TECHNISCHE UNIVERSITÄT CAROLO-WILHELMINA ZU BRAUNSCHWEIG
Institut für Theoretische Informatik Prof. Dr. D. Wätjen, Dr. J. Koslowski

Braunschweig, 2003-09-30

Vordiplomsklausur zur Vorlesung „Theoretische Informatik I/II“ im WS 2002/SS 2003

Aufgabe 1

Die Übergangsfunktion des EA $A = \langle\{z_0, z_1, z_2, z_3, z_4, z_5, z_6, z_7\}, \{a, b\}, \delta, z_0, \{z_3\}\rangle$ sei

δ	a	b
z_0	z_2	z_1
z_1	z_1	z_5
z_2	z_2	z_5
z_3	z_3	z_3
z_4	z_2	z_5
z_5	z_0	z_6
z_6	z_6	z_3
z_7	z_4	z_3

[17 PUNKTE] Konstruieren Sie einen zu A äquivalenten Minimalautomaten $A_{\min}$. Berechnen Sie dazu so viele Äquivalenzrelationen $\approx_i$ wie nötig, und geben Sie jeweils die Partitionen $X/\approx_i$ an. Skizzieren Sie den Zustandsüberführungsgraphen von $A_{\min}$. Wenn Sie aus $A_{\min}$ *ohne Anwendung des Kleene'schen Algorithmus* einen regulären Ausdruck für $L(A)$ direkt ablesen können, gibt es [6 SONDERPUNKTE].

Lösung: Die “Zeugen” minimaler Länge für die Erreichbarkeit der Zustände sind gegeben durch

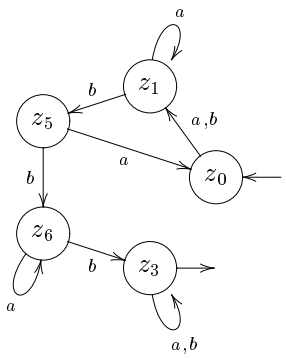
Zustand	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7
Zeuge	ε	b	a	$bbbb$	–	bb	bbb	–

Damit sind die Zustände z_4 und z_7 nicht erreichbar und werden entfernt. [4 PUNKTE]

Wir berechnen nun die zu den Äquivalenzrelationen $\approx_i$ gehörigen Partitionen [8 PUNKTE]:

	Partition von Q
$\approx_0$	$\{z_0, z_1, z_5, z_6, z_2\}, \{z_3\}$
$\approx_1$	$\{z_0, z_1, z_5, z_2\}, \{z_6\}, \{z_3\}$
$\approx_2$	$\{z_0, z_1, z_2\}, \{z_5\}, \{z_6\}, \{z_3\}$
$\approx_3$	$\{z_0\}, \{z_1, z_2\}, \{z_5\}, \{z_6\}, \{z_3\}$
$\approx_4$	$\{z_0\}, \{z_1, z_2\}, \{z_5\}, \{z_6\}, \{z_3\}$

Der resultierende EA $A_{\min}$ hat nach Identifikation von z_1 mit z_2 dann die Form [5 PUNKTE]



Daraus können wir sofort ablesen [6 SONDERPUNKTE]:

$$L(A) = ((a \cup b)a^*ba)^*\,(a \cup b)a^*bba^*b(a \cup b)^*$$

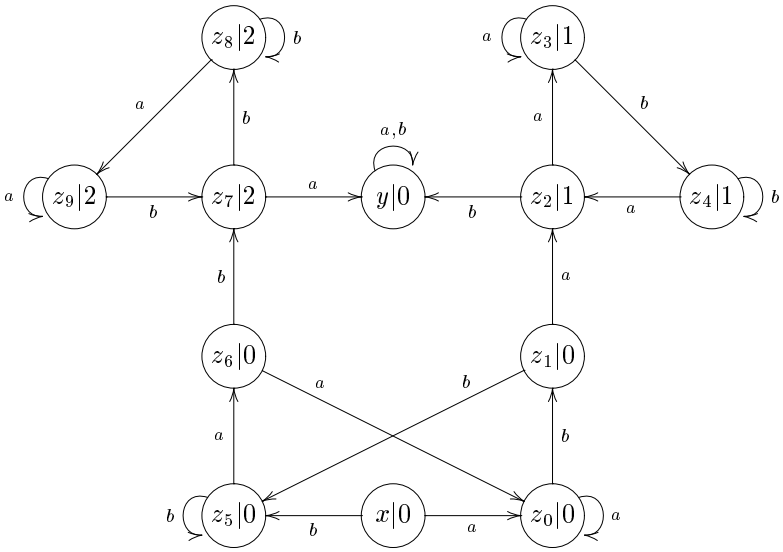
Aufgabe 2

[18 PUNKTE] Konstruieren Sie einen Moore-Automaten $M = (Z, \{a,b\}, \{0,1,2\}, \delta, \beta)$, der einen Zustand $x \in Z$ mit folgender Antwortfunktion besitzt:

$$M_x(w) = \begin{cases} 1 & \text{falls } aba \text{ Teilwort und } bab \text{ nicht Teilwort von } w \text{ ist;} \\ 2 & \text{falls } bab \text{ Teilwort und } aba \text{ nicht Teilwort von } w \text{ ist;} \\ 0 & \text{sonst} \end{cases}$$

Beweisen Sie, daß Ihr Automat das Gewünschte leistet. Hinweis: es sind weniger als 15 Zustände nötig.

Lösung: Ein Wort $w \in \{a,b\}^*$ enthält genau dann weder aba noch bab als Teilwort, wenn auf jedes Teilwort ab ein b und auf jedes Teilwort ba ein a folgt. Ausgehend vom Zustand x sind mit solchen Wörtern nur die Zustände z_0, z_1, z_5 sowie z_6 des folgenden Automaten erreichbar.



Tritt erstmals aba (bab) als Teilwort auf, so erreicht man z_2 (z_7). Tritt anschließend noch bab (aba) als Teilwort auf, so landet man im Zustand y und bleibt dort. Anderfalls kann man die Zustandsmenge $\{z_2, z_3, z_4\}$ ($\{z_7, z_8, z_9\}$) nicht verlassen.

Aufgabe 3

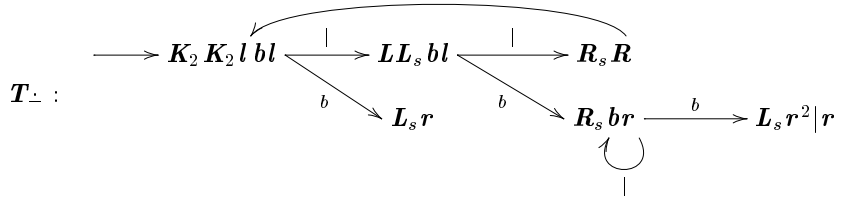
Die *monus-Operation* $\mathbb{N} \times \mathbb{N} \xrightarrow{\dot{-}} \mathbb{N}$ ist bekannterweise definiert durch

$$m \dot{-} n := \max\{0, m - n\}$$

[20 PUNKTE] Setzen Sie aus in der Vorlesung angegebenen einfachen Turingmaschinen eine Maschine $\boldsymbol{T}_{\dot{-}}$ zusammen, die die monus-Operation *normiert*(!) berechnet. Zur Verfügung stehen dabei die Rechtsmaschine $\boldsymbol{r}$, die große Rechtsmaschine $\boldsymbol{R}$, die Rechts-Suchmaschine $\boldsymbol{R}_s$, die entsprechenden Linksmaschinen $\boldsymbol{l}$, $\boldsymbol{L}$ sowie $\boldsymbol{L}_s$, die linke Translationsmaschine $\boldsymbol{T}_L$, die Verschiebemaschine $\boldsymbol{V}$, die m-Kopiermaschine $\boldsymbol{K}_m$, $m \in \mathbb{N}$, und die Schreibmaschinen $\boldsymbol{a}$, $a \in X$, sowie $\boldsymbol{b}$.

Geben Sie die jeweils erzeugten Folgen von Bandinhalten für die Berechnungen von $2 \dot{-} 3$ bzw $3 \dot{-} 2$ an, wobei Sie die Ergebnisse unverzweigter sequentieller Teilmaschinen zusammenfassen dürfen.

Lösung:



$2 \dot{-} 3$:	
Teilmaschine	Bandinschrift
	$b\|\|b\|\|\|\underline{b}bbbbbbbb$
$\boldsymbol{K_2 K_2}$	$b\|\|b\|\|\|b\|\|\|b\|\|\|\underline{b}$
$\boldsymbol{lbl}$	$b\|\|b\|\|\|b\|\|\|b\|\|\underline{b}b$
$\boldsymbol{LL_s bl}$	$b\|\|b\|\|\|b\|\underline{b}b\|\|\overline{b}b$
$\boldsymbol{R_s Rlbl}$	$b\|\|b\|\|\|b\|\overline{b}b\|\underline{b}bb$
$\boldsymbol{LL_s bl}$	$b\|\|b\|\|\|b\|bbb\|\overline{b}\|\|bb$
$\boldsymbol{R_s Rlbl}$	$b\|\|b\|\|\|b\overline{b}bb\|\underline{b}bbb$
$\boldsymbol{LL_s bl}$	$b\|\|b\|\|\|\underline{b}bbbb\|\overline{b}bbb$
$\boldsymbol{R_s br}$	$b\|\|b\|\|\|bbbbbb\underline{b}bbb$
$\boldsymbol{L_s r^2 \| r}$	$b\|\|b\|\|\|b\|\underline{b}bbbbbb$

$3 \dot{-} 2$:	
Teilmaschine	Bandinschrift
	$b\|\|\|b\|\|\underline{b}bbbbbbbb$
$\boldsymbol{K_2 K_2}$	$b\|\|\|b\|\|b\|\|\|b\|\|\underline{b}$
$\boldsymbol{lbl}$	$b\|\|\|b\|\|b\|\|\|b\|\underline{b}b$
$\boldsymbol{LL_s bl}$	$b\|\|\|b\|\|b\|\|\underline{b}b\|\overline{b}b$
$\boldsymbol{R_s Rlbl}$	$b\|\|\|b\|\|b\|\|\overline{b}b\|bbb$
$\boldsymbol{LL_s bl}$	$b\|\|\|b\|\|b\|\underline{b}bb\|\overline{b}bb$
$\boldsymbol{R_s Rlbl}$	$b\|\|\|b\|\|b\|\|bb\underline{b}bbb$
$\boldsymbol{L_s r}$	$b\|\|\|b\|\|b\|\|\underline{b}bbbbbb$

Aufgabe 4

Zeigen Sie unter Verwendung des

(a) [8 PUNKTE] μ-Operators, daß die Funktion $\mathbb{N}_0^2 \xrightarrow{f} \mathbb{N}_0$ μ-rekursiv ist:

$$f(m,n) = \begin{cases} \frac{3m}{4n}, & \text{falls } \frac{3m}{4n} \in \mathbb{N}_0; \\ \perp, & \text{sonst.} \end{cases}$$

(b) [6 PUNKTE] beschränkten μ-Operators, daß die Funktion $\mathbb{N}_0^2 \xrightarrow{f_0} \mathbb{N}_0$ primitiv rekursiv ist:

$$f_0(m,n) = \begin{cases} \frac{3m}{4n}, & \text{falls } \frac{3m}{4n} \in \mathbb{N}_0; \\ 0, & \text{sonst.} \end{cases}$$

(c) [4 PUNKTE] Ergebnisses in (b), daß die Funktion $\mathbb{N}_0^2 \xrightarrow{f_2} \mathbb{N}_0$ primitiv rekursiv ist:

$$f_2(m,n) = \begin{cases} \frac{3m}{4n}, & \text{falls } \frac{3m}{4n} \in \mathbb{N}_0; \\ 2, & \text{sonst.} \end{cases}$$

Lösung: Vergleiche Aufgabe 3 der Klausur von 2003-02-08. Allerdings ist hier keine dritte Wurzel erforderlich...

(a) Wir definieren die Funktion $\mathbb{N}_0^3 \xrightarrow{g} \mathbb{N}$ vermöge

$$g(m,n,x) := (3m \dot{-} 4nx) + (4nx \dot{-} 3m) \qquad [3 \text{ PUNKTE}]$$

Da g aus bekanntermaßen primitiv rekursiven Funktionen zusammengesetzt ist, sind sowohl g als auch das Prädikat

$$\rho(m,n,x) \quad :\Longleftrightarrow \quad g(m,n,x) = 0 \quad \Longleftrightarrow \quad 4nx = 3m \quad \Longleftrightarrow \quad x = \frac{3m}{4n}$$

primitiv rekursiv [3 PUNKTE]. Daraus ergibt sich (Definition 3.3.5)

$$f(m,n) = \mu x\, \rho(m,n,x) = \mu(g)(m,n)$$

was nach Definition 3.3.6 eine μ-rekursive Funktion ist [2 PUNKTE].

(b) Die primitiv rekursive Funktion $\mathbb{N}^3 \xrightarrow{\mu(\leq)(g)} \mathbb{N}$ bildet $(m,n,x) \in \mathbb{N}^3$ auf die kleinste Zahl $i \leq x$ ab, die $4ni = 3m$ erfüllt, bzw. auf 0, falls keine solche Zahl $i \leq x$ existiert [3 PUNKTE]. Da $i = 3m/4n$ sofort $i \leq 3m/4 < m$ impliziert, ergibt sich

$$f_0 = \mu(\leq)(g) \circ \left(U_1^{(2)}, U_2^{(2)}, U_1^{(2)}\right) \qquad [3 \text{ PUNKTE}]$$

(c) Achtung: da $f_0(m,n)$ auch für $m = 0$ den Wert 0 annimmt, ist es falsch, genau dann 2 zu $f_0(m,n)$ zu addieren, wenn $f_0(m,n) = 0$ gilt. Die Addition darf nur stattfinden, wenn außerdem $m \neq 0$ gilt. Dies läßt sich z.B. wie folgt formalisieren:

$$f_2 = f_0 + C_2^{(2)} * \overline{\text{sign}} \circ \left(f_0 + \overline{\text{sign}} \circ U_1^{(2)}\right) \qquad [4 \text{ PUNKTE}]$$

Aufgabe 5

Wir betrachten die Grammatik $G = (\{X_0, X_1, X_2\}, \{u, v\}, X_0, F)$ mit den Produktionen

$$\begin{array}{lll} 1. & X_0 \to uX_1u & 5. \quad X_2v \to vX_2 \\ 2.-4. & X_1 \to X_1v \mid v \mid uX_1X_2 & 6. \quad X_2u \to uu \end{array}$$

(a) [4 PUNKTE] Welche Typen hat diese Grammatik und welche nicht? Ist sie monoton?

(b) [14 PUNKTE] Bestimmen Sie $L(G)$ (mit Beweis).

(c) [10 PUNKTE] Finden Sie eine Typ-2 Grammatik, die $L(G)$ erzeugt (mit Beweis).

(d) [6 PUNKTE] Gibt es eine Typ-3 Grammatik, die $L(G)$ erzeugt? (Begründung!)

Lösung:

(a) Da es linke Seiten von Produktionen gibt, die mehr als ein Symbol enthalten, kann die Grammatik nicht vom Typ 3 oder 2 sein. Die Produktion Nr. 5 ist nicht kontextsensitiv, also ist die Grammatik vom Typ 0. Da ε nicht als rechte Seite einer Produktion vorkommt, läßt sich die Monotonie allein mittels des Längenvergleichs der linken und rechten Seiten überprüfen. Offenbar ist G monoton (und kann folglich mit Hilfe einer Typ-1 Grammatik erkannt werden).

(b) Die Produktionen Nr. 5 und 6 sorgen dafür, daß jedes im Verlauf einer Herleitung erzeugte X_2 mit einem rechts benachbarten v vertrauscht werden kann, solange, bis als rechter Nachbar ein u auftritt; dann wird X_2 ebenfalls in ein u verwandelt.

Die einzige Produktion, die ein X_2 erzeugt, ist Nr. 4, daher kann niemals ein X_1 rechts von einem X_2 auftauchen. Damit können rechts von einem X_2 nur die Symbole X_2, u und v auftreten. Da die erste Produktion am rechten Rand ein u erzeugt, kann nach der obigen Überlegung jedes X_2 in ein u umgewandelt werden, rechts davon können nur weitere u's auftreten.

Nach dieser Vorüberlegung behaupten wir: $L(G) = \{\, u^n v^m u^n \mid m, n \geq 1 \,\}$ [4 PUNKTE]

Beweis: Jedes derartige Wort kann hergeleitet werden. Nach der ersten Produktion wenden wir die vierte Produktion $(n-1)$-mal an, dann die zweite $(m-1)$-mal und die dritte einmal. Das resultiert in $u^n v^m X_1^{n-1} u$. Mittels $(n-1)$-maliger Anwendung der letzten Produktion erhalten wir nun $u^n v^m u^n$. [4 PUNKTE]

Umgekehrt muss in jeder terminierenden Herleitung die dritte Produktion genau einmal vorkommen. Vorher sind die zweite und die vierte Produktion anwendbar, hinterher nicht. Bei p-maliger Anwendung der zweiten und q-maliger Anwendung der vierten Produktion erhalten wir nach Anwendung der dritten Produktion ein Wort der Form $u^{q+1} W u$, wobei W das Symbol v p-mal und das Symbol X_2 q-mal enthält, in beliebiger Reihenfolge. Mit dem oben beschriebenen Mechanismus können dann sämtliche Symbole X_2 nach rechts getauscht und dort durch u's ersetzt werden, was in $u^{q+1} v^{p+1} u^{q+1}$ resultiert.

Die Rechtsvertauschung der X_2's und ihre Umwandlung in u's kann allerdings schon beginnen, bevor die dritte Produktion angewendet wird, weil kein X_1 rechts von einem X_2 auftauchen kann. Also liegt im Allgemeinen nach Anwendung der dritten Produktion ein Wort der Form $u^{q+1} W' u^r$ vor, wobei $r \leq q+1$ gilt und in W' $(p+1)$-mal das Symbol v und $(q+1-r)$-mal das Symbol X_2 vorkommt. [6 PUNKTE]

(c) Die Grammatik $G' = (\{X_0, X_1, X_2\}, \{u, v\}, X_0, F')$ habe die folgenden Produktionen [4 PUNKTE]

$$\begin{array}{ll} 1. & X_0 \to uX_1u \\ 2.-3. & X_1 \to uX_1u \mid X_2 \\ 4. & X_2 \to vX_2 \mid v \end{array}$$

Ein Wort der Form $u^n v^m u^n$ läßt sich herleiten, indem nach der ersten die zweite Produktion $(n-1)$-mal angewendet wird, dann die dritte, gefolgt von $(m-1)$ Anwendungen der vierten Produktion und abschließend der fünften. [3 PUNKTE]

Umgekehrt können die vierte und fünfte Produktion erst nach dritten angewendet werden. Die dritte und fünfte Produktion sind unverzichtbar, denn anders können die Nichtterminalsymbole nicht eliminiert werden. Damit sind die Exponenten n und m um 1 größer als die Anzahlen der Anwendungen der zweiten bzw. vierten Produktion. [3 PUNKTE]

(d) Behauptung: $L(G)$ ist nicht regulär. Anderfalls könnte $L(G)$ durch einen EA mit k Zuständen erzeugt werden. Bei Wörtern der Länge $> k$ treten also nach spätestens k Buchstaben Schleifen auf, die weggelassen oder vervielfältigt werden können, ohne die Akzeptanz zu beeinflussen. Insbesondere beim Wort $u^{k+1}vu^{k+1}$ muß die erste Schleife beginnen, bevor die ersten $k+1$ Symbole u erzeugt wurden. Sie darf aber nicht das Symbol v umfassen, da andernfalls durch Weglassen der Schleife ein Wort ohne v erzeugt werden könnte. Aber selbst wenn die Schleife nur u's erzeugt, entsteht durch ihr Weglassen ein Wort, das nicht zu $L(G)$ gehört, Widerspruch.

Damit kann $L(G)$ nicht von einer Typ-3 Grammatik erzeugt werden.

Aufgabe 6

[15 PUNKTE] Das E-Problem k-CLIQUE (mit fest vorgegebener Zahl k) hat als Eingabe einen ungerichteten Graphen $G = \langle V, E\rangle$. Zu entscheiden ist, ob G eine Clique der Größe k besitzt, d.h., eine Teilmenge $U \subseteq V$ von k Knoten, die alle miteinander verbunden sind.

Untersuchen Sie k-CLIQUE auf Zugehörigkeit zu **NP** und **P**.

Lösung: Eine Menge V mit n Elementen hat $\binom{n}{k}$ viele Teilmengen mit k Elementen. Dabei gilt

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n}{k} \cdot \frac{n-1}{k-1} \cdots \frac{n-k+1}{1} < n^k$$

Jede dieser Teilmengen ist darauf zu untersuchen, ob der induzierte Untergraph vollständig ist, dies erfordert jeweils $(k-1)^2$ viele Tests. Damit ist die Anzahl der benötigten Schritte durch ein Polynom in n beschränkt, und k-CLIQUE gehört zu **P**. Achtung: sobald die Zahl k Teil der Eingabe ist, mit der Größe $\log k$, ist der Ausdruck n^k doppelt exponentiell in $\log k$. Daher liegt kein Widerspruch zu dem bekannten Ergebnis vor, daß CLIQUE **NP**-vollständig ist.

Aufgabe 7

(a) [9 PUNKTE] Aus den Übungen wissen wir, daß die Relation der **FP**-Reduzierbarkeit reflexiv und transitiv, aber nicht notwendig antisymetrisch auf der Klasse aller E-Probleme ist.

Wie lassen sich von diesem Gesichtspunkt aus die **NP**-vollständigen Probleme charakterisieren? Erläutern sie stichpunktmäßig, wie man mit Hilfe eines bekanntermaßen **NP**-vollständigen E-Problems A versuchen kann, die **NP**-Vollständigkeit eines E-Problems B nachzuweisen.

(b) [23 PUNKTE] Das E-Problem INZIDENZ SYSTEM (IS) hat als Eingabe eine Menge S, eine endliche Familie von Teilmengen $S_j \subseteq S$, $j < n$, und eine ganze Zahl $k \leq |S|$. Zu entscheiden ist, ob es eine Teilmenge $T \subseteq S$ mit höchstens k Elementen gibt, die jede der Mengen S_j nichtleer schneidet.

Überlegen Sie sich zunächst eine vernünftige binäre Codierung von Instanzen dieses Problems als Eingabe für eine Turingmaschine.

Beweisen Sie dann detailliert die **NP**-Vollständigkeit dieses Problems.

[Hinweis: das aus den Hausaufgaben bekannte E-Problem KNOTENÜBERDECKUNG (KÜ) ist **NP**-vollständig. Für die Eingabe $\langle G, m\rangle$, bestehend aus einem ungerichteten Graphen $G = \langle V, E\rangle$ und einer Zahl $m \in \mathbb{N}$, ist zu entscheiden, ob eine höchstens m-elementige Knotenmenge $U \subseteq V$ existiert, so daß mindestens ein Endpunkt jeder Kante in E zu U gehört.]

Lösung:

(a) Die **NP**-vollständigen E-Probleme sind gerade die *maximalen Elemente* der Klasse **NP** in der Klasse aller E-Probleme bzgl. der durch **FP**-Reduktion gegebenen Relation. [3 PUNKTE]

Die **NP**-Vollständigkeit von B ist also äquivalent zu

1. $A \leq B$, was wir vermöge einer **FP**-Reduktion von A nach B zeigen können; [3 PUNKTE]

2. $B \leq A$, was wir vermöge einer **FP**-Reduktion von B nach A zeigen können, oder alternativ direkt durch den Nachweis von $B \in$ **NP**, denn A ist eine obere Schranke von **NP**. [3 PUNKTE]

(b) Zunächst ist zu klären, wie die Eingabe für (IS) codiert werden soll. Wir wählen als Eingabe eine Binärzahl $s \in \mathbb{N}$, zu intepretieren als die Größe der Menge S, gefolgt von der Binärzahl k, gefolgt von n binären s-Tupeln zur Bestimmung der Teilmengen. Diese Werte sind jeweils durch ein Blankzeichen getrennt.[3 PUNKTE]

Um nachzuweisen, daß (IS) **NP**-vollständig ist, geben wir zunächst eine Reduktion von (KÜ) auf (IS) an: einer Instanz $(\langle V, E\rangle, k)$ von (KÜ) ordnen wir die Instanz (V, E, k) von (IS) zu, mit Grundmenge V, durch die Kanten von G bestimmten 2-elementigen Teilmengen $\{u, v\} \subseteq V$ mit $\{u, v\} \in E$, und derselben Zahl k. [4 PUNKTE]

Korrektheit der Reduktion: Jede Knotenüberdeckung $W \subseteq V$ der Größe k von G schneidet jede Kante $\{u, v\} \in E$ nichtleer. Umgekehrt ist jede k-elementige Menge $U \subseteq V$, die einen Endpunkt jeder Kante enthält, eine Knotenüberdeckung. [4 PUNKTE]

Die Reduktion gehört zu **FP**: Die Reduktion extrahiert aus dem oberen Dreieck der Adjazenzmatix für G die Zahl $s = |V|$, und schreibt dann für jede 1 in dieser Matrix ein s-Tupel mit zwei Einsen an entsprechenden Koordinaten, und Nullen sonst. Schließlich wird k kopiert. Dazu sind größenordnungsmäßig $O(|V|(|E|+1)+\log k)$ viele Schritte erforderlich. Dies ist polynomial in den Eingabegrößen $|V|$, $|E|$ und $\log k$. [6 PUNKTE]

Es bleibt zu zeigen, daß (IS) zu **NP** gehört. Aber dieses Problem läßt sich mit einer 2-Band NTM M lösen: sie schreibt zufällig eine Teilmenge von S in Form eines binären s-Tupels auf Band 2, was s Schritte erfordert. Dann testet sie in s weiteren Schritten, ob dieses Zufallstupel höchstens k Einsen enthält, andernfalls hält die Maschine, ohne zu akzeptieren. Schließlich wird jedes binäre s-Tupel der Eingabe mit dem Zufallstupel verglichen, ob an mindestens einer Stelle eine Übereinstimmung besteht. Genau in diesem Fall akzeptiert die Maschine. Dies erfordert maximal weitere $n \cdot s$ viele Vergleiche, sowie Kopfbewegungen in der gleichen Größenordnung. Bei einer Eingabegröße von $\log s + n \cdot s + \log k$ ergibt sich ein Zeitaufwand von $O(n \cdot s)$, was linear in der Eingabegröße ist. [6 PUNKTE]

Aufgabe 8

[20 PUNKTE] Schreiben Sie ein Programm, daß in polylogarithmischer Zeit arbeitet und für die Eingabe der positiven Zahlen

$$z_1, z_2, \ldots, z_n, p, q$$

mit $p > q$ als Ausgabe die gemäß $\geq$ geordnete Liste aller Zahlen z_i liefert, die echt zwischen p und q liegen. Beschreiben Sie das benutzte PRAM-Model ausführlich, einschließlich Zeitkomplexität, verrichteter Arbeit und Anzahl der benutzten Prozessoren.

Lösung: Der Algorithmus 10.2.5 aus dem Skript kann nahezu unverändert übernommen werden:
Eingabe: n unsortierte Zahlen $z_i = A[i]$, $i < n$;
Ausgabe: $k \leq n$ sortierte Zahlen $p > S[0] \geq S[1] \geq \ldots \geq S[k-1] > q$;
Modell: common CRCW PRAM mit n^2 Prozessoren;
Hilfsgrößen: Arrays $a[i,j]$, $\bar{a}[i,j]$, $b[j]$ und $\bar{b}[j]$ mit $i, j < k$

```
for i < n pardo
    if A[i] ≥ p or q ≥ A[i] then A[i] := 0 fi
    odrap
for (i,j) < (n,n) pardo
    if A[i] ≥ A[j] then a[i,j] := 1 else a[i,j] := 0 fi
    if A[i] > A[j] then ā[i,j] := 1 else ā[i,j] := 0 fi
    odrap
for j < n pardo
    b_j := Σ_{i<n} a[i,j]
    b̄_j := Σ_{i<n} ā[i,j]
    odrap
for i < n pardo
    for b̄_i ≤ j < b_i pardo
        S[j] := A[i]
        odrap
    odrap
```

Zunächst werden die Zahlen außerhalb des Intervalls $]q, p[$ in $O(1)$ Zeit auf den Wert 0 gesetzt, mit $O(n)$ Arbeit. Dann werden die Größenvergleiche bzgl. $\geq$ bzw. $>$ der modifizierten Einträge von $\boldsymbol{A}$ in $O(1)$ Zeit in den Arrays $\boldsymbol{a}$ sowie $\bar{\boldsymbol{a}}$ eingetragen, was $O(n^2)$ Arbeit erfordert. Die n Spaltensummen $b[j]$ und $\bar{b}[j]$ dieser Hilfsmatizen lassen sich jeweils in $O(\log n)$ Zeit mittels n Prozessoren bestimmen (gemeinsamer Lesezugriff), mit $O(n)$ Arbeit, und beschreiben die Anzahl der modifizierten Werte $A[i] \geq A[j]$ bzw. $A[i] > A[j]$. Sie legen damit die Positionen im Array S fest, in denen die Zahl $A[j]$ auftreten

muss. Diese werden in der letzten Doppelschleife eingetragen, evtl. mit mehrfachem Schreibzugriff, aber dann immer mit demselben Wert. Somit erfordert die Doppelschleife nur $O(1)$ viel Zeit, leistet aber wieder $O(n^2)$ Arbeit.

Insgesamt beträgt der Zeitaufwand $O(\log n)$ und die Arbeit $O(n^2)$.

Schlußendlich können die Einträge von S mit Wert 0 entfernt werden, sie treten ohnehin am Ende auf.

Es scheint möglich zu sein, mit n Prozessoren eine Laufzeit von $O((\log n)^2)$ zu realisieren.
