Übungen

Theoretische Informatik 2

Niels Lohmann

Wintersemester 2002/2003

nlohmann@informatik.hu-berlin.de
http://www.informatik.hu-berlin.de/~nlohmann

Inhaltsverzeichnis

Serie 1

Institut für Informatik
Algorithmen und Komplexität
Dr. Till Nierhoff

WS 2002
16. Oktober 2002

Theoretische Informatik II

1. Serie — korrigiert

Abgabe bis zum 23. Oktober 2002

Aufgabe 1 [3+4+3 Punkte]

Beweisen Sie die folgenden mathematischen Aussagen für $0 < a < 1$, $n \in N = \{0, 1, 2, \ldots\}$:

a) $\sum_{i=0}^{n} a^i = \frac{1-a^{n+1}}{1-a}$
(Hinweis zu a) und b): vollständige Induktion)

b) $1 - an + \frac{a^2n^2}{2} \geq (1-a)^n \geq 1 - an$

c) $(1+a)^n \leq e^{an}$ (Hinweis: Ableitung)

Aufgabe 2 [10 Punkte]

Nach der Vorlesung von Prof. Starke ist $T = (X, Z, z_0, \delta)$ eine Turing-Maschine, wenn

- X ein endliches Alphabet mit $\square \notin X$,
- Z eine endliche Menge von Zuständen und $z_0 \in Z$ der Anfangszustand
- und $\delta : Z \times (X \cup \{\square\}) \to (X \cup \{\square\}) \times \{R, L, N, S\} \times Z$ die Überführungsfunktion ist.

Die erkannte Sprache dieser Maschine ist die Menge $L(T) = \{x \in X^* | \, T \text{ stoppt bei Eingabe } x\}$.

Zeigen Sie die Äquivalenz zur 1-DTM nach folgender Definition:
Eine deterministische k-Band-Turingmaschine (kurz k-DTM) wird durch ein Tupel $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$ beschrieben, wobei

- Z eine endliche Menge von Zuständen und $q_0 \in Z$ der Startzustand,
- Σ das Eingabealphabet mit $\square \notin \Sigma$,
- Γ das Arbeitsalphabet mit $\Sigma \cup \{\square\} \subseteq \Gamma$

- $\delta : Z \times \Gamma^k \to \left(Z \times \Gamma^k \times \{L, R, N\}^k \right) \cup \oslash$ die Überführungsfunktion

- und E die Menge der Endzustände ist.

Die Maschine M erkennt die Sprache $L = \{x \in \Sigma^* | \ M \text{ gelangt bei Eingabe } x \text{ in einen Endzustand}\}$.

Aufgabe 3 [4+6 Punkte]

Beschreiben Sie in Worten welche Sprache die angegebene Turingmaschine M erkennt und wie sie funktioniert.

a) $M = (\{z_0, z_1\}, \{a, b\}, \{a, b, \Box\}, \delta, z_0, \{z_1\})$, wobei

$$\delta(z_0, a) = (z_1, b, R), \quad \delta(z_0, b) = (z_1, a, R), \quad \delta(z_0, \Box) = (z_1, \Box, N)$$

b) $M = (\{q, r_0, r_1, r_0^*, r_1^*, s_0, s_1, l, ja, nein\}, \{0, 1\}, \{0, 1, \Box\}, \delta, q, ja)$, mit der Überführungsfunktion

$\delta\,(z, x)$	$x = \Box$	0	1
$z = q$	$ja, \Box, N$	$r_0, \Box, R$	$r_1, \Box, R$
r_0	$ja, \Box, N$	$r_0^*, 0, R$	$r_0^*, 1, R$
r_1	$ja, \Box, N$	$r_1^*, 0, R$	$r_1^*, 1, R$
r_0^*	$s_0, \Box, L$	$r_0^*, 0, R$	$r_0^*, 1, R$
r_1^*	$s_1, \Box, L$	$r_1^*, 0, R$	$r_1^*, 1, R$
s_0		$l, \Box, L$	$nein, \Box, N$
s_1		$nein, \Box, N$	$l, \Box, L$
l	$q, \Box, R$	$l, 0, L$	$l, 1, L$

Aufgabe 4 [10 Punkte]

Geben Sie eine 1-TM mit dem Eingabealphabet $\Sigma = \{|\}$ an, die zwei unär kodierte natürliche Zahlen x und y multipliziert.
Aus der Eingabe $\underbrace{|\ldots|}_{x+1\text{-mal}} \Box \underbrace{|\ldots|}_{y+1\text{-mal}}$ soll das Produkt $\underbrace{|\ldots|}_{x \cdot y+1\text{-mal}}$ erzeugt werden.
Argumentieren Sie, weshalb die von Ihnen erstellte TM ihre Aufgabe erfüllt.

Aufgabe 1

Sei $0 < a < 1$ und $n \in \mathbb{N} = \{0, 1, 2, \dots\}$.

Teilaufgabe a

Behauptung:

$$\sum_{i=0}^{n} a^i \quad = \quad \frac{1 - a^{n+1}}{1 - a}$$

Beweis:
durch vollständige Induktion über n.

Induktionsanfang ($n = 0$):

$$\sum_{i=0}^{0} a^i = a^0 = 1 = \frac{1-a}{1-a} = \frac{1 - a^{0+1}}{1-a}$$

ist erfüllt.

Induktionsschritt ($n \to n+1$):

$$\begin{aligned}
\sum_{i=0}^{n+1} a^i &= \sum_{i=0}^{n} a^i + a^{n+1} \\
&\overset{\text{Ind.-Vor.}}{=} \frac{1 - a^{n+1}}{1-a} + a^{n+1} \\
&= \frac{1 - a^{n+1} + (a^{n+1})(1-a)}{1-a} \\
&= \frac{1 - a^{n+1} + a^{n+1} - a^{n+2}}{1-a} \\
&= \frac{1 - a^{n+2}}{1-a}
\end{aligned}$$

□

Teilaufgabe b

Behauptung:

$$1 - an + \frac{a^2 n^2}{2} \quad \geq (1-a)^n \quad \geq 1 - an$$

Beweis:
durch vollständige Induktion über n.

Induktionsanfang ($n = 0$):

$$1 \geq (1-a)^0 \geq 1 \quad \Leftrightarrow \quad 1 \geq 1 \geq 1$$

ist erfüllt.

Induktionsschritt ($n \to n+1$):

$$\begin{aligned} 1 - a(n+1) + \frac{a^2(n+1)^2}{2} &= 1 - an - a + \frac{a^2(n^2+2n+1)}{2} \\ &= 1 - an + \frac{a^2n^2}{2} + \frac{a^2(2n+1)}{2} - a \\ &= 1 - an + \frac{a^2n^2}{2} + \frac{a^2(2n+1)}{2} - a \\ &= 1 - an + \frac{a^2n^2}{2} + na^2 + \frac{a^2}{2} - a \end{aligned}$$

$$\begin{aligned} (1-a)^{n+1} &= (1-a)^n \cdot (1-a) \\ &= (1-a)^n - a(1-a)^n \end{aligned}$$

$$1 - a(n+1) = 1 - an - a$$

Es genügt, die einzelnen Reste abzuschätzen:

Behauptung:

$$-a + na^2 + \frac{a^2}{2} \geq -a(1-a)^n \geq -a$$

1. Vergleich des 2. und 3. Terms:
   $-a(1-a)^n \geq -a \Leftrightarrow (1-a)^n \leq 1$ ist erfüllt, da $a \in]0,1[$.
2. Vergleich des 1. und 2. Terms:

$$\begin{aligned} & -a + na^2 + \tfrac{a^2}{2} && \geq -a(1-a)^n \\ \Leftrightarrow \quad & -a\left(1 - na - \tfrac{a}{2}\right) && \geq -a(1-a)^n \\ \Leftrightarrow \quad & 1 - an - \tfrac{a}{2} && \leq (1-a)^n \end{aligned}$$

3. Laut Voraussetzung gilt: $1 - an + \frac{a^2n^2}{2} \geq (1-a)^n$

wegen 2. und 3. genügt zu überprüfen: $1 - an + \frac{a^2n^2}{2} \geq 1 - an - \frac{a}{2}$

$$\begin{aligned} & 1 - an + \tfrac{a^2n^2}{2} && \geq 1 - an - \frac{a}{2} \\ \Leftrightarrow \quad & \tfrac{a^2n^2}{2} && \geq -\frac{a}{2} \\ \Leftrightarrow \quad & a^2n^2 && \geq -a \\ \Leftrightarrow \quad & an^2 && \geq -1 \end{aligned}$$

ist erfüllt, da $a \in]0,1[$ und $n \in \mathbb{N}$. □

Teilaufgabe c

Behauptung:

$$(1+a)^n \quad \leq \quad e^{an}$$

Beweis:
durch vollständige Induktion über n.

Induktionsanfang ($n = 0$):

$$(1+a)^0 \leq e^0 \quad \Leftrightarrow \quad 1 \leq 1$$

ist erfüllt.

Induktionsschritt ($n \to n+1$):

$$(1+a)^{n+1} = (1+a)^n \cdot (1+a) \text{ und } e^{a(n+1)} = e^{an+a} = e^{an} \cdot e^a$$

Da laut Voraussetzung $(1+a)^n \leq e^{an}$ gilt, genügt es, die Faktoren ($1+a$ und e^a) zu überprüfen.

Behauptung:
$1 + a \leq e^a$ für $a \in]0,1[$

1. Sei $f(a) = e^a - 1 - a$. Dann ist $f'(a) = e^a - 1 \geq 0 \ \forall a \in]0,1[$
   $\Rightarrow f$ ist monoton steigend

2. $\lim\limits_{a \to 0+} f(a) = \lim\limits_{a \to 0+} (e^a - 1) = 0$

Aus 1. und 2. folgt: $f(a) \geq 0 \; \forall a \in]0, 1[$, also gilt:

$$\begin{aligned} & f(a) && \geq 0 \\ \Leftrightarrow \quad & e^a - 1 - a && \geq 0 \\ \Leftrightarrow \quad & e^a && \geq 1 + a \end{aligned}$$

□

Aufgabe 2

Behauptung:
Die Turing-Maschine $T = (X, Z, z_0, \delta)$ (nach Prof. Starke) ist äquivalent zur 1-DTM $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$.

Um die Behauptung zu beweisen, wird nacheinander gezeigt, dass sich die Turing-Maschine (nach Prof. Starke) als 1-DTM darstellen lässt und umgekert.

Beweis (Turing-Maschine → 1-DTM):
Es kann ein Algorithmus angegeben werden, mit dem aus einer Turing-Maschine T eine 1-DTM M konstruiert werden kann, sodass $L(T) = L(M)$:

1. **Bandbeschriftung:** Das Band mit seiner Beschriftung von T kann ohne Änderung von M übernommen werden.

2. **Alphabete:** Sei $\square \notin X$ das endliche Alphabet von T. Für M sei $\Sigma = X$ das Eingabealphabet und $\Gamma = X \cup \{\square\}$ das Arbeitsalphabet.

3. **Zustände:** Sei Z die endliche Menge der Zustände von T mit dem Startzustand $q_0 \in Z$, sowie allen Zuständen, in denen T stoppt $E_T \subseteq Z$. Für M sei Z die endliche Zustandsmenge, $q_0 \in Z$ der Startzustand und $E_M = E_T \subseteq Z$ die Menge der Endzustände von M, die im Folgenden noch modifiziert wird.

4. **Überführungsfunktion:** Sei δ_T die Überführungsfunktion von T mit $\delta_T = Z \times (X \cup \{\square\}) \to (X \cup \{\square\}) \times \{R, L, N, S\} \times Z$. Sei nun $\delta_M = Z \times \Gamma \to (Z \times \Gamma \times \{L, R, N\})$ die Überführungsfunktion für M für alle Zustände $z \notin E_T$, wobei E_T die Menge der Zustände ist, in denen T stoppt (bzw. die Kopfbewegung S ausführen muss).

   Dabei werden die 3-Tupel von T vom Format [„neues Zeichen“, „Kopfbewegung“, „neuer Zustand“] für M in das Format [„neuer Zustand“, „neues Zeichen“, „Kopfbewegung“] geändert.

5. **Stoppzustände:** Die Stoppzustände $E_T \subseteq Z$ von T in der Form $\delta_T(z,x) = (x',S,z)$ $(z \in E_T; x,x' \in X \cup \{\square\})$ werden für M in $\delta_M(z,x) = (x',N,z_e)$ $(z \in Z \backslash E_M; x,x' \in \Gamma)$ mit $z_e \in E_M$ und $\delta_M(z_e,\hat{x}) = \emptyset$ $(\hat{x} \in \Gamma)$ geändert. Dabei wird aus dem Stoppzustand $z \in E_T$ von T ein „normaler“ Zustand von M, der jedoch in jedem Fall in einen neuen Zustand $z_e \in E_M$ übergeht, der zu den Endzuständen E_M von M gehört.

T und M arbeiten nun mit den gleichen Alphabeten und unterscheiden sich nur im Format der Überführungsfunktion und der Darstellung der (insbesondere Stopp- bzw. End-) Zustände. Jede Eingabe, bei der T stoppt, führt auch M in einen Endzustand.

Beweis (1-DTM → Turing-Maschine):
Des Weiteren kann ein Algorithmus angegeben werden, mit dem aus einer 1-DTM M eine Turingmaschine T konstruiert werden kann, sodass $L(M) = L(T)$:

1. **Bandbeschriftung:** Die Bandbeschriftung von M kann ohne Änderung von T übernommen werden.

2. **Alphabete:** Sei $\square \notin \Sigma$ das endliche Eingabealphabet von M. Für T sei $X = \Sigma$ das Alphabet.

3. **Zustände:** Sei Z die endliche Menge der Zustände von M mit dem Startzustand $q_0 \in Z$. Für T sei Z ebenfalls die endliche Zustandsmenge, $q_0 \in Z$ der Startzustand. Die Endzustände von M E_M werden im Folgenden untersucht.

4. **Überführungsfunktion:** Sei δ_M die Überführungsfunktion von M mit $\delta_M = Z \times \Gamma \to (Z \times \Gamma \times \{L,R,N\}) \cup \emptyset$. Sei nun $\delta_T = Z \times (X \cup \{\square\}) \to (X \cup \{\square\}) \times \{R,L,N,S\} \times Z$ die Überführungsfunktion für T für alle Zustände $z \notin E_M$. (Ausgenommen zunächst also alle Endzustände von M, sowie alle Zustände $z_\emptyset \in Z$: $\delta_M(z_\emptyset,x) = \emptyset$ $(x \in \Gamma)$. Diese Zustände werden später seperat behandelt.).

   Dabei werden die geordneten 3-Tupel von M vom Format [„neuer Zustand“, „neues Zeichen“, „Kopfbewegung“) für T in das Format [„neues Zeichen“, „Kopfbewegung“, „neuer Zustand“] geändert.

5. **Endzustände:** Die Endzustände E_M von M in der Form $\delta_M(z,x) = (z,x',\kappa)$ $(z \in E_M; x,x' \in \Gamma; \kappa \in \{L,R,N\})$ werden für T in $\delta_T(z,x) = (x',\kappa,z_e)$ $(z,z_e \in Z; x,x' \in \Gamma; \kappa \in \{L,R,N\})$ mit $\delta_T(z_e,\hat{x}) = (\hat{x},S,z_e)$ $(\hat{x} \in X \cup \{\square\})$ geändert. Außerdem gilt für alle Zustände mit $\delta_M(z_\emptyset,x) = \emptyset$ für T $\delta_T(z_\emptyset,x) = \{x,S,z_\emptyset\}$.

M und T arbeiten nun wieder mit den gleichen Alphabeten und unterscheiden sich nur im Format der Überführungsfunktion und der Darstellung der (insbesondere Stopp- bzw. End-) Zustände. Jede Eingabe, bei der M einen Endzustand erreicht, bringt auch T zum Stopp.

□

Beispiel (anhand der 1-DTM aus Aufgabe 3a):
Die gegebene 1-DTM M wird anhand des zuletzt beschriebenen Algorithmus in eine Turing-Maschine T (nach Prof. Starke) überführt:

1. Die Bandbeschriftung wird übernommen.
2. $X = \{a, b\}$ ist das Alphabet von T.
3. $Z = \{z_0, z_1\}$ ist die Zustandsmenge von T, deren Zustände im Folgenden behandelt werden.
4. $\delta_T : \{z_0, z_1\} \times \{a, b, \square\} \to (\{a, b, \square\} \times \{R, L, N, S\} \times \{z_0, z_1\})$ ist die Überführungsfunktion von T, die zunächst wiefolgt definiert ist: $\delta_T(z_0, a) = (b, R, z_1)$, $\delta_T(z_0, b) = (a, R, z_1)$ und $\delta_T(z_0, \square) = (\square, N, z_0)$.
5. Der Endzustand z_1 von M mit $\delta_M(z_1, x) = \emptyset$ $(x \in \{a, b, \square\})$ wird für T wie folgt definiert: $\delta_T(z_1, x) = (x, S, z_1)$, $(x \in \{a, b, \square\})$.

Aufgabe 3

Teilaufgabe a

Die gegebene Turinmaschine M startet im Zustand z_0 und geht unabhängig vom gelesenen Zeichen in den Endzustand z_1 über. Dabei ändert sie das Zeichen vor dem Übergang in den Zustand z_1 wie folgt: aus einem a wird ein b und umgekehrt wird aus einem b ein a. Nach der Zeichenänderung geht M außerdem einen Schritt nach rechts. Des Weiteren akzepziert M das leere Wort ε, bei dem sie ohne Kopfbewegung in den Endzustand z_1 übergeht.

$\Rightarrow L(M) = \{a, b, \varepsilon\}$

Teilaufgabe b

Die gegebene Turingmaschine M akzeptiert alle möglichen Palindrome aus dem Eingabealphabet $\{0, 1\}$ (also z.B. 00111100), sowie das leere Wort ε. Sei $x = x_1x_2x_3 \dots x_n$ eine Eingabe, geht M wie folgt vor:

1. x_1 wird gelesen:
   - $x_1 = \varepsilon \to$ Übergang in Endzustand $\boxed{ja}$, da es sich beim leeren Wort ε um ein Palindrom handelt
   - $x_1 = i$ $(i = 0, 1) \to$ Zeichen mit Blank $(\square)$ überschreiben, Kopf nach rechts bewegen und Übergang in Zustand r_i
2. x_2 wird im Zustand r_i gelesen:
   - $x_2 = \varepsilon \to$ Übergang in Endzustand $\boxed{ja}$, da ein einzelnes Zeichen (x_1) ein Palindrom ist

- $x_2 = 0$ oder $x_2 = 1 \rightarrow$ Zeichen unverändert lassen (mit gleichem Zeichen überschreiben), Kopf nach rechts bewegen und Übergang in Zustand r_i^*

3. x_3 bis x_n werden im Zustand r_i^* gelesen:

   - $x_3 = \varepsilon \rightarrow$ letztes Zeichen der Eingabe passiert: Kopf nach links bewegen und in den Zustand s_i übergehen
   - $x_3 = 0$ oder $x_3 = 1 \rightarrow$ Zeichen unverändert lassen, Kopf nach rechts bewegen und im gleichen Zustand (r_i^*) bleiben, um nächstes Zeichen einzulesen

4. letztes Zeichen x_n wird im Zustand s_i gelesen:

   - $x_n = i \rightarrow$ letztes Zeichen löschen (mit Blank überschreiben), Kopf nach links bewegen und in den Zustand l übergehen
   - $x_n \neq i$ Zeichen mit Blank überschreiben und ohne Kopfbewegung in den Zustand $\boxed{nein \notin E}$ übergehen und abbrechen, da das letzte Zeichen nicht mit dem ursprünglich ersten Zeichen übereinstimmt ($x_n \neq x_1 = i$) und es sich deshalb bei der Eingabe nicht um ein Palindrom handelt

5. Zustand l: M geht Zeichen für Zeichen – ohne die Zellen zu verändern – das Band nach links, bis sie den Anfang erreicht hat. Anschließend geht sie wieder in den Startzustand q über. Der Zustand l kann nur erreicht werden, nachdem das erste Zeichen (im Startzustand q) und letzte Zeichen (im Zustand s_m) auf dem Band gelöscht, das mit einem Blank ($\square$) überschrieben wurden. Die im zweiten Durchgang zu verarbeitende Eingabe ist nun um zwei Zeichen kürzer und sieht wie folgt aus: $x' = x_2x_3x_4 \ldots x_{n-1}$

In den folgenden Durchgängen wird die Eingabe durch das Löschen des jeweils ersten und letzten Zeichens verkürzt, bis das leere Wort (siehe 1.) oder ein einzelnes Zeichen (siehe 2.) übrigbleibt und M in den Endzustand $\boxed{ja}$ übergeht oder vorher wegen einer ungültigen Eingabe (siehe 4.) abgebrochen wird.

$\Rightarrow L(M) = \{x | x$ ist Palindrom über $\{0,1\}\}$, wobei „Palindrom über $\{0,1\}$“ wie folgt definiert sei:

1. 0, 1 und das leere Wort ε sind Palindrome über $\{0,1\}$.
2. Wenn p ein Palindrom über $\{0,1\}$ ist, so auch $0p0$ und $1p1$.
3. Nichts anderes ist ein Palindrom über $\{0,1\}$.

Aufgabe 4

Sei M eine 1-TM mit $M = (\{q_1, \ldots, q_{21}\}, \{|\}, \{|, \square\}, \delta, q_1, \{q_{21}\})$, mit der Überführungsfunktion

$\delta(z,x)$	$x = \square$	$\mid$	Beschreibung
$z = q_1$		$q_2, \square, R$	Startzustand
q_2	$q_3, \square, R$	$q_2, \mid, R$	
q_3		$q_3, \mid, R$	
q_4	$q_5, \square, L$	$q_5, \square, L$	Unäre Kodierung (x+1)
q_5	$q_8, \square, L$	$q_6, \mid, L$	bei beiden Faktoren
q_6	$q_7, \square, L$	$q_6, \mid, L$	entfernen
q_7	$q_9, \square, R$	$q_7, \mid, L$	
q_8	$q_9, \square, R$	$q_8, \mid, L$	
q_9	$q_{18}, \square, N$	$q_{10}, \square, R$	
q_{10}	$q_{11}, \square, R$	$q_{10}, \mid, R$	
q_{11}	$q_{16}, \square, L$	$q_{12}, \square, R$	
q_{12}	$q_{13}, \square, R$	$q_{12}, \mid, R$	
q_{13}	$q_{14}, \mid, L$	$q_{13}, \mid, R$	Die eigentliche
q_{14}	$q_{15}, \square, L$	$q_{14}, \mid, L$	Multiplikation
q_{15}	$q_{11}, \mid, R$	$q_{15}, \mid, L$	
q_{16}	$q_{17}, \square, L$	$q_{16}, \mid, L$	
q_{17}	$q_9, \mid, R$	$q_{17}, \mid, L$	
q_{18}	$q_{19}, \square, R$		Das Ergebnis unär
q_{19}	$q_{20}, \square, R$	$q_{19}, \mid, R$	kodieren
q_{20}	$q_{21}, \mid, R$	$q_{20}, \mid, R$	
q_{21}	$q_{21}, \square, S$	$q_{21}, \mid, S$	Endzustand

Beschreibung der einzelnen Zustände

q_1 Ersten Strich des 1. Faktors (x) mit $\square$ überschreiben, Kopf nach rechts und weiter mit q_2.

q_2 Bewegt den Kopf auf das 1. | des 2. Faktors (y) und geht in q_3 über.

q_3 Bewegt den Kopf auf das letzte Zeichen des 2. Faktors (y). Weiter mit q_4.

q_4 Überschreibt letztes Zeichen des 2. Faktors (y) mit $\square$ und bewegt den Kopf nach links. Weiter mit q_5.

q_5 Kopf nach links und bei | weiter nach q_6, sonst q_8.

q_6 Bewegt Kopf auf letzten | von y, weiter mit $q7$.

q_7 Bewegt Kopf auf letzten | von x, weiter mit $q9$.

q_8 Bewegt Kopf auf letzten | von x, weiter mit $q9$. (identisch mit q_7...)

q_9 Löscht | in x, geht nach rechts in q_{10}. Bei $\square$ an dieser Stelle ohne Kopfbewegung Übergang zu q_{18}.

q_{10} Bewegt Kopf auf ersten | von y und geht in q_{11} über.

q_{11} Bei $\square$ an dieser Stelle Kopf nach links und weiter mit q_{16}, ansonsten $\square$ schreiben, nach rechts und weiter mit q_{12}.

q_{12} Bewegt Kopf hinter y und noch eine Zelle weiter nach rechts. Weiter mit q_{13}.

q_{13} Schreibt |, wenn $\square$ in der betrachteten Zelle (bereits Ende des Ergebnisses) geht nach links und weiter mit q_{14}. Bei | in der Zelle befindet sich Kopf noch nicht am Ende des Ergebnisses. $\rightarrow$ bewegt Kopf mittels q_{13} ans Ende des Ergebnisses.

q_{14} Bewegt den Kopf auf den letzten | des y. Weiter mit q_{15}.

q_{15} Bewegt Kopf dahin, wo in q_{11} der | (in y) gelöscht wurde und schreibt ihn wieder an die gleiche Position. Weiter nach rechts und q_{11}.

q_{16} y ist nun vollständig kopiert (bzw. einmal zum Ergebnis addiert). Bringt Kopf nun auf das letzte | des x. Weiter mit q_{17}.

q_{17} Bewegt Kopf dahin, wo in q_9 der | (in x) gelöscht wurde und schreibt ihn wieder an die gleiche Position. Weiter nach rechts und q_9.

q_{18} Bewegt Kopf nach rechts, weiter mit q_{19}.

q_{19} Bewegt Kopf auf den ersten | des Ergebnisses.

q_{20} Bewegt Kopf hinter das Ergebniss, und fügt dort ein | ein, um das Ergebnis unär darzustellen. Dann in q_{21}.

q_{21} Endzustand.

Zusammenfassung

Zusammenfassend lässt sich sagen, dass die Turing Maschine zuerst die $(x+1)$-Striche in x-Striche umwandelt, dies passiert mit beiden Faktoren. Dann folgt erst die eigentliche Multipikation. Es wird der erste | von x gelöscht, und ein y an das Ende des Bandes kopiert, und der vorher gelöschte | wieder an die selbe Stelle geschrieben. Danach wird gegebenfalls der zweite | von x gelöscht, y wiederrum kopiert, usw. Nachdem der letzte | des ersten Faktors nach dem Löschen wieder hinzugefügt wurde, wurde das y x-mal an das Ende des Bandes kopiert. Nun bleibt nur noch, das Ergebnis unär zu kodieren, d.h. es wird ein zusätzlicher | an das Ende des $x\dot{y}$ geschrieben. Nun stoppt die Maschine und hat die Multiplikation erfolgreich ausgeführt.

Serie 2

Institut für Informatik
Algorithmen und Komplexität
Dr. Till Nierhoff

WS 2002
23. Oktober 2002

Theoretische Informatik II

2. Serie — korrigiert

Abgabe bis zum 30. Oktober 2002

Aufgabe 5 [8 Punkte]

Zeigen Sie, dass das Intervall $[0, 1]$ überabzählbar ist. Führen Sie dazu die Annahme zum Widerspruch, es gäbe eine abzählbare Aufzählung aller reellen Zahlen zwischen 0 und 1.
Hinweis: Untersuchen Sie die Binärdarstellung der Zahlen dieser Aufzählung. Sei x_i das i-te Bit der i-ten Zahl der Aufzählung. Betrachten Sie nun jene Zahl, deren i-tes Bit die Negation von x_i ist.

Aufgabe 6 [16 Punkte]

Die sogenannte „Groß-O-Notation“ ist folgendermaßen definiert:
Sei $f : \mathbb{N} \to \mathbb{R}$ eine zahlentheoretische Funktion.
Alle Funktionen, die bis auf einen konstanten Faktor höchstens so schnell wachsen wie f, sind in der folgenden Menge zusammengefasst:
$O(f) := \{g : \mathbb{N} \to \mathbb{R} \mid \exists c \in \mathbb{R}^{>0},\ \exists n_0 \in \mathbb{N} :\ g(n) \leq c \cdot f(n),\ \forall n \geq n_0\}$.
Die vereinfachte Schreibweise „$g(n) = O(f(n))$“ bedeutet: $g \in O(f)$.

Prüfen Sie die Korrektheit der folgenden Aussagen und begründen Sie Ihr Urteil:

a) $3n + 4 + \log n = O(n)$

b) $(3n + \sqrt{n})^2 = O(n^2)$

c) $\log n = O(\log(\sqrt{n}))$

d) $\log n = O(\sqrt{n})$

e) $\sum_{i=1}^{n} i^{-2} = O(1)$

f) $\sum_{i=1}^{n} i = O(n)$

g) $\frac{n - n^{3/4}}{\sqrt{n}+5} = O\left(n^{1/4}\right)$

h) $\left(1 + \frac{2}{n-4}\right)^n = O(1)$

Definition. *Für eine k-Band-Turingmaschine und ein Eingabewort x sei*

$$K_x \vdash K_1 \vdash \ldots \vdash K_t$$

eine akzeptierende Rechnung mit $K_i = (q_i, u_{i1}, v_{i1}, \ldots, u_{ik}, v_{ik})$.
Die Laufzeit dieser Rechnung ist dann t und der benötigte Platz sei

$$s = \max\{|u_{ij}| + |v_{ij}| \ : \ i = 1 \ldots t, \ j = 1 \ldots k\}.$$

Aufgabe 7 [6 Punkte]

Gegeben sei folgende 2-Band-Turingmaschine T:
$T = (\{z_a, z_e, z_0, z_1\}, \{0,1\}, \{0,1,\Box\}, \delta, z_a, \{z_e\})$, wobei δ durch folgende Übergänge definiert ist:

$$\begin{array}{ll} (z_a,(0,\Box)) \to (z_0,(0,\Box),(R,N)), & (z_a,(1,\Box)) \to (z_e,(1,1),(N,N)) \\ (z_0,(0,\Box)) \to (z_1,(0,\Box),(R,N)), & (z_0,(1,\Box)) \to (z_e,(1,0),(N,N)) \\ (z_1,(0,\Box)) \to (z_0,(0,\Box),(R,N)), & (z_1,(1,\Box)) \to (z_e,(1,1),(N,N)) \end{array}$$

a) Welche Funktion wird von T berechnet? Begründen Sie Ihre Antwort.

b) Welche ist die kürzeste und welche die längste Laufzeit für Eingabewörter aus $E_n = \{0^m 10^{n-m-1} \mid 0 \leq m \leq n-1\}$?

Aufgabe 8 [10 Punkte]

Zeigen Sie: Eine k-Band-Turingmaschine kann so durch eine 1-TM simuliert werden, dass die Simulation einer Rechnung mit Laufzeit t und Platzbedarf s eine Laufzeit von $O(t^2)$ und den Platzbedarf $O(s)$ hat.

Aufgabe 5

Behauptung:
Das Intervall $[0, 1]$ ist überabzählbar.

Beweis (indirekt):
Wir nehmen an, es gäbe eine abzählbare Aufzählung aller $x \in \mathbb{R} \in [0, 1]$. Dazu untersuchen wir zunächst den ganzen Körper der reellen Zahlen $\mathbb{R}$:

Behauptung:
Der Körper $\mathbb{R}$ ist nicht abzählbar.

Beweis (indirekt):
Wir nehmen an, es gäbe eine Abzählung

$$\mathbb{R} = \{x_1, x_2, x_3, \ldots\}$$

Zu dieser konstruieren wir eine Intervallschaltung (I_n) mit:

$$x_n \notin I_n \forall n \tag{0.1}$$

Die I_n werden rekursiv definiert. Wir beginnen mit $I_1 := \left[x_1 + 1, x_1 + \frac{4}{3}\right]$. Aus I_n konstruieren wir dann I_{n+1} wie folgt: Wir teilen I_n in drei gleichlange Intervalle und wählen als I_{n+1} ein solches abgeschlossenes Teilintervall, das x_{n+1} nicht enthält. Offenbar hat I_n die Länge 3^{-n}. Die Folge (I_n) ist also tatsächlich eine Intervallschachtelung.
Sei dann x die Zahl mit $x \in I_n \forall n$. Hat x die Nummer k, $x = x_k$, so gilt insbesondere $x \in I_k$. Dieses widerspricht Annahme (0.1). Also gibt es keine Abzählung von $\mathbb{R}$. □

Da der gesamte Körper der reellen Zahlen $\mathbb{R}$ nicht abzählbar ist, gilt dies insbesondere für das Intervall $[0, 1]$. □

Aufgabe 6

Seien $f : \mathbb{N} \to \mathbb{R}$ und $g : \mathbb{N} \to \mathbb{R}$ zahlentheoretische Funktionen sowie $n \in \mathbb{N}$.
$O(f)$ sei wie folgt definiert:

$$\begin{aligned} O(f) &= \{g | \exists c > 0 \in \mathbb{R}, \exists n_0 \in \mathbb{N} : g(n) \leq c \cdot f(n), \forall n \geq n_0\} \\ &= \{g | \exists c > 0 \in \mathbb{R}, \exists n_0 \in \mathbb{N} : \frac{g(n)}{f(n)} \leq c, \forall n \geq n_0\} \\ &= \{g | \exists c > 0 \in \mathbb{R} : \lim_{n \to \infty} \frac{g(n)}{f(n)} \leq c\} \end{aligned} \tag{0.2}$$

Die Formel (0.2) wurde in dieser Form in der Vorlesung „Praktische Informatik 2“ eingeführt.

Teilaufgabe a
Sei $g(n) = 3n + 4 + \log n$. Zu überprüfen: $g \in O(n)$.

$$\begin{aligned}
\lim_{n\to\infty} \frac{3n+4+\log n}{n} &= \lim_{n\to\infty} \left(\frac{\log n}{n} + \frac{3n+4}{n} \right) \\
&= \lim_{n\to\infty} \frac{\log n}{n} + \lim_{n\to\infty} \frac{3n+4}{n} \\
&= 0 + 3 \\
&= 3
\end{aligned}$$

Es kann ein $c > 0 \in \mathbb{R}$ angegeben werden, für das gilt: $\lim_{n\to\infty} \frac{g(n)}{f(n)} \leq c$, also z.B. $c = 4$.
Es gilt: $g \in O(n)$.

Teilaufgabe b
Sei $g(n) = (3n + \sqrt{n})^2$. Zu überprüfen: $g \in O(n^2)$.

$$\begin{aligned}
\lim_{n\to\infty} \frac{(3n+\sqrt{n})^2}{n^2} &= \lim_{n\to\infty} \left(\frac{9n^2 + 6n^{\frac{3}{2}} + n}{n^2} \right) \\
&= \lim_{n\to\infty} \left(9 + \frac{6}{\sqrt{n}} + \frac{1}{n} \right) \\
&= 9
\end{aligned}$$

Es kann ein $c > 0 \in \mathbb{R}$ angegeben werden, für das gilt: $\lim_{n\to\infty} \frac{g(n)}{f(n)} \leq c$, also z.B. $c = 10$.
Es gilt: $g \in O(n^2)$.

Teilaufgabe c
Sei $g(n) = \log n$. Zu überprüfen: $g \in O(\log(\sqrt{n}))$.

$$\begin{aligned}
\lim_{n\to\infty} \frac{\log n}{\log(\sqrt{n})} &\overset{\text{L'Hôpital}}{=} \lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{2n}} \\
&= \lim_{n\to\infty} \frac{2n}{n} \\
&= 2
\end{aligned}$$

Es kann ein $c > 0 \in \mathbb{R}$ angegeben werden, für das gilt: $\lim_{n\to\infty} \frac{g(n)}{f(n)} \leq c$, also z.B. $c = 3$.
Es gilt: $g \in O(\log(\sqrt{n}))$.

Teilaufgabe d
Sei $g(n) = \log n$. Zu überprüfen: $g \in O(\sqrt{n})$.

$$\begin{aligned}
\lim_{n\to\infty} \frac{\log n}{\sqrt{n}} &\overset{\text{L'Hôpital}}{=} \lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{2\sqrt{n}}} \\
&= \lim_{n\to\infty} \frac{2\sqrt{n}}{n} \\
&= \lim_{n\to\infty} \frac{2}{\sqrt{n}} \\
&= 0
\end{aligned}$$

Es kann ein $c > 0 \in \mathbb{R}$ angegeben werden, für das gilt: $\lim_{n\to\infty} \frac{g(n)}{f(n)} \leq c$, also z.B. $c = 1$. Es gilt: $g \in O(\sqrt{n})$.

Teilaufgabe e

Sei $g(n) = \sum_{i=1}^{n} i^{-2}$. Zu überprüfen: $g \in O(1)$.

$$\begin{aligned}
\lim_{n\to\infty} \frac{\sum_{i=1}^{n} i^{-2}}{1} &= \sum_{i=1}^{\infty} i^{-2} \\
&= \zeta(2) \\
&= \frac{\pi^2}{6} \\
&\approx 1,64
\end{aligned} \tag{0.3}$$

Bei Gleichung (0.3) handelt es sich um die Riemann'sche Zeta-Funktion, die als so genanntes „Baseler Problem“ von Euler 1734 berechnet worden ist. Es kann also ein $c > 0 \in \mathbb{R}$ angegeben werden, für das gilt: $\lim_{n\to\infty} \frac{g(n)}{f(n)} \leq c$, also z.B. $c = 2$. Es gilt: $g \in O(1)$.

Teilaufgabe f

Sei $g(n) = \sum_{i=1}^{n} i$. Zu überprüfen: $g \in O(n)$.

$$\begin{aligned}
\lim_{n\to\infty} \frac{\sum_{i=1}^{n} i}{n} &= \lim_{n\to\infty} \frac{\frac{n(n+1)}{2}}{n} \\
&= \lim_{n\to\infty} \frac{n+1}{2} \\
&= +\infty
\end{aligned}$$

Es kann kein $c > 0 \in \mathbb{R}$ angegeben werden, für das gilt: $\lim_{n\to\infty} \frac{g(n)}{f(n)} \leq c$, da $c < \infty, \forall c \in \mathbb{R}$. Demnach gilt: $g \notin O(n)$.

Teilaufgabe g

Sei $g(n) = \frac{n-n^{3/4}}{\sqrt{n}+5}$. Zu überprüfen: $g \in O(n^{1/4})$.

$$\begin{aligned}
\lim_{n\to\infty} \frac{\frac{n-n^{3/4}}{\sqrt{n}+5}}{n^{1/4}} &= \lim_{n\to\infty} \frac{n-n^{3/4}}{(\sqrt{n}+5)(n^{1/4})} \\
&= \lim_{n\to\infty} \frac{n^{3/4}-n^{1/2}}{\sqrt{n}+5} \\
&= \lim_{n\to\infty} \frac{n^{1/2}\left(n^{1/4}-1\right)}{\sqrt{n}+5} \\
&\overset{\text{L'Hôpital}}{=} \lim_{n\to\infty} \frac{\frac{3n^{1/4}-2}{4\sqrt{n}}}{\frac{1}{2\sqrt{n+5}}} \\
&= \lim_{n\to\infty} \frac{3n^{1/4}-2}{2} \\
&= +\infty
\end{aligned}$$

Es kann kein $c > 0 \in \mathbb{R}$ angegeben werden, für das gilt: $\lim_{n\to\infty} \frac{g(n)}{f(n)} \leq c$, da $c < \infty, \forall c \in \mathbb{R}$. Demnach gilt: $g \notin O(n^{1/4})$.

Teilaufgabe h

Sei $g(n) = \left(1 + \frac{2}{n-4}\right)^n$. Zu überprüfen: $g \in O(1)$.

$$\begin{aligned}
\lim_{n\to\infty} \frac{\left(1+\frac{2}{n-4}\right)^n}{1} &= \lim_{n\to\infty} \left(1+\frac{2}{n-4}\right)^n \\
&= e^2
\end{aligned}$$

Es kann also ein $c > 0 \in \mathbb{R}$ angegeben werden, für das gilt: $\lim_{n\to\infty} \frac{g(n)}{f(n)} \leq c$, also z.B. $c = 8$. Es gilt: $g \in O(1)$.

Aufgabe 7

Teilaufgabe a

Behauptung:
Sei m die Anzahl der führenden Nullen der Eingabe der Form $0^m 1z$ mit $z \in \{0,1\}^*$. Dann berechnet die gegebene Turingmaschine T die Funktion

$$f_T(m) = 1 - (m \bmod 2)$$

Das heißt, sie schreibt eine 1 auf das zweite Band, wenn die Anzahl der Nullen m gerade (oder $m = 0$) ist bzw. eine 0 auf das zweite Band, wenn die Anzahl der Nullen m ungerade ist.

Beweis:
Sei im Folgenden m die Anzahl der führenden Nullen in der Eingabe. Es werden zunächst spezielle Fälle ($m = 0 \dots 2$) untersucht und dann der allgemeine Fall überprüft.

- $m = 0$: Bei einer Eingabe, die mit der Ziffer 1 beginnt, geht T direkt in den Endzustand z_e über und schreibt eine 1 auf das zweite Band. Das erste Band bleibt dabei unverändert. Es gilt $1 = 1 - (0 \bmod 2)$.

- $m = 1$: Bei einer Eingabe der Form $01z$ mit $z \in \{0,1\}^*$ geht T aus dem Startzustand z_a in den Zustand z_0 über, nachdem der Kopf auf dem ersten Band nach rechts bewegt wurde. Anschließend schreibt T eine 0 auf das zweite Band und terminiert im Endzustand z_e. Dabei bleibt das erste Band unverändert. Es gilt $0 = 1 - (1 \bmod 2)$.

- $m = 2$: Bei einer Eingabe der Form $001z$ mit $z \in \{0,1\}^*$ geht T aus dem Startzustand z_a in den Zustand z_0 über, nachdem der Kopf nach rechts bewegt wurde. Anschließend geht T weiter in Zustand z_1, nachdem der Kopf erneut nach rechts bewegt wurde. Abschließend schreibt T eine 1 auf das zweite Band und terminiert im Endzustand z_e. Dabei bleibt das erste Band unverändert. Es gilt $1 = 1 - (2 \bmod 2)$.

- $m > 2$: Bei der Eingabe der Form 0^m1z mit $z \in \{0,1\}^*$ und $m > 2$ liest T zunächst die ersten zwei Nullen und verhält sich wie im vorherigen Fall, allerdings geht sie nicht in den Endzustand z_e über, sondern bewegt den Kopf nach rechts und betritt von z_1 aus eine Schleife zwischen den Zuständen z_0 und z_1:

  Die Zustände z_0 und z_1 werden nun getrennt betrachtet werden:

  - Zustand z_0: Die Maschine konnte in diesem Fall den Zustand z_0 nur von z_1 aus erreichen, d.h. T ging von z_1 aus nicht in den Endzustand über. Dies bedeutet, dass die Eingabe mit einer ungeraden Anzahl von Nullen begann. Liest T nun eine 1 auf dem ersten Band, muss sie sich dementsprechend verhalten, also eine 0 auf das zweite Band schreiben und terminieren.
  - Zustand z_1: Die Maschine konnte in diesem Fall den Zustand z_1 nur von z_0 aus erreichen, d.h. T ging von z_0 aus nicht in den Endzustand über. Dies bedeutet, dass die Eingabe mit einer geraden Anzahl von Nullen begann. Liest T nun eine 1 auf dem ersten Band, muss sie sich dementsprechend verhalten, also eine 1 auf das zweite Band schreiben und terminieren. □

Es wird bei diesem Beweis davon ausgegangen, dass die Eingabe überhaupt eine 1 enthält, da T sonst nicht stoppt und f_T nicht definiert wäre.

Teilaufgabe b

Für eine Eingabe aus $E_n = \{0^m10^{n-m-1} | 0 \leq m \leq n-1\}$ ist die kürzte mögliche Laufzeit für Eingaben für $m = 0$ (d.h. keine führenden Nullen) und beliebige n gegeben, da T

schon nach dem Lesen des ersten Zeichens der Eingabe (eine 1) terminiert es muss nur ein Schritt ausgeführt werden: Dann gilt für die Laufzeit

$$s = \max\{|u_i| + |v_i| : i = 1 \dots t, t = 1\} = \max\{|\square| + |1|\} = 1$$

Da T erst nach dem Lesen der ersten 1 terminiert, benötigt er bei Eingaben der Form $E_n = \{0^m 10^{n-m-1} | 0 \leq m \leq n-1\}$ genau $m+1$ Schritte (m Schritte um alle Nullen zu lesen und einen Schritt um die 1 zu verarbeiten) um zu terminieren. Für die maximale Laufzeit gilt also: $s = m + 1$. Die Endkonfiguration sieht dann wie folgt aus:

$$K_t = (z_e, 0^m, 10^{n-m-1})$$

Auch bei dieser Aufgabe wird davon ausgegangen, dass die Eingabe überhaupt eine 1 enthält, da T sonst nicht stoppt und keine Endkonfiguration erreichen würde.

Aufgabe 8

Die Überführung einer $k-TM$ in eine $1-TM$ wurde in Satz 1.1 aus der Vorlesung (Teil (ii) $\Rightarrow$ (iii)) bewiesen. Dabei wurde wie folgt vorgegangen:

> Sei M eine k-TM mit $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$, $L(M) = A$
>
> 1. Konstruiere 1-TM $M' = (Z' \cup Z, \Sigma, \Gamma', \delta', q_0, E)$
> 2. Dann läuft M' nach rechts über das Band und merkt sich im Zustand auf welchen Zeichen die Köpfe stehen: $\hat{a}_1, \dots, \hat{a}_k$ (endlich viele) bis sie die letzte Kopfmarkierung gefunden hat.
> 3. M' wählt eine Anweisung aus $\delta_M(q, a_1, \dots, a_k)$.
> 4. M' läuft wieder nach links und realisiert die Anweisungen (Zeichen mit Hut austauschen, Hut auf jeweiliges Nachbarzeichen).
> 5. M' geht in den neuen Zustand der Anweisung ($\in Z$).
>
> Es gilt: $L(M') = L(M)$, da die Konfiguration, in denen der Kopf ganz links steht, einer Konfigurationsfolge von M entspricht.

Für den Rechen- bzw. Platzbedarf ergibt sich also folgendes:

- Da M' die Bandbeschriftung von M durch k Spuren realisiert, entspricht der Platzbedarf von M' dem von M.
- Die Konstruktion der 1-TM M ist in konstanter Laufzeit $O(1)$ möglich (siehe Punkt 1).
- Bei jedem Rechenschritt von M können sich die (äußersten) Köpfe um maximal zwei Felder voneinander entfernen (indem ein Kopf einen Schritt nach rechts der andere ein Schritt nach links ausführt). Nach t Rechenschritten können sie also maximal $2t$ Felder auseinander liegen.

- Also braucht M' zur Simulation von t Schritten maximal

$$\sum_{i=1}^{t} 2i = t(t+1) = t^2 + t$$

  Schritte.

Der Platzbedarf ist also von s abhängig, also gilt für M' $O(s)$. Für den Rechenaufwand ergibt bei einer Laufzeit t von M für M' eine Laufzeit von $t^2 + t$ Schritten, also $O(t^2)$.
□

Serie 3

Institut für Informatik
Algorithmen und Komplexität
Dr. Till Nierhoff

WS 2002
30. Oktober 2002

Theoretische Informatik II

3. Serie

Abgabe bis zum 6. November 2002

Aufgabe 9 [4+6 Punkte]

Zeigen Sie folgende Eigenschaften der Reduktionsrelation $\leq$:
a) $A \leq B \quad \Leftrightarrow \quad \overline{A} \leq \overline{B}$
b) $\leq$ ist transitiv

Aufgabe 10 [4+6 Punkte]

a) Zeigen Sie, dass eine Sprache A genau dann rekursiv aufzählbar ist, wenn es eine Turing-berechenbare, partielle Funktion f gibt, so dass A der Definitionsbereich von f ist.

b) Beweisen sie die folgende Variante des Satzes von Rice: Sei $\mathcal{L} \subset \mathcal{RE}$ mit $\mathcal{L} \notin \{\emptyset, \mathcal{RE}\}$. Dann ist die Sprache $K(\mathcal{L}) := \{w \mid L(M_w) \in \mathcal{L}\}$ unentscheidbar.
Hinweis:
Nutzen Sie die Reduktion $K(\mathcal{F}) \leq K(\mathcal{L})$ für $\mathcal{F} = \mathcal{F}(\mathcal{L}) = \{M_w(.) \mid L(M_w) \in \mathcal{L}\}$.

Aufgabe 11 [2+2+2+2 Punkte]

Sind die folgenden Sprachen entscheidbar? Begründen Sie Ihre Antwort.
a) $\{w \mid L(M_w)$ ist endlich $\}$
b) $\{w \mid \overline{L(M_w)} = \overline{L(M_w)}\}$
c) $\{w \mid \overline{L(M_w)}$ ist rekursiv aufzählbar $\}$
d) $\{w \mid \exists w' \neq w : L(M_w) = L(M_{w'})\}$

Aufgabe 12 [4+4+4 Punkte]

Betrachten Sie die Sprache $\text{Eq} = \{v\#w \mid L(M_v) = L(M_w)\}$. Zeigen Sie:
a) Das Halteproblem lässt sich auf Eq reduzieren.
b) Das Halteproblem lässt sich auch auf $\overline{\text{Eq}}$ reduzieren.
c) Weder Eq noch $\overline{\text{Eq}}$ sind rekursiv aufzählbar.

Hinweis: Betrachten Sie Kodierungen von drei Maschinen:
eine tut das Gleiche wie M_w bei Eingabe x, eine akzeptiert immer und eine nie.

Aufgabe 9

Teilaufgabe a

Behauptung:
$A \leq B \Leftrightarrow \overline{A} \leq \overline{B}$

Beweis:
Sei $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$.

1. Richtung $\Rightarrow$:

   Voraussetzung:
   $A \leq B$, d.h. nach Definition: $\exists f : \Sigma^* \to \Gamma^* : x \in A \Leftrightarrow f(x) \in B$

   Laut Voraussetzung gilt: $x \notin A \Leftrightarrow f(x) \notin B$.
   Also gilt laut Definition des Komplementes: $x \in \overline{A} \Leftrightarrow f(x) \in \overline{B}$.
   f ist Reduktion: $\overline{A} \leq \overline{B}$.

2. Richtung $\Leftarrow$:

   Voraussetzung:
   $\overline{A} \leq \overline{B}$, d.h. nach Definition: $\exists f : \Sigma^* \backslash A \to \Gamma^* \backslash B : x \in \overline{A} \Leftrightarrow f(x) \in \overline{B}$

   Laut Voraussetzung gilt: $x \notin \overline{A} \Leftrightarrow f(x) \notin \overline{B}$.
   Also gilt lauf Definition des Komplementes: $x \in A \Leftrightarrow f(x) \in B$.
   f ist Reduktion: $A \leq B$.

□

Teilaufgabe b

Behauptung:
$\leq$ ist transitiv, d.h. $A \leq B \wedge B \leq C \Rightarrow A \leq C$.

Beweis:
Sei $A \subseteq \Sigma^*$, $B \subseteq \Gamma^*$ und $C \subseteq \Psi^*$.

Voraussetzung:
$A \leq B$, d.h. nach Definition: $\exists f_1 : \Sigma^* \to \Gamma^* : x \in A \Leftrightarrow f_1(x) \in B$
$B \leq C$, d.h. nach Definition: $\exists f_2 : \Gamma^* \to \Psi^* : x \in B \Leftrightarrow f_2(x) \in C$

$$x \in A \Leftrightarrow f_1(x) \in B$$

$$x \in B \Leftrightarrow f_2(x) \in C$$

$$x \in A \Leftrightarrow f_2(f_1(x)) \in C$$

Sei $f_3 = f_1 \circ f_2$ Reduktion: $A \leq C$.

□

Aufgabe 10

Teilaufgabe a

Behauptung:
A rekursiv aufzählbar $\Leftrightarrow \exists f : f$ partiell, f Turing-berechenbar, $D(f) = A$

Beweis:
Sei $A \subseteq \Sigma^*$.

1. Richtung $\Rightarrow$

   Voraussetzung:
   A rekursiv aufzählbar.

   Nach Satz 1.1 aus der Vorlesung ist A semi-entscheidbar, d.h. es exisitiert eine charakteristische Funktion $\hat{\chi}_A : A \to \{1, \uparrow\}$ mit

   $$\hat{\chi}_A = \begin{cases} 1, & \text{falls } x \in A \\ \uparrow, & \text{falls } x \notin A \end{cases}$$

   $\hat{\chi}_A$ hat A als Definitionsbereich, ist Turing-berechenbar und partiell.

2. Richtung $\Leftarrow$

   Voraussetzung:
   f ist Turing-berechenbar, partiell und hat A als Definitionsbereich.

   f ist Turing-berechenbar, d.h. es exisitiert eine Turingmaschine M, die f berechnet. Diese Funktion ist nur für die Menge A definiert (weil f partiell), d.h. M akzeptiert nur die Sprache A und stoppt für Eingaben $x \notin A$ nie. Die charakteristische Funktion $\hat{\chi}_A$ mit

   $$\hat{\chi}_A = \begin{cases} 1, & \text{falls } x \in A \\ \uparrow, & \text{falls } x \notin A \end{cases}$$

   ist berechenbar. Laut Definition und Satz 1.1 aus der Vorlesung ist also $A \in \mathcal{RE}$, d.h. rekursiv aufzählbar.

□

Teilaufgabe b

- Eine Sprache aus $\mathcal{RE}$ ist rekursiv aufzählbar, d.h. kann durch die charakteristische Funktion $\hat{\chi}$ entschieden werden. Dabei ist $\hat{\chi}$ Turing-berechenbar und partiell.
- Nach Folgerung 3.4.5 aus dem Starke-Skript entspricht die Menge der Turing-berechenbaren Funktionen der Menge der partiell-rekursiven Funktionen, d.h.

$$\mathcal{RE} = \text{PREK}$$

- Da für die Menge $\mathcal{L} \in \mathcal{RE} = \text{PREK}$ Nichttrivialität verlangt wird ($\mathcal{L} \notin \{\emptyset, \mathcal{RE}\}$), kann der Satz von Rice auf die Sprache $K(\mathcal{L})$ angewendet werden, d.h. $K(\mathcal{L}) \notin \mathcal{REC}$.
- $K(\mathcal{L})$ ist also unentscheidbar. □

Aufgabe 11

Teilaufgabe a

$\{w \mid L(M_w)$ ist endlich$\}$, d.h. M_w akzeptiert eine endliche Sprache. Damit hat die von M_w berechnete Funktion einen endlichen Definitionsbereich. Da die Menge der Funktionen mit endlichem Definitionsbereich eine nichttrivale Teilmenge von PREK ist, sind die Voraussetzungen für den Satz von Rice erfüllt:
$\Rightarrow \{w \mid L(M_w)$ ist endlich$\} \notin \mathcal{REC}$

Teilaufgabe b

$\{w \mid L(M_w) = \overline{L(M_w)}\}$ ist die Menge aller Turingmaschinen, deren akzeptierte Sprachen ihrem Komplement entspricht. Da dies für keine Menge gelten kann, kann die Voraussetzung $L(M_w) = \overline{L(M_w)}$ nie erfüllt werden. $\{w \mid L(M_w) = \overline{L(M_w)}\}$ entspricht also der leeren Menge $\emptyset$.
Laut Definition 4.2c (Köbler-Skript) ist die leere Menge rekursiv aufzählbar, also nach Satz 1.1 aus der Vorlesung semi-entscheidbar.
$\Rightarrow \{w \mid L(M_w) = \overline{L(M_w)}\} \notin \mathcal{REC}$

Teilaufgabe c

Zu überprüfen ist die Menge $\{w \mid \overline{L(M_w)}$ ist rekursiv aufzählbar$\}$. Das heißt, dass die Sprache $\overline{L(M)}$ mittels der Funktion

$$\hat{\chi}_{\overline{L(M_w)}} = \begin{cases} 1, & \text{falls } x \in \overline{L(M_w)} \\ \uparrow, & \text{sonst} \end{cases}$$

semi-entschieden werden kann. Damit ist $\{w \mid \overline{L(M_w)}$ ist rekursiv aufzählbar$\} \notin \mathcal{REC}$

Teilaufgabe d

$\{w \mid \exists w' \neq w : L(M_w) = L(M_{w'})\}$ ist die Menge aller Turingmaschinen M_w, deren akzeptierte Sprache $L(M_w)$ auch von einer Maschine $M_{w'}$ mit von w unterschiedlicher Kodierung w' akzeptiert wird.
Nun kann die Überführungsfunktion einer jeden Turingmaschine um einen Zustand erweitert werden, der von den ursprünglichen Zuständen nicht erreicht werden kann und somit nichts an der akzeptierten Sprache, allerdings an ihrer Kodierung ändert.
Also kann zu jeder gültigen Turingmaschine M_w eine Maschine $M_{w'}$ gefunden werden, deren Kodierungen (w und w') unterschiedlich sind, allerdings die gleiche Sprache akzeptieren.
Damit ist die Menge entscheidbar, da die geforderte Eigenschaft für alle Turingmaschinen gilt: $\{w \mid \exists w' \neq w : L(M_w) = L(M_{w'})\} \in \mathcal{REC}$

Aufgabe 12

Teilaufgabe a

Sei $H = \{w\#x \mid M_w(x) \neq \uparrow\}$ das Halteproblem und $\text{Eq} = \{v\#w \mid L(M_v) = L(M_w)\}$.
Das Halteproblem H lässt sich auf Eq wie folgt reduzieren:
Zu zeigen: $H \leq \text{Eq}$, d.h. nach Definition gilt: $x' \in H \Leftrightarrow f(x') \in \text{Eq}$.

$$\begin{aligned} w\#x = x' \in H &\Leftrightarrow f(x') \in \text{Eq} \\ &\Leftrightarrow f(w\#x) \in \text{Eq} \\ &\Leftrightarrow f(w\#x) = v\#w \in \text{Eq} \\ &\Leftrightarrow L(M_v) = L(M_w) \end{aligned}$$

Sei nun $v = w$, sodass $L(M_v) = L(M_w)$. Also existiert eine Reduktion von H auf Eq:

$$f_{\text{Eq}}(w\#x) = \begin{cases} w\#w, & \text{falls } w\#x \in H \\ w' \notin \text{Eq}, & \text{sonst} \end{cases}$$

f_{Eq} ist berechenbar.

Teilaufgabe b

Sei $H = \{w\#x \mid M_w(x) \neq \uparrow\}$ das Halteproblem und $\overline{\text{Eq}} = \{v\#w \mid L(M_v) \neq L(M_w)\}$.
Das Halteproblem H lässt sich auf $\overline{\text{Eq}}$ wie folgt reduzieren:
Zu zeigen: $H \leq \overline{\text{Eq}}$, d.h. nach Definition gilt: $x' \in H \Leftrightarrow f(x') \in \overline{\text{Eq}}$.

$$\begin{aligned} w\#x = x' \in H &\Leftrightarrow f(x') \in \overline{\text{Eq}} \\ &\Leftrightarrow f(w\#x) \in \overline{\text{Eq}} \\ &\Leftrightarrow f(w\#x) = v\#w \in \overline{\text{Eq}} \\ &\Leftrightarrow L(M_v) \neq L(M_w) \end{aligned}$$

Laut Voraussetzung ist $w\#x \in H$, d.h. $\exists x : M_w(x) \neq \uparrow$. Also ist $L(M_w) \neq \emptyset$. Sei nun $L(M_v) = \emptyset$, d.h. M_v die Turingmaschine, die nie akzeptiert.
Also existiert eine Reduktion von H auf $\overline{\text{Eq}}$:

$$f_{\overline{\text{Eq}}}(w\#x) = \begin{cases} v\#w, & \text{falls } w\#x \in H \text{ (sei } L(M_v) = \emptyset) \\ w' \notin \overline{\text{Eq}}, & \text{sonst} \end{cases}$$

$f_{\overline{\text{Eq}}}$ ist berechenbar.

Teilaufgabe c

Behauptung: Eq $\notin \mathcal{RE}$ und $\overline{\text{Eq}} \notin \mathcal{RE}$.

- In Teilaufgabe a und b haben wir gezeigt, dass sich das Halteproblem sowohl auf Eq, als auch auf $\overline{\text{Eq}}$ reduzieren lässt.
- Wäre nun Eq bzw. $\overline{\text{Eq}}$ entscheidbar, dann würde sich diese Entscheidbarkeit nach Satz 4.10 (Köbler-Skript) auch auf das Halteproblem übertragen.
- Da wir jedoch wissen, dass $K \notin \mathcal{REC}$, gilt auch: Eq $\notin \mathcal{REC}$ und $\overline{\text{Eq}} \notin \mathcal{REC}$.

  Behauptung: Es existiert eine Turingmaschine M, die $\overline{\text{Eq}}$ semi-entscheidet.

  Beweis: Sei M eine Turingmaschine, die folgendes leistet:

  - M simuliert M_v und M_w auf allen möglichen Eingaben $x \in \Sigma^*$.
  - Wenn x von M_v und M_w akzeptiert wird, dann simuliere mit nächster Eingabe aus Σ^*. M soll nicht stoppen, wenn bereits für alle möglichen Eingaben simuliert wurde.
  - Wenn z.B. M_v eine bestimmte Eingabe x akzeptiert, die M_w nicht akzeptiert (oder umgekehrt), dann gilt: $v\#w \in \overline{\text{Eq}}$, also soll M akzeptieren.

  M semi-entscheidet also $\overline{\text{Eq}}$, d.h. $\overline{\text{Eq}} \in \mathcal{RE}$. □

Wenn nun $\overline{\text{Eq}} \in \mathcal{RE}$, dann muss Eq $\notin \mathcal{RE}$, weil sonst nach Satz 4.4 (Köbler-Skript) Eq $\in \mathcal{REC}$ gelten würde, was ein Widerspruch zur Reduktion $H \leq$ Eq darstellt (siehe oben). Also gilt die zu zeigende Behauptung nicht.

Serie 4

Institut für Informatik
Algorithmen und Komplexität
Dr. Till Nierhoff

WS 2002
6. November 2002

Theoretische Informatik II

4. Serie

Abgabe bis zum 13. November 2002

Aufgabe 13 [10 Punkte]

Sei Σ ein durch $<$ geordnetes Alphabet. Die lexikographische Ordnung auf Σ^* ist dann durch $x < y :\Leftrightarrow \begin{cases} |x| < |y| & \text{oder} \\ |x| = |y|,\ x_1 \cdots x_{i-1} = y_1 \cdots y_{i-1} \text{ und } x_i < y_i \text{ für ein } i \leq |x| \end{cases}$ gegeben. Eine Menge $A \subseteq \Sigma^*$ heißt *in lexikographischer Ordnung rekursiv aufzählbar*, falls $A = \emptyset$ oder falls es eine berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ mit Wertebereich A gibt, so daß $f(x) \leq f(y)$, falls $x < y$. Zeigen Sie:

A ist genau dann in lexikographischer Ordnung rekursiv aufzählbar, wenn A entscheidbar ist.
Hinweis für eine Richtung: Betrachten Sie den Beweis für „iv) $\Longrightarrow$ v)“ des Satzes 1.1 der Vorlesung.

Aufgabe 14 [5+5 Punkte]

a) In der Vorlesung wurde H$\leq$MPKP gezeigt (Satz 2.5). Wie muss der Beweis modifiziert werden, um $L \leq$ MPKP zu zeigen, wobei $L = L(M)$ und M eine 1-DTM seien?

b) Gegeben ist die TM $T = (\{s, z\}, \{0, 1\}, \{0, 1, \Box\}, \delta, s, \{z\})$ mit

$$\delta = \{(s, 0) \to (s, 1, R), \quad (s, 1) \to (s, 0, R), \quad (s, \Box) \to (z, \Box, N)\}.$$

Geben Sie die Kopier-, Überführungs-, Lösch- und Abschlussregeln aus Ihrer Lösung zu a) an. Geben Sie außerdem die Startregel zur Eingabe $w = 011010$ an.

Aufgabe 15 [4+4+4 Punkte]

Entscheiden Sie, ob die folgenden Wörter zur Sprache PKP gehören oder nicht. Begründen Sie Ihr Urteil.

a) $w_1 = \begin{pmatrix} aa & a & abab & ab \\ a & bababa & ba & a \end{pmatrix}$

b) $w_2 = \begin{pmatrix} 000 & 00 \\ 0 & 00000 \end{pmatrix}$

c) $w_3 = \begin{pmatrix} aa & aaa \\ a & aa \end{pmatrix}$

Aufgabe 16 [8 Punkte]

Zeigen Sie: Die Sprache PKP auf einem einelementigen Alphabet ist entscheidbar.

Aufgabe 13

Zu zeigen: A in lexikografischer Reihenfolge rekursiv aufzählbar $\Leftrightarrow$ A entscheidbar.

Beweis von Richtung $\Rightarrow$

Voraussetzung:
A sei in lexikografischer Reihenfolge rekursiv aufzählbar.

Behauptung:
A ist entscheidbar.

Beweis:
Wenn A in lexikografischer Reihenfolge rekursiv aufzählbar ist, dann kann für jeweils zwei verschiedene Wörter $x, y \in A$ entschieden werden, ob $a < b$ oder $b < a$. Da dies jedoch laut Vorausetzung für alle Wörter aus Σ^* entschieden werden kann (da Σ ein durch $<$ geordnetes Alphabet ist), ist die Funktion $f : \Sigma^* \to \Sigma^*$ mit $W_f = A$ total, d.h. für alle $x, y \in \Sigma^*$ definiert.

Es gibt also für alle $x < y$ $(x, y \in \Sigma^*)$ Funktionswerte, für die gilt $f(x) \leq f(y)$ und $f(x), f(y) \in A$.

Dementsprechend ist $\chi_A(x) = \begin{cases} 1, & \text{falls } x \in \Sigma^* \\ 0, & \text{sonst} \end{cases}$ und damit $A \in \mathcal{REC}$.

Beweis von Richtung $\Leftarrow$

Voraussetzung:
A sei entscheidbar.

Behauptung:
A ist in lexikografischer Reihenfolge rekursiv aufzählbar.

Beweis:
Laut Köbler 4.2c kann es sich bei A nicht um die leere Menge handeln, da diese nicht entscheidbar ist.

Sei nun T eine Turingmaschine, die die Sprache A lexikographisch ordnet. Dazu zählt sie zunächst A auf dem 1. Band (getrennt durch $\square$) auf. Dies ist möglich, da A entscheidbar ist, d.h. für alle $x \in \Sigma^*$ kann entschieden werden, ob der Fall $x \in A$ oder $x \notin A$ gilt.

Nun liest sie das erste Wort auf dem 1. Band ein und schreibt es auf das 3. Band.

Anschließend geht sie wie folgt vor: Sie liest das nächste Wort auf dem 1. Band ein und sucht auf dem 3. Band die Stelle, an der das gelesene Wort nach der Definition der lexikografischen Ordnung einsortiert werden muss. Steht an dieser Stelle bereits ein anderes

Wort, so wird dieses Wort auf das 2. Band kopiert, und durch durch das eingelesene Wort ersetzt. Anschließend wird das Wort auf dem 2. Band nach dem oben angegebenen Verfahren einsortiert. Anschließend wird das 2. Band gelöscht.

Die Maschine stoppt, wenn alle Wörter auf dem 1. Band auf dem 3. Band in lexikografischer Reihenfolge aufgezählt sind.

> **Anmerkung:** Wir gehen bei der Turingmaschine T davon aus, dass sie erkennt, wenn sie das letzte Wort auf dem 1. Band gelesen hat (beispielsweise wenn sie zwei $\square$ hintereinander gelesen hat). Des Weiteren soll sie beim Einsortieralgorithmus verschieden lange Worte entsprechend verarbeiten können, d.h. auch ein kürzeres durch ein längeres und umgekehrt ersetzen können. Dies kann beispielsweise durch zeichenweises Kopieren auf dem 3. Band realisiert werden.

Also ist A in lexikografischer Reihenfolge rekursiv aufzählbar, wenn $A \in \mathcal{REC}$. $\square$

Aufgabe 14

Teilaufgabe a

Sei $L = L(M)$ mit M ist 1-DTM gegeben. Sei des Weiteren w_M die Gödelzahl der 1-DTM M.

Behauptung:
$L \leq \text{MPKP}$

Beweis:
Die Reduktionsrelation $\leq$ ist transitiv. Da $H \leq \text{MPKP}$ bereits bewiesen worden ist (Satz 2.5), genügt also zu zeigen, dass gilt: $L \leq H$. Laut Definition der Reduzierung gilt: $x \in L(M) \Leftrightarrow f(x) \in H$.

$$\begin{aligned} x \in L(M) &\Leftrightarrow f(x) \in H \\ &\Leftrightarrow w_M \# x \in H \end{aligned}$$

Also existiert eine Reduktion von $L = L(M)$ auf H:

$$f(x) = w_M \# x$$

wobei wie oben definiert w_M die Gödelzahl der 1-DTM M ist. f ist berechenbar.

$(L \leq H) \wedge (H \leq \text{MPKP}) \Rightarrow L \leq \text{MPKP}$ $\square$

Teilaufgabe b

Da in Teilaufgabe a die Reduktion $L \leq \text{MPKP}$ gezeigt worden ist, kann nun analog zum Beweis $H \leq \text{MPKP}$ zu der gegebenen 1-DTM $T = (\{s, z\}, \{0, 1\}, \{0, 1, \square\}, \delta, s, \{z\})$ die Kopier-, Überführungs-, Lösch und Abschlussregeln, sowie die Startregel bei der Eingabe $w = 011010$ angegeben werden.

Startregel

Für Startzustand s und Eingabe $w = 011010$:

$$\begin{aligned} x_1 &= \$ \\ y_1 &= \$s011010\$ \end{aligned}$$

Kopierregeln

Es gibt insgesamt 4 Kopierregeln:

$$\begin{array}{cccc} x = 0 & x = 1 & x = \square & x = \$ \\ y = 0 & y = 1 & y = \square & y = \$ \end{array}$$

Überführungsregeln

Es gibt folgende Überführungsfunktionen:

$$\begin{array}{rl} \begin{aligned} x &= s0 \\ y &= 1s \end{aligned} & \text{für die Überführung } (s, 0) \mapsto (s, 1, R) \\ \begin{aligned} x &= s1 \\ y &= 0s \end{aligned} & \text{für die Überführung } (s, 1) \mapsto (s, 0, R) \\ \begin{aligned} x &= s\$ \\ y &= z\square\$ \end{aligned} & \text{für die Überführung } (s, \square) \mapsto (z, \square, N) \end{array}$$

Löschregeln

Es gibt 8 Löschregeln:

$$\begin{array}{cccccccc} x = 0z & x = z0 & x = 1z & x = z1 & x = \square z & x = z\square & x = \$z & x = z\$ \\ y = 0 & y = 0 & y = 1 & y = 1 & y = \square & y = \square & y = \$ & y = \$ \end{array}$$

Abschlussregel

Es gibt die Abschlussregel

$$\begin{aligned} x &= z\$\$ \\ y &= \$ \end{aligned}$$

Aufgabe 15

Teilaufgabe a

Gegeben: $w_1 = \begin{pmatrix} aa & a & abab & ab \\ a & bababa & ba & a \end{pmatrix}$

Das Wort w_1 gehört zur Sprache PKP. Es hat eine Lösung (4,3,3,2). Es gilt:

$$x_4x_3x_3x_2 = abababababa = y_4y_3y_3y_2$$

Teilaufgabe b

Gegeben: $w_2 = \begin{pmatrix} 000 & 00 \\ 0 & 00000 \end{pmatrix}$

Das Wort w_2 gehört zur Sprache PKP. Es hat eine Lösung (1,2,1,1,2). Es gilt:

$$x_1x_2x_1x_1x_2 = 0000000000000 = y_1y_2y_1y_1y_2$$

w_2 gehört sogar zur Sprache MPKP.

Teilaufgabe c

Gegeben: $w_3 = \begin{pmatrix} aa & aaa \\ a & aa \end{pmatrix}$

Das Wort w_3 gehört nicht zur Sprache PKP, da es keine Lösung gibt: Jede der beiden Regeln sorgt bei der x-Folge für einen Vorsprung von einem a, der nicht aufgeholt werden kann.

Aufgabe 16

Behauptung:
Die Sprache PKP auf einem einelementigem Alphabet ist entscheidbar.

Beweis:
Bei einem Wort der Form $\begin{pmatrix} x_1 & x_2 & \cdots & x_k \\ y_1 & y_2 & \cdots & y_k \end{pmatrix}$ auf einem einelementigem Alphabet unterscheiden sich die einzelnen Regeln nur durch die Anzahl der Zeichen in der x-Folge und der y-Folge.

Bei der Suche einer Lösung für ein Problem ist jedoch nur die Differenz der der Anzahl der Zeichen der x- und y-Folge interessant. So kann jeder Regel diese Differenz zugeordnet werden. Es gibt also eine Abbildung von den einzelnen Regeln auf die ganzen Zahlen:

$$f\left((x_i, y_i)\right) = |x_i| - |y_i|$$

Die Funktionswerte geben dabei den Vorsprung an, den die x-Folge nach Ausführung der jeweiligen Regel auf- (für $|x_i| - |y_i| > 0$) bzw. abbaut (für $|x_i| - |y_i| < 0$).

Um zu entscheiden, ob ein Wort zum PKP auf einem einelementigen Alphabet gehört, muss also überprüft werden, ob es eine Summe von Funktionswerten (oder einen einzigen Funktionswert) gibt, die Null ergibt, d.h. die x- und y-Folge sind gleich lang und durch das einelementige Alphabet auch gleich.
Nun gibt es folgende Möglichkeiten:

1. Alle Funktionswerte sind positiv. Es gibt also keine Lösung, da die x-Folge stets einen Vorsprung hat, der nicht aufgeholt werden kann.

2. Alle Funktionswerte sind negativ. Analog gibt es wieder keine Lösung, da nun die y-Folge einen uneinholbaren Vorsprung hat.

3. Es gibt eine Regel a mit dem Funktionswert $f(a) = 0$, d.h. eine Lösung ergibt sich aus eben dieser Regel a.

4. Es gibt negative und positive Funktionswerte also beispielsweise $f(a) < 0$ und $f(b) > 0$. Dann führt die $f(b)$-fache Wiederholung der Regel a, gefolgt von der $|f(a)|$-fachen Wiederholung der Regel b zu einer Lösung, da der Vorsprung der y-Folge, der zunächst aufgebaut wird, durch die Ausführung der b-Regeln aufgeholt wird und es gilt: $f(b) \cdot a + |f(a)| \cdot b = 0$.

Es gibt also nur vier Möglichkeiten, die zu überprüfen sind, d.h. es kann ein endlicher Algorithmus definiert werden, der jede Möglichkeit untersucht.

Dafür berechnet eine Turingmaschine von allen Regeln $a_1, a_2, \ldots, a_n$ alle möglichen Funktionswerte (endlich viele) und überprüft, ob anhand der beschriebenen Möglichkeiten eine Lösunge gefunden werden kann. Sie stoppt dabei in jedem Fall, nämlich entweder, wenn ein Fall mit Lösung gefunden wurde (erfolgreich), oder wenn alle Möglichkeiten überprüft worden sind und kein Lösungsfall gefunden wurde (erfolglos).

Demnach ist die Sprache PKP auf einem einelementigen Alphabet entscheidbar. □

Anmerkung: Das Wort w_2 aus Aufgabe 15b entspricht der 3. beschriebenen Möglichkeit ($f((000, 0)) = 2 > 0$, $f((00, 00000)) = -3 < 0$) und das Wort w_3 auf Aufgabe 15c entspricht der 4. beschriebenen Möglichkeit ($f((aa, a)) = 1 > 0$, $f((aaa, aa)) = 1 > 0$).

Serie 5

Institut für Informatik
Algorithmen und Komplexität
Dr. Till Nierhoff

WS 2002
13. November 2002

Theoretische Informatik II

5. Serie

Abgabe bis zum 20. November 2002

Aufgabe 17 [3+3+4 Punkte]

Sei $G = (\{S, A, B\}, \{a, b\}, P, S)$ eine Grammatik mit P:

$$S \to aB, bA$$
$$A \to a, aS, bAA$$
$$B \to b, bS, aBB.$$

Geben Sie für das Wort *aaabbabbba*

a) eine Linksableitung, d.h. eine Ableitung, bei der jeweils das am weitesten links stehende Nichtterminalsymbol in einer Satzform ersetzt wird, und

b) eine Rechtsableitung, d.h. eine Ableitung, bei der jeweils das am weitesten rechts stehende Nichtterminalsymbol in einer Satzform ersetzt wird, an.

c) Ist die angegebene Grammatik mehrdeutig, d.h gibt es ein Wort $w \in L(G)$, für das mindestens zwei verschiedene Linksableitungen existieren?

Aufgabe 18 [5+5 Punkte]

Entscheiden Sie zu den folgenden Grammatiken, von welchem Typ (höchstmöglich) sie sind und welche Sprache sie erzeugen. Begründen Sie Ihre Antwort.

a) $G_1 = (\{A, S\}, \{(,), [,]\}, P, S)$ mit P:

$$S \to \varepsilon, (S), A], SS$$
$$A \to (SA, (S$$

b) $G_2 = (\{S, A, B, C\}, \{0, 1\}, P, S)$ mit P:

$$S \to 0, 1A$$
$$A \to 1B, 0C$$
$$B \to \varepsilon, 1A, 0B$$
$$C \to 1C, 0A$$

Aufgabe 19 [4+6 Punkte]

Die Palindrome über einem Alphabet Σ sind definiert durch:
ε und a mit $a \in \Sigma$ sind Palindrome.
Ist w ein Palindrom, dann auch awa mit $a \in \Sigma$.

a) Erstellen Sie eine Grammatik G, die alle Palindrome über $\Sigma = \{a, b, c\}$ erzeugt.

b) Beweisen Sie, dass $L(G)$ die Menge aller Palindrome über $\{a, b, c\}$ ist.

Aufgabe 20 [2+2+2+4 Punkte]

Sei $G = (\{S\}, \{a, b\}, P, S)$ eine Grammatik mit $P : \; S \to aSb, bSa, SS, \varepsilon$. Zeigen Sie, dass

a) $abab$ zu $L(G)$ gehört

b) für jedes $w \in \Sigma^*$ gilt: wenn $w \in L(G)$, dann sind auch $awb, bwa \in L(G)$,

c) für alle $w, w' \in \Sigma^*$ gilt: wenn $w, w' \in L(G)$, dann ist auch $ww' \in L(G)$,

d) $L(G)$ die Menge aller Wörter ist, die die gleiche Anzahl von a's und b's enthalten.
Hinweis zur Rückrichtung: Induktion über die Wortlänge; Teile b) und c) verwenden.

Aufgabe 17

Teilaufgabe a

Das Wort $aaabbabbba$ kann wie folgt durch Linksableitung erzeugt werden:

$$
\begin{aligned}
S &\Rightarrow aB \\
&\Rightarrow aaBB \\
&\Rightarrow aaaBBB \\
&\Rightarrow aaabSBB \\
&\Rightarrow aaabbABB \\
&\Rightarrow aaabbaBB \\
&\Rightarrow aaabbabB \\
&\Rightarrow aaabbabbS \\
&\Rightarrow aaabbabbbA \\
&\Rightarrow aaabbabbba
\end{aligned}
$$

Teilaufgabe b

Das Wort $aaabbabbba$ kann wie folgt durch Rechtsableitung erzeugt werden:

$$
\begin{aligned}
S &\Rightarrow aB \\
&\Rightarrow aaBB \\
&\Rightarrow aaBaBB \\
&\Rightarrow aaBaBbS \\
&\Rightarrow aaBaBbbA \\
&\Rightarrow aaBaBbba \\
&\Rightarrow aaBabbba \\
&\Rightarrow aaaBBabbba \\
&\Rightarrow aaaBbabbba \\
&\Rightarrow aaabbabbba
\end{aligned}
$$

Teilaufgabe c

Das Wort $aaabbabbba$ kann – alternativ zu Teilaufgabe a – wie folgt durch Linksableitung erzeugt werden:

$$
\begin{aligned}
S &\Rightarrow aB \\
&\Rightarrow aaBB \\
&\Rightarrow aaaBBB \\
&\Rightarrow aaabBB \\
&\Rightarrow aaabbSB \\
&\Rightarrow aaabbaBB \\
&\Rightarrow aaabbabB \\
&\Rightarrow aaabbabbS \\
&\Rightarrow aaabbabbbA \\
&\Rightarrow aaabbabbba
\end{aligned}
$$

Die gegebene Grammatik ist also mehrdeutig.

Aufgabe 18

Teilaufgabe a:

Die gegebene Grammatik G_1 ist vom Typ 0, da sie die Regel $S \to \varepsilon$ enthält und somit von der ε-Sonderregel gebraucht machen müsste, um von höherem Typ zu sein. Allerdings steht das Startsymbol S auch auf der rechten Seite der Regeln.

Die von der Grammatik G_1 beschriebene Sprache $L(G_1)$ beschreibt die Sprache alle Klammerausdrücke, die wie folgt charakterisiert ist:

- Jedes nicht-leere Wort aus $L(G_1)$ beginnt mit (.
- Das Terminalzeichen [kommt in $L(G_1)$ nicht vor.
- Sobald die A-Regel benutzt wird, ist die Anzahl der (-Klammern größer als die Anzahl der)-Klammern. Wenn nur die S-Regel benutzt wird, ist die Anzahl der öffnenden (-Klammern gleich der Anzahl der schließenden)-Klammern.

Teilaufgabe b:

Die gegebene Grammatik G_2 ist vom Typ 3, da für alle Regeln $u \to v$ gilt:
$u \in V = \{S, A, B, C\}$ und $v \in \Sigma V \cup \Sigma \cup \{\varepsilon\}$ mit $\Sigma = \{0, 1\}$.

Die von der Grammatik G_2 beschriebene Sprache $L(G_2)$ beschreibt alle Binärzahlen, die ohne Rest durch 3 teilbar sind.

$$
\begin{aligned}
S &\Rightarrow 0 \\
S &\Rightarrow^* 11 \quad (3_{10}) \\
S &\Rightarrow^* 110 \quad (6_{10}) \\
S &\Rightarrow^* 1001 \quad (9_{10}) \\
S &\Rightarrow^* 1100 \quad (12_{10}) \\
S &\Rightarrow^* 1111 \quad (15_{10}) \\
S &\Rightarrow^* 10010 \quad (18_{10}) \\
S &\Rightarrow^* 10101 \quad (21_{10}) \\
S &\Rightarrow^* 11000 \quad (24_{10}) \\
&\vdots
\end{aligned}
$$

Es ist nicht möglich, andere Kombinationen von Nullen und Einsen abzuleiten, die als Binärzahl dargestellt nicht durch drei teilbar sind.

Aufgabe 19

Teilaufgabe a:

Die Grammatik $G = (\{S\}, \{a, b, c\}, P, S)$ mit P:

$$
\begin{aligned}
S &\rightarrow \varepsilon \\
S &\rightarrow a, b, c \\
S &\rightarrow aSa, bSb, cSc
\end{aligned}
$$

erzeugt alle Palindrome über $\Sigma = \{a, b, c\}$.

Teilaufgabe b:

Behauptung:
$x \in L(G) \Leftrightarrow x$ ist Palindrom über $\Sigma = \{a, b, c\}$

Beweis:
Richtung „⇒“:

> Voraussetzung: $x \in L(G)$
> Behauptung: x ist Palindrom über $\Sigma = \{a, b, c\}$
>
> **Beweis:**
> durch Induktion über die Wortlänge $n = |x|$:

Induktionsanfang ($n = 0$, $n = 1$):
Das leere Wort $x = \varepsilon$ mit $|x| = 0$ kann nur durch einmaliges Anwenden der Regel $S \to \varepsilon$ aus G abgeleitet werden, da dies die einzige Regel ist, die ε enthält. Es ist per Definition ein Palindrom.

Alle Worte der Länge 1, also a, b und c für das Alphabet $\{a, b, c\}$ können nur durch einmaliges Anwenden der Regeln $S \to a, b, c$ aus G abgeleitet werden, da alle anderen Regeln entweder die Zeichenkette verlängern oder auf das leere Wort ε ableiten. Alle Worte der Länge 1 über einem Alphabet Σ, also in diesem Fall a, b und c sind per Definition Palindrome.

Induktionsschritt ($n \to n + 1$):
Sei $x \in L(G)$ mit $|x| > 1$. Das Wort x besteht also ausschließlich aus Terminalsymbolen und ist länger als ein Zeichen. Da die Regeln $S \to \varepsilon$ und $S \to a, b, c$ keine Worte y mit $|y| > 1$ erzeugen können, muss bei der Ableitung von x mindestens eine der Regeln $S \to aSa, bSb, cSc$ angewedendet worden sein, da nur sie eine Verlängerung der Ableitung zulassen.

Die Anwendung der Regeln $S \to aSa, bSb, cSc$ erhalten den Charakter eines Palindromes, da sie den gleichen Buchstaben an eine Zeichenfolge, in deren Mitte die Variable S steht, anhängt. Da Wort x kann nur ein Palindrom sein, wenn aus der Variable S letztlich ein Palindrom abgeleitet werden kann. Dies ist im Induktionsanfang bewiesen worden.

Also ist jedes Wort $x \in L(G)$ ein Palindrom über $\Sigma = \{a, b, c\}$.

Beweis:
Richtung „$\Leftarrow$“:

Voraussetzung: x ist Palindrom über $\Sigma = \{a, b, c\}$
Behauptung: $x \in L(G)$

Beweis (über die Definition der Palindrome):
Palindrome über $\Sigma = \{a, b, c\}$ sind wie folgt definiert:

1. Das leere Wort ε ist ein Palindrom.
2. Die einelementigen Worte a, b und c sind Palindrome.
3. Ist w ein Palindrom über $\Sigma = \{a, b, c\}$, so auch awa, bwb und cwc.

- zu 1.: Das leere Wort ε kann mit der Regel $S \to \varepsilon$ abgeleitet werden: $\varepsilon \in L(G)$.
- zu 2.: Die einelementigen Worte a, b und c können mittels der Regeln $S \to a$, $S \to b$ und $S \to c$ abgeleitet werden: $a, b, c \in L(G)$.
- zu 3.: Sei w ein Palindrom, das wie in 1. und 2. beschrieben aus dem Startsymbol S abgeleitet werden kann. Wendet man vor dieser Ableitung die Regeln $S \to aSa$, $S \to bSb$, oder $S \to cSc$ an, so ist letztlich auch awa, bwb bzw. cwc ein Palindrom.

Es gibt keine weiteren Ableitungsregeln, also können keine anderen Worte durch die Grammatik G beschrieben werden. Also gilt:

Alle Palindrome über $\Sigma = \{a, b, c\}$ x (und nur diese) können durch die Grammatik G erzeugt werden, d.h. $x \in L(G) \Leftrightarrow x$ ist Palindrom über $\Sigma = \{a, b, c\}$ □

Aufgabe 20

Teilaufgabe a

Das Wort $abab$ gehört zur Sprache $L(G)$, da es sich wie folgt aus dem Startsymbol S ableiten lässt:

$$S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$$

Teilaufgabe b

Voraussetzung:
$w \in L(G)$

Behauptung:
$awb, bwa \in L(G)$

Beweis:
Wenn w zur Sprache $L(G)$ gehört, bedeutet dies, dass sich w durch k Ableitungsschritte aus dem Startsymbol S ableiten lässt: $w \in L(G) \iff S \Rightarrow^k w$, $(k \in \mathbb{N})$

Da $S \rightarrow aSb \in P$ kann awb wie folgt aus dem Startsymbol abgeleitet werden:

$$S \Rightarrow aSb \Rightarrow^k awb$$

Außerdem gilt: $S \rightarrow bSa \in P$, sodass bwa folgendermaßen aus dem Startsymbol S abgeleitet werden kann:

$$S \Rightarrow bSa \Rightarrow^k bwa$$

□

Teilaufgabe c

Voraussetzung:
$w, w' \in L(G)$

Behauptung:
$ww' \in L(G)$

Beweis:
Wenn w und w' zur Sprache $L(G)$ gehören, bedeutet dies, dass sich w' und w' durch k bzw. l Ableitungsschritte aus dem Startsymbol S ableiten lassen: $w \in L(G) \iff S \Rightarrow^k w$ bzw. $w' \in L(G) \iff S \Rightarrow^l w$ $(k, l \in \mathbb{N})$

Da $S \to SS \in P$ kann ww' wie folgt aus dem Startsymbol abgeleitet werden:

$$S \Rightarrow SS \Rightarrow^k wS \Rightarrow^l ww'$$

□

Teilaufgabe d

Behauptung:
$x \in L(G) \Leftrightarrow x$ hat die gleiche Anzahl von a's und b's

Beweis:
Richtung „⇒“:

> Voraussetzung: $x \in L(G)$
> Behauptung: x hat die gleiche Anzahl von a's und b's
>
> **Beweis:**
> durch Induktion über die Wortlänge $n = |x|$:
>
> **Induktionsanfang ($0 \leq n \leq 2$):**
> Das Wort der Länge 0, nämlich das leere Wort ε kann mittels der Regel $S \to \varepsilon$ abgeleitet werden. Es hat per Definition die gleiche Anzahl von a's und b's.
>
> Aus der Grammatik G lassen sich keine Worte der Länge 1 (oder einer sonstigen ungeraden Länge) ableiten, da die einzigen Regeln, in denen auf der rechten Seite Terminale vorkommen ($S \to aSb$ und $S \to bSa$) die Satzform um zwei Zeichen (bzw. die Regel $S \to \varepsilon$ um kein Zeichen) verlängern.
>
> Die Worte ab und ba (durch die Ableitungen $S \Rightarrow aSb \Rightarrow ab$ bzw. $S \Rightarrow bSa \Rightarrow ba$) sind die einzigen Worte mit 2 Zeichen, die aus S ableitbar sind. Sie sind ebenfalls per Defintion Worte mit der gleichen Anzahl von a's und b's.
>
> **Induktionsvoraussetzung:**
> Für alle Wörter v mit $|v| \leq n$ und n gerade gilt: $v \in L(G)$ und v hat die gleiche Anzahl von a's und b's.
>
> **Induktionsschritt ($n \to n + 2$):**
> Sei v ein Wort, das der Induktionsvoraussetzung genügt, und somit aus der Sprache $L(G)$ ist und die gleiche Anzahl von a's und b's hat. Da $v \in L(G)$

ist, lässt es sich aus dem Startsymbol ableiten ($S \Rightarrow^* v$). S ist die einzige Variable in G und kommt in den Regeln $S \to aSb$, $S \to bSa$ und $S \to SS$ vor.

Analog zu Teilaufgabe b kann nun zunächst eine der ersten beiden Regeln $S \to aSb$ (bzw. $S \to bSa$) angewendet werden, bevor aus aSb (bzw. bSa) avb (bzw. bva) abgeleitet wird. Da v laut Induktionsvoraussetzung die gleiche Anzahl von a's und b's hat, hat auch avb (bzw. bva) die gleiche Anzahl von a's und b's.

Seien nun v, v' Worte, die der Induktionsvoraussetzung genügen, und somit aus der Sprache $L(G)$ sind und jeweils die gleiche Anzahl von a's und b's haben.

Analog zu Teilaufgabe c kann nun zunächst die Regeln $S \to SS$ angewendet werden, bevor mit $SS \Rightarrow vS \Rightarrow vv'$ bzw. $SS \Rightarrow v'S \Rightarrow v'v$ angewenden werden. Da laut Induktionsvoraussetzung v und v' die gleiche Anzahl von a's und b's haben, haben auch die Worte vv' und $v'v$ die gleiche Anzahl von a's und b's.
Damit sind alle Regeln aus P beschrieben worden.

Beweis:
Richtung „$\Leftarrow$“:

Voraussetzung: x hat die gleiche Anzahl von a's und b's
Behauptung: $x \in L(G)$

Beweis:
durch Induktion über die Wortlänge $n = |x|$:

Induktionsanfang ($0 \leq n \leq 2$):
Das Wort mit der Länge 0, nämlich das leere Wort ε ist ein Wort mit der gleichen Anzahl von a's und b's. Es kann mittels der Regel $S \to \varepsilon$ abgeleitet werden.

Ein Wort der Länge 1 (oder einer sonstigen ungeraden Länge) über dem Alphabet $\{a, b\}$ kann per Definition nicht die gleiche Anzahl von a's und b's haben.

Die einzigen Worte der Länge 2, die gleiche Anzahl von a's und b's haben, sind ab und ba. Sie können wie folgt abgeleitet werden:

$S \to aSb \to ab$ bzw. $S \to bSa \to ba$.

Induktionsvoraussetzung:
Für alle Wörter v mit $|v| \leq n$ und n gerade gilt: v hat die gleiche Anzahl von a's und b's und gehört zur Sprache $L(G)$.

Induktionsschritt $(n \to n+2)$**:**
Sei v ein Wort, das der Induktionsvoraussetzung genügt, und somit die gleiche Anzahl von a's und b's und $v \in L(G)$. Dann haben auch die Worte avb und bva die gleiche Anzahl von a's und b's.

In Teilaufgabe b wurde gezeigt, dass für jedes Wort $w \in \Sigma^*$ gilt: wenn $w \in L(G)$, dann sind auch $awb, bwa \in L(G)$. Da $v \in L(G)$ gilt auch $avb, bva \in L(G)$.

Seien des Weiteren v, v' Worte, die der Induktionsvoraussetzung genügen, und somit jeweils die gleiche Anzahl von a's und b's und $v \in L(G)$. Dann hat auch das Wort vv' die gleiche Anzahl von a's und b's.

In Teilaufgabe c wurde gezeigt, dass für jedes Wort $w, w' \in \Sigma^*$ gilt: wenn $w, w' \in L(G)$, dann ist auch $ww' \in L(G)$. Da $v, v' \in L(G)$ gilt auch $vv' \in L(G)$.

Also gilt: $x \in L(G) \Leftrightarrow x$ hat die gleiche Anzahl von a's und b's $\square$

Serie 6

Institut für Informatik
Algorithmen und Komplexität
Dr. Till Nierhoff

WS 2002
20. November 2002

Theoretische Informatik II

6. Serie

Abgabe bis zum 27. November 2002

Aufgabe 21 [9 Punkte]

Erstellen Sie eine Grammatik G über dem Alphabet $\Sigma = \{0, 1\}$ mit höchstens 2 Variablen, die genau die Wörter erzeugt, die der folgende LBA akzeptiert:
$M = (\{q, r, s, t\}, \Sigma', \Gamma, \delta, q, \{s\})$ mit $\Sigma' = \Sigma \cup \{\hat{0}, \hat{1}\}$, $\Gamma = \Sigma' \cup \{\square\}$
und der Überführungsfunktion:

$\delta(z,a)$	$a=0$	1	$\hat{0}$	$\hat{1}$
$z=q$	$(r,0,R)$	$(s,1,N)$	$(t,\hat{0},N)$	$(s,\hat{1},N)$
r	$(q,0,R)$	$(t,1,N)$	$(s,\hat{0},N)$	$(t,\hat{1},N)$
s				
t				

Beweisen Sie die Korrektheit Ihrer Konstruktion.

Aufgabe 22 [10 Punkte]

Nach Satz 3.1 der Vorlesung kann aus der Grammatik $G = (\{S\}, \{0, 1\}, P, S)$ mit den Regeln $P = \{S \to 0, 1, 0S0, 1S1\}$ ein LBA konstruiert werden, der genau die Wörter akzeptiert, die von G erzeugt werden.
Geben Sie für die Eingabe 0110110 eine akzeptierende Konfigurationsfolge dieser Maschine an. Erläutern Sie dabei ihre Funktionsweise.

Aufgabe 23 [2+2+2+3 Punkte]

Für zwei nichtleere Sprachen A und B sei die *markierte Vereinigung* $A \oplus B$ definiert durch

$$A \oplus B = \{0x \mid x \in A\} \cup \{1x \mid x \in B\}$$

Zeigen Sie:

a) $A \leq A \oplus B$ und $B \leq A \oplus B$.

b) $A \oplus B \in \mathcal{REC} \Leftrightarrow A \in \mathcal{REC} \wedge B \in \mathcal{REC}$

c) $A \oplus B \in \mathcal{RE} \Leftrightarrow A \in \mathcal{RE} \wedge B \in \mathcal{RE}$

d) Für alle Sprachen C gilt: $A \oplus B \leq C \Leftrightarrow A \leq C \wedge B \leq C$.

Aufgabe 24 [4+4+4 Punkte]

Definition: Einen Graphen, den man dadurch erhält, dass man zu einem Pfad von x nach y noch die Kante $\{y, x\}$ hinzunimmt, nennt man Kreis. Dabei muss der x-y-Pfad mindestens drei Knoten beinhalten.

Zeigen Sie, dass für jeden Baum T mit $n \geq 2$ Knoten gilt:

a) T enthält keinen Kreis,

b) T besitzt mindestens zwei Knoten vom Grad 1, *Hinweis*: a) verwenden

c) T besitzt genau $n - 1$ Kanten. *Hinweis*: Induktion

Aufgabe 21

Aus M kann folgende Grammatik erstellt werden:
$G = (\{S, A\}, \{0, 1\}, P, S)$ mit den Regeln $P = (S \to 1A, 00S, 00; A \to 0A, 1A, \varepsilon)$.

Beweis

- M startet im Zustand q und geht bei einer gelesenen 1 (bzw. $\hat{1}$, wenn das Bandende erreicht ist) in den einzigen akzeptierten Endzustand s über.
- Liest M hingegen eine 0 am Bandende (also $\hat{0}$) in Zustand q, geht M in den nicht akzeptierenden Zustand t über.
- Bei einer gelesenen 0, die nicht am Bandende steht, geht M von q nach r über. Wird nun in r wieder eine 0 gelesen, geht M wieder in q über.
- Bei einer 0 am Bandende ($\hat{0}$) geht M hingegen von r in den Endzustand s über.
- Liest M hingegen eine 1 (oder $\hat{1}$ am Bandende) in Zustand r, geht M in den nicht akzeptierenden Zustand t über.

Um vom Startzustand q in den akzeptierten Zustand s zu gelangen, bleiben also folgende Möglichkeiten:

1. M liest in q ein Wort, das mit 1 beginnt. Dabei ist es irrelevant, ob die Eins das einzige Zeichen der Eingabe ist, da M in den akzeptierten Endzustand s übergeht ohne die weiteren Zeichen zu lesen.
2. M liest eine gerade Anzahl von Nullen (und pendelt so zwischen q und r)
3. M liest eine gerade Anzahl von Nullen, gefolgt von einer Eins. Auch dabei ist es irrelevant, ob die Eins das letzte Zeichen der Eingabe ist, da M in den akzeptierten Endzustand s übergeht ohne die weiteren Zeichen zu lesen.

Da q der Startzustand von M ist, ist im folgenden S das Startsymbol der Grammatik G.

- Aus 1. folgen die Regeln $S \to 1A$ und $A \to \varepsilon$ (für die Eingabe $x = 1$), bzw. $A \to 1A, 0A$ (für alle anderen Eingaben, die mit 1 beginnen).
- Aus 2. folgt die Regel $S \to 00S$. Zusammen mit der Regel $S \to 00$ können so Worte erzeugt werden, die aus einer gerade Anzahl von Nullen bestehen.
- Aus 3. folgt die Regel $S \to 00S$ (aus 2.) (beliebig oft wiederholt), gefolgt von der Regel $S \to 1A$ (aus 1.) mit den anderen Regeln für die Variable A (aus 1.).

Aufgabe 22

Sei M ein LBA mit $M = (\{q, r, s, t, u, v, w, x, y, z\}, \Sigma', \Gamma, \delta, q, \{z\})$ mit $\Sigma = \{0, 1\}$, $\Sigma' = \Sigma \cup \{\hat{0}, \hat{1}, \bar{0}, \bar{1}\}$ und $\Gamma = \Sigma' \cup \{\square\}$, sowie der Überführungsfunktion δ wie folgt:

$\delta(z, a)$	$a = 0$	1	$\hat{0}$	$\hat{1}$	$\bar{0}$	$\bar{1}$
q	$(r, \bar{0}, R)$	$(s, \bar{1}, R)$	$(z, \hat{0}, N)$	$(z, \hat{1}, N)$		
r	$(t, 0, R)$	$(t, 1, R)$				
s	$(u, 0, R)$	$(u, 1, R)$				
t	$(r, 0, R)$	$(r, 1, R)$	$(v, \square, L)$			
u	$(s, 0, R)$	$(s, 1, R)$		$(w, \square, L)$		
v	$(x, \hat{0}, L)$	$(x, \hat{1}, L)$				
w	$(y, \hat{0}, L)$	$(y, \hat{1}, L)$				
x	$(x, 0, L)$	$(x, 1, L)$			$(q, \square, R)$	
y	$(y, 0, L)$	$(y, 1, L)$				$(q, \square, R)$
z						

Arbeitsweise

Vor der Rechnung hat die Eingabe auf dem Band des LBA die folgende Form:

$$x_1 x_2 \dots \hat{x_n} (x \in \Sigma'^*)$$

- Verhalten des LBA falls die Eingabe w aus G abgeleitet werden kann:

  Falls die Eingabe im Zustand q nur ein Zeichen enthält (Zeichen hat einen Hut), akzeptiert der Automat die Eingabe. ($S \to 0$, $S \to 1$)

  Falls die Eingabe länger ist, wird das erste Zeichen mit dem Überstrich markiert (q). Danach bewegt sich der Kopf bis zum letzten Zeichen der Eingabe, dass mit einem Hut markiert ist. (Zustände r, t, wenn erstes Zeichen 0 und s, u für 1) Es werden zwei Zustände benötigt, um zu testen, ob das Eingabewort eine ungerade Anzahl von Zeichen hat (wie in der Grammatik gefordert). Nun wird das letzte Zeichen gelöscht, bzw. mit einem Blank überschrieben.

  Anschließend wird das ehemals vorletzte Zeichen mit dem Hut markiert, da es nach dem Löschen das letzte Zeichen ist, und der Kopf bewegt sich zum ersten Zeichen der Eingabe zurück und löscht dieses und der Kopf bewegt sich ein Zeichen nach rechts (Zustände t, v für 0 und u, w für 1). Danach zurück zum Zustand q.

  Dieses Prinzip setzt die Regeln $S \to 0S0$,$S \to 1S1$ um, da jeweils die beiden äußersten Nullen oder Einsen gelöscht werden, und danach das innere S mit den gleichen Regeln bearbeitet wird, bis das S nur noch aus einem Element besteht. Dann geht die Maschine in den akzeptierenden Zustand z.

- Verhalten des LBA falls die Eingabe w nicht aus G abgeleitet werden kann:

  Es gibt mehrere Möglichkeiten der falschen Eingabe:

– Das Eingabewort hat eine gerade Anzahl von Zeichen:
  Das leere Wort ε wird im Startzustand q nicht akzeptiert. Eine Eingabe mit einer geraden Anzahl von Zeichen ($n = 2, 4, 6 \ldots$) kann nicht akzeptiert werden, da r und s für das jeweils letzte Zeichen (ein Zeichen mit Hut) des Wortes nicht definiert sind.
– Das Wort endet nicht mit den selben Zeichen:
  Das einelementige Wort mit $w = \hat{0}$ oder $w = \hat{1}$ wird im Startzustand q akzeptiert. Falls nun das Wort nicht mit den selben Zeichen endet, wird das Wort nicht von der Maschine akzeptiert, da es zwei Schleifen für das Durchlaufen des Wortes gibt. Die eine erwartet eine $\hat{0}$ am Ende des Wortes, falls das erste Zeichen eine 0 war, die andere Schleife funktioniert analog wie die der Eins. Falls die Eingabe nun nicht dieser Form entspricht, wird die Maschine keinen akzeptierenden Zustand erreichen.

Also akzeptiert der LBA Worte, die mit Hilfe der Grammatik G gebildet werden können und nichts anderes.

Konfigurationsfolge für die Eingabe w=0110110
$q0110110\hat{0} \vdash \bar{0}r11011\hat{0} \vdash \bar{0}1t1011\hat{0} \vdash \bar{0}11r011\hat{0} \vdash \bar{0}110t11\hat{0} \vdash \bar{0}1101r1\hat{0} \vdash \bar{0}11011t\hat{0} \vdash \bar{0}1101v1\square \vdash \bar{0}110x1\hat{1}\square \vdash \bar{0}11x01\hat{1}\square \vdash \bar{0}1x101\hat{1}\square \vdash \bar{0}x1101\hat{1}\square \vdash x\bar{0}1101\hat{1}\square \vdash \square q1101\hat{1}\square \vdash \square\bar{1}s101\hat{1}\square \vdash \square\bar{1}1u01\hat{1}\square \vdash \square\bar{1}10s1\hat{1}\square \vdash \square\bar{1}101u\hat{1}\square \vdash \square\bar{1}10w1\square\square \vdash \square\bar{1}1y0\hat{1}\square\square \vdash \square\bar{1}y10\hat{1}\square\square \vdash \square y\bar{1}10\hat{1}\square\square \vdash \square\square q10\hat{1}\square\square \vdash \square\square\bar{1}s0\hat{1}\square\square \vdash \square\square\bar{1}0u\hat{1}\square\square \vdash \square\square\bar{1}w0\square\square\square \vdash \square\square y\bar{1}\hat{0}\square\square\square \vdash \square\square\square q\hat{0}\square\square\square \vdash \square\square\square z\hat{0}\square\square\square$

Nun ist die Maschine in einen akzeptierten Endzustand übergegangen.

Aufgabe 23

Teilaufgabe a

Zu zeigen: $A \leq A \oplus B$ bzw. $B \leq A \oplus B$

$A \leq A \oplus B \Leftrightarrow \exists f_1 : A \to A \oplus B : x \in A \Leftrightarrow f_1(x) \in A \oplus B$ mit f_1 berechenbar, bzw.
$B \leq A \oplus B \Leftrightarrow \exists f_2 : B \to A \oplus B : x \in B \Leftrightarrow f_2(x) \in A \oplus B$ mit f_2 berechenbar.

Sei $f_1(x) = \begin{cases} 0x, & \text{wenn } x \in A \\ \uparrow, & \text{sonst} \end{cases}$ und $f_2(x) = \begin{cases} 1x, & \text{wenn } x \in B \\ \uparrow, & \text{sonst} \end{cases}$

Die Reduktionen f_1 und f_2 sind berechenbar, da Turingmaschinen konstruiert werden kann, die f_1 bzw. f_2 wie folgt berechnet:

Sei T_1 eine 2-DTM, die f_1 wie folgt berechnet: T_2 kopiert zunächst die Eingabe x auf das zweite Band und berechnet anschließend die semi-charakteristische Funktion $\hat{\chi}_A(x)$ von A und – falls $\hat{\chi}_A \neq \uparrow$ – löscht anschließend das 1. Band und überschreibt es mit einer

0, gefolgt von der Eingabe x, die vom 2. Band kopiert wird. Falls $x \notin A$ kommt T_1 nie zum Stopp.

Analog wird die Turingmaschine konstruiert, die f_2 berechnet (und eine 1 vor die Eingabe x schreibt.).

Teilaufgabe b

Zu zeigen: $A \oplus B \in \mathcal{REC} \Leftrightarrow A \in \mathcal{REC} \wedge B \in \mathcal{REC}$

Beweis (Richtung „⇒“):
Sei $A \oplus B \in \mathcal{REC}$. Nach Teilaufgabe a) gilt: $A \leq A \oplus B$ und $B \leq A \oplus B$.

Weiterhin gilt nach Satz 4.10 (Köbler):
$A \leq A \oplus B \wedge A \oplus B \in \mathcal{REC} \Rightarrow A \in \mathcal{REC}$ und $B \leq A \oplus B \wedge A \oplus B \in \mathcal{REC} \Rightarrow B \in \mathcal{REC}$

Beweis (Richtung „⇐“):
Sei $A \in \mathcal{REC}$ und $B \in \mathcal{REC}$.

Es gilt: $A \oplus B \leq A \cup B \Leftrightarrow \exists f : A \oplus B \to A \cup B : x \in A \oplus B \Leftrightarrow f(x) \in A \cup B$ mit f berechenbar.
Sei $f(x = x_1 x_2 \dots x_n) = \begin{cases} x_2 \dots x_n, & \text{wenn } x \in A \oplus B \\ c \notin A \cup B, & \text{sonst} \end{cases}$.
$f(x)$ ist durch eine Turingmaschine berechenbar, die zunächst mittels der charakteristischen Funktion $\chi_{A \oplus B}$ überprüft, ob die Eingabe $x \in A \oplus B$, und wenn dies der Fall ist, das erste Zeichen der Eingabe (welches die Markierung darstellt) löscht und das so entstandene Wort ausgibt.

Des Weiteren kann leicht gezeigt werden, dass gilt: $A \leq A \cup B$ und $B \leq A \cup B$, da als Reduktion die Identität angegeben werden kann, da gilt: $x \in A \Leftrightarrow Id(x) \in A \cup B$ bzw. $x \in B \Leftrightarrow Id(x) \in A \cup B$. Die Identität ist berechenbar.

Nach Teilaufgabe d) gilt: $A \oplus B \leq C \Leftrightarrow A \leq C \wedge B \leq C$. Da $A \leq A \cup B \wedge B \leq A \cup B \Leftrightarrow A \oplus B \leq A \cup B$.

Da laut Voraussetzung $A \in \mathcal{REC}$ und $B \in \mathcal{REC}$, ist auch die Vereinigung $A \cup B \in \mathcal{REC}$, da die Turingmaschine, die entscheiden soll, ob $x \in A \cup B$ zunächst überprüfen kann, ob $x \in A$ und dann ob $x \in B$.

Laut Satz 4.10 (Köbler) gilt nun: $A \oplus B \leq A \cup B \wedge A \cup B \in \mathcal{REC} \Rightarrow A \oplus B \in \mathcal{REC}$. □

Teilaufgabe c

Zu zeigen: $A \oplus B \in \mathcal{RE} \Leftrightarrow A \in \mathcal{RE} \wedge B \in \mathcal{RE}$

Beweis (Richtung „⇒“):
Der Beweis wird analog zu Teilaufgabe b) geführt:

Sei $A \oplus B \in \mathcal{RE}$. Nach Teilaufgabe a) gilt: $A \leq A \oplus B$ und $B \leq A \oplus B$.

Weiterhin gilt nach Satz 4.10 (Köbler):
$A \leq A \oplus B \wedge A \oplus B \in \mathcal{RE} \Rightarrow A \in \mathcal{RE}$ und $B \leq A \oplus B \wedge A \oplus B \in \mathcal{RE} \Rightarrow B \in \mathcal{RE}$

Beweis (Richtung „⇐“):
Der Beweis wird analog zu Teilaufgabe b) geführt:

Sei $A \in \mathcal{RE}$ und $B \in \mathcal{RE}$.

Es gilt: $A \oplus B \leq A \cup B \Leftrightarrow \exists f : A \oplus B \to A \cup B : x \in A \oplus B \Leftrightarrow f(x) \in A \cup B$ mit f berechenbar.
Sei $f(x = x_1 x_2 \dots x_n) = \begin{cases} x_2 \dots x_n, & \text{wenn } x \in A \oplus B \\ \uparrow, & \text{sonst} \end{cases}$.
$f(x)$ ist durch eine Turingmaschine berechenbar, die zunächst mittels der semi-charakteristischen Funktion $\hat{\chi}_{A \oplus B}$ überprüft, ob die Eingabe $x \in A \oplus B$, und wenn dies der Fall ist, das erste Zeichen der Eingabe (welches die Markierung darstellt) löscht und das so entstandene Wort ausgibt. Wenn $x \notin A \oplus B$, so stoppt die Turingmaschine nie.

Des Weiteren kann leicht gezeigt werden, dass gilt: $A \leq A \cup B$ und $B \leq A \cup B$, da als Reduktion die Identität angegeben werden kann, da gilt: $x \in A \Leftrightarrow Id(x) \in A \cup B$ bzw. $x \in B \Leftrightarrow Id(x) \in A \cup B$. Die Identität ist berechenbar.

Nach Teilaufgabe d) gilt: $A \oplus B \leq C \Leftrightarrow A \leq C \wedge B \leq C$. Da $A \leq A \cup B \wedge B \leq A \cup B \Leftrightarrow A \oplus B \leq A \cup B$.

Da laut Voraussetzung $A \in \mathcal{RE}$ und $B \in \mathcal{RE}$, ist auch die Vereinigung $A \cup B \in \mathcal{RE}$, da die Turingmaschine, die entscheiden soll, ob $x \in A \cup B$ parallel überprüfen kann, ob $x \in A$ und ob $x \in B$ und stoppt, sobald einer der Fälle erfolgreich ist, bzw. sonst nie stoppt.

Laut Satz 4.10 (Köbler) gilt nun: $A \oplus B \leq A \cup B \wedge A \cup B \in \mathcal{RE} \Rightarrow A \oplus B \in \mathcal{RE}$. □

Teilaufgabe d

Zu zeigen: Für alle Sprachen C gilt: $A \oplus B \leq C \Leftrightarrow A \leq C \wedge B \leq C$

Beweis (Richtung „⇒“):
Sei $A \oplus B \leq C$, d.h. $\exists f : A \oplus B \to C : x \in A \oplus B \Leftrightarrow f(x) \in C$ mit f berechenbar.

Nach Definition der markierten Vereinigung gilt: $\forall x \in A : \exists 0x \in A \oplus B$. Jedes Element aus A ist also „mit Markierung“ in $A \oplus B$ enhalten. Außerdem ist laut Voraussetzung

$A \oplus B$ auf C reduzierbar.

Also kann A auf C wie folgt reduziert werden:

$A \leq C \Leftrightarrow \exists f_1 : A \to C : x \in A \Leftrightarrow f_1(x) \in C$, wobei f_1 wie folgt definiert sei:

$$f_1(x) = \begin{cases} f(x), & \text{falls } 0x \in A \oplus B \\ y \notin C, & \text{sonst} \end{cases}$$

Analog kann mit Hilfe der Reduzierung $f : A \oplus B \to C$ eine Reduzierung von B auf C angegeben werden:

$$f_2(x) = \begin{cases} f(x), & \text{falls } 1x \in A \oplus B \\ y \notin C, & \text{sonst} \end{cases}$$

Da f laut Voraussetzung berechenbar ist, sind auch f_1 und f_2 berechenbar.

Beweis (Richtung „⇐“):
Sei $A \leq C$ und $B \leq C$, d.h.
$\exists f_1 : A \to C : x \in A \Leftrightarrow f_1(x) \in C$ und $\exists f_2 : B \to C : x \in B \Leftrightarrow f_2(x) \in C$. f_1, f_2 berechenbar.

$A \oplus B$ lässt sich mit Hilfe dieser Reduktionen auf C reduzieren, da $A \oplus B$ alle Elemente aus A und B „mit Markierung“ enthält:

$$f_3 : A \oplus B \to C : f_3(x = x_1 x_2 \dots x_n) = \begin{cases} f_1(x_2 \dots x_n), & \text{falls } x_1 = 0 \wedge x_2 \dots x_n \in A \\ f_2(x_2 \dots x_n), & \text{falls } x_1 = 1 \wedge x_2 \dots x_n \in B \\ y \notin C, & \text{sonst} \end{cases}$$

Dabei wird für ein Wort aus $A \oplus B$ entschieden, ob es ursprünglich aus A oder B ist (anhand der Markierung, dem 1. Zeichen) und dementsprechend entweder f_1 oder f_2 berechnet, die nach C abbilden.

Da f_1 und f_2 laut Voraussetzung berechenbar sind, ist auch f_3 berechenbar.

Aufgabe 24

Sei T Baum mit $n \geq 2$ Knoten.

Teilaufgabe a

Behauptung:
T enthält keinen Kreis.

Beweis:
Laut Definition des Baumes sind je zwei Knoten x und y genau durch einen Pfad verbunden. Sind x und y benachbart, so enthält dieser Pfad genau die Kante $\{x, y\}$. Es kann sich bei diesem Pfad nicht um einen Kreis handeln, da er nur zwei Knoten enthält und dies ein Widerspruch zur Definition ist.
Seinen x und y also nicht benachbart. Da es sich bei T um einen Baum handelt, sind also x und y genau durch einen Pfad verbunden. Wenn dieser Pfad ein Kreis wäre, so würde er auch die Kante $\{y, x\}$ enthalten. Dies steht allerdings im Widerspruch zu unserer Forderung, dass x und y nicht benachbart sind. □

Teilaufgabe b und Teilaufgabe c

Behauptung:
T besitzt mindestens zwei Knoten vom Grad 1. T besitzt genau $n - 1$ Kanten.

Beweis (durch vollständige Induktion über die Knotenanzahl n):

Induktionsanfang (n=2):
Wenn der Baum T_2 genau zwei Knoten a und b besitzt, so sind diese durch die Kante $\{a, b\}$ verbunden und haben jeweils einen Nachbar, also den Grad 1. Des Weiteren stimmt die Annahme, dass T_2 bei $n = 2$ Knoten $n - 1 = 1$ Kante (nämlich $\{a, b\}$) enthält.

Induktionsvoraussetzung:
Für alle Bäume T_n mit n Knoten gilt: T_n besitzt mindestens zwei Knoten vom Grad 1 und hat genau $n - 1$ Kanten.

Induktionsschritt ($n \to n + 1$):
Sei T ein Baum, der der Induktionsvoraussetzung genügt, also mindestens zwei Knoten vom Grad 1 und genau $n - 1$ Kanten hat.
Wenn nun der Knoten x zu T hinzugefügt werden soll, so muss x durch genau eine Kante mit einem beliebigen Knoten y von T verbunden werden, da sonst die Baum-Charakteristik verletzt würde. Da x nur den Nachbar y hat, ist x vom Grad 1. Da T bereits mindestens zwei Knoten vom Grad 1 hatte, hat T nach dem Hinzufügen von x entweder einen Knoten mehr, der vom Grad 1 ist (falls $|\Gamma(y)| \geq 1$ war) oder die gleiche Anzahl (falls $|\Gamma(y) = 1|$ war und nun $|\Gamma(y)| = 2$, aber $|\Gamma(x) = 1|$).
Des Weiteren hat T nach dem Hinzufügen eine Kante mehr als vor dem Hinzufügen, also genau n Kanten bei $n + 1$ Knoten. □

Serie 7

Institut für Informatik
Algorithmen und Komplexität
Dr. Till Nierhoff

WS 2002
27. November 2002

Theoretische Informatik II

7. Serie

Abgabe bis zum 4. Dezember 2002

Allgemeine Bemerkung: Für die Konstruktion eines endlichen Automaten genügt die Angabe eines Diagramms. Behauptungen über die erkannte Sprache müssen bewiesen werden.

Aufgabe 25 [10 Punkte]

Konstruieren Sie zu einer gegebenen Turingmaschine $M = (Z, \Sigma, \Gamma, \delta, q, E)$ eine Grammatik G mit $L(G) = L(M)$. Erläutern Sie die Funktionen der Produktionsregeln. Zeigen Sie, dass die Grammatik im allgemeinen nicht vom Typ 1 ist.
Hinweis: vgl. den Beweis von Satz 3.1 der Vorlesung.

Aufgabe 26 [5+5 Punkte]

Konstruieren Sie

a) einen DFA, der diejenigen Binärzahlen erkennt, die bei Division durch 3 den Rest 1 ergeben und aus einer geraden Anzahl von Zeichen bestehen.

b) einen NFA, der die Sprache
$L := \{u01^{2n}0v \mid u, v \in \{0,1\}^*, n \in \mathbb{N}, n \geq 1\}$ erkennt.

Aufgabe 27 [5+5 Punkte]

Konstruieren Sie

a) einen NFA mit maximal vier Zuständen, der die Sprache
$L := \{uzzv \mid u, v \in \{0,1\}^*, z \in \{0,1\}\}$ erkennt.

b) gemäß der Vorlesung einen DFA, der den in a) konstruierten NFA simuliert.

Aufgabe 28 [4+4+2 Punkte]

Seien $M_1 = (Z_1, \Sigma, \delta_1, q_1, E_1)$ und $M_2 = (Z_2, \Sigma, \delta_2, q_2, E_2)$ zwei DFAs.
Konstruieren Sie

a) einen DFA $\overline{M_1}$ mit $L\left(\overline{M_1}\right) = \overline{L\left(M_1\right)}$,

b) einen DFA M_{12} mit $L\left(M_{12}\right) = L\left(M_1\right) \cap L\left(M_2\right)$.

Damit haben Sie gezeigt, dass die Menge der regulären Sprachen abgeschlossen ist unter Schnittmengenbildung und Komplementbildung.

c) Seien A und B zwei reguläre Sprachen. Ist im allgemeinen auch $A \setminus B$ regulär? Begründen Sie Ihre Antwort.

Aufgabe 25

Sei $M = (Z, \Sigma, \Gamma, \delta, q, E)$ eine Turingmaschine. Daraus soll ähnlich zu Satz 3.1 aus der Vorlesung eine Grammatik G gebildet werden mit $G = (V, \Sigma, P, S)$ und $L(G) = L(M)$. Dabei sei $V = \{S, (c, a), ((q, c), a) \mid a, b \in \Sigma,\ c, d \in \Gamma, q \in Z\}$ und P mit folgenden Regeln:

1. **Startregeln**
   Das Nichtterminal A aus Satz 3.1 kann weggelassen werden, da bei Turingmaschinen auch über die Eingabe hinweggelesen werden kann, und so Regeln für Zeichen $\hat{a}$ ($a \in \Sigma$) nicht benötigt werden. Stattdessen:

   $S \to S(a, a), S(\square, \varepsilon), ((q_0, a), a)$ (für $a \in \Sigma$ und den Startzustand $q_0 \in Z$)

   Außerdem $S \to \varepsilon$, falls $\varepsilon \in L(M)$.

   Die Regel $S \to S(\square, \varepsilon)$ ist notwendig, damit der durch die Eingabe benutzte Bereich verlassen werden kann, was bei einer Turingmaschine im Vergleich zu einem LBA erlaubt ist.

2. **Überführungsregeln**
   Entsprechen den Regeln des LBA aus Satz 3.1. Allerdings kann die TM M bei verkürzenden Produktionen den durch die Eingabe benutzten Bereich verlassen.

   - keine Kopfbewegung:
     $((q, c), a) \to ((q', c'), a)$ $(a \in \Sigma,\ ((q', c', N) \in \delta(q, c))$
   - Kopfbewegung nach rechts:
     $((q, c), a)(b, d) \to (c', a)((q', b), d)$ $(a, d \in \Sigma,\ (q', c', R) \in \delta(q, c))$
   - Kopfbewegung nach links:
     $(d, b)((q, c), a) \to ((q', d), b)(c', a)$ $(a, d \in \Sigma,\ (q', c', L) \in \delta(q, c))$

3. **Abschlussregeln**
   Auch die Abschlussregeln entsprechen denen des Satzes 3.1:

   $((q, c), a) \to a$
   $(c, a) \to a$ $(a \in \Sigma, q \in E)$

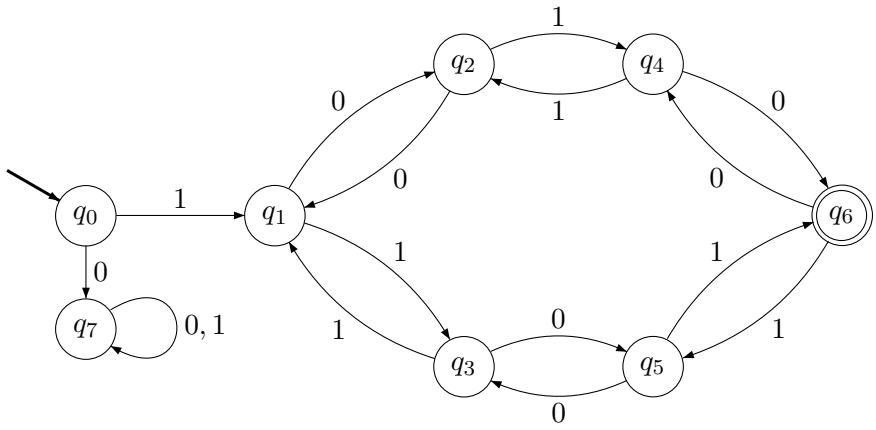
Wie schon bei den Überführungsregeln angesprochen, kann der durch die Eingabe benutzte Bereich verlassen werden, falls es verkürzende Regeln $\alpha \to \beta$ mit $|\alpha| > |\beta|$ gibt.

Dadurch ist die Grammatik G mit $L(G) = L(M)$ nicht notwendigerweise vom Typ 0.

Aufgabe 26

Teilaufgabe a

DFA $M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}, \{0, 1\}, \delta, q_0, \{q_6\})$:



Der Automat M akzeptiert nur Binärzahlen mit einer geraden Anzahl von Zeichen, da vom Startzustand q_0 zum Zustand q_1 ein Übergang, und von q_1 zum Endzustand q_6 eine ungerade Anzahl von Übergängen benötigt wird. Eingaben mit führenden Nullen werden verworfen (Übergang in q_7).

Da es keine zweistelligen Binärzahlen gibt, die bei der Division durch 3 den Rest 1 ergeben, betrachten wird zunächst vierstellige Zahlen. Beim Lesen einer solchen Binärzahl gibt es zunächst zwei Möglichkeiten zum Endzustand zu gelangen, nämlich die Zahlen 1010 und 1101, die den Dezimalzahlen 10 und 13 entsprechen.

Betrachten wir zunächst den Konfigurationsübergang für die Zahl 1010: Beim Lesen der ersten beiden Ziffern wird vom Startzustand q_0 über den Zustand q_1 in den Zustand q_2 übergegangen. Dies entspricht der Dezimalzahl 2 mit $2 \equiv 2 \mod 3$. Anschließend wird vom Zustand q_2 über q_4 in den Endzustand q_6 übergegangen. Betrachtet man diese beiden Ziffern seperat, ergibt sich wieder die Dezimalzahl 2 mit $2 \equiv 2 \mod 3$. Addiert man nun diese Reste bei der Division durch 3, ergibt sich $4 \equiv 1 \mod 3$.

Analog der Konfigurationsübergang für die Zahl 1101: die ersten beiden Ziffern 11 (Übergang von q_0 über q_1 nach q_3) entspechen der Dezimalzahl 3 mit $0 \equiv 3 \mod 3$. Die weiteren Ziffern 01 (Übergang von q_3 nach q_6 über q_5) entsprechen der Dezimalzahl 1 mit $1 \equiv 1 \mod 3$. Addiert man wieder diese Reste der Division durch 3, ergibt sich $1 \equiv 1 \mod 3$.

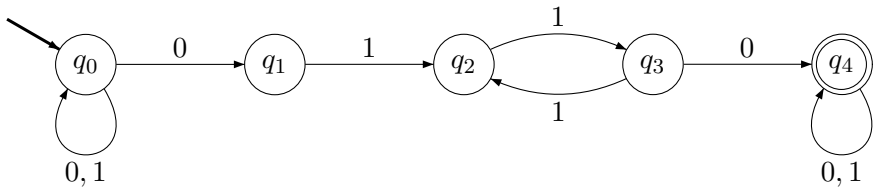
Bei der Betrachtung der Funktionsübergänge für längere Binärzahlen (mit der geraden Ziffernzahl größer als 4) fällt auf, dass zusätzlich zu dem oben beschriebenen Übergang

Schleifen zwischen zwei Zuständen durchlaufen werden, die eine gerade Anzahl von Nullen oder Einsen an die bis dorthin konstruierte Zahl anhängen.

Das Anhängen von zwei Nullen an eine Binärzahl entspricht im Dezimalsystem der Multiplikation mit $2^2 = 4$. Diese Multiplikation ändert nichts am Rest, der sich bei der Division durch 3 ergibt. Analog entspricht das Anhängen zweier Einsen eine Multiplikation mit 4, gefolgt von einer Addition mit 3, die ebenfalls nichts am Rest, der sich bei der Division durch 3 ergibt, ändert.

Teilaufgabe b

NFA $N = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, q_0, \{q_4\})$:



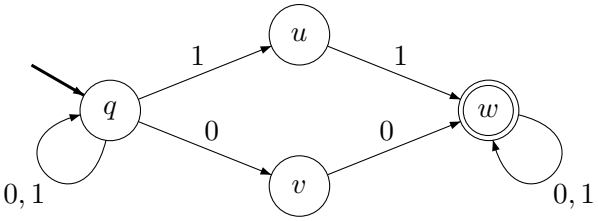
N akzeptiert die Sprache $L = \{u01^{2n}0v \mid u, v \in \{0,1\}^*, n \in \mathbb{N}, n \geq 1\}$:

Im Startzustand q_0 wird das Teilwort u, also eine beliebige Folge von Nullen und Einsen erkannt. Anschließend muss eine 0 gelesen werden (Übergang in q_1). Es muss nun eine 1 gelesen werden (Übergang in q_2) und eine ungeraden Anzahl von Einsen (Pendeln zwischen q_3 und q_2). Dies beschreibt das Teilwort 1^{2n} ($n \in \mathbb{N}$). Nun muss eine 0 gelesen werden, um in den Endzustand q_4 zu gelangen. Hier kann nun das letzte Teilwort v (wieder eine beliebige Folge von Nullen und Einsen) erkannt werden.

Aufgabe 27

Teilaufgabe a

NFA $N = (\{q, u, v, w\}, \{0, 1\}, q, \{w\})$:



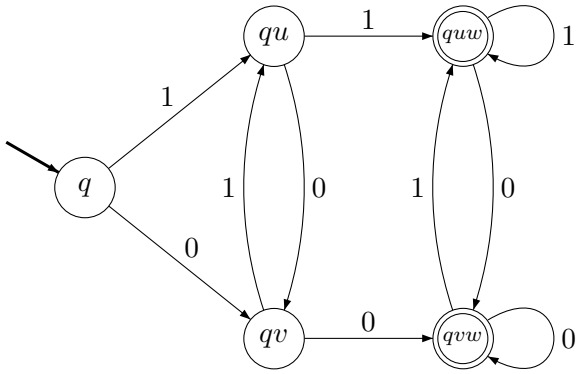
N akzeptiert die Sprache $L = \{uzzv \mid u, v \in \{0,1\}^*, z \in \{0,1\}\}$:

Im Zustand q wird das Teilwort u (also eine beliebige Folge von Einsen und Nullen) erkannt. Anschließend wird abhängig vom gelesenen Zeichen in den Zustand u (bzw. v) übergegangen. Dort muss das gleiche Zeichen noch einmal gelesen werden, um in den Endzustand w überzugehen. Anschließend wird wieder eine beliebige Folge von Einsen und Nullen erkannt.

Teilaufgabe b

Analog zur Vorlesung wird nun der NFA N aus Teilaufgabe a in einen DFA umgewandelt. Dazu wird vom Startzustand q aus eine Tiefensuche durchgeführt und jeweils zunächst der Fall für eine gelesen 0, und dann für eine gelesene 1 analysiert.

Nach Anwendung des Alogrithmus ergibt sich der DFA M mit $L(M) = L(N) = L$ (nach Satz 4.1):



Aufgabe 28

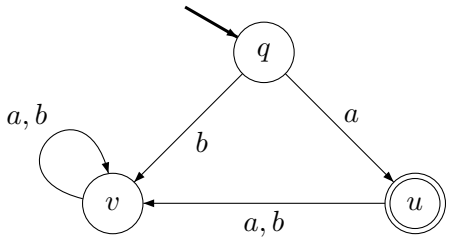
Teilaufgabe a

Der DFA $\overline{M_1}$ mit $L(\overline{M_1}) = \overline{L(M_1)}$ sei wie folgt definiert: $\overline{M_1} = (Z_1, \Sigma, \delta_1, q_1, Z_1 \backslash E_1)$, wobei $\overline{M_1}$ genau dann ein Wort w akzeptiert, wenn $\overline{M_1}$ in einem Endzustand stehenbleibt. Genau dann aber bleibt M_1 in einem Nicht-Endzustand stehen und akzeptiert w nicht.

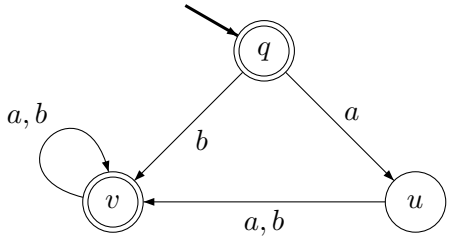
Jeder Zustand, der bei M_1 ein Endzustand war, ist bei $\overline{M_1}$ kein Endzustand, und umgekehrt ist jeder Nicht-Endzustand von M_1 ein Endzustand bei $\overline{M_1}$.

Für jedes Wort w, das von $\overline{M_1}$ akzeptiert wird, gilt also $w \notin L(M_1)$ und für jedes Wort, das von $\overline{M_1}$ nicht akzeptiert wird, gilt $w \in L(M_1)$. Also gilt: $L(\overline{M_1}) = \overline{L(M_1)}$.

Beispiel
Sei $M_1 = (\{q, u, v\}, \{a, b\}, \delta, q, \{u\})$. $L(M_1) = \{a\}$



Daraus folgt $\overline{M_1} = (\{q, u, v\}, \{a, b\}, \delta, q, \{q, v\})$. $L(\overline{M_1}) = \{a, b\}^* \backslash \{a\}$



Teilaufgabe b

Um zu zeigen, dass für zwei DFAs M_1 und M_2 ein DFA M_{12} existiert mit $L(M_{12}) = L(M_1) \cap L(M_2)$, werden die De Morgan'schen Formeln benutzt:

$$A \cup B = \overline{\overline{A} \cap \overline{B}} \text{ und } A \cap B = \overline{\overline{A} \cup \overline{B}}$$

Da in Teilaufgabe a) gezeigt wurde, dass die regulären Sprachen unter Komplementbildung abgeschlossen sind, genügt zu zeigen, dass die regulären Sprachen außerdem unter der Vereinigung abgeschlossen sind:

$$\exists \text{ DFA } M'_{12} : L(M'_{12}) = L(M_1) \cup L(M_2) \in \mathcal{REG}$$

Dazu wird aus den DFAs M_1 und M_2 zunächst ein NFA $N = (Z, \Sigma, \delta, Q_0, E)$ konstruiert:

- $Z = Z_1 \dot{\cup} Z_2$ die vereinigten Zustandsmengen von M_1 und M_2
- Σ das Alphabet von M_1 und M_2
- $\delta(q, a) = \begin{cases} \delta_1(q, a), & \text{falls } q \in Z_1 \\ \delta_2(q, a), & \text{falls } q \in Z_2 \end{cases}$
- $Q_0 = \{q_1, q_2\}$ die Startzustände von M_1 und M_2

- $E_1 \dot{\cup} E_2$ die Endzustände von M_1 und M_2

Der NFA N akzeptiert nun alle Worte, die M_1 oder M_2 akzeptieren. Da verlangt ist, dass die Zustandsmengen disjunkt sind, finden keine Übergänge von Zuständen aus Z_1 zu Zuständen aus Z_2 (und umgekehrt) statt.

Nach Satz 4.1 aus der Vorlesung lässt sich aus dem NFA N ein DFA M'_{12} konstruieren, für den gilt: $L(M'_{12}) = L(N) = L(M_1) \cup L(M_2)$.

Damit wurde gezeigt, dass die Menge der regulären Sprachen unter Vereinigung und Komplementbildung abgeschlossen ist.

Da M_1 und M_2 DFAs sind, gilt: $L(M_1), L(M_2) \in \mathcal{REG}$, und insbesondere:

$$L(M_1) \cap L(M_2) = \overline{\overline{L(M_1)} \cup \overline{L(M_2)}} \in \mathcal{REG}$$

Nach Satz 4.1 aus der Vorlesung gibt es zu dieser regulären Sprache einen DFA M_{12} mit $L(M_{12}) = L(M_1) \cap L(M_2)$. □

Teilaufgabe c

Voraussetzung: $A, B \in \mathcal{REG}$
Behauptung: $A \backslash B \in \mathcal{REG}$

Beweis:
$A \backslash B = A \backslash (A \cup B)$. In Teilaufgabe a) und b) wurde gezeigt, dass die Menge der regulären Sprachen unter Schnittmengenbildung, Vereinigung und Komplementbildung abgeschlossen ist.

Daher ist insbesondere $A \backslash (A \cup B) \in \mathcal{REG}$ für $A, B \in \mathcal{REG}$. Für den Sonderfall $A = B$ gilt: $A \backslash B = \emptyset \overset{\text{Def}}{\in} \mathcal{REG}$. □

Serie 8

Institut für Informatik
Algorithmen und Komplexität
Dr. Till Nierhoff

WS 2002
4. Dezember 2002

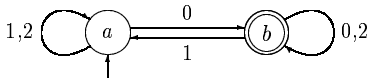
Theoretische Informatik II

8. Serie

Abgabe bis zum 11. Dezember 2002

Aufgabe 29 [8 Punkte]

Konstruieren Sie zu dem folgenden DFA M einen regulären Ausdruck γ mit $L(\gamma) = L(M)$. Verwenden Sie dafür das Verfahren aus der Vorlesung.



Aufgabe 30 [4+4+4 Punkte]

Entscheiden Sie, ob die folgenden Sprachen regulär sind. Begründen Sie Ihr Urteil.

a) $\{a^{(n^3)} \mid n \in \mathbb{N}\}$

b) $\{x \in \{(,)\}^* \mid x \text{ stellt eine korrekte Klammerung dar}, |x| > 0\}$

c) $\{a^{(n \bmod 2)}b^n \mid n \in \mathbb{N}\}$

Aufgabe 31 [3+3+4 Punkte]

Beweisen Sie:

a) $L = \{ab^jc^j \mid j \in \mathbb{N}\} \notin REG$

b) $A = \{a^ib^jc^k \mid i = 0 \vee j = k\}$ besitzt die Pumping-Zahl 1. Als Pumpingzahl bezeichnet man die Zahl l des Pumping-Lemmas (Satz 4.3 der Vorlesung).

c) A ist dennoch nicht regulär. *Hinweis*: Aufgabe 28 verwenden.

Damit haben Sie gezeigt, dass die Umkehrung des Pumping-Lemmas nicht gilt.

Aufgabe 32 [4+6 Punkte]

Sei $k : \mathbb{N} \to \mathbb{N}$ eine Funktion.

a) Zeigen Sie: Wenn $L = \{a^{k(n)}b^n \mid n \in \mathbb{N}\}$ eine reguläre Sprache ist, ist k beschränkt, d.h. $\exists k_0 \in \mathbb{N}\ \forall n \in \mathbb{N} : k(n) \leq k_0$.

b) Gilt auch die Umkehrung der Aussage? Begründen Sie Ihr Urteil.

Aufgabe 29

Gesucht wird für den DFA M ein regulärer Ausdruck γ mit $L(\gamma) = L(M)$.

Nummeriert man die Zustände a und b so durch, dass $a = z_1$ und $b = z_2$ gilt, so wird der reguläre Ausdruck $L^2_{1,2} = \gamma^2_{1,2}$ gesucht, da $E = \{z_2\}$ und $z_1 = a$ der Startzustand ist.

Definitionen

$$L^0_{pq} = \begin{cases} \{a \mid \delta(p,a) = q\} & \text{falls } p \neq q \\ \{a \mid \delta(p,a) = q\} \cup \{\varepsilon\} & \text{sonst} \end{cases}$$

$$L^{r+1}_{pq} = L^r_{pq} \cup L^r_{p,r+1}(L^{r+1}_{p,r+1})^* L^r_{r+1,q} \overset{\text{induktiv}}{=} \gamma^{r+1}_{pq} = \gamma^r_{pq} | \gamma^r_{p,r+1}(\gamma^{r+1}_{p,r+1})^* \gamma^r_{r+1,q}$$

$l = 0$:

$$\begin{array}{lll} L^0_{1,1} = \{1,2\} \cup \{\varepsilon\} & \Rightarrow & \gamma^0_{1,1} = (\varepsilon|1|2) \\ L^0_{1,2} = \{0\} & \Rightarrow & \gamma^0_{1,2} = 0 \\ L^0_{2,1} = \{1\} & \Rightarrow & \gamma^0_{2,1} = 1 \\ L^0_{2,2} = \{0,2\} \cup \{\varepsilon\} & \Rightarrow & \gamma^0_{2,2} = (\varepsilon|0|2) \end{array}$$

$l = 1$:

$$\begin{aligned} \gamma^1_{1,2} &= \gamma^0_{1,2} | \gamma^0_{1,1}(\gamma^0_{1,1})^* \gamma^0_{1,2} \quad = (0)|(\varepsilon|1|2)(\varepsilon|1|2)^*0 \\ &= (1|2)^*0 \end{aligned}$$

$$\begin{aligned} \gamma^1_{2,2} &= \gamma^0_{2,2} | \gamma^0_{2,1}(\gamma^0_{1,1})^* \gamma^0_{1,2} \quad = (\varepsilon|0|2)|1(\varepsilon|1|2)^*0 \\ &= (\varepsilon|0|2)|1(1|2)^*0 \end{aligned}$$

$l = 2$:

$$\begin{aligned} \gamma^2_{1,2} &= \gamma^1_{1,2} | \gamma^1_{1,2}(\gamma^1_{2,2})^* \gamma^1_{2,2} \\ &= (1|2)^*0 \mid (1|2)^*0((\varepsilon|0|2)|1(1|2)^*0)^*(\varepsilon|0|2)|1(1|2)^*0 \\ &= (1|2)^*0(0|2|1(1|2)^*0)^* \end{aligned}$$

Der gesuchte reguläre Ausdruck γ mit $L(\gamma) = L(M)$ ist also:

$$\gamma = (1|2)^*0(0|2|1(1|2)^*0)^*$$

Aufgabe 30

Teilaufgabe a

Die Sprache

$$L = \{a^{(n^3)} \mid n \in \mathbb{N}\}$$

ist nicht regulär: Angenommen doch, dann gibt es gemäß Pumping-Lemma eine entsprechende Zahl l, sodass sich jedes Wort der Form a^{n^3}, $n^3 \geq l$, $n \in \mathbb{N}$ zerlegen lässt in uvw mit den Eigenschaften

1. $v \neq \varepsilon$

2. $|uv| \leq l$

3. $uv^i w \in L \forall i \geq 0$

Wähle speziell $x = a^{(l^3)}$. Betrachte eine beliebige Zerlegung $x = uvw$, die die Bedingungen 1, 2 und 3 erfüllt. Wegen Bedingung 1 und 2 gilt:

$$1 \leq |v| \leq |uv| \leq l$$

Mit Bedingung 3, $i = 2$, gilt dann:

$$uv^2w \in L$$

Andererseits gilt die Abschätzung:

$$l^3 = |x| = |uvw| < |uv^2w| = l^3 + |v| \leq l^3 + l < l^3 + 3l^2 + 3l + 1 = (l+1)^3$$

Somit kann die Zahl $|uv^2w|$, also die Anzahl der a's, keine Kubikzahl sein, da sie echt zwischen zwei aufeinanderfolgenden Kubikzahlen liegt. Somit gehört das Wort uvw nicht zu L. Dies ist ein Widerspruch zur Annahme, L sei regulär. Also ist gezeigt, dass L nicht regulär ist.

Teilaufgabe b

Die Sprache

$$L = \{x \in \{(,)\}^* \mid x \text{ stellt eine korrekte Klammerung dar}, |x| < 0\}$$

ist nicht regulär. Angenommen doch, dann gibt es gemäß Pumping-Lemma eine entsprechende Zahl l, sodass sich jedes Wort der Form $x \in L$, $|x| \geq l$ zerlegen lässt in uvw mit den Eigenschaften

1. $v \neq \varepsilon$

2. $|uv| \leq l$

3. $uv^i w \in L \forall i \geq 0$

Wähle speziell $x = ({}^l)^l$. Betrachte eine beliebige Zerlegung $x = uvw$, die die Bedingungen 1, 2 und 3 erfüllt. Wegen Bedingung 1 und 2 gilt:

$$1 \leq |v| \leq |uv| \leq l$$

Also ist $v = ({}^j \ (1 \leq j \leq l)$, besteht also aus mindestens einer öffnenden Klammer. Für $i = 2$ gilt dann:

$$uv^2w \in L$$

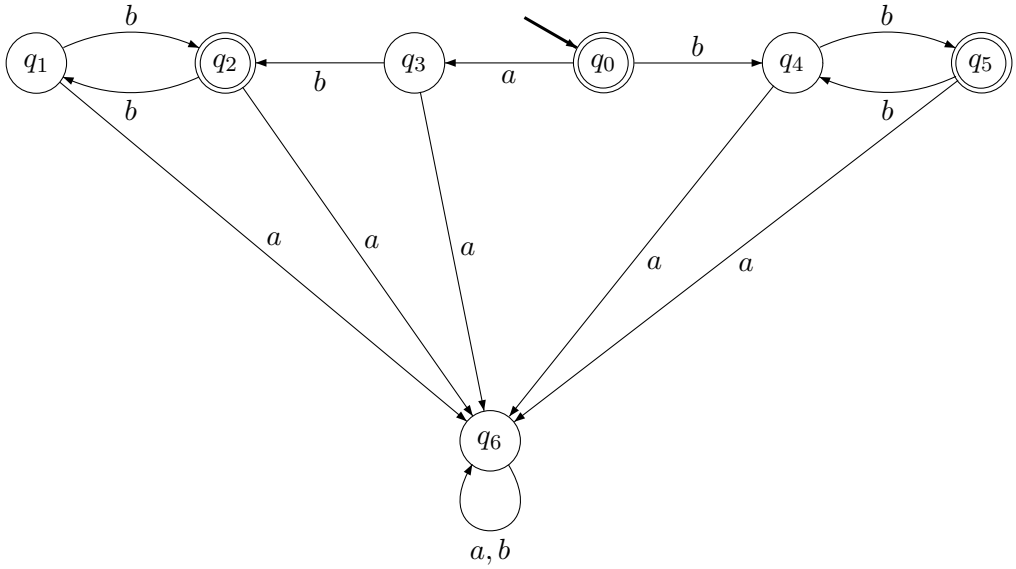
Offensichtlich ist dies jedoch nicht der Fall, da das Wort mehr öffnende als schließende Klammern hat, somit keine korrekte Klammerung darstellt und per Definition nicht zu L gehört. Dies ist ein Widerspruch zur Annahme, L sei regulär. Also ist gezeigt, dass L nicht regulär ist.

Teilaufgabe c

Die Sprache

$$L = \{a^{(n \bmod 2)} b^n \mid n \in \mathbb{N}\}$$

ist regulär, da ein DFA M exisitiert mit $L(M) = L$:



Wenn ein Wort x zur Sprache L gehört, dann gibt es drei Fälle:

1. x hat die Form ab^{2n-1}
2. x hat die Form b^{2n}
3. $x = \varepsilon$

Im 1. Fall hat das Wort $1(= n \mod 2)$ a, also ist n ungerade. In diesem Fall wird vom Startzustand q_0 aus ein a gelesen und in q_3 übergegangen. Der DFA M geht nun nur in den Endzustand q_2, wenn ein b oder eine beliebige andere ungerade Anzahl von b's gelesen werden (Pendeln zwischen q_2 und q_1).

Im 2. Fall hat das Wort $0(= n \mod 2)$ a's, also ist n gerade. In diesem Fall wird vom Startzustand q_0 aus mit einem b in Zustand q_4 übergegangen. Der DFA M geht nun nur in den Endzustand q_5, wenn ein b oder eine beliebige andere ungerade Anzahl von b's gelesen werden (Pendeln zwischen q_4 und q_5). Da im Startzustand schon ein b gelesen wurde, ist die Anzahl der gelesenen b's bei Erreichen des Endzustandes q_5 gerade.

Im 3. Fall ist das Wort das leere Wort, das also kein b und kein a enthält. Dies ist ein Sonderfall des 2. Falles, da die Anzahl der b's gerade ist. In diesem Fall akzeptiert der Automat direkt, weswegen der Startzustand q_0 gleichzeitig ein Endzustand ist.

Jedes andere Wort, das nicht von den oben beschriebenen Fällen abgedeckt wird, gehört nicht zur Sprache L und wird gleichzeitig nicht vom DFA M erkannt.

Es gilt: $L = L(M)$ und nach Satz 4.1 aus der Vorlesung: $L \in \mathcal{REG}$

Aufgabe 31

Teilaufgabe a

Die Sprache

$$L = \{ab^j c^j \mid j \in \mathbb{N}\}$$

ist nicht regulär. Angenommen doch, dann gibt es gemäß Pumping-Lemma eine entsprechende Zahl l, sodass sich jedes Wort der Form $x = ab^j c^j$, $j \in \mathbb{N}$,$|x| \geq l$ zerlegen lässt in uvw mit den Eigenschaften

1. $v \neq \varepsilon$
2. $|uv| \leq l$
3. $uv^i w \in L \forall i \geq 0$

Wähle speziell $x = ab^l c^l$. Betrachte eine beliebige Zerlegung $x = uvw$, die die Bedingungen 1, 2 und 3 erfüllt. Wegen Bedingung 1 und 2 gilt:

$$1 \leq |v| \leq |uv| \leq l$$

- Fall 1: Das Teilwort u ist leer und $v = ab^j$ $(j \leq l - 1)$
- Fall 2: Das Teilwort u ist nicht leer, also enthält v mindestens ein b.

Für $i = 2$ gilt dann:

$$uv^2 w \in L$$

Offensichtlich ist dies jedoch nicht der Fall, im 1. Fall das Wort uv^2w zwei a's enthält und im 2. Fall das Wort uv^2w mehr b's als c's hat (da v aus mindestens einem b besteht) und in beiden Fällen das Wort per Definition nicht zu L gehört. Dies ist ein Widerspruch zur Annahme, L sei regulär. Also ist gezeigt, dass L nicht regulär ist.

Teilaufgabe b

Die Sprache

$$A = \{a^i b^j c^k \mid i = 0 \vee j = k\}$$

besitze die Pumping-Zahl $l = 1$, sodass sich jedes Wort der Form $x = a^i b^j c^k$, $i = 0 \vee j = k$,$|x| \geq 1$ zerlegen lässt in uvw mit den Eigenschaften

1. $v \neq \varepsilon$

2. $|uv| \leq 1$

3. $uv^e w \in L \forall e \geq 0$

Daraus folgt:

$$u = \varepsilon, |v| = 1$$

- Fall 1: $i = 0$, d.h. ein Wort $x \in L$ hat die Form $x = b^j c^k$. Für $u = \varepsilon$ und $|v| = 1$ ergibt sich die Zerlegung $x = uvw$ mit $u = \varepsilon, v = b, w = b^{j-1}c^k$ $(j, k \geq 1$, da sonst $|x| = 0 < 1 = l)$.
- Fall 2: $j = k$, d.h. ein Wort $x \in L$ hat die Form $x = a^i b^j c^k$ $(j = k)$. Für $u = \varepsilon$ und $|v| = 1$ ergibt sich die Zerlegung $x = uvw$ mit $u = \varepsilon, v = a, w = a^{i-1}b^j c^k$ $(j = k)$

Für alle $e \geq 1$ gilt dann:

$$uv^e w \in L$$

- Im 1. Fall ergibt sich $uv^e w = b^e b^{j-1} c^k = b^{j+e-1} c^k$, was laut Definition zu L gehört.
- Im 2. Fall ergibt sich $uv^e w = a^e a^{i-1} b^j c^k = a^{e+i-1} b^j c^k$ $(j = k)$, was laut Definition zu L gehört.

Also gilt das Pumping-Lemma für die Pumping-Zahl $l = 1$ für die Sprache L.

Teilaufgabe c

A lässt sich wie folgt darstellen:

$$\underbrace{\{a^i b^j c^j\}}_{j=k} \cup \underbrace{\{b^j c^k\}}_{i=0}$$

Annahme: A ist regulär. Da die regulären Sprachen unter Vereinigung abgeschlossen sind, müsste nun ebenfalls die Sprache

$$\{a^i b^j c^j\}$$

regulär sein. Der Fall $i = 1$ $(\{ab^j c^j\})$ wurde jedoch in Teilaufgabe a) widerlegt.

Dies ist ein Widerspruch zur Annahme, A sei regulär. Also ist gezeigt, dass A nicht regulär ist.

Aufgabe 32

Sei $k : \mathbb{N} \to \mathbb{N}$ eine Funktion. $L = \{a^{k(n)} b^n \mid n \in \mathbb{N}\}$

Teilaufgabe a

Voraussetzung:
$L \in \mathcal{REG}$

Behauptung:
k ist beschränkt.

Beweis:
Nach [Köbler] 1.5 (Seite 13) gilt:

$$L \in \mathcal{REG} \Leftrightarrow R_L \text{ hat einen endlichem Index}$$

Da $L \in \mathcal{REG}$ existiert ein DFA M mit $L(M) = L$. Für M seien die Worte $x = a^{k(n)b^n}$ und $y = a^{k(m)}b^m$ bezüglich R_L genau dann äquivalent, wenn $k(n) = k(m)$, wenn also x und y die gleiche Anzahl von a's haben.

Offensichtlich ist R_L eine Äquivalenzrelation, die L in anhand der Anzahl der a's in verschiedene Äquivalenzklassen aufteilt. Die Anzahl dieser Äquivalenzklassen muss endlich sein, da L sonst nicht regulär wäre.

Also ist die Anzahl der Äquivalenzklassen endlich, und damit auch die Anzahl der möglichen Funktionswerte von $k(n)$. Also ist die Funktion $k(n)$ beschränkt. □

Teilaufgabe b

Voraussetzung:
k ist beschränkt.

Behauptung:
$L \in \mathcal{REG}$

Beweis:
Da die Funktion k beschränkt ist, gibt es nur endlich viele Funktionswerte $k(n)$, die die Anzahl der a's in einem Wort $x \in L$ festlegen.

Anhand dieser Funktionswerte lässt sich die Sprache L in endlich viele Äquivalenzklassen aufteilen. Sei R_L dabei die Äquivalenzrelation, sodass die Worte $x = a^{k(n)b^n}$ und $y = a^{k(m)}b^m$ bezüglich R_L genau dann äquivalent sind, wenn $k(n) = k(m)$, also x und y die gleiche Anzahl von a's haben.

R_L hat also einen endlichen Index und somit ist L nach dem Satz aus Teilaufgabe a) regulär. □

Serie 9

Institut für Informatik
Algorithmen und Komplexität
Dr. Till Nierhoff

WS 2002
11. Dezember 2002

Theoretische Informatik II

9. Serie

Abgabe bis zum 18. Dezember 2002

Aufgabe 33 [4+4+4 Punkte]

Entscheiden Sie, ob für alle Sprachen $L \in \mathcal{REG}$ über dem Alphabet Σ gilt:

a) $inv(L) := \{x_1 \dots x_n \mid x_n \dots x_1 \in L\} \in \mathcal{REG}$

b) $L' \in \mathcal{REG}, \forall L' \subseteq L$

c) $postfix(L) := \{v \in \Sigma^* \mid \exists u \in \Sigma^* : uv \in L\} \in \mathcal{REG}$

Begründen Sie Ihr Urteil.

Aufgabe 34 [5+5 Punkte]

Sei $k \geq 1$. Bestimmen Sie die minimale Anzahl von Zuständen eines DFA für die Sprachen:

a) $L_k := \{x = x_1 \dots x_n \in \Sigma^* \mid n \geq k,\ x_k = 1\}$ (k-ter Buchstabe gleich 1)

b) $inv(L_k)$.

Aufgabe 35 [8 Punkte]

Konstruieren Sie nach dem Verfahren der Vorlesung aus der folgenden Grammatik G einen PDA: $G = (\{A, S\}, \{(,), [,]\}, P, S)$ mit P:

$$S \to \varepsilon, (S), A], SS$$
$$A \to (SA, (S$$

Geben Sie eine akzeptierende Berechnung für das Eingabewort (()(()] an.

Aufgabe 36 [5+5 Punkte]

Zeigen Sie:

a) Für jede Sprache $L \in \mathcal{CFL}$ mit $\varepsilon \notin L$ gibt es eine kontextfreie Grammatik ohne ε-Produktionen.

b) Besitzt eine kontextfreie Grammatik G keine ε-Produktionen und keine Produktionen der Form $A \to B$, dann gibt es eine kontextfreie Grammatik G' mit $L(G') = L(G)$, sodass gilt: G' besitzt nur Regeln der Form $A \to a$ bzw. $A \to BC$, wobei a ein Terminal und B, C Nichtterminale sind.

Erläutern Sie in beiden Aufgaben Ihre Vorgehensweise an einem sinnvollen Beispiel.

Aufgabe 33

Sei $L \subseteq \Sigma^* \in \mathcal{REG}$.

Teilaufgabe a

Behauptung:
Wenn $L \in \mathcal{REG}$, so auch $inv(L) \in \mathcal{REG}$.

Beweis:
Wenn L regulär ist, dann kann L durch einen regulären Ausdruck γ ausgedrückt werden. Durch Induktion über die Definition der regulären Ausdrücke kann gezeigt werdenn, dass es einen Ausdruck $inv(\gamma)$ gibt, der $inv(L)$ ausdrückt.

Induktionsanfang:
Wenn $\gamma = \varepsilon$, so $inv(\gamma) = \varepsilon$.
Wenn $\gamma = \emptyset$, so $inv(\gamma) = \emptyset$.
Wenn $\gamma = a$, so $inv(\gamma) = a$ $(a \in \Sigma)$.

Induktionsschritt:
Wenn $\gamma = (\alpha|\beta)$, so $inv(\gamma) = (inv(\alpha) \mid inv(\beta))$.
Wenn $\gamma = \alpha\beta$, so $inv(\gamma) = inv(\beta)inv(\alpha)$.
Wenn $\gamma = \alpha^*$, so $inv(\gamma) = inv(\alpha)^*$.

Also gilt: $L \in \mathcal{REG} \Rightarrow inv(L) \in \mathcal{REG}$ □

Teilaufgabe b

Behauptung:
$L' \notin \mathcal{REG}$, $\forall L' \subseteq L$

Beweis:
Sei $L = \{a^i b^j \mid i, j \in \mathbb{N}\} \subseteq \{a, b\}^*$ eine reguläre Sprache und $L' = \{a^i b^i \mid i \in \mathbb{N}\}$ eine Teilmenge von L. L' ist jedoch nicht regulär (vgl. Vorlesung 4.3). □

Teilaufgabe c

Behauptung:
$L \in \mathcal{REG} \Rightarrow postfix(L) \in \mathcal{REG}$

Beweis:
Sei $M = (Z, \Sigma, \delta, Q_0, E)$ ein NFA mit $L(M) = L$. Es kann ein NFA $N = (Z, \Sigma, \delta, Q_0, E)$ mit $Q_0 = Z$ konstruiert werden, für den gilt: $L(N) = postfix(L)$.

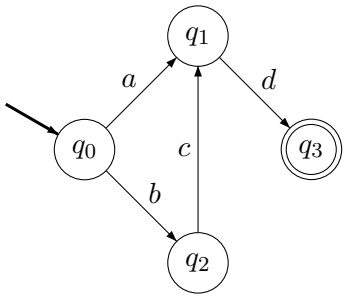
Die Sprache $postfix(L)$ enthält alle Teilworte v, für die $uv \in L$. In jedem Zustand von M wurde vom Startzustand aus schon ein gewisses Teilwort u gelesen. Falls $uv \in L(M)$, müsste um in einen Endzustand zu gelangen, noch das Teilwort v gelesen werden.

Der NFA N entspricht dem NFA M, jedoch ist hier jeder Zustand ein Startzustand. Nun akzeptiert N genau die Worte v, die im NFA M jene Teilworte v waren, sodass $uv \in L(M) = L$.

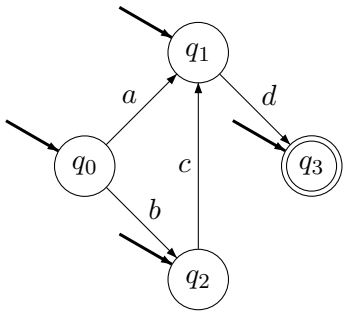
Also ist $L(N) = postfix(L(M)) = postfix(L)$.

Da N ein NFA ist, gilt $L(N) = postfix(L(M)) = postfix(L) \in \mathcal{REG}$ □

Beispiel
Sei $M = (\{q_0, q_1, q_2, q_3\}, \{a, b, c, d\}, \{q_0\}, \{q_3\})$ ein NFA mit $L(M) = \{ad, bcd\}$:



Daraus wird wie beschrieben der NFA $N = (\{q_0, q_1, q_2, q_3\}, \{a, b, c, d\}, \{q_0, q_1, q_2, q_3\}, \{q_3\})$ gebildet:



Es gilt: $L(N) = \{bcd, ad, d, cd, \varepsilon\} = postfix(L(M))$

Aufgabe 34

Sei $k \geq 1$.

Teilaufgabe a

Sei $x_1 \dots x_{k-1} x_k x_{k+1} \dots x_n$ die Eingabe von M.

Der Automat M soll genau dann akzeptieren, wenn das k-te Zeichen eine Eins ist. Um alle Zeichen bis dorthin ($x_1 \dots x_{k-1}$) zu lesen, benötigt der DFA M mit $L(M) = L_k$ genau $k - 1$ Zustandsübergänge, also k Zustände.

Nun wird das k-te Zeichen gelesen und entweder bei einer 1 in einen akzeptierenden Endzustand (ein weiterer Zustand), oder bei jedem anderen Zeichen in einen nicht-akzeptierenden Zustand (noch ein weiterer Zustand) übergegangen, von dem der Endzustand erreichbar ist.

Also benötigt der DFA $M = (Z, \Sigma, q_0, E)$ mit $L(M) = L_k$ mindestens $k + 2$ Zustände ($|Z| = k + 2$).

Teilaufgabe b

Sei $x_n \dots x_{k+1} x_k x_{k-1} \dots x_1$ die Eingabe von M'.

Entsprechend soll der Automat M' mit $L(M') = inv(L_k)$ genau dann akzeptieren, wenn bei einem gegebenen Eingabewort das k-te Zeichen der Eingabe von hinten eine Eins ist. Um alle Zeichen bis dorthin ($x_n \dots x_{k+1}$) zu lesen, benötigt der DFA M' genau $n - k$ Zustandsübergänge, also $n - k + 1$ Zustände.

Analog wird das k-te Zeichen von hinten (x_k) gelesen und entweder bei einer 1 in einen akzeptierenden Endzustand (ein weiterer Zustand), oder bei jedem anderen Zeichen in einen nicht-akzeptierenden Zustand (noch ein weiterer Zustand) übergegangen, von dem der Endzustand erreichbar ist.

Also benötigt der DFA $M' = (Z, \Sigma, q_0, E)$ mit $L(M) = inv(L_k)$ mindestens $k + 3$ Zustände ($|Z| = k + 3$).

Aufgabe 35

Sei G eine Grammatik mit $G = (\{A, S\}, \{(,), [,]\}, P, S)$ und P :

$$\begin{aligned} S &\rightarrow \varepsilon, (S), A], SS \\ A &\rightarrow (SA, (S \end{aligned}$$

Entsprechend Satz 5.1 aus der Vorlesung sei M ein PDA mit $L(M) = L(G)$ mit

$$M = (\{q\}, \{(,), [,]\}, \{A, S\} \cup \{(,), [,]\}, \delta, q, S)$$

und der Überführungsfunktion δ:

$$\begin{aligned} \delta(q, \varepsilon, S) &= \{(q, \varepsilon), (q, (S)), (q, A]), (q, SS)\} \\ \delta(q, \varepsilon, A) &= \{(q, (SA), (q, (S)\} \\ \delta(q, (, () &= \{(q, \varepsilon)\} \\ \delta(q,),)) &= \{(q, \varepsilon)\} \\ \delta(q, [, [) &= \{(q, \varepsilon)\} \\ \delta(q,],]) &= \{(q, \varepsilon)\} \end{aligned}$$

Das Wort (()(()] wird wie folgt vom PDA akzeptiert:

$$\begin{aligned}
(q, (()(()], S) &\vdash (q, (()(()], A]) \\
&\vdash (q, (()(()], (SA]) \\
&\vdash (q, ()(()], SA]) \\
&\vdash (q, ()(()], (S)A]) \\
&\vdash (q,)(()], S)A]) \\
&\vdash (q,)(()],)A]) \\
&\vdash (q, (()], A]) \\
&\vdash (q, (()], (S]) \\
&\vdash (q, ()], S]) \\
&\vdash (q, ()], (S)]) \\
&\vdash (q,)], S)]) \\
&\vdash (q,)],)]) \\
&\vdash (q,],]) \\
&\vdash (q, \varepsilon, \varepsilon)
\end{aligned}$$

Da $(q, (()(()], S) \vdash^* (q, \varepsilon, \varepsilon)$ wird die Eingabe (()(()] nach Definition akzeptiert.

Aufgabe 36

Sei $G = (V, \Sigma, P, S)$ eine kontextfreie Grammatik mit $L(G) = L \in \mathcal{CFL}$ und $\varepsilon \notin L$.

Teilaufgabe a

Sei $G' = (V, \Sigma, P', S)$ eine kontextfreie Grammatik ohne ε-Produktionen mit $L(G') = L(G)$, die wie folgt gebildet wird. Zunächst ist $P' = P$.

Fall 1: $A \to \varepsilon$ ist die einzige Regel, in der das Nichtterminal A auf der linken Seite steht:

> Jede Regel $B \to \alpha A \beta$ wird durch die Regel $B \to \alpha\beta$ ersetzt. Außerdem wird die Regel $A \to \varepsilon$ aus P' entfernt.

Fall 2: $A \to \varepsilon$ und $A \to \gamma \in P$.

> Zu jeder Regel $B \to \alpha A \beta$ wird die Regel $B \to \alpha\beta$ zu P' hinzugefügt. Außerdem wird die Regel $A \to \varepsilon$ aus P' entfernt.

Da sowohl α als auch β leer sein können, können Regeln der Form $B \to \varepsilon \in P'$ entstehen. Die beschriebenen Ersetzungen und Streichungen werden daher solange wiederholt, bis keine Regeln der Form $B \to \varepsilon$ zu P' gehören.

Die erzeugte Sprache wird durch den beschriebenen Algorithmus nicht verändert:

- Enthält eine (alte) Ableitung $B \to \alpha A \beta$, dann auch $A \to \varepsilon$ (und zwar später). Falls beide nicht direkt aufeinander folgen, verschiebe $A \to \alpha A \beta$ nach hinten. Dann ersetze beide durch $B \to \alpha\beta$.

- Enthält eine (neue) Ableitung $B \to \alpha\beta$, dann ersetze sie durch $B \to \alpha A\beta$, $A \to \varepsilon$.

Beispiel
Sei $G = (\{A, B, S\}, \{a, b, c\}, P, S)$ mit P :

$$\begin{array}{lcl} S & \to & aA, bB \\ A & \to & \varepsilon \\ B & \to & \varepsilon, c \end{array}$$

Daraus wird nun $G' = (\{A, B, S\}, \{a, b, c\}, P', S)$ gebildet. Zunächst ist $P' = P$.

Zunächst soll die ε-Produktion der Regel A entfernt werden. Sie ist die einzige Regel, in der A auf der linken Seite steht (Fall 1), also wird P' wie folgt manipuliert:

$$\begin{array}{lcl} S & \to & a, bB \\ B & \to & \varepsilon, c \end{array}$$

Nun soll die ε-Produktion der Regel B entfernt werden. Da es mehrere Regeln gibt, bei denen B auf der linken Seite steht (Fall 2), wird P' wie folgt manipuliert:

$$\begin{array}{lcl} S & \to & a, bB, b \\ B & \to & c \end{array}$$

G' enthält nun keine ε-Produktionen.

Teilaufgabe b

Nach Voraussetzung hat jede Regel von G die Formen $A \to a$ $(a \in \Sigma)$ oder die Form $A \to \alpha$ $(\alpha \in (V \cup \Sigma)^*, |\alpha| \geq 2)$.

Sei $G' = (V', \Sigma, P', S)$ die Grammatik in der gewünschten Form (also nur mit Regeln der Form $A \to a$ und $A \to BC$), die wie folgt bebildet wird. Zunächst ist $P' = P$ und $V' = V$.

Für jedes Terminalzeichen $a \in \Sigma$ fügen wir eine neue Variable X_a zu V' hinzu, sowie zu P' die Regel $X_a \to a$ hinzu.

Als nächstes ersetzen wir jedes Terminalzeichen a auf der rechten Seite einer Regel durch die zugehörige, gerade eingeführte Variable X_a (außer die Regel hat bereits die Form $A \to a$).

Nun haben alle Regeln die Form $A \to a$ oder $A \to B_1 B_2 \dots B_k, k \geq 2$.

Alle Regeln $A \to B_1 B_2 \dots B_k$ mit $k > 2$ haben noch nicht die gewünschte Form. Für jene Regeln führen wir weitere Variablen $C_2, \dots C_{k-1}$ ein und ersetzen solche Regeln durch

$$\begin{array}{rcl} A & \to & B_1 C_2 \\ C_2 & \to & B_2 C_3 \\ & \vdots & \\ C_{k-1} & \to & B_{k-1} B_k \end{array}$$

Nun haben alle Regeln die Form $A \to a$ bzw. $A \to BC$.

Die erzeugte Sprache wird durch den beschriebenen Algorithmus nicht verändert:

1.
   - Enthält eine (neue) Ableitung $A \to BX_a$ und (später) $X_a \to a$ (wobei X_a eine neu hinzugefügte Variable aus Schritt 1 ist), so ersetze diese durch $A \to Ba$. Wenn die Ableitungen nicht direkt hintereinander stehen, so verschiebe sie zunächst nach hinten.
   - Enthält eine (alte) Ableitung $A \to Ba$, so ersetze dies durch $A \to BX_a, X_a \to a$.

2.
   - Enthält eine (neue) Ableitung $A \to B_1 C_2, C_2 \to B_2 C_3, \dots, C_{k-1} \to B_{k-1} B_k$, so ersetze diese durch $A \to B_1 B_2 \dots B_k$. Wenn die Ableitungen nicht direkt hintereinander stehen, so verschiebe sie zunächst nach hinten.
   - Enthält eine (alte) Ableitung die $A \to B_1 \dots B_k$, dann ersetze dies durch $A \to B_1 C_2, C_2 \to B_2 C_3, \dots C_{k-1} \to B_{k-1} B_k$ (wobei $C_2, \dots C_{k-1}$ neue Variablen sind).

Beispiel
Sei $G = (\{A, B, S\}, \{a, b\}, P, S)$ mit P :

$$\begin{array}{rcl} S & \to & aA, aB \\ A & \to & bAA, aS, a \\ B & \to & aBBB, bS, b \end{array}$$

Daraus wird nun $G' = (V', \{a, b\}, P', S)$ gebildet. Zunächst ist $P' = P$ und $V' = V$.

Für die Terminalzeichen a und b werden nun die Variablen X_a, X_b zu V' und die Regeln $X_a \to a$ und $X_b \to b$ zu P' hinzugefügt.

Die Produktionen P' sehen nun wie folgt aus:

$$\begin{array}{rcl} S & \to & aA, aB \\ A & \to & bAA, aS, a \\ B & \to & aBBB, bS, b \\ X_a & \to & a \\ X_b & \to & b \end{array}$$

Nun ersetzen wir die Terminalzeichen a bzw. b auf der rechten Seite durch die entsprechenden neuen Regeln X_a bzw. X_b (außer die Regel hat bereits die Form $X \to a$ bzw. $X \to b$).

Die Produktionen P' sehen nun wie folgt aus:

$$\begin{array}{rcl} S & \to & X_aA, X_aB \\ A & \to & X_bAA, X_aS, a \\ B & \to & X_aBBB, X_bS, b \\ X_a & \to & a \\ X_b & \to & b \end{array}$$

Alle Regeln bis auf $A \to X_bAA$ und $B \to X_aBBB$ haben bereits die gewünschte Form. Für diese Regeln führen wir weitere neue Variablen E_2, F_2 und F_3 zu V' hinzu und ersetzen diese Regeln durch $A \to X_bE_2$, $E_2 \to AA$, $B \to X_aF_2$, $F_2 \to BF_3$ und $F_3 \to BB$.
Die Produktionen P' sehen nun wie folgt aus:

$$\begin{array}{rcl} S & \to & X_aA, X_aB \\ A & \to & X_aS, a, X_bE_2 \\ B & \to & X_bS, b, X_bF_2 \\ X_a & \to & a \\ X_b & \to & b \\ E_2 & \to & AA \\ F_2 & \to & BF_3 \\ F_3 & \to & BB \end{array}$$

$G' = (\{A, B, X_a, X_b, E_2, F_2, F_3, S\}, \{a, b\}, P', S)$ hat nun nur noch Regeln der Form $A \to a$ und $A \to BX_a$.

Serie 10

Institut für Informatik
Algorithmen und Komplexität
Dr. Till Nierhoff

WS 2002/03
8. Januar 2003

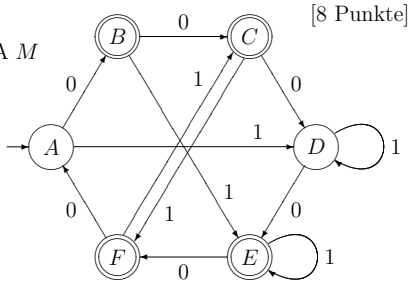
Theoretische Informatik II

10. Serie

Abgabe bis zum 15. Januar 2003

Aufgabe 37 [8 Punkte]

Minimieren Sie den nebenstehenden DFA M mit dem Verfahren aus der Vorlesung. (Geben Sie die einzelnen Schritte an.)



Aufgabe 38 [7 Punkte]

Sei A eine kontextfreie Sprache und B eine reguläre Sprache.

a) Zeigen Sie, dass $A \cap B$ kontextfrei ist.

b) Ist $A \setminus B$ kontextfrei?

Aufgabe 39 [3+3+4 Punkte]

Beweisen Sie:

a) $L = \{ab^j c^j d^j \mid j \in \mathbb{N}\} \notin \mathcal{CFL}$

b) $A = \{a^i b^j c^k d^l \mid i = 0 \vee j = k = l\}$ erfüllt die Folgerung des Pumping-Lemmas für kontextfreie Sprachen mit der Pumping-Zahl 1.

c) A ist dennoch nicht kontextfrei. *Hinweis*: Aufgabe 38 verwenden.

Damit haben Sie gezeigt, dass die Umkehrung des Pumping-Lemmas nicht gilt.

Aufgabe 40 [5+5+5 Punkte]

Sei $Rep(L) := \{aa \mid a \in L\}$ für eine Sprache L.

a) Finden Sie eine unendliche Sprache $L_1 \in \mathcal{CFL}$, sodass $Rep(L_1) \in \mathcal{CFL}$.

b) Finden Sie eine Sprache $L_2 \in \mathcal{CFL}$, sodass $Rep(L_2) \notin \mathcal{CFL}$.

c) Zeigen Sie, dass für alle Sprachen $L \in \mathcal{CFL}$ gilt: $Rep(L) \in \mathcal{CSL}$.
Hinweis: LBA konstruieren.

Aufgabe 37

Gegeben: DFA $M = (Z = \{A, B, C, D, E, F\}, \Sigma = \{0, 1\}, \delta, A, E' = \{B, C, E, F\})$
Gesucht: minimaler DFA M' mit $L(M') = L(M)$

Algorithmus, um M in M' zu überführen:
Im folgenden sind U, U' Listen, in denen die Zustände gesammelt werden, die nicht zusammengefasst werden können.

- **Initialsierung:**

  $U \leftarrow \{\{p, q\} \mid p \in E', q \in Z \backslash E'$

- **Schleife:**

  Wiederhole

  - $U' \leftarrow \{\{p, q\} \mid \exists a \in \Sigma : \{\delta(p, a), \delta(q, a) \in U\} \backslash U$
  - $U \rightarrow U \cup U'$

  bis $U' = \emptyset$

Für den gegebenen DFA M ergibt sich:

Initialsierung:
$U \leftarrow \{A, B\}$, da $A \in Z \backslash E'$ und $B \in E'$
$U \leftarrow \{A, C\}$, da $A \in Z \backslash E'$ und $C \in E'$
$U \leftarrow \{A, E\}$, da $A \in Z \backslash E'$ und $E \in E'$
$U \leftarrow \{A, F\}$, da $A \in Z \backslash E'$ und $F \in E'$
$U \leftarrow \{B, D\}$, da $D \in Z \backslash E'$ und $B \in E'$
$U \leftarrow \{C, D\}$, da $D \in Z \backslash E'$ und $C \in E'$
$U \leftarrow \{D, E\}$, da $D \in Z \backslash E'$ und $E \in E'$
$U \leftarrow \{D, F\}$, da $D \in Z \backslash E'$ und $F \in E'$

$U = \{\{A, B\}, \{A, C\}, \{A, E\}, \{A, F\}, \{B, D\}, \{C, D\}, \{D, E\}, \{D, F\}\}$

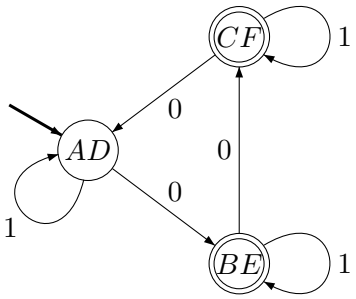
Schleife:
$U' \leftarrow \{B, C\}$, da $\{\delta(B, 0), \delta(C, 0)\} = \{C, D\} \in U$
$U' \leftarrow \{B, F\}$, da $\{\delta(B, 0), \delta(F, 0)\} = \{C, A\} \in U$
$U' \leftarrow \{C, E\}$, da $\{\delta(C, 0), \delta(E, 0)\} = \{D, F\} \in U$
$U' \leftarrow \{E, F\}$, da $\{\delta(E, 0), \delta(F, 0)\} = \{A, F\} \in U$

Es können keine weiteren Paare zu U' hinzugefügt werden.
$U' = \{\{B, C\}, \{B, F\}, \{C, E\}, \{E, F\}\}$
$U = U \cup U'$

Eine Wiederholung der Schleife führt zu keinen neuen Elementen, die zu U' bzw. später zu U hinzugefügt werden könnten.

Die Zustandspaare aus U können nicht zusammengefasst werden. Übrig bleiben die Zustandspaare $\{A, D\}$, $\{B, E\}$ und $\{C, F\}$. Also können die Zustände A und D zu AD, B und E zu BE und C und F zu CF zusammengefasst werden.

Dadurch ergibt sich der minimale DFA M' mit $M' = (Z = \{AD, BE, CF\}, \Sigma = \{0, 1\}, \delta, AD, E' = \{BE, CF\})$ und folgendem Zustandsgraphen:



Aufgabe 38

Sei $A \in \mathcal{CFL}$ und $B \in \mathcal{REG}$

Teilaufgabe a

Behauptung: $A \cap B \in \mathcal{CFL}$

Beweis: Sei $M_1 = (Z_1, \Sigma_1, \Gamma, \delta_1, q_{0_1}, \#, E_1)$ ein Kellerautomat mit $L(M_1) = A$ und $M_2 = (Z_2, \Sigma_2, \delta_2, q_{0_2}, E_2)$ ein DFA mit $L(M_2) = B$.

Die Endzustände des Kellerautomaten sind wie folgt charakterisiert: Eine vom leeren Keller akzeptierte Eingabe x hat folgende Konfigurationsübergänge:

$$(q_{0_1}, x, \#) \vdash^* (q_n, \varepsilon, \#) \vdash (q_e, \varepsilon, \varepsilon) \quad (q \in Z_1, q_e \in E_1)$$

Dies bedeutet, dass im letzten Schritt nach Löschen des Kelleranfangszeichens (und nur dann) in einen Zustand q_e übergegangen wird, der als Endzustand ausgezeichnet ist.

Der Kellerautomat M_{12} mit $M_{12} = (Z_1 \times Z_2, \Sigma_1 \cup \Sigma_2, \Gamma, \delta_{12}, (q_{0_1}, q_{0_2}), \#, E_1 \times E_2)$ und $\delta_{12}((q, r), a, \beta) := \{((q', r'), \gamma) \mid (q', \gamma) \in \delta_1(q, a, \beta) \wedge r' = \hat{\delta}_2(r, a)\}$ akzeptiert $A \cap B$ über die Endzustände $E_1 \times E_2$.

Dabei simuliert der Kellerautomat M_{12} den Kellerautomaten M_1, sowie den DFA M_2, auf denen parallel die Eingabe abgearbeitet wird. Dabei ist zu beachten, dass bei spontanten Übergängen von M_1 vom DFA M_2 kein Zeichen gelesen wird (dann gilt: $\hat{\delta}(q, \varepsilon) = q$). Letztlich akzeptiert M_{12} eine Eingabe x genau dann, wenn gilt:

$$((q_{0_1}, q_{0_2}), x, \#) \vdash^* ((q_{e_1}, q_{e_2}), \varepsilon, \varepsilon) \quad ((q_{e_1}, q_{e_2}) \in E_1 \times E_2)$$

Da $A \cap B$ von einem Kellerautomaten akzeptiert wird, gilt: $A \cap B \in \mathcal{CFL}$ □

Teilaufgabe b

Behauptung: $A \backslash B \in \mathcal{CFL}$

Beweis: $A \backslash B = A \cap \bar{B}$. $A \backslash B \in \mathcal{CFL}$, da in Teilaufgabe a gezeigt wurde, dass die kontextfreien Sprachen abgeschlossen sind bezüglich des Durchschnitts mit regulären Sprachen und die regulären Sprachen bezüglich des Komplements abgeschlossen sind. □

Aufgabe 39

Teilaufgabe a

Die Sprache

$$L = \{ab^j c^j d^j \mid j \in \mathbb{N}\}$$

ist nicht kontextfrei. Angenommen doch, dann gibt es gemäß Pumping-Lemma eine entsprechende Zahl l, sodass sich jedes Wort der Form $x = ab^j c^j d^j$, $j \in \mathbb{N}$,$|x| \geq l$ zerlegen lässt in $uvwxy$ mit den Eigenschaften

1. $vx \neq \varepsilon$
2. $|vwx| \leq l$
3. $uv^i wx^i y \in L \forall i \in \mathbb{N}$

Wähle speziell $x = ab^l c^l d^l$. Betrachte eine beliebige Zerlegung $x = uvwxy$, die die Bedingungen 1, 2 und 3 erfüllt. Wegen Bedingung 1 und 2 gilt:

$$1 \leq |vx| \leq |vwx| \leq l$$

- Fall 1: Das Teilwort vx enthält jedes der Zeichen b, c, d. Dieser Fall wegen der obigen Betrachtung nicht eintreten.
- Fall 2: Das Teilwort vx enthält nicht jedes der Zeichen b, c, d. Ein Wort $z' = uv^i wx^i y$ gehört nicht zu L, da es nach dem Pumpen nicht mehr gleich viele b's, c's und d's hat.

Damit ist gezeigt, dass L nicht kontextfrei ist.

Teilaufgabe b

Die Sprache

$$A = \{a^i b^j c^k d^l \mid i = 0 \vee j = k = l\}$$

besitze die Pumping-Zahl $l = 1$, sodass es sich jedes Wort der Form $a^i b^j c^k d^l$, $i = 0 \vee j = k = l$ zerlegen lässt in $uvwxy$ mit den Eigenschaften

1. $vx \neq \varepsilon$
2. $|vwx| \leq 1$
3. $uv^e wx^e y \in L \forall e \in \mathbb{N}$

- Fall 1: $i = 0$, d.h. ein Wort $x \in A$ hat die Form $x = b^j c^k d^l$. Wir wählen die Zerlegung $z = uvwxy$ mit $u = \varepsilon$, $v = \varepsilon$, $w = \varepsilon$, $x = b$ und $y = b^{j-1} c^k d^l$.
- Fall 2: $j = k = l$, d.h. ein Wort $x \in A$ hat die Form $x = a^i b^j c^k d^l$ $(j = k = l)$. Wir wählen die Zerlegung $z = uvwxy$ mit $u = \varepsilon$, $v = \varepsilon$, $w = \varepsilon$, $x = a$ und $y = a^{i-1} b^j c^k d^l$.

Für alle $e \geq 1$ gilt dann:

$$uv^e wx^e y \in A$$

- Im 1. Fall ergibt sich $uv^e wx^e y = b^e b^{j-1} c^k d^l$, was laut Definition zu A gehört.
- Im 2. Fall ergibt sich $uv^e wx^e y = a^e a^{i-1} b^j c^k d^l$ $(j = k = l)$, was laut Definition ebenfalls zu A gehört.

Also gilt das Pumping-Lemma für die Pumping-Zahl $l = 1$ für die Sprache A.

Teilaufgabe c

Behauptung: $A \notin \mathcal{CFL}$

Beweis (durch Kontraposition): Sei $A \in \mathcal{CFL}$ und $B = \{a^w b^x c^y d^z \mid w, x, y, z \in \mathbb{N}\} \in \mathcal{REG}$. In Aufgabe 38a wurde gezeigt, dass die kontextfreien Sprachen bezüglich des Durchschnitts mit regulären Sprachen abgeschlossen sind. Also müsste insbesondere gelten: $A \cap B = \{a^i b^j c^k d^l \mid j = k = l\}$. Für diese Sprache wurde allerdings schon bewiesen, dass sie nicht kontextfrei ist.

Also ist die Voraussetzung, die Sprache A sei kontextfrei, falsch. Es gilt: $A \notin \mathcal{CFL}$. □

Aufgabe 40

Sei $Rep(L) := \{aa \mid a \in L\}$ für eine Sprache L.

Teilaufgabe a

Sei $L = \{a^i \mid i \in \mathbb{N}^+\}$ eine kontextfreie Sprache mit $L = L(G)$ mit $G = (S, a, P, S)$ und $P : S \to aS, a$, die alle nichtleeren Worte über dem Alphabet $\{a\}$ enhält.

Zu jedem Wort $a^i \in L$ $(i \geq 0)$ muss gelten: $a^i a^i = a^{2i} \in Rep(L)$.

Die Sprache $Rep(L) = \{a^{2i} \mid i \in \mathbb{N}^+\}$ ebenso kontextfrei, da sie durch eine kontextfreie Grammatik $G' = (S, a, P, S)$ mit $P : S \to Saa, aa$ beschrieben werden kann, sodass gilt: $Rep(L) = L(G')$.

Teilaufgabe b

Sei $L = \{a^i b^i \mid i \in \mathbb{N}\}$ eine kontextfreie Sprache.

Die Sprache

$$Rep(L) = \{a^i b^i a^i b^i \mid i \in \mathbb{N}\}$$

ist nicht kontextfrei. Angenommen doch, dann gibt es gemäß Pumping-Lemma eine entsprechende Zahl l, sodass sich jedes Wort der Form $x = a^i b^i a^i b^i$, $i \in \mathbb{N}$,$|x| \geq l$ zerlegen lässt in $uvwxy$ mit den Eigenschaften

1. $vx \neq \varepsilon$
2. $|vwx| \leq l$
3. $uv^i wx^i y \in L \forall i \in \mathbb{N}$

Wähle speziell $x = a^l b^l a^l b^l$. Betrachte eine beliebige Zerlegung $x = uvwxy$, die die Bedingungen 1, 2 und 3 erfüllt. Wegen Bedingung 1 und 2 gilt:

$$1 \leq |vx| \leq |vwx| \leq l$$

- Fall 1: Das Teilwort vx enthält nur eines der Zeichen a oder b. Ein Wort $z' = uv^i wx^i y$ gehört nicht zu $Rep(L)$, da es nach dem Pumpen nicht mehr gleich viele a's wie b's hat.
- Fall 2: Das Teilwort vx enthält sowohl a's als auch b's.
  - Fall 2a: Das Teilwort vx enthält nicht genauso viele a's wie b's. Ein Wort $z' = uv^i wx^i y$ gehört nicht zu $Rep(L)$, da es nach dem Pumpen nicht mehr gleich viele a's wie b's hat.
  - Fall 2b: Das Teilwort vx enthält genauso viele a's wie b's, d.h. $vx = a^k b^k$ $(2k \leq l)$. Also kommen für eine Zerlegung $z = uvwxy$ nur zwei Fälle in Betracht:
    1. $u = a^{l-k}, w = \varepsilon, x = b^{l-k} a^l b^l$
    2. $u = a^l b^l a^{l-k}, w = \varepsilon, x = b^{l-k}$

In beiden Fällen gehört $z = uv^0wx^0y$ nicht zu $Rep(L)$, da es sich nicht mehr in zwei gleich lange Teilworte s mit $z' = ss$ zerlegen lässt, z'

Damit ist gezeigt, dass L nicht kontextfrei ist: $Rep(L) \notin \mathcal{CFL}$

Teilaufgabe c

Um zu zeigen, dass für alle $L \in \mathcal{CFL}$ gilt: $Rep(L) \in \mathcal{CSL}$, wird ein LBA M konstruiert, der ein gegebenes Wort x genau dann akzeptiert, wenn $x \in Rep(L)$ für eine kontextfreie Sprache L.

Der LBA M nützt dabei aus, dass unabhängig von einer gewählten Sprache L ein Wort $x \in Rep(L)$ die Form $x = aa$ hat. Um dies zu überprüfen, geht der LBA wie folgt vor:

Zunächst liest M die Eingabe x bis zum Ende und merkt sich die Anzahl der Zeichen in endlicher Kontrolle. Bei n Zeichen markiert M nun das 1. Zeichen mit einer Unterstreichung, sowie das Zeichen an der Stelle $\frac{n}{2} + 1$ mit einer Überstreichung. Dazu wird das Bandalphabet Γ um die Zeichen $\{\underline{a} \mid a \in \Sigma\}$ und $\{\overline{a} \mid a \in \Sigma\}$ erweitert.

Nun vergleicht M in jedem Durchlauf das 1. Zeichen auf der 1. Bandhälfte (unterstrichen) mit dem 1. Zeichen der 2. Bandhälfte (überstrichen). Anschließend werden die Markierungen verschoben, d.h. es wird das 2. Zeichen unter- und das Zeichen an der $\frac{n}{2} + 2$. Stelle überstrichen. Des Weiteren wird das Zeichen $\frac{n}{2} + 1$. Stelle durch einen Punkt gekennzeichnet. Dazu wird das Bandalphabet noch durch die Zeichen $\{\dot{a} \mid a \in \Sigma\}$ erweitert.

In den folgenden Schritten liest M vom linken Bandrand bis zum unterstrichenen Zeichen und vergleicht dies mit dem überstrichenem Zeichen. Bei Übereinstimmung werden die Markierungen ein Zeichen nach rechts verschoben.

Das gesamte Wort x wird von M akzeptiert, sobald das punktierte Zeichen mit dem Zeichen am rechten Bandrand erfolgreich verglichen wurde, da dann sichergestellt ist, dass für eine Eingabe $x = uv$ gilt $u_1 = v_1, u_2 = v_2, \ldots, u_m = v_m$ $(2m = n)$.

Erreicht M zu einem anderen Zeitpunkt das rechte Bandende oder schlägt ein Vergleich fehl, so soll M nicht akzeptieren.

Da $Rep(L)$ von einem LBA erkannt werden kann gilt: $Rep(L) \in \mathcal{CSL}$. □

Serie 11

Institut für Informatik
Algorithmen und Komplexität
Dr. Till Nierhoff

WS 2002/03
15. Januar 2003

Theoretische Informatik II

11. Serie

Abgabe bis zum 22. Januar 2003

Aufgabe 41 [5+5 Punkte]

a) Zeigen Sie, dass in einem gerichteten Graphen mit m Kanten gilt

$$\sum_{v \in V} d^+(v) = \sum_{v \in V} d^-(v) = m.$$

b) Wieviele ungerichtete Graphen mit der Knotenmenge $[n]$ gibt es?

Aufgabe 42 [5+5 Punkte]

a) Begründen Sie, warum der Q_n für alle $n \geq 2$ den C_{2^n} enthält.

b) *Definition*: In einem Graphen heißt eine Kante $e = \{u, v\}$ *Brücke*, falls es von u nach v nur einen Pfad gibt.
Gibt es einen Graphen G auf n Knoten, der sowohl eine Brücke als auch den C_n enthält?

Aufgabe 43 [10 Punkte]

Sei G ein Graph auf n Knoten. Beweisen Sie die Äquivalenz der folgenden Aussagen:

i) G ist ein Baum

ii) G ist (kanten-)minimal zusammenhängend

iii) G ist (kanten-)maximal kreisfrei

Aufgabe 44 [10 Punkte]

Seien $d_1, \ldots, d_n \in \mathbb{N} \backslash \{0\}$. Zu zeigen ist, dass es genau dann einen Baum $T = ([n], E)$ mit $d_T(i) = d_i$ gibt, wenn $\sum_{i=1}^{n} d_i = 2n - 2$.

Aufgabe 41

Teilaufgabe a

Zu zeigen:

$$\sum_{v \in V} d^+(v) = \sum_{v \in V} d^- = m$$

Beweis (durch vollständige Induktion über die Anzahl der Kanten m):

- Induktionsanfang $m = 1$:

  Bei einem Graph mit einer Kante (a, b) gilt:

  $$\sum_{v \in V} d^+(v) = \underbrace{d^+(a)}_{0} + \underbrace{d^+(b)}_{1} = 1 = \sum_{v \in V} d^-(v) = \underbrace{d^-(a)}_{1} + \underbrace{d^-(b)}_{0} = 1$$

- Induktionsvoraussetzung:

  Für einen Graphen mit m Kanten gilt:

  $$\sum_{v \in V} d^+(v) = \sum_{v \in V} d^- = m$$

- Induktionsschritt $m \to m + 1$:

  Sei G ein Graph mit m Kanten und l nach Induktionsvoraussetzung eine Zahl

  $$l := \sum_{v \in V} d^+(v) = \sum_{v \in V} d^- = m$$

  Zu G wird eine zusätzliche Kante $e = (u, v)$ hinzugefügt. Dementsprechend werden die Grade $d^-(u)$ und $d^+(v)$ werden nach Hinzufügen der Kante e um 1 erhöht.

  Dann gilt:

  $$\sum_{v \in V} d^+(v) = \sum_{v \in V} d^- = l + 1 = m + 1$$

□

Teilaufgabe b

Über der Knotenmenge $[n]$ gibt es genau $2^{\binom{n}{2}}$ ungerichtete Graphen:

Bei n Knoten gibt es $\binom{n}{2}$ Teilmengen mit genau zwei verschiedenen Knoten. Dies sind alle möglichen Kanten, die bei jedem möglichen Graphen entweder zur Kantenmenge gehören oder nicht.

Daraus ergibt sich die Anzahl $2^{\binom{n}{2}}$.

Aufgabe 42

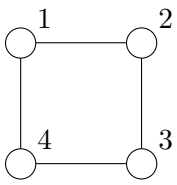
Teilaufgabe a

Zu zeigen: für $n \geq 2$ enthält der Hyperwürfel Q_n den Kreis C_{2^n}.

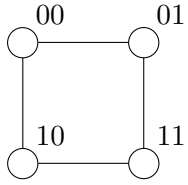
Beweis (durch vollständige Induktion über n):

- Induktionsanfang ($n = 2$):
  $Q_2 \hat{=} C_{2^2} = C_4$

  C_4 :



  Q_2 :



- Induktionsvoraussetzung: Für Q_n gilt: Q_n enthält C_{2^n}.
- Induktionsschritt ($n \to n + 1$):
  Konstruktion des Q_{n+1} wie folgt:
  - Hyperwürfel Q_n mit Knoten $1, \ldots, 2^n$ und Hyperwürfel Q'_n mit Knoten $1', \ldots, 2'^n$
  - verbinde Q_n und Q'_n über 2^n neue Kanten $\{u, u'\}$ mit $u = 1, \ldots, n$ und $u' = 1', \ldots, n'$ (also $\{1, 1'\}, \{2, 2'\}, \ldots$)
  - Q_n enthält laut Induktionsvoraussetzung C^{2^n}: $(1, \ldots, 2^n, 1)$ und Q'_n enthält C^{2^n}: $(1', \ldots 2'^n, 1')$
  - Der konstruierte Hyperwürfel C_{n+1} hat $2^n + 2^n = 2^{n+1}$ Knoten und enthält $C_{2^{n+1}}$ wie folgt: $(1, \ldots, 2^n, 2'^n, \ldots, 1', 1)$. □

Teilaufgabe b

Der Graph $G = (V, E)$ kann keine Brücke enthalten, da alle Knoten über mindestens zwei Pfade miteinander verbunden sind:

Enthält ein Graph mit n Knoten den Kreis C_n, so liegen alle Knoten auf diesem Kreis, d.h. für alle $u, v \in V$ existieren mindestens zwei verschiedene $u - v$-Pfade:

- Fall 1: u und v sind Nachbarn. Pfad 1 über Kante $\{u, v\}$ und Pfad 2 beginnt mit Kante $\{u, x\}$, wobei x ein Nachbar von u ($u \neq v$) auf dem Kreis C_n ist. Dann ist der Pfad wie folgt: $(u, x, \ldots, v)$.
- Fall 2: u und v sind keine Nachbarn. Sei x ein Nachbar von u auf dem Kreis C_n und y ein Nachbar von u ($x \neq y$) auf dem Kreis C_n. Pfad 1 über $(u, x, \ldots, v)$ und Pfad 2 über $(u, y, \ldots, v)$. □

Aufgabe 43

Beweis:
Sei $x, y \in V$.

- (i) $\Rightarrow$ (ii):

  $G = (V, E)$ sei ein Baum. Laut Baumdefinition ist G zusammenhängend, d.h. für je zwei Knoten $u, v \in V$ existiert genau ein $u - v$-Pfad.

  Annahme: G ist minimal zusammenhängend.

  Beweis durch Kontraposition: Wenn G nicht minimal zusammenhängend wäre, könnte eine Kante $e = \{x, y\}$ entfernt werden. Da zwei beliebige Knoten $x, y \in V$ jedoch nur über einen einzigen $x - y$-Pfad verbunden waren, existiert nach Entfernen von e kein solcher mehr. Also ist G nach Entfernen von e nicht mehr zusammenhängend.

- (ii) $\Rightarrow$ (iii):

  $G = (V, E)$ sei minimal zusammenhängend, d.h. für je zwei Knoten $u, v \in V$ existiert genau ein $u - v$-Pfad.

  Annahme: G ist maximal kreisfrei.

  Beweis durch Kontraposition: Wenn G nicht maximal kreisfrei wäre, so könnte eine neue Kante $e = \{x, y\}$ zu G hinzugefügt werden, ohne einen Kreis zu G hinzuzufügen. Allerdings existierte vor dem Hinzufügen von e schon (genau) ein $x - y$-Pfad. Durch die Kante $e = \{x, y\}$ würde ein Kreis entstehen.

- (iii) $\Rightarrow$ (i):

  $G = (V, E)$ sei maximal kreisfrei.

  Annahme: G ist zusammenhängend.

  Beweis durch Kontraposition: Wenn G nicht zusammenhängend wäre, so könnte eine Kante $e = \{x, y\}$ zu G hinzugefügt werden, die zwei Komponenten verbindet. Nach Hinzufügen von e wäre G immernoch kreisfrei, also vor dem Hinzufügen nicht maximal kreisfrei.

  Somit ist G (maximal) kreisfrei und zusammenhängend und somit per Definition ein Baum.

□

Aufgabe 44

Der Baum $T = ([n], E)$ ist nach Aufgabe 43 minimal zusammenhängend, hat also genau $n - 1$ Kanten.

Die Summe $\sum_{i=1}^{n} d_i$ zählt die Nachbarn jedes Knotens. Die Kante $\{u, v\}$ wird genau zweimal mit $d(u)$ und $d(v)$ gezählt. Somit entspricht die Summe der doppelten Kantenanzahl, also

$$\sum_{i=1}^{n} d_i = 2(n-1) = 2n - 2$$

□

Serie 12

Institut für Informatik
Algorithmen und Komplexität
Dr. Till Nierhoff

WS 2002/03
22. Januar 2003

Theoretische Informatik II

12. Serie

Abgabe bis zum 29. Januar 2003

Aufgabe 45 [8 Punkte]

Vervollständigen Sie den Beweis von Satz 1.1 der Vorlesung, indem Sie für das Verfahren *Suche* zeigen: Wenn der Eingabegraph G einen 1-k-Pfad enthält, enthält die Ausgabe T ebenfalls einen 1-k-Pfad. *Hinweis*: Induktion

Aufgabe 46 [4+4+4 Punkte]

Definition: In einem gerichteten Graphen $D = (V, A)$ heißt ein Knoten v_k von v_0 aus in k Schritten erreichbar, wenn es Knoten $v_1, \ldots, v_{k-1} \in V$ gibt, sodass $(v_{i-1}, v_i) \in A, \forall i \in [k]$.
Der Abstand von x nach y sei $\text{dist}(x, y) := \min\{k \mid y \text{ ist in } k \text{ Schritten von } x \text{ erreichbar}\}$.

a) Beschreiben Sie, wie eine Adjazenzliste für gerichtete Graphen aussieht.

b) Geben Sie einen Algorithmus an, der zu einem gegebenen Knoten die Abstände aller von ihm aus erreichbaren Knoten bestimmt. Beweisen Sie seine Korrektheit.

c) Bestimmen Sie die Laufzeit ihres Algorithmus.

Aufgabe 47 [4+4+4 Punkte]

Sei G ein Graph mit n Knoten und m Kanten.
Definition: Eine Partition der Knoten in zwei disjunkte Teile heiße *toll*, falls zwischen den Teilen mindestens $m/2$ Kanten verlaufen.
Eine Partition sei umso *besser*, je mehr Kanten zwischen den beiden Teilen verlaufen.

a) Zeigen Sie, dass in jedem Graphen eine bestmögliche Partition stets toll ist.

b) Geben Sie einen Algorithmus an, der eine tolle Partition findet. Beweisen Sie seine Korrektheit.

c) Wie kann eine tolle Partition in Laufzeit $O(n + m)$ gefunden werden?

Aufgabe 48 [8 Punkte]

Gegeben sei ein Graph G auf n Knoten als Adjazenzliste. Geben Sie einen Algorithmus an, der mit Laufzeit $O(n)$ testet, ob G ein Baum ist.

Aufgabe 46

Teilaufgabe a

Sei $G = ([n], E)$ ein gerichteter Graph. Die Adjazenzliste dieses Graphen enthält eine Liste alle Knoten $1 \dots n$, jedoch sind an die einzelnen Knoten u nur jene Nachbarn v in der Nachbarliste angehängt, für die eine Kante (u, v) existiert. Ebenso enthält die Kantenliste nun lediglich geordnete Paare von Knoten.

Teilaufgabe b

Gegeben sei ein Graph $G = (V, E)$ und ein Knoten $u \in V$. Der folgende Algorithmus leistet das Verlangte:

```
ALGORITHMUS Abstand(v):
  Initialiserung: Feld A: A[v] := ∞ ∀v ∈ V, v ≠ u, A[u] := 0;
  d := 0;
  N ← {u};
  M ← ∅;
  WHILE M ≠ N
       M ← M ∪ N;
       N ← {y ∈ Γ(x) | x ∈ N}\M;
       d := d + 1;
       A[x] := d ∀x ∈ N;
  ENDWHILE
  Ausgabe: Feld A mit A[v] := dist(u, v)
```

Dabei geht der Algorithmus wie folgt vor: zunächst wird das Feld A, das zu jedem Knoten v die Entfernung $dist(u, v)$ speichern soll für alle von u unterschiedlichen Knoten mit ∞, sowie für u selbst mit 0 initialisiert. Im Folgenden werden in jedem Schleifendurchlauf die Knoten mit wachsendem Abstand von u zu N hinzugefügt, also im ersten Durchlauf jene Knoten v mit $dist(u, v) = 1$ usw. Dieser Abstand wird an der entsprechenden Stelle im Feld A gespeichert.

Der Algorithmus terminiert, d.h. verlässt die Schleife, sobald keine weiteren Knoten gefunden werden können. Man beachte, dass der Abstand von u zu Knoten, die nicht erreicht werden können ∞ ist.

Teilaufgabe c

Die Laufzeit wird durch die maximale Länge eines $u - v$-Pfades bestimmt, der im worst case alle Kanten enthält. In diesem Fall müsste die Schleife m mal durchlaufen werden,

bis der am weitesten entfernteste Knoten v zu M hinzugefügt werden kann. Somit ergibt sich die Laufzeit $\mathcal{O}(m)$.

Aufgabe 47

Teilaufgabe a

Sei G ein Graph mit einer bestmöglichen Partition in A und B, mit a bzw. b Knoten. Somit hat G insgesamt $a + b$ Knoten und maximal $\binom{a+b}{2}$ Kanten. Da die Partition bestmöglich ist, existieren zwischen den Knoten aus A und B die maximale Anzahl an Kanten, nämlich $a \cdot b$.

Zu zeigen: $ab \geq \frac{\binom{a+b}{2}}{2}$

Es gilt: $\frac{\binom{a+b}{2}}{2} = \frac{a+b}{4(a+b-2)!}$.
Außerdem ist $4(a+b-2)! > 0$ für alle $a, b \in \mathbb{N}_+$.

Somit gilt $ab \geq \frac{\binom{a+b}{2}}{2}$ für alle $a, b \in \mathbb{N}_+$. □

Teilaufgabe b

Der folgende Algorithmus leistet das Verlangte:

```
ALGORITHMUS TolleParition(G):
  Eingabe: G = ([n], E);
  E' ← ∅;
  M ← {1};
  M' ← ∅;
  WHILE |E'| < |E|/2
       M ← M';
       M ← M ∪ {Γ(x) | x ∈ M};
       E' ← {{u, v} | u ∈ M, v ∈ Γ(u), v ∉ M'};
  ENDWHILE;
  IF E' ≠ E
       THEN Ausgabe: tolle Partition mit Knotenliste M'
       ELSE Ausgabe: keine tolle Parition gefunden
```

Dabei sei G o.B.d.A. zusammenhängend, da nicht zusammenhängende Knoten die Kantenanzahl nicht beeinflussen. In E' werden nach und nach die Kanten gespeichert, die von die markierte Knotenmenge M bzw. M' verlassen. Falls diese Knotenmenge größer oder gleich der Hälfte der gesamten Knotenmenge E ist und sich E' trotzdem noch von

E unterscheidet, enthält nach der WHILE-Schleife M' alle Knoten der tollen Partition.

Der Graph G lässt sich also in die disjunkten Knotenmengen M' und $[n]\backslash M'$ zu einer tollen Partition zerlegen.

Teilaufgabe c

Im worst-case wird in jedem Schleifendurchlauf nur ein einziger Knoten zu M und eine einzige Kante zu E' hinzugefügt. Da nur die Knotenanzahl die Abbruchbedingung ($|E'| \geq \frac{|E|}{2}$) beeinflusst, ist die Komplexität des Algorithmus $\mathcal{O}(m)$.

Aufgabe 48

Um bei einem Graphen G auf n Knoten zu überprüfen, ob G ein Baum ist, kann wie folgt auf den Algorithmus der Tiefensuche zurückgegriffen werden:

```
ALGORITHMUS Baum(G):
  Eingabe: G = ([n], E);
  IF |E| ≠ n − 1
      THEN Ausgabe: kein Baum;
      ELSE Tiefensuche(G);
  Ausgabe Tiefensuche: T = (M, E')
  IF M = [n] ∧ E' = E
      THEN Ausgabe: Baum;
      ELSE Ausgabe: kein Baum;
```

Dabei nutzt der Algorithmus aus, dass ein Baum über n Knoten $n-1$ haben muss, um zusammenhängend und kreisfrei zu sein. Also wird in der Adjazenzliste die Kantenliste durchgezählt und bei weniger als $n-1$ Kanten (nicht zusammenhängend) oder mehr als $n-1$ Kanten (nicht kreisfrei) direkt der Misserfolg ausgegeben. Für diesen Test benötigt der Algorithmus $\mathcal{O}(n)$.

Anschließend wird der Algorithmus Tiefensuche auf den Graphen G angesetzt. Er hat nach Satz 1.2 aus der Vorlesung die Komplexität $\mathcal{O}(m)$. Da bereits vorher sichergestellt wurde, dass nur im Falle $m = n-1$ fortgefahren wird, gilt $\mathcal{O}(m) = \mathcal{O}(n-1) = \mathcal{O}(n)$.

Die Tiefensuche ermittelt auf dem gegebenen Graphen G einen spannenden Baum T. Falls es sich bei dem Eingabegraphen G um einen Baum handelt, müsste er mit T identisch sein. Dies wird nach der Tiefensuche überprüft. Dazu müssen die Kanten- und Knotenmengen (E und E' bzw. $[n]$ und M) übereinstimmen. Dies ist über die Zuweisung $E'' = E\backslash E'$ und den anschließenden Test $E'' = \emptyset$? in $\mathcal{O}(1)$ möglich.

Dadurch ergibt sich eine Gesamtkomplexität für den gegebenen Algorithmus von $\mathcal{O}(n)$.

Serie 13

Institut für Informatik
Algorithmen und Komplexität
Dr. Till Nierhoff

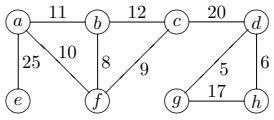
WS 2002/03
29. Januar 2003

Theoretische Informatik II

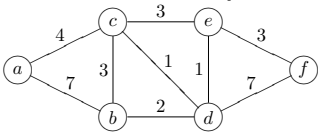
13. Serie

Abgabe bis zum 5. Februar 2003

Aufgabe 49 [4+4 Punkte]



zu a)



zu b)

a) Bestimmen Sie anhand des Algorithmus von Prim einen MST im gegebenen Graphen.

b) Ermitteln Sie mit dem Algorithmus von Dijkstra im gegebenen Graphen die Abstände aller Knoten von a.

Geben Sie bei beiden Aufgaben die einzelnen Schritte an.

Aufgabe 50 [8 Punkte]

Sei $G = (V, E)$ mit $w : E \to \mathbb{R}^+$ ein kantengewichteter Graph.
Ein kürzester Kreis in G ist ein Kreis C, der $w(C) := \sum_{e \in E(C)} w(e)$ minimiert.

Formulieren Sie einen Algorithmus, der einen kürzesten Kreis findet, und beweisen Sie seine Korrektheit. Die Laufzeit sollte $O(n^4)$ nicht überschreiten (mit kurzer Begründung).

Aufgabe 51 [10 Punkte]

Sei $G = (V, E)$ ein Graph mit Kanten-Kapazitätsfunktion $c : E \to \mathbb{R}^+$.
Für jeden Pfad P in G sei seine Kapazität $c(P) := \min_{e \in E(P)} c(e)$.

Formulieren Sie einen Algorithmus, der zu einem gegebenen Knoten s für alle anderen Knoten v die maximale Kapazität eines s-v-Pfades findet. Beweisen Sie seine Korrektheit. Die Laufzeit sollte $O(n^2)$ nicht überschreiten (mit kurzer Begründung).

Aufgabe 52 [4+3+3+4 Punkte]

Sei $G = ([n], E)$ ein Graph.
Definitionen:
Eine Knotenmenge $S \subseteq V$ heißt *stabil* in G, falls $E \cap \binom{S}{2} = \emptyset$.
Gibt es keine stabile Menge S' mit $|S'| > |S|$, so ist S eine *größte* stabile Menge.
Der *Maximalgrad* von G ist $\Delta(G) := \max_{v \in V} d(v)$.

Der Greedy-Algorithmus zur Findung einer möglichst großen stabilen Menge S geht folgendermaßen vor:

Input: $G = (V, E), V = [n]$
Output: S
(1) $S \leftarrow \emptyset$
(2) **while** $V \neq \emptyset$
(3) $s \leftarrow \min V$
(4) $S \leftarrow S \cup \{s\}$
(5) $V \leftarrow V \setminus (\{s\} \cup \Gamma(s))$
(6) $G \leftarrow G[V]$

a) Zeigen Sie, dass dieser Greedy-Algorithmus stets eine stabile Menge S der Größe $|S| \geq \frac{n}{\Delta(G)+1}$ findet.

b) Sei $0 < \Delta < n$. Geben Sie einen Graphen $G = ([k], E)$ mit $\Delta(G) = \Delta$ und $k \geq n$ an, bei dem der Greedy-Algorithmus nicht die größte stabile Menge findet.

c) Sei $0 < \Delta < n$. Geben Sie einen Graphen $G = (V, E)$ mit $\Delta(G) = \Delta$ und $|V| \geq n$ an, bei dem der Greedy-Algorithmus unabhängig von der Knotennummerierung stets eine größte stabile Menge findet.

d) Zeigen Sie: Für jeden Graphen gibt es eine Knotennummerierung, sodass der Greedy-Algorithmus eine größte stabile Menge findet.

Aufgabe 49

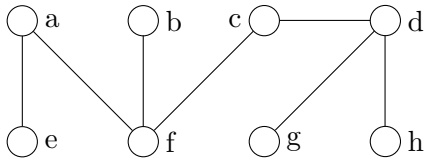
Teilaufgabe a

Wir nehmen für den Graphen die Abbildung φ wie folgt an, sodass die Knotenmenge E der Menge [8] entspricht:
$\varphi(a) = 1, \varphi(b) = 2, \varphi(c) = 3, \varphi(d) = 4, \varphi(e) = 5, \varphi(f) = 6, \varphi(g) = 7, \varphi(h) = 8$

Im Folgenden werden die Werte der markierten Knotenmenge M, der Ausgabekantenmenge E' und der jeweils betrachteten Kante e in den jeweiligen Durchläufen der WHILE-Schleife betrachtet:

Schleifendurchlauf	M	E'	e
Init.	$\{a\}$	$\emptyset$	
1	$\{a, f\}$	$\{\{a, f\}\}$	$\{a, f\}$
2	$\{a, b, f\}$	$\{\{a, f\}, \{f, b\}\}$	$\{f, b\}$
3	$\{a, b, c, f\}$	$\{\{a, f\}, \{f, b\}, \{f, c\}\}$	$\{f, c\}$
4	$\{a, b, c, d, f\}$	$\{\{a, f\}, \{f, b\}, \{f, c\}, \{c, d\}\}$	$\{c, d\}$
5	$\{a, b, c, d, f, g\}$	$\{\{a, f\}, \{f, b\}, \{f, c\}, \{c, d\}, \{d, g\}\}$	$\{d, g\}$
6	$\{a, b, c, d, f, g, h\}$	$\{\{a, f\}, \{f, b\}, \{f, c\}, \{c, d\}, \{d, g\}, \{d, h\}\}$	$\{d, h\}$
7	$\{a, b, c, d, e, f, g, h\}$	$\{\{a, f\}, \{f, b\}, \{f, c\}, \{c, d\}, \{d, g\}, \{d, h\}, \{a, e\}\}$	$\{a, e\}$

Ausgabe: $T = (M, E')$:



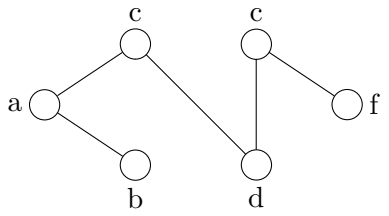
Teilaufgabe b

Im folgenden werden die Werte eines Nachbarknotens v, der markierten Knotenmenge M, des Entfernungsfeldes d und des Vorgängerfeldes p in den jeweiligen Durchläufen der WHILE-Schleife betrachtet:

S.durchlauf	v	M	Feld d						Feld p					
Init		$\emptyset$	0	∞	∞	∞	∞	∞	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
1	a	$\{a\}$	0	7	4	∞	∞	∞	$\perp$	a	a	$\perp$	$\perp$	$\perp$
2	c	$\{a, c\}$	0	7	4	5	7	∞	$\perp$	a	a	c	c	$\perp$
3	d	$\{a, c, d\}$	0	7	4	5	6	12	$\perp$	a	a	c	d	d
4	e	$\{a, c, d, e\}$	0	7	4	5	6	9	$\perp$	a	a	c	d	e
5	b	$\{a, b, c, d, e\}$	0	7	4	5	6	9	$\perp$	a	a	c	d	e
6	f	$\{a, b, c, d, e, f\}$	0	7	4	5	6	9	$\perp$	a	a	c	d	e

$E' = \{\{a,b\},\{a,c\},\{c,d\},\{d,e\},\{e,f\}\}$

Ausgabe: $T = (M, E')$:



Aufgabe 51

Der Algorithmus, die maximale Kapazität eines $s - v$ Pfades zu gegebenen s zu finden, basiert auf dem Algorithmus von Dijkstra:

```
PROCEDURE Capacity (G, c, s);
BEGIN
 EINGABE G, c, s;
 M ← ∅;
 d[s] ← ∞;
 d[v] ← 0   (∀v ∈ V\{s});
 p[v] ←⊥    (∀v ∈ V);
 WHILE M ≠ V
   v ← argmin{d[v'] | v' ∈ V\M};
   M ← M ∪ {v};
   FORALL v' ∈ Γ(v)\M
     IF d[v'] < min{c({p[v], v}), c({v, v'})}                    (*)
       d[v'] ← min{c({p[v], v}), c({v, v'})};
     ENDIF;
   ENDFOR;
 ENDWHILE;
 E' ← {{p[v], v} | v ∈ V\{u}};
 AUSGABE T = (M, E');
END;
```

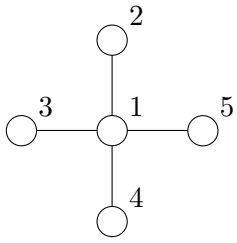
Dabei wird das Feld d nun dazu benutzt, die maximal mögliche Kapazität des Pfades von s für den jeweiligen Knoten zu speichern. Dazu wird das Feld für den Knoten s selbst mit ∞ (undendlicher Kapazität zu sich selbst) und für alle anderen Knoten zunächst mit 0 initialisiert. In der IF-Zeile (*) wird nun die größtmögliche Kapazität des Pfades von s wie folgt ermittelt: wenn das Minimum der Kapazitäten der Kante zu v und der Kante $\{v, v'\}$ größer als die bisher ermittelte maximale Kapazität zum Knoten v' ($d[v']$), so wird ersteres in $d[v']$ gespeichert.

Der Algorithmus CAPACITY unterscheidet sich nur in der Initialisierung und des IF-Vergleichs vom Algorithmus von DIJKSTRA, sodass bei einer Realisierung mittels eines Heaps die Laufzeit von CAPACITY nach 1.3 aus der Vorlesung bei $\mathcal{O}((n+m)\cdot\log n)$ liegt, also unter $\mathcal{O}(n^2)$.

Aufgabe 52

Teilaufgabe b

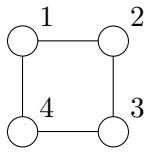
Der Algorithmus kann im gegebenen Graphen



nicht die größte stabile Menge S finden, da er stets bei dem Knoten mit der geringsten Nummerierung beginnt und im nächsten Schritt alle Nachbarn dieses Knotens aus V entfernt. Da der Knoten 1 mit allen anderen benachbart ist, kann nur eine stabile Menge mit $|S| = 1$ gefunden werden. Für die größte stabile Menge gilt $S_{max} = \{2, 3, 4, 5\}$ mit $|S_{max}| = 4$.

Teilaufgabe c

Beim Graphen



hingegen kann der Algorithmus eine größte stabile Menge finden, da alle Knoten gleich viele Nachbarn haben, sodass es irrelevant ist, bei welchem Knoten begonnen wird und somit das Problem mit der Nummerierung (Teilaufgabe b) nicht auftritt.

Teilaufgabe d

Die Knotennummerierung müsste so geändert werden, dass vom Knoten mit dem kleinsten Grad aus aufsteigend nummeriert wird, also eine hohe Nummerierung einen höheren Grad bedeutet.

Der Algorithmus löscht nach wie vor die Nachbarn des betrachteten Knotens. Durch die oben beschriebenen Nummerierung wird jedoch vermieden, dass ungünstige Fälle wie in Teilaufgabe a auftreten.

Bewertungen

- Aufgabe 1a: alles OK [3/3]
- Aufgabe 1b: Kommentar: „falsche Seite“ [2/4]
- Aufgabe 1c: alles OK [3/3]
- Aufgabe 2: fehlt: Fehlerbehandlung für Fehleingaben [8/10]
- Aufgabe 3a: korrekt: $L(M) = \{a, b\}^*$ [2/4]
- Aufgabe 3b: für $x = \varepsilon$ gibt es kein x_1 [5/6]
- Aufgabe 4: fehlt: Fehlerbehandlung für Fehleingaben [8/10]
- Aufgabe 5: zwar aus Buch abgeschrieben, aber unvollständig (?) [4/8]
- Aufgabe 6: alles OK [16/16]
- Aufgabe 7: Kommentar: „falls eine 1 vorkommt“ [5/6]
- Aufgabe 8: mehrere, kleinere Fehler [7/10]
- Aufgabe 9a: Kommentar: „muss auch Wörter aus A abbilden“ [3/4]
- Aufgabe 9b: Kommentar: „Wieso ist f_3 berechenbar?“ [5/6]
- Aufgabe 10a: Kommentar: „Warum berechenbar?“ [3/4]
- Aufgabe 10b: Kommentar: „Das definiert noch keine Abb. für Teilmengen von $\mathcal{RE}$“, Korrektur: $\mathcal{L} \subseteq \mathcal{RE}$ [3/6]
- Aufgabe 11: kleiner Fehler bei 11c [7/8]
- Aufgabe 12a: Fehler: f_{Eq} ist nicht berechenbar, da H semi-entscheidbar ist [2/4]
- Aufgabe 12b: siehe oben [3/4]
- Aufgabe 12c: Kommentar: „Wie ermittle ich das?“ [3/4]
- Aufgabe 13: richtig viele Fehler... [3/10]
- Aufgabe 14a: der Beweis aus der Vorlesung sollte benutzt werden [0/5]

- Aufgabe 14b: Fehler bei Überführungs- und Löschregel [3/5]
- Aufgabe 15: alles OK [12/12]
- Aufgabe 16: alles OK [8/8]
- Aufgabe 17: alles OK [10/10]
- Aufgabe 18a: Fehler: braucht keine ε-Sonderregel. Korrekte Klammerung ist enthalten [2/5]
- Aufgabe 18b: Kommentar: „Warum?“ [4/5]
- Aufgabe 19a: alles OK [4/4]
- Aufgabe 19b: Kommentar: „Es ist nicht klar, dass jedes Palindrom abgeleitet werden kann.“ [4/6]
- Aufgabe 20: Kommentar bei 20d: „Auch hier wurde nicht gezeigt, dass alle Wörter erzeugt werden können“ – sonst alles OK [8/10]
- Aufgabe 21: nur $L(G) \subseteq L(M)$ gezeigt (keine Rückrichtung) [6/9]
- Aufgabe 22: Kommentar: „falsche Aufgabe“ – trotzdem ein Punkt... [1/10]
- Aufgabe 23a: Kommentar: „A entscheidbar?“ [0/2]
- Aufgabe 23b: Kommentar: „falsch, falls $A \cap B \neq \emptyset$“ [1/2]
- Aufgabe 23c: ohne Kommentar [1/2]
- Aufgabe 23d: alles OK [3/3]
- Aufgabe 24a: alles OK [4/4]
- Aufgabe 24b+c: Kommentar: „inwiefern wird Baumcharakteristik verletzt?“ [4/8]
- Aufgabe 25: Kommentar: „Wo sind $\square$?“ [4/10]
- Aufgabe 26a: Kommentar: „Das sind nicht alle Übergänge“ [3/5]
- Aufgabe 26b: Kommentar: „Äquivalenz?“ [4/5]
- Aufgabe 27a: Kommentar: „Äquivalenz?“ [4/5]
- Aufgabe 27b: alles OK [5/5]
- Aufgabe 28a: alles OK [4/4]
- Aufgabe 28b: alles OK [4/4]
- Aufgabe 28c: Verpeiler: Ergebnis enthält immer noch „\“ [0/2]

- Aufgabe 29: alles OK [8/8]
- Aufgabe 30: alles OK [12/12]
- Aufgabe 31a: alles OK [3/3]
- Aufgabe 31b: alles OK [3/3]
- Aufgabe 31c: Kommentar: „Nur kleiner Teil gezeigt“ [2/4]
- Aufgabe 32: Lösung unklar [1/10]
- Aufgabe 33a: nicht Äquivalenz bewiesen [2/4]
- Aufgabe 33b: alles OK [4/4]
- Aufgabe 33c: Kommentar: „präziser beweisen“ [3/4]
- Aufgabe 34a: Kommentar: „Wieso nicht weniger?“ [2/5]
- Aufgabe 34b: falsch: Zustandszahl hängt von n ab [0/5]
- Aufgabe 35: alles OK [8/8]
- Aufgabe 36: alles OK – Kommentar: „Beispiel mit $\alpha = \beta = \varepsilon$ wäre angebracht“ [9/10]
- Aufgabe 37: alles OK [8/8]
- Aufgabe 38a: Kommentar: „Kellerautomaten haben keine Endzustände“ [3/5]
- Aufgabe 38b: alles OK [2/2]
- Aufgabe 39a: Kommentar: „$vx = a$?“ [2/3]
- Aufgabe 39b: alles OK [3/3]
- Aufgabe 39c: Kommentare: „$A \cap B = A$“ und „$0 \in \mathbb{N}$“ [2/4]
- Aufgabe 40a: alles OK [5/5]
- Aufgabe 40b: alles OK [5/5]
- Aufgabe 40c: Kommentare: „Wie soll er sich die Anzahl der Zeichen merken?“ und „Die Erkennung, ob die erste Hälfte L ist, fehlt“ [2/5]
- Aufgabe 41: alles OK [10/10]
- Aufgabe 42a: Kommentare: „Wieso ist das Q_{n+1}?“ und „Welcher Pfad wird benutzt?“ [4/5]
- Aufgabe 42b: alles OK [5/5]

- Aufgabe 43: alles OK [10/10]
- Aufgabe 44: Kommentar: „Äquivalenz?“ [3/10]
- Aufgabe 45: nicht gemacht [0/8]
- Aufgabe 46a: alles OK [4/4]
- Aufgabe 46b: Korrektur: „$y \in \Gamma^*(x)$“ (nur ausgehende) und Kommentar: „Wieso wird jeder Knoten nur einmal besucht?“ [2/4]
- Aufgabe 46c: Kommentar: „Die Initialisierung benötigt $\mathcal{O}(n)$ [10/10]
- Aufgabe 47: Algorithmus ist fehlerhaft [4/12]
- Aufgabe 48: alles OK [8/8]
- Aufgabe 49: alles OK [8/8]
- Aufgabe 50: nicht gemacht [0/8]
- Aufgabe 51: Korrektur: „$v \rightarrow$ argmax“ und Kommentare: „Wo wird p gesetzt?“ und „Warum funktioniert das?“ [4/8]
- Aufgabe 52: viele Fehler [3/10]

Musterlösungen

Institut für Informatik
Algorithmen und Komplexität
Dr. Till Nierhoff

WS 2002
26. November 2002

Theoretische Informatik II

Musterlösungen

Aufgabe 8 [10 Punkte]

Zeigen Sie: Eine k-Band-Turingmaschine kann so durch eine 1-TM simuliert werden, dass die Simulation einer Rechnung mit Laufzeit t und Platzbedarf s eine Laufzeit von $O(t^2)$ und den Platzbedarf $O(s)$ hat.

Wir betrachten die Simulation aus der Vorlesung. Die Ausgangsmaschine sei M und für eine Eingabe x sei der Platzbedarf s und der Zeitbedarf t. Die Simulationsmaschine sei M' mit s' und t' analog.
Die Übergangsfunktion von M' war so gemacht, dass die Simulation einer Rechnung unterteilt war in eine „Initialisierung“, einen „Rechtslauf“, eine „Auswahl“ und einen „Linkslauf“.
Die Initialisierung braucht $2|x| + 2 \leq 2s + 2 = O(s)$ Schritte, bis hierher ist der Platzbedarf höchstens $|x| + 1 \leq s + 1 = O(s)$. Für jeden der t Schritte der ursprünglichen Berechnung macht die Simulation einen Linkslauf, eine Auswahl und einen Rechtslauf. Dabei besteht die Auswahl aus einem Schritt, ein Rechtslauf aus höchstens s' und ein Linkslauf aus höchstens $s' + 3k$ Schritten. Das letztere liegt daran, dass die Realisierung einer Anweisung mit Rechtsschritten für alle Köpfe (k) immer noch mal einen Schritt zurück machen muss. Insgesamt also höchstens $t \cdot (2s' + 3k + 1)$ Schritte.
Bandinschriften werden immer nur länger. Jede der k Inschriften von M enthält ein Zeichen, das in dem gleichen Zeichen von M' kodiert ist (sozusagen die jeweils erste Zelle auf die die Köpfe zeigen.) Außerdem ist jede zu jeder Zeit höchstens s lang. Also ist jede Inschrift von M' zu jeder Zeit höchstens $2s + 2$ lang: Deshalb $s' \leq 2s + 2 = O(s)$. Weil jede Maschine nur so viele Zellen besuchen kann, wie sie Schritte macht, gilt $s \leq t$. Damit gilt für t':

$$t' \leq t \cdot (2s' + 3k + 1) + O(s) \leq tO(s) + O(s) = O(st) = O(t^2).$$

Aufgabe 12 [4+4+4 Punkte]

Betrachten Sie die Sprache $\text{Eq} = \{v\#w \mid L(M_v) = L(M_w)\}$. Zeigen Sie:
a) Das Halteproblem lässt sich auf Eq reduzieren.
b) Das Halteproblem lässt sich auch auf $\overline{\text{Eq}}$ reduzieren.
c) Weder Eq noch $\overline{\text{Eq}}$ sind rekursiv aufzählbar.

Hinweis: Betrachten Sie Kodierungen von drei Maschinen:
eine tut das Gleiche wie M_w bei Eingabe x, eine akzeptiert immer und eine nie.

Sei w_0 die Kodierung einer Maschine, die nichts akzeptiert, w_1 die Kodierung einer Maschine, die alles akzeptiert und $f(w,x)$ die Kodierung einer Maschine, die das Gleiche macht, wie M_w bei Eingabe x. So eine läßt sich aus w und x berechnen, da w nur eine Initialisierung vorgeschaltet werden braucht, die das erste Band löscht, dann x draufschreibt und den Kopf richtig positioniert. Die akzeptierte Sprache ist

$$L(M_{f(w,x)}) = \begin{cases} \emptyset = L(M_{w_0}), & \text{falls } x \in L(M_w) \\ \Sigma^* = L(M_{w_1}), & \text{sonst.} \end{cases}$$

Die Reduktion zu a) ist dann $w\#x \mapsto f(w,x)\#w_1$ (Wörter, die nicht von der Form $w\#x$ sind, werden auf $\varepsilon \notin \text{Eq}$ abgebildet). Die Reduktion zu b) ist $w\#x \mapsto f(w,x)\#w_0$ (andere Wörter als $w\#x$ genauso wie bei a). Beide Reduktionen sind berechenbar, weil f berechenbar und w_0 und w_1 konstant sind. Es gilt

$$\begin{aligned} w\#x \in H &\implies x \in L(M_w) \implies L(M_{f(w,x)}) = L(M_{w_1}) \neq L(M_{w_0}) \\ &\implies f(w,x)\#w_1 \in \text{Eq},\ f(w,x)\#w_0 \notin \text{Eq} \end{aligned}$$

und

$$\begin{aligned} w\#x \notin H &\implies x \notin L(M_w) \implies L(M_{f(w,x)}) = L(M_{w_0}) \neq L(M_{w_1}) \\ &\implies f(w,x)\#w_0 \in \text{Eq},\ f(w,x)\#w_1 \notin \text{Eq} \end{aligned}$$

Insbesondere also

$$f(w,x)\#w_1 \in \text{Eq} \iff w\#x \in H \iff f(w,x)\#w_0 \in \overline{\text{Eq}}.$$

Zu c): wäre $\overline{\text{Eq}}$ rekursiv aufzählbar, dann auch $\overline{H}$ nach a), Aufgabe 9 und Satz 2.3 der Vorlesung. Dann wäre aber nach Satz 1.2 der Vorlesung H entscheidbar, im Widerspruch zu einem Korollar zu Satz 2.2 der Vorlesung. Wäre Eq rekursiv, dann nach b), Aufgabe 9 und Satz 1.2 der Vorlesung auch $\overline{H}$. Gleicher Widerspruch wie oben.