Mächtigkeit und Komplexität von Berechnungen mit der ganzzahligen Division

Dissertation

zur Erlangung des akademischen Grades

eines Doktors der Naturwissenschaften

dem Fakultätsrat der Fakultät

Elektrotechnik, Informatik und Mathematik

der Universität Paderborn vorgelegt von

Katharina Lürwer-Brüggemeier

Eingereicht: 15.09.2008

Erster Gutachter: Prof. Dr. Friedhelm Meyer auf der Heide

Zweiter Gutachter: Privatdozent Dr. Martin Ziegler

Inhaltsverzeichnis

1 Einleitung

In dieser Arbeit wird die Mächtigkeit, d.h. was kann überhaupt berechnet werden, und die Komplexität (d.h. wie schnell können die Berechnungen durchgeführt werden) über verschiedene Operationsmengen $S \subseteq \{+, -, *, ...\}$ mit der Eingabemenge $\mathbb{Z}^n$ betrachtet. Sowohl Berechnungsmächtigkeit als auch Komplexität hängen stark von dem zugrundeliegenden Rechenmodell ab.

Die Turingmaschine als uniformes Rechenmodell wird allgemein als das geeignete Modell für diese Art der Betrachtungen angesehen. Sie berechnet eine Funktion $f : A^* \to A^*$ für ein endliches Alphabet A, i.a $A = \{0, 1\}$. Bei ihr werden die Kosten bitweise berechnet, als eine Funktion $T : \mathbb{N} \to \mathbb{N}$ mit $T(n)$ der Worst Case über alle Eingaben der Länge n.

Aber gerade bei der Entwicklung von Algorithmen zur Ableitung oberer Schranken und insbesondere beim Beweis unterer Schranken bedient man sich häufig algebraischer Modelle, wie der uniformen Registermaschine RAM, die auf den ganzen Zahlen (oder $\mathbb{R}$ oder $\mathbb{Q}$) mit dem Einheitskostenmaß und nicht bitweise operiert, d.h. bei ihr werden Funktionen $f : \mathbb{Z}^* \to \mathbb{Z}^*$(bzw. $f : \mathbb{R}^* \to \mathbb{R}^*$ oder $f : \mathbb{Q}^* \to \mathbb{Q}^*$) berechnet und die Komplexität $T(n)$ wird als eine Funktion $T : \mathbb{N} \to \mathbb{N}$ als Worst Case über alle Eingaben aus $\mathbb{Z}^n$ (bzw. $\mathbb{Q}^n$ oder $\mathbb{R}^n$) berechnet. Die Durchführung einer Operation entspricht in diesem Modell einem Zeitschritt und die Laufzeit wird nur in Abhängigkeit von der Dimension der Eingabe und nicht von der Bitlänge bestimmt.

Die Berechenbarkeit und auch die Komplexität solch einer Registermaschine hängt stark von der Wahl der Grundoperationen ab, z.B. Inkrementation, Addition, Subtraktion, Multiplikation, Vergleiche „=" oder „<", ganzzahlige Division, bitweise Konjunktion „&", Shifts „$x \leftarrow y = x \cdot 2^y$" und „$x \rightarrow y = x \,\mathrm{DIV}\, 2^y$", indirekte Adressierung usw.. Die bitweise Konjunktion und die ganzzahlige Division (wenn der Dividend kein Vielfaches des Divisors ist) gehören zwar nicht zu den klassischen Grundoperationen, werden aber von digitalen Computern unterstützt.

Wie sehr die Wahl der Operationsmenge die Mächtigkeit bestimmt, zeigt sich darin, dass für RAMs mit der Eingabemenge $\{0, 1\}^*$ eine Berechnung mit der

Operationsmenge $\{+,-,*,=\}$ trotz exponentiell langer Zwischenergebnisse in RP [39] simuliert werden kann, dagegen Berechnungen mit der Operationsmenge $\{+,-,\mathrm{DIV},=\}$ bereits ganz NP abdecken [39] und mit $\{+,-,*,\&,=\}$ sogar PSPACE [34], vergleiche auch [4,3] und [41].

Um untere Schranken zu beweisen, werden statt der uniformen Rechenmodelle häufig die entsprechenden nichtuniformen Modelle betrachtet, d.h. es werden nicht Eingaben beliebiger Länge, sondern n-stellige Eingaben betrachtet. Eine nichtuniforme RAM über den ganzen Zahlen bestimmt für eine Eingabe $\bar{x}\in \mathbb{Z}^n$ in Abhängigkeit von n, welche RAM M_n ihr Programm auf $\bar{x}\in \mathbb{Z}^n$ ausführt.

Alle bekannten unteren Schranken für uniforme RAMs über $\mathbb{Z}^n$ gelten auch für nichtuniforme RAMs und es gibt bisher keine unteren Schranken für uniforme RAMs, die explizit die Uniformität ausnutzen, sich also von unteren Schranken für nichtuniforme RAMs unterscheiden. Als adäquates nichtuniformes Rechenmodell zum Beweis unterer Schranken dient der S- Berechnungsbaum, kurz: S-CT (CT = computation tree). Die Berechnungsmächtigkeit für n-stellige Eingaben bei S-RAMs und S-CTs ist identisch, jedoch sind die Komplexitäten unterschiedlich. Da die Registermaschine zusätzlich die Möglichkeit der indirekten Adressierung hat, hat der S-Berechnungsbaum zur Simulation einer nichtuniformen S-RAM einen zusätzlichen logarithmischen Faktor [31, Lemma 1].

Ziel dieser Arbeit ist es, die Berechnungsmächtigkeit und die Komplexität von S-Berechnungsbäumen mit der Eingabemenge $\mathbb{Z}^n$ für $n>1$ für verschiedene Operationsmengen $S \subseteq \{+,-,*,...\}$, die die ganzzahlige Division oder die ganzzahlige Division mit Konstanten enthalten, zu untersuchen und Algorithmen zu entwickeln, die durch die Hinzunahme der ganzzahligen Division und auch weiterer nicht klassischer Operationen wie der bitweisen Konjunktion und dem größten gemeinsamen Teiler mit dem Einheitskostenmaß beschleunigt werden.

1.1 Die Berechnungsmodelle

Ein *(S,C)-Berechnungsbaum*, kurz ein (S, C)-CT, mit der Operationsmenge $S \subseteq \{+, -, *, *_c, \mathrm{DIV}_c, \mathrm{DIV}\}$ und Konstanten aus der Menge C, $\{1\} \subseteq C \subseteq \mathbb{Q}$, für Eingaben $x_1, \ldots, x_n$ ist ein endlicher binärer Baum. „$*_c$, DIV_c" bezeichnen die Multiplikation und die Division, bei denen ein Faktor oder der Divisor konstant ist, d.h. er hängt also nicht von den Eingabewerten ab.

