lösungsvorschlag

Fortsetzung von Aufgabe 7

Lösung

Der Huffman*baum* ist eindeutig (alle Möglichkeiten sind zueinander isomorph), nur bei der Zuweisung von 0 und 1 auf linke und rechte Kinder besteht Spielraum. Ein möglicher Huffmancode wurde bereits in der vorherigen Teilaufgabe konstruiert.

Da die Topologie des Baumes eindeutig ist, ist die Codelänge von e gleich 4 Zeichen und e muss mit 0010 codiert worden sein. Ähnlich erfahren wir, dass b mit 11 codiert wurde. Diese Information reicht aus, um den verwendeten Code durch Vertauschen der Kantenbeschriftungen eindeutig zu rekonstruieren, und wir erhalten den Codebaum in der Abbildung unten.



Der Code codiert also a mit 01, b mit 11, c mit 10, d mit 000, e mit 0010 und f mit 0011. Dementsprechend lautet das decodierte Wort $eaddcb$.

Lösungsende

c. Gilt die folgende Aussage? Wenn die Wahrscheinlichkeit jedes Zeichens verschieden ist, so sind die Codelängen eines entsprechenden Huffman Codes eindeutig. Begründen Sie kurz. [2 Punkte]

Lösung

Gegenbeispiel rechts: Zuerst werden c und d kombiniert. Danach kann b entweder mit a oder mit cd kombiniert werden. Im ersten Fall haben alle Codes Länge 2, im zweiten Fall hat a Codelänge 1, b 2, und c und d 3.

Zeichen	Wahrscheinlichkeit
a	0.35
b	0.3
c	0.2
d	0.15

Lösungsende

Lösungsvorschlag

Aufgabe 8. Kleinaufgaben [7 Punkte]

a. Unter welcher der Operationen $\cup$, $\cap$, $\bar{\cdot}$ (Komplement), $*$ (Kleenescher Abschluss) ist die Klasse der rekursiv aufzählbaren Sprachen **nicht** abgeschlossen? Begründen Sie kurz. [1 Punkt]

Lösung

Chomsky-0 ist nur unter $\bar{\cdot}$ nicht abgeschlossen. Für eine Sprache $\mathcal{L}$ vom Chomsky-Typ 0 ist $\overline{\mathcal{L}}$ genau dann rekursiv aufzählbar, wenn $\mathcal{L}$ entscheidbar ist. Dies folgt aus der Semientscheidbarkeit des Wortproblems für rekursiv aufzählbare Sprachen.

Lösungsende

b. Ist es entscheidbar, ob zwei nichtdeterministische Kellerautomaten die gleiche Sprache erkennen? Begründen Sie kurz. [1 Punkt]

Lösung

Das Problem ist für NKellerA unentscheidbar (für DKellerA aber entscheidbar). In der Vorlesung wurde durch Reduktion vom PKP gezeigt, dass es bereits unentscheidbar ist, ob der Schnitt zweier kontextfreie Grammatiken *nicht leer* ist. Da kontextfreie Grammatiken und NKellerA gleich mächtig sind, ist die Gleichheit der erkannten Sprachen ebenso unentscheidbar.

Lösungsende

c. Kann eine linear beschränkte nichtdeterministische Turingmaschine (LBA) das Problem 3SAT lösen? Begründen Sie kurz! [1 Punkt]

Lösung

Ja: Es wird nur linearer Zusatzplatz benötigt, um eine Belegung der Variablen zu speichern (rate diese nichtdeterministisch). Die Erfüllbarkeit der Klauseln unter einer Belegung kann trivial in Polynomialzeit getestet werden. Der Platz für die Belegung der Variablen lässt sich auch durch einen Bandalphabet-Blowup einsparen.

Lösungsende

d. Kann 2SAT in Polynomialzeit auf HAMILTON CYCLE reduziert werden? Liegt es in NP? Ist es NP-vollständig? Begründen Sie kurz. [1 Punkt]

Lösung

Da 2SAT $\in$ P $\subseteq$ NP kann es offensichtlich polynomiell auf HAMILTON CYCLE reduziert werden, wir können in der Reduktion die Instanz lösen und eine triviale Ja-/Nein-Instanz ausgeben. Es ist aber nicht NP-vollständig, dazu müssten wir die Gegenreduktion zeigen.

Lösungsende

(weitere Teilaufgaben auf dem nächsten Blatt)

Lösungsvorschlag

Fortsetzung von Aufgabe 8

e. Gegeben sei eine ternäre (= 3 Zeichen) gedächtnislose Quelle. Welche Wahrscheinlichkeiten müssen die Zeichen annehmen, um die Entropie der Quelle zu maximieren? Wie hoch ist die Entropie dann? [1 Punkt]

Lösung

Alle Zeichen müssen gleich wahrscheinlich sein, d.h. mit Wahrscheinlichkeit $\frac{1}{3}$ auftreten. Die Entropie beträgt dann $\log_2 3 \approx 1.58$ Bits pro Zeichen.

Lösungsende

f. Suffixfreie Codes sind analog zu präfixfreien Codes definiert: Kein Code ist ein Suffix eines anderen. Sind suffixfreie Codes immer eindeutig decodierbar? *Begründen Sie!* [2 Punkte]

Lösung

Ja, denn ein präfixfreier Code rückwärts ist suffixfrei. Wenn es also unterschiedliche Eingaben gäbe, die mit dem gleichen String codiert würden, dann würden Sie immer noch mit dem gleichen String codiert, wenn man den Eingabestring und jedes Codewort umdreht. Dies steht im Widerspruch zur eindeutigen Decodierbarkeit präfixfreier Codes.

Lösungsende