- Knoten v vom Grad 1 berechnen eine Funktion $g_v : \mathbb{Q}^n \to \mathbb{Q}$. g_v ist entweder x_i für ein $i \in \{1, \ldots, n\}$ oder g_v ist c für ein $c \in C$ oder g_v ist von der Form $g_{v_1} op\, g_{v_2}$ mit v_1, v_2 Knoten auf dem Weg von der Wurzel zu v, und $op \in S$.
- Knoten v vom Grad 2, die Verzweigungen, sind mit Vergleichen „$g(x_1, \ldots, x_n) > 0$" für eine Funktion g, die auf dem Weg zum Knoten v berechnet wurde, beschriftet.
- Knoten vom Grad 0 sind die Blätter. Sie sind mit „akzeptiere" oder „verwerfe" beschriftet.

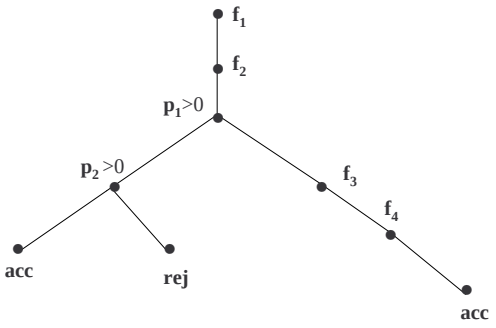

Abbildung 1. Berechnungsbaum

Eine Eingabe $\overline{x} = (x_1, \ldots, x_n) \in \mathbb{Z}^n$ folgt einem Weg von der Wurzel zu einem Blatt. An einem Vezweigungsknoten v folgt sie dem linken Zweig, falls „$g(x_1, \ldots, x_n) > 0$" wahr ist, sonst dem rechten Zweig. Die Eingabe $\overline{x}$ wird akzeptiert, falls sie zu einem mit „akzeptiere" beschrifteten Blatt gelangt.

Die Menge der Eingaben, die zu den mit „akzeptiere" beschrifteten Blättern gelangen, ist die von dem Berechnungsbaum erkannte Sprache $L \subseteq \mathbb{Z}^n$.

Die *Komplexität* eines S- Berechnungsbaumes ist seine Tiefe.

Werden zur Tiefe nur die Verzweigungsknoten gerechnet, spricht man von der *Verzweigungstiefe* des Baumes.

Der *Grad* eines $(\{+,-,*\},C)$-CT ist der maximale Grad der Polynome, die in seinen Knoten berechnet werden.

Die Berechnungen entlang eines Pfades im S- Berechnungsbaum, d.h. ohne Verzweigungen, werden als *Straight-Line- Programm,* kurz SLP bezeichnet.

Die Familie $CC_n(S)$ der Sprachen $L \subseteq \mathbb{Z}^n$, die von einem Berechnungsbaum mit der Operationsmenge S erkannt werden, bezeichnet die *Berechnungsmächtigkeit* einer Operationsmenge S für n- dimensionale Eingaben.

Bemerkung 1 *Da angenommen wird, dass* $\{+,-\} \subseteq S$ *gilt, ist* $CC_n(S)$ *unabhängig von der Wahl von* C, $\{1\} \subseteq C \subseteq \mathbb{Q}$. *Daher schreiben wir* S*-CTs statt* (S,C)*-CTs, falls die Wahl von* C *nicht von Bedeutung ist.*

Die Sprachen, die ein S- Berechnungsbaum für eine Operationsmenge S erkennt, lassen sich folgendermaßen charakterisieren.

Eine Funktion $f : \mathbb{Z}^n \to \mathbb{Z}$ wird als S*-Funktion* bezeichnet, falls sie durch ein Straight- Line- Programm mit Operationen aus S und Konstanten aus $\mathbb{Q}$ berechnet werden kann.

Bemerkung 2 *Eine Sprache* $L \subseteq \mathbb{Z}^n$ *kann genau dann durch einen* S*-CT entschieden werden, wenn* L *eine Boolsche Kombination endlich vieler Mengen* $\{\overline{x} \in \mathbb{Z}^n,\ f(\overline{x}) > 0\}$ *für* S*-Funktionen* f *ist.*

Dies ist bei wohlbekannten S-Funktionen eine hinreichende Charakterisierung. Dies ist im Allgemeinen der Fall für $S = \{+,-,*_c\}$, $S = \{+,-,*\}$, $S = \{+,-,*,/\}$ (die S-Funktionen sind lineare Funktionen, Polynome, rationale Funktionen).

In den Aufsätzen von David DOBKIN und Richard J. LIPTON [16] und Michael BEN-OR [12] werden Argumente aus der algebraischen Geometrie benutzt, um untere Schranken für $\{+,-,*,/\}$-CTs mit rationalen oder reellen Eingaben

zu beweisen. Die Beweise basieren auf der Zahl der Zusammenhangskomponenten der Sprache, die erkannt werden soll. Dies führt zu einer unteren Schranke von z.B. $\Omega(n^2)$ für das Rucksackproblem (knapsack problem). In dem Aufsatz von Andrew YAO [42] werden diese Schranken auf eine große Sprachklasse mit ganzzahligen Eingaben übertragen, in [33], [23] und [30] werden diese Schranken auf Registermaschinen, bei denen auch die indirekte Adressierung zulässig ist, für den Fall $S = \{+, -\}$ übertragen.

Falls DIV oder DIV_c in S ist, ist über die S-Funktionen viel weniger bekannt, so dass Charakterisierungen der Sprachklassen und der Komplexitäten schwieriger sind.

In [2] wird ein sehr allgemeines Ergebnis für ein noch strengeres Modell, in dem DIV und andere analytische Funktionen mit konstanten Kosten berechnet werden, vorgestellt, nämlich dass im Allgemeinen die ganzzahlige lineare Programmierung mit n Variablen und m Ungleichungen nicht in einer nur von n und m abhängigen Zeit, (aber nicht von der binären Eingabelänge) in diesem Modell berechnet werden kann. Das gilt auch für die Berechnung des größten gemeinsamen Teilers ggT mit der Operationsmenge $\{+, -, *, \text{DIV}\}$ in [27].

Aus technischen Gründen, um die Theoreme 2 und 4 zu beweisen, wird noch der etwas künstlich anmutende Modulo-Verzweigungs- Baum (kurz: MBT=modulo branching tree) eingeführt.

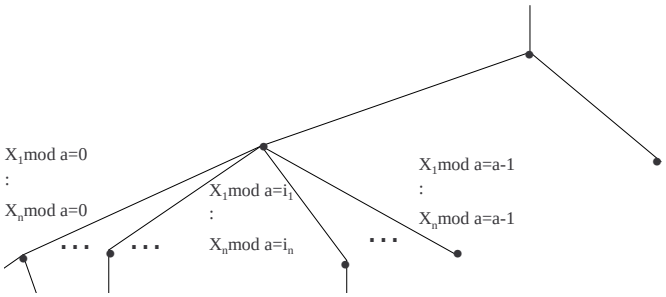

Abbildung 2. Modulo-Verzweigungs-Baum

Ein *Modulo-Verzweigungs-Baum* ist ein $\{+, -, *\}$- bzw. $\{+, -, *_c\}$- Berechnungsbaum, der zu einer Eingabe $\overline{x} = (x_1, \ldots, x_n) \in \mathbb{Z}^n$ zusätzliche Verzwei-

gungsknoten von beliebigem endlichem Grad enthält. Gelangt die Eingabe $\overline{x}$ zu einem Knoten vom Grad a^n, folgt die Eingabe $\overline{x}$ genau dann dem $\overline{i}$-ten Zweig, falls $x_j \bmod a = i_j$ für $\overline{i} = (i_1, \ldots, i_n) \in \{0, \ldots, a-1\}^n, j = 1, .., n$ gilt. Eine Eingabe $\overline{x} \in \mathbb{Z}^n$ wird akzeptiert, falls der zugehörige Pfad in einem akzeptierenden Blatt endet. Die *Komplexität* eines Modulo-Verzweigungs-Baumes ist seine Tiefe.

1.2 Überblick

Da bei den klassischen Programmiersprachen die häufigsten Operationen auf den ganzen Zahlen „$+, -, *, *_c, \mathrm{DIV}, \mathrm{DIV}_c$" sind, wird im ersten Teil dieser Arbeit die Berechnungsmächtigkeit und die Komplexität von Berechnungsbäumen mit den obigen Operationen und Verzweigungen betrachtet. DIV bezeichnet die ganzzahlige Division, DIV_c die ganzzahlige Division durch Konstanten und $*_c$ die Multiplikation mit Konstanten. Für die Operationsmenge $\{+, -, *, /\}$ ohne ganzzahlige Division sind die Berechnungsmächtigkeit und die Komplexitäten weitestgehend bekannt, da entlang der Pfade im Berechnungsbaum je nach Wahl der Operationsmenge lineare Funktionen, Polynome oder rationale Funktionen berechnet werden. Es werden für die Komplexitätsbetrachtungen Methoden aus der algebraischen Geometrie über Zusammenhangskomponenten im Reellen auf diskrete Mengen übertragen [42].

Dies ist bei Hinzunahme der ganzzahligen Division nicht mehr möglich.

Die Charakterisierungen mit den daraus abgeleiteten Schranken für Berechnungsbäume mit einer Operationsmenge $S \subseteq \{+, -, *, *_c, \mathrm{DIV}_c, \mathrm{DIV}\}$ werden im nächsten Kapitel vorgestellt. Um den Zusammenhang zwischen den einzelnen Theoremen, Sätzen, Korollaren usw. zu verdeutlichen und die Lesbarkeit zu erleichtern, werden die Beweise, da sie sehr technisch und aufwändig sind, in den beiden folgenden Kapiteln für den eindimensionalen und mehrdimensionalen Fall geführt.

Für den eindimensionalen Fall ist in [22] eine vollständige Charakterisierung mit den daraus abgeleiteten Schranken gezeigt worden. Die Berechenbarkeit und die Komplexität von Berechnungsbäumen mit einer Operationsmenge $S \subseteq$

$\{+, -, *, *_c, \mathrm{DIV}_c, \mathrm{DIV}\}$ im eindimensionalen Fall werden im ersten Teil des nächsten Kapitels vorgestellt. Dieser Abschnitt folgt weitestgehend den Ausführungen in [22].

Im zweiten Teil des nächsten Kapitels werden für den mehrdimensionalen Fall bei Hinzunahme der ganzzahligen Division durch Konstanten eine vollständige und für die allgemeine ganzzahlige Division eine partielle Charakterisierung angegeben. Aus diesen Charakterisierungen werden Schranken abgeleitet und Sprachklassen unterschieden. Dies sind die ersten neuen Ergebnisse. Es wird die Berechenbarkeit im mehrdimensionalen Fall für die ganzzahlige Division durch Konstanten vollständig charakterisiert und es werden daraus Schranken, die ohne ganzzahlige Division bewiesen wurden, übertragen. Bei der allgemeinen ganzzahligen Division wird nicht wie im eindimensionalen Fall eine vollständige Charakterisierung der Sprachklassen angegeben, sondern sie werden nur teilweise charakterisiert. Aber aus diesen partiellen Charakterisierungen werden wiederum untere Schranken abgeleitet, im Fall der mächtigsten Operationsmenge $S = \{+, -, *, \mathrm{DIV}\}$ sogar die erste untere Schranke bei Konstanten aus $\mathbb{Q}$. Bis dahin waren nur untere Schranken bei der Konstantenmenge $C = \{0; 1\}$ bekannt. Über diese Charakterisierungen werden die Beziehungen der Sprachklassen $CC_n(S)$ für n>1 und Teilmengen $S \subseteq \{+, -, *, *_c, \mathrm{DIV}_c, \mathrm{DIV}\}$ vollständig bewiesen.

Die Beweise zu den Charakterisierungen und den Schranken im Fall n=1 folgen in Kapitel 3 dem Aufsatz von Friedhelm MEYER AUF DER HEIDE u.a. in [22] mit Ausnahme von dem Beweis zu Lemma 3, der den Ausführungen von Joao MEIDANIS in [29] folgt.

In Kapitel 4 werden die Beweise zu den Charakterisierungen und den Schranken im Fall n> 1 geführt und außerdem werden noch Sprachen angegeben, die die Sprachklassen unterscheiden, die im eindimensionalen Fall zusammenfallen.

Aus der ersten unteren Schranke für die Operationsmenge $S = \{+, -, *, \mathrm{DIV}\}$ bei Konstanten aus $\mathbb{Q}$ in diesem Kapitel kann gefolgert werden, dass der Algorithmus von Nader BSHOUTY [6] zur Polynomauswertung über einer endlichen Menge in $\mathbb{N}$ nicht über ganz $\mathbb{N}$ konstant sein kann. Dieser Algorithmus

wertet ein univariates Polynom mit ganzzahligen Koeffizienten über einem endlichen Eingabebereich in 15 Schritten aus, d.h. unabhängig vom Grad des Polynoms und von der Eingabe, aber durch Nutzung sehr großer Konstanten im Verhältnis zum Grad und zur Eingabe.

Eine genauere Betrachtung der Komplexität der Polynomauswertung wird in Kapitel 5 vorgenommen. Bekannt ist, dass es irreduzible Polynome vom Grad d gibt, die über $\{+, -, *\}$ $\Omega(d)$ Schritte zur Auswertung benötigen [8, Theorem 6.5]. Es wird gezeigt, dass bei ganzzahligen Polynomen mit kleinen Koeffizienten sich die Auswertung über $\{+, -, *\}$ beschleunigen lässt. Über einem endlichen Bereich können Polynome in konstant vielen Schritten unabhängig von dem Grad des Polynoms ausgewertet werden. Der von Nader BSHOUTY geführte Beweis für univariate Polynome wird auf multivariate Polynome übertragen.

Die Polynomauswertung über $\mathbb{Z}^n$ kann durch die Hinzunahme der bitweisen Konjunktion „&" als weitere Operation beschleunigt werden. Mit dieser neuen Operationsmenge $\{+, -, *, \text{DIV}, \&\}$ kann jedes Polynom über $\mathbb{Z}$ in einer Variablen mit einer leichten (doppellogarithmischen) Abhängigkeit von der Eingabe unabhängig vom Grad des Polynoms oder aber mit einer logarithmischen Abhängigkeit vom Grad berechnet werden. Bei Polynomen mit mehreren Variablen kommt die Anzahl der Variablen als Faktor beim Aufwand hinzu. Wird zusätzlich eine im Verhältnis zur Eingabe sehr große ganze Zahl eingegeben, so kann die Berechnung auf die Quadratwurzel der Laufzeit ohne diese Eingabe im univariaten Fall beschleunigt werden. Im multivariaten Fall benötigt man wiederum die Anzahl der Variablen als zusätzlichen Faktor.

Ohne „&" bleibt die doppellogarithmische Lücke zwischen der oberen Schranke $\mathcal{O}(d)$ und der unteren Schranke $\Omega(log\, log\, d)$. Daher stellt sich die Frage, ob jedes Polynom vom Grad d über $\{+, -, *, \text{DIV}\}$ in $o(d)$ Schritten berechnet werden kann. Die Antwort ist positiv, falls die Reihe $\sum_{n=0}^{\infty} 2^{-dn^2}$ algebraisch vom Grad kleiner d ist. Dies ist aber ein in der Zahlentheorie ungelöstes Problem.

In Kapitel 6 werden Anwendungen in der Linearen Algebra betrachtet. Dazu zählen die Matrixmultiplikation und die Berechnung der Determinante, die

beide mit der ganzzahligen Division (aber ohne bitweise Konjunktion) eine optimale quadratische Laufzeit haben. Das klassische Verfahren zur Matrixmultiplikation benötigt kubische Laufzeit. Eingeleitet von Volker STRASSEN wurde die Suche nach schnelleren Verfahren mit dem derzeitigen Rekord von $\mathcal{O}(\mathrm{n}^{\omega})$ mit $\omega <$ 2,38, aufgestellt von Don COPPERSMITH und Shmuel WINOGRAD, siehe [8, Abschnitt 15]. In diesem Modell werden mit dem Einheitskostenmaß die arithmetischen Operationen $\{+, -, *\}$ benutzt, aber auch die Hinzunahme der Division kann bewiesenermaßen vergl. [8, Theorem 7.1] die Laufzeit nicht verbessern. Wird aber nun die ganzzahlige Division als weitere Operation hinzugenommen, so wird gezeigt, dass die Matrixmultiplikation über $\mathbb{Z}$ quadratische Laufzeit hat.

Über den Operationen $(+, -, *)$ sind die asymptotischen Komplexitäten der Matrixmultiplikation und der Determinantenberechnung beliebig nahe beieinander [8, Abschnitt 16.4], wenn auch, wie oben erwähnt, nicht bekannt ist, wo zwischen $\mathcal{O}(\mathrm{n}^2)$ und $\mathcal{O}(\mathrm{n}^{\omega})$ mit $\omega <$ 2,38 die genaue Komplexität liegt. Es wird gezeigt, dass dieser Zusammenhang zwischen den Komplexitäten von Matrixmultiplikation und Determinantenberechnung auch gilt, wenn die ganzzahlige Division hinzugenommen wird. Dieser Zusammenhang wird nicht durch Reduktion, sondern durch die Angabe expliziter Algorithmen für beide Probleme bewiesen. Der Algorithmus zur Matrixmultiplikation nutzt eine geschickte Kodierung der zu multiplizierenden Matrizen und Dekodierung der Produktmatrix aus.

Die Determinante einer $n \times n$ Matrix A kann relativ einfach in Polynomialzeit $\mathcal{O}(n^3)$ mithilfe des Gaußschen Eliminationsverfahren berechnet werden. Solch eine einfache Berechnung der Permanente in Polynomialzeit ist nicht bekannt. Dieses Problem ist VALIANTNP-vollständig in diesem algebraischen Modell [8, Theorem 21.17] (und sogar #P-vollständig im Bitmodell). Wird jedoch die ganzzahlige Division als weitere Operation hinzugenommen, kann auch die Permanente sogar in quasi-optimaler Polynomialzeit $\mathcal{O}(n^2)$ berechnet werden [1, Proposition 2.4]. Die Tatsache, dass die Permanente in quadratischer Laufzeit berechnet werden kann, wird zur Algorithmenentwicklung für die Determinantenberechnung mit derselben Laufzeit verwendet.

Durch das k-fache wiederholte Quadrieren einer $n \times n$ Matrix erhält man die 2^k-fache Potenz der Matrix und man benötigt also die k-fache quadratische Laufzeit $\mathcal{O}(k \cdot n^2)$. Bei der Hinzunahme des größten gemeinsamen Teilers als weitere Operation und zusätzlicher Eingabe einer Matrix B mit großen, aber nicht zu großen Einträgen lässt sich die 2^k-fache Potenz einer Matrix mit dem $\sqrt{k}-$fachen Aufwand der Matrixmultiplikation durchführen.

Im folgenden Abschnitt wird gezeigt, dass es unendlich viele solcher Matrizen B gibt, die die dort geforderten Eigenschaften für den größten gemeinsamen Teiler der Differenzen dieser Matrix B mit allen Matrizen, deren Einträge kleiner als eine von k und n abhängige Konstante sind, erfüllen. Es wird eine obere Schranke für die Größe der Matrizeneinträge und eine obere Schranke für den Aufwand zur Bildung solch einer Matrix B angegeben.

Da die Zahlen bei der Aufwandsabschätzung erheblich größer sind als die obere Schranke für die Größe der Einträge, die dort als Primzahlen gewählt wurden, stellt sich die Frage, ob die Laufzeit bei der Konstruktion von Primzahlen durch Hinzunahme der ganzzahligen Division beschleunigt werden kann. Im letzten Abschnitt von Kapitel 6 wird gezeigt, dass dies der Fall ist für randomisierte Verfahren, aber der vorgestellte deterministische Algorithmus zur Primzahlbildung findet zu einer Zahl N nur dann in $\mathcal{O}(\log N)$ Schritten eine Primzahl größer N, falls Mills Konstante $\theta \in \mathbb{R}$ algebraisch ist. Aber auch dies ist ein ungelöstes Problem der Zahlentheorie

2 S-Berechnungsbäume für $S \subseteq \{+, -, *, *_c, \mathrm{DIV}, \mathrm{DIV}_c\}$

In diesem Kapitel werden im ersten Abschnitt die vollständige Charakterisierung der Berechnungsmächtigkeit der S-CTs im Fall n=1 und daraus abgeleitete untere Schranken und im zweiten Abschnitt für n>1 eine vollständige Charakterisierung der Berechnungsmächtigkeit der S-CTs für $S = \{+, -, \mathrm{DIV}_c\}$ und $S = \{+, -, *, \mathrm{DIV}_c\}$ und eine partielle Charakterisierung der Berechnungsmächtigkeit der S-CTs für $S = \{+, -, \mathrm{DIV}\}$ und $S = \{+, -, *, \mathrm{DIV}\}$

gezeigt. Auch im Fall n>1 werden aus den Charakterisierungen untere Schranken abgeleitet. Die Beweise zu den Theoremen, Sätzen, Korollaren usw. dieses Kapitels sind im Fall n=1 im 3. Kapitel und im Fall n>1 im 4. Kapitel, da sie sehr umfangreich und technisch sind und den Lesefluss stark beeinträchtigen würden.

2.1 S-Berechnungsbäume im Fall n=1

Im Fall einer einzigen Eingabevariable wird eine vollständige Charakterisierung der Berechnungsmächtigkeit der S-CTs gegeben [22]. Diese Sprachklassen sind äquivalent für $\{+,-,*,\mathrm{DIV}\}$ und $\{+,-,\mathrm{DIV}_c\}$; es sind genau die Sprachen $L \subseteq \mathbb{Z}$, die aus einer endlichen Menge und endlich vielen arithmetischen Progressionen bestehen.

Definition 1. *Seien* $a_1, a_2 \in \mathbb{N}$, $A_1, A_2, B \subset \mathbb{Z}$, A_1, A_2, B *endlich.*
$L(a,A,B) := B \cup \{d + \lambda a_1 | \lambda \in \mathbb{N}, d \in A_1\} \cup \{d - \lambda a_2 | \lambda \in \mathbb{N}, d \in A_2\}$ *heißen AP-Sprachen (AP = arithmetische Progression).*

Theorem 1. Sei L⊂ℤ. *Folgende Aussagen sind äquivalent:*

(i) L ist eine AP-Sprache.

(ii) L ist durch einen MBT entscheidbar.

(iii) L ist durch einen $\{+,-,\mathrm{DIV}_c\}$*-CT entscheidbar.*

(iv) L ist durch einen $\{+,-,*,\mathrm{DIV}\}$*-CT entscheidbar.*

Aber die Komplexitäten sind verschieden: Falls die Konstanten aus $\mathbb{Q}$ sind, kann jede Sprache $L \subseteq \mathbb{Z}$, die überhaupt erkannt werden kann, bereits in konstanter Zeit über $\{+,-,*_c,\mathrm{DIV}\}$ erkannt werden. Dies folgt unmittelbar aus der obigen Charakterisierung der Sprachklassen als AP-Sprache, da A und B endlich sind und die Zugehörigkeit zu einer arithmetischen Progression über $\{+,-,*_c,\mathrm{DIV}\}$ in konstanter Zeit entschieden werden kann. Aber die aus demTheorem abgeleitete Konstante aus [22] ist abhängig von der Sprache L. Darüber hinaus zeige ich sogar, dass erstaunlicherweise jede Sprache L in

$CC_1(\{+,-,*,\mathrm{DIV}\},\mathbb{Q})$ in 40 Schritten, d.h. unabhängig von L, erkannt werden kann. Der Beweis basiert auf dem bereits eingangs erwähnten Algorithmus zur Polynomauswertung in 15 Schritten von Nader BSHOUTY.

Satz 1 *Jede Sprache $L \subseteq \mathbb{Z}$, die von einem $(\{+,-,*,\mathrm{DIV}\},\mathbb{Q})$-CT erkannt wird, kann in 40 Schritten, unabhängig von L, von einem $(\{+,-,*,\mathrm{DIV}\},\mathbb{Q})$-CT erkannt werden.*

Auf der anderen Seite gibt es für einige Sprachen der Größe n untere Schranken $\Omega(log(n)/loglog(n))$ [22], falls nur Operationen aus $\{+,-,\mathrm{DIV}_c\}$ benutzt werden, (aber immer noch beliebige Konstanten aus $\mathbb{Q}$).

Satz 2 *Sei L$\subset \mathbb{Z}$, #L=n. Falls L keine arithmetische Progression der Länge k+1 enthält, hat ein $(\{+,-,\mathrm{DIV}_c\},\mathbb{Q})-$ CT, der L erkennt, die Tiefe $\Omega(\log n/\log\log n)$, falls k$\leq$log n, ansonsten $\Omega(\log n/\log k)$.*

Aus diesem Satz ergeben sich für nachstehende Beispiele untere Schranken für $(\{+,-,\mathrm{DIV}_c\},\mathbb{Q})-$ CTs.

Beispiel 1

- $\mathrm{L_n} := \{2^{\mathrm{i}}, i = 1,...,n\}$ hat n Elemente und keine arithmetische Progression der Länge 3, daher gilt eine untere Schranke von $\Omega(\log n/\log\log n)$.
- $\mathrm{L_m} := \{i^2, i = 1,...,m\}$ hat m Elemente und keine Progression der Länge 4, daher gilt ebenfalls eine untere Schranke von $\Omega(\log n/\log\log n)$.
- $\mathrm{L_{l,k}} := \{j \cdot (k+1)^{\mathrm{i}}, i = 0,...,l-1; j = 1,...,k\}$ hat n= l·k Elemente und keine Progression der Länge k+1, daher gilt eine untere Schranke von $\Omega(\log n/\log k)$.

Weitere untere Schranken sind nur bekannt, falls die Konstanten auf $\{0,1\}$ beschränkt werden. In diesem Fall benötigt die Berechnung des größten gemeinsamen Teilers von zwei N-Bit Zahlen mit den Operationen $\{+,-,*,\mathrm{DIV}\}$ $\Omega(\log\log N)$ Zeit, siehe [27], und $\Omega(N)$ mit den Operationen $\{+,-,*_c,\mathrm{DIV}_c\}$, siehe [5].

2.2 S-Berechnungsbäume im Fall n>1

Es werden Sprachklassen $CC_n(S)$ mit DIV oder DIV_c in S unterschieden und untere Schranken bewiesen.

Zunächst betrachte ich die Operationsmengen $\{+,-,*,\mathrm{DIV}_c\}$ und $\{+,-,*_c,\mathrm{DIV}_c\}$. Mit $\{+,-,*\}$ können Polynome und mit $\{+,-,*_c\}$ nur lineare Funktionen berechnet werden.

Für $a \in \mathbb{N}$, $\overline{b} \in \mathbb{Z}^n$ bezeichne ich die Menge $a\mathbb{Z}^n + \overline{b} = \{a\overline{x} + \overline{b},\ x \in \mathbb{Z}^n\}$ als a-Gitter. Die a-Gitter $a\mathbb{Z}^n + \overline{b}$ für $\overline{b} \in \{0,\ldots,a-1\}^n$ bilden eine Zerlegung des $\mathbb{Z}^n$ in a-Gitter der Länge a.

In $\mathbb{Z}$ ist solch ein a-Gitter nichts anderes als eine arithmetische Progression mit der Schrittlänge a.

Theorem 2. *Sei $S = \{+,-,*,\mathrm{DIV}_c\}$ oder $S = \{+,-,*_c,\mathrm{DIV}_c\}$.*

a) $L \subseteq \mathbb{Z}^n$ kann genau dann durch einen S-CT D entschieden werden, wenn es ein $a \in \mathbb{N}$ gibt, so dass der $\mathbb{Z}^n$ in a-Gitter zerlegt werden kann, so dass L auf jedem einzelnen a-Gitter durch einen $(S - \{\mathrm{DIV}_c\})$-CT erkannt wird.

b) Ist D ein $(S,\mathbb{Q})$-CT der Tiefe T, haben die obigen $((S - \{\mathrm{DIV}_c\},\mathbb{Q})$-CTs die Tiefe $\mathcal{O}(T)$.

c) Ist D ein $(S,\{0,1\})$-CT der Tiefe T, haben die obigen $(S-\{\mathrm{DIV}_c\},\mathbb{Q})$-CTs die Tiefe $\mathcal{O}(T)$, und die Schrittweite a der Gitter ist höchstens $2^{2^{2T}}$.

Das obige Ergebnis gibt eine „Normalform" für Berechnungsbäume mit DIV_c: Zunächst bestimme mit DIV_c, auf welchem Gitter die Eingabe liegt. Danach entscheide mit einem Berechnungsbaum ohne DIV_c, ob die Eingabe in L liegt. Die untere Schranke in b) zeigt, dass die Tiefe sich nur um einen konstanten Faktor unterscheidet, falls diese Normalform benutzt wird.

In $\mathbb{Z}$ ist die Darstellung als AP- Sprache solch eine „Normalform", jedoch gilt sie dort, anders als im Fall n>1, nicht nur für DIV_c, sondern auch für DIV.

Als Korollar zu Theorem 1b) kann gezeigt werden, dass die untere Schranke über die Zahl der Zusammenhangskomponenten für $\{+,-,*\}$-CTs in [12],

die in [42] auf ganzzahlige Eingaben übertragen wird, auch für ganzzahlige Eingaben und $\{+,-,*,\mathrm{DIV}_c\}$-CTs gilt und daher auf eine große Klasse von Sprachen übertragen werden kann. Hierzu zählen folgende Beispiele

Korollar 1 *Für $(\{+,-,*,\mathrm{DIV}_c\},\mathbb{Q})$- CTs gelten folgende unteren Schranken:*

- $\Omega(n\log(n))$ *für Element Distinctness (überprüfe, ob alle Eingaben $x_1,\ldots,x_n$ verschieden sind, (siehe [12]).*
- $\Omega(n^2)$ *für das Rucksackproblem (Eingabe $a_1,\ldots,a_n,b$; überprüfe, ob es $y_1,\ldots,y_n\in\{0,1\}$ gibt mit $\Sigma_{i=1}^n a_iy_i=b$, siehe [16]).*
- $\Omega(n^2\log(k+1))$ *für lineare diophantische Gleichungen mit k-beschränkten Lösungen (Eingabe $a_1,\ldots,a_n,b$, überprüfe, ob es $y_1,\ldots,y_n\in\{0,\ldots,k\}$ gibt mit $\Sigma_{i=1}^n a_iy_i=b$, siehe [30]).*

Nun komme ich zu Berechnungen mit der allgemeinen ganzzahligen Division. In diesem Fall kann keine vollständige Charakterisierung der Sprachklassen gegeben werden, sondern nur eine partielle. Zunächst betrachte ich $\{+,-,*_c,\mathrm{DIV}\}$-CTs.

Theorem 3.

a) Falls $L\subseteq\mathbb{Z}^n$ von einem $\{+,-,*_c,\mathrm{DIV}\}$-CT D erkannt wird, gilt: Zu jeder irrationalen Zahl β gibt es eine Pyramide $P:=\{(x_1,..,x_n)\in\mathbb{Z}^n,\ c_1<\frac{x_{i+1}}{x_i}<c_2,\ i=2,...,n\}$ mit $c_1,c_2\in\mathbb{Q}$, $c_1<\beta<c_2$ und $a\in\mathbb{N}$, so dass es zu jedem $\bar{b}\in\{0,\ldots,a-1\}^n$ einen $\{+,-,*_c\}$-CT gibt, der L auf $(a\mathbb{Z}^n+\bar{b})\cap P$ erkennt.

b) Falls D ein $(\{+,-,*_c,\mathrm{DIV}\},\{0,1\})$-CT der Tiefe T ist, gibt es $a\in\mathbb{N}$ und eine Pyramide $P:=\{(x_1,..,x_n)\in\mathbb{Z}^n,\ c_1<\frac{x_{i+1}}{x_i}<c_2,\ i=2,...,n\}$ mit $c_1,c_2\in\mathbb{Q}$, so dass es zu jedem $\bar{b}\in\{0,\ldots,a-1\}^n$ einen $(\{+,-,*_c\},\mathbb{Q})$-CT der Tiefe $\mathcal{O}(T)$ gibt, der L auf $(a\mathbb{Z}^n+\bar{b})\cap P$ erkennt.

c) Falls D ein $(\{+,-,*_c,\mathrm{DIV}\},\{0,1\})$-CT der Tiefe T ist, der eine Sprache $L \subseteq \mathbb{Z}^2$ erkennt, gibt es $a \in \mathbb{N}$, $a \leq 2^{2^{3T}}$, und eine Pyramide $P := \{(x,y) \in \mathbb{Z}^2,\ c_1 < \frac{x}{y} < c_2,\ i = 2,...,n\}$ mit $c_1, c_2 \in \mathbb{Q}$, $c_1 < c_2, c_2 - c_1 \geq \frac{1}{2^{2^{3T}}}$, so dass es zu jedem $\bar{b} \in \{0,\ldots,a-1\}^2$ einen $(\{+,-,*_c\},\mathbb{Q}\})$-CT der Tiefe $\mathcal{O}(T)$ gibt, der L auf $(a\mathbb{Z}^2 + \bar{b}) \cap P$ erkennt.

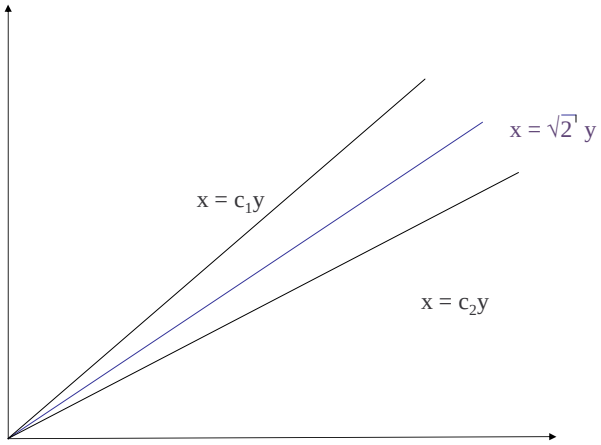

Abbildung 3. Pyramide

In [10] wird ein Algorithmus vorgestellt, der den ggT von 2 N-Bit Zahlen in der Zeit $\mathcal{O}(N)$ berechnet. Er benutzt nur Operationen aus $\{+,-,\mathrm{DIV}_c\}$. Aus Ergebnissen in [5] und [22] folgt, dass für die Operationsmenge $\{+,-,\mathrm{DIV}_c\}$ die untere Schranke auch bei $\Omega(N)$ ist, d.h. eine scharfe Schranke ist. Aus Theorem 2b) folgt, dass der Algorithmus aus [10] auch bei der Operationsmenge $\{+,-,\mathrm{DIV}\}$ optimal ist.

Korollar 2 *Ein $(\{+,-,\mathrm{DIV}\},\{0,1\})$-CT, der die Teilerfremdheit von 2 N-Bit Zahlen x,y überprüft, hat eine Tiefe von $\Omega(N)$.*

Nun komme ich zur mächtigsten Operationsmenge, nämlich $\{+,-,*,\mathrm{DIV}\}$.

Wie bereits vorher erwähnt, ist es sehr schwierig, in diesem Fall untere Schranken zu finden. Es wird die erste untere Schranke bei Konstanten $\mathbb{Q}$ bewiesen .

Theorem 4.

a) Falls $L \subseteq \mathbb{Z}^n$ *durch einen* $\{+,-,*,\mathrm{DIV}\}$*-CT* D *erkannt werden kann, gibt es für jedes* $c_1,..,c_n \in \mathbb{N}$ *Polynome* $p_i : \mathbb{Z}^{n-i} \to \mathbb{Q}$*, i=1,...,n-1 und* $k_1,...,k_n$, $K \in \mathbb{N}$*, so dass eingeschränkt auf* $\{(x_1,...,x_n) \in \mathbb{N}^n \; x_i \geq x_{i+1}^{k_i}$, $x_i \equiv c_i \bmod p_i(x_{i+1},...,x_n)$, $i = 1,...,n-1$, $x_n \equiv c_n \bmod k_n$, $x_n > K,\}$ L *durch einen* $\{+,-,*\}$*-CT erkannt werden kann.*

b) Falls D *ein* $(\{+,-,*,\mathrm{DIV}\},\mathbb{Q})$*-CT der Tiefe* T *ist, haben die* p_i *einen Grad* $\leq 2^{2^{nT}}$, $k_i \leq 2^{2^{(n-1)T}}$ *und der* $(\{+,-,*\},\mathbb{Q})$*-CT hat eine Verzweigungstiefe* $\mathcal{O}(T)$ *und Grad* $\mathcal{O}(2^{2^{nT}})$.

Aus Teil b) kann die erste nicht-triviale untere Schranke für die Tiefe von $(\{+,-,*,\mathrm{DIV}\},\mathbb{Q})$-CTs abgeleitet werden. Diese Sprachklasse ist sehr mächtig, wenn man daran denkt, dass auch im mehrdimensionalen Fall jede endliche Sprache, unabhängig von der Größe der Sprache, in konstanter Zeit erkannt werden kann.

Bemerkung 3 *Jede endliche Sprache* $L \subseteq \mathbb{Z}^n$ *kann durch einen* $(\{+,-,*,\mathrm{DIV}\},\mathbb{Q})$*-CT in konstanter Zeit unabhängig von* L *erkannt werden.*

Korollar 3 *Sei* $r : \mathbb{Z} \to \mathbb{Z}$ *ein irreduzibles Polynom vom Grad* d *mit positivem Leitkoeffizienten. Jeder* $(\{+,-,*,\mathrm{DIV}\},\mathbb{Q})$*-CT, der* $L_r = \{(x,y) \in \mathbb{Z}^2, r(y) > x\}$ *erkennt, hat eine Tiefe* $\Omega(log\,log\,(d))$.

Diese untere Schranke zeigt, dass anders als im eindimensionalen Fall die Aussage, dass jede Sprache L, die überhaupt von einem $(\{+,-,*,\mathrm{DIV}\},\mathbb{Q})$*-CT* erkannt werden kann, bereits in konstanter Zeit erkannt wird, im mehrdimensionalen Fall nicht mehr gilt.

Ebenfalls anders als im Falle $n = 1$ zeigen die Teile a) der obigen drei Theoreme, dass nicht nur Sprachklassen mit oder ohne ganzzahlige Division unterschieden werden.

Eine vollständige Übersicht über die Beziehungen dieser Sprachklassen zeigt das folgende Theorem. In diesem Theorem bedeutet ein Pfeil $S \to S'$ für

Operationsmengen S, S', dass $CC_n(S') \subsetneq CC_n(S)$ für $n \geq 2$ gilt. $S - - - S'$ bedeutet, dass $CC_n(S)$ und $CC_n(S')$ nicht vergleichbar sind.

Theorem 5. *Folgende Beziehungen gelten für die Sprachklassen $CC_n(S)$ für $n \geq 2$:*

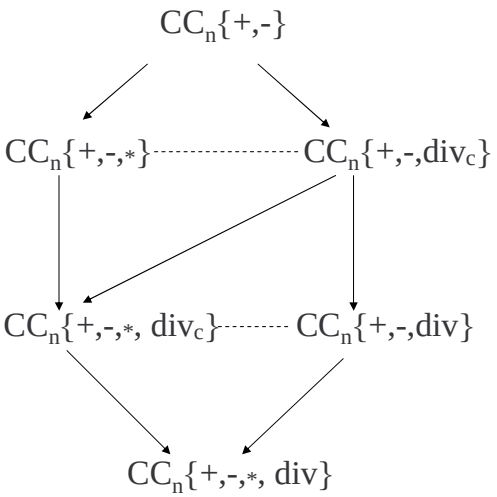

Abbildung 4. Sprachklassen für n>1

Bemerkung 4 *Das obige Diagramm zerfällt in zwei Sprachklassen mit* DIV *oder* DIV_c *bzw. ohne* DIV *oder* DIV_c*, falls der Fall $n = 1$ betrachtet wird, siehe Theorem 1.*

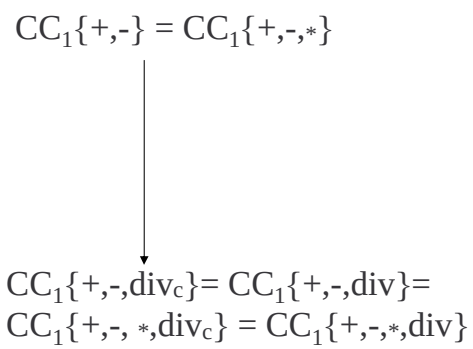

Abbildung 5. Sprachklassen für n=1

3 Beweise im Fall n= 1

Dieses Kapitel enthält die Beweise zu Theorem 1, zu Satz 1 und zu der unteren Schranke in Satz 2, die bis auf Lemma 4 in [29] und Satz 1 in [25] den Beweisen in [22] folgen.

Beweis. [zu Theorem 1]
Es wird der Beweis in folgender Reihenfolge ausgeführt:
(i) ⇒(iii) ⇒(iv) ⇒(ii) ⇒ (i)

(i) ⇒(iii)

Sei L = L(a,A,B) eine AP-Sprache.

- Überprüfe durch Binärsuche, ob x ∈B gilt.

- Falls ja, akzeptiere, ansonsten überprüfe für jedes d ∈A_1 oder d ∈A_2, ob $x \in \{d + \lambda a_1, \lambda \in \mathbb{N}\}$ oder $x \in \{d - \lambda a_2, \lambda \in \mathbb{N}\}$ gilt.

Dies ist durch $a_i \cdot((x - d)\ \mathrm{DIV}_c a_i) = x - d$, i∈{1,2} möglich.

Da a_i Konstanten sind, kann $a_i \cdot((x - d)\ \mathrm{DIV}_c a_i)$ ohne Multiplikation berechnet werden.

Der obige $\{+, -, *, \mathrm{DIV}\}$- CT entscheidet L. □

(iii) ⇒(iv)

gilt trivialerweise, da $\{+, -, \mathrm{DIV}_c\} \subset \{+, -, *, \mathrm{DIV}\}$. □

(ii) ⇒ (i)

Sei T ein MBT, der eine Sprache L ⊂ ℤ entscheidet, v ein akzeptierendes Blatt von T und c(v) die Eingabemenge, die zu v gelangt. Da durch $\{+, -, *\}$ nur Polynome berechnet werden können, sind die binären Verzweigungsknoten mit „p(x) >0" oder „p(x) ≤0" für Polynome p beschriftet, die auf dem Weg zu v berechnet wurden. Daher ist $c(v)$ entweder endlich oder $c(v) = B_v \cup I_v$. B_v ist dabei eine endliche Menge, die alle Elemente von c(v) enthält, die zu beschränkten Zusammenhangskomponenten der Mengen $\{x|p(x) > (\leq)0\}$, die von den

binären Verzweigungen kommen, gehören. Die Menge I_v hat die Form I_v= {x|x >β_v, x mod δ_j= i_j, j = 1,...,r}, wobei die r Modulo-Verzweigungsknoten (kurz: MB-Knoten) auf dem Weg zu v vom Grad δ_1,...,δ_r sind, bei der j-ten Verzweigung der i_j-te Zweig gewählt wird und β_v= max B_v ist.

Die Menge I_v kann als eine einzige Progression dargestellt werden, d.h. I_v= $\{d_v + \lambda a_v, \lambda \in \mathbb{N}\}$ für geeignete d_v, a_v. Sei V die Menge der Blätter, die unendlich viele Eingaben akzeptieren.

Dann ist I = $\bigcup_{\nu \in V} I_v$Vereinigung von endlich vielen arithmetischen Progressionen.

Bekannterweise folgt aus der Zahlentheorie, dass I = B' $\cup\{d+\lambda a, \lambda \in \mathbb{Z}, d \in A\}$ für ein a $\in \mathbb{Z}$ und endliche Mengen B' und A gilt.

Sei nun B" die endliche Eingabemenge, die zu den akzeptierenden Blättern mit endlicher Eingabemenge gelangt, und B"':= $\bigcup_{\nu \in V} B_v$, B:=B' $\cup$B" $\cup$B"', dann ist L = L(a,A,B), also eine AP-Sprache. □

(iv) ⇒ (ii)

Diese Beweisrichtung ist die umfangreichste.

Sei T ein $\{+, -, *, \mathrm{DIV}\}$- CT, der L erkennt.

Es wird gezeigt, dass die DIV-Knoten durch MB-Knoten ersetzt werden können, so dass der so erhaltene MBT eine Sprache L' akzeptiert, die für betragsmäßig genügend große x mit L übereinstimmt. Für betragsmäßig kleine x werden x ∈L durch Binärsuche akzeptiert, so dass man einen MBT für L erhält.

Um die DIV-Knoten durch MB-Knoten zu ersetzen, wird das folgende Lemma benötigt.

Lemma 1. *Seien* p, q :$\mathbb{Z} \to \mathbb{Q}$ *Polynome mit rationalen Koeffizienten, deg(p)* ≥*deg(q). Dann gibt es* $\beta, z \in \mathbb{N}$, *so dass es zu jedem* $i \in \{0, ..., \beta - 1\}$ *ein Polynom* r_i :$\mathbb{Z} \to \mathbb{Q}$ *mit rationalen Koeffizienten gibt, so dass* $p(x)\,\mathrm{DIV}\,q(x) = r_i(x)$ *gilt.*

Mit Hilfe dieses Lemmas wird zunächst der Beweis (iv) ⇒ (ii) beendet.

Sei υ der erste DIV-Knoten auf einem Weg von der Wurzel zu einem Blatt in T. Dann ist dieser Knoten mit $p(x)\,\mathrm{DIV}\,q(x)$ beschriftet, wobei p,q Polynome mit rationalen Koeffizienten sind, die auf dem Weg zu υ berechnet wurden. Falls deg(p) < deg(q) gilt, wird $p(x)\,\mathrm{DIV}\,q(x)$ durch 0 ersetzt. Dies ist korrekt für betragsmäßig genügend große x.

Im Folgenden sei deg(p) ≥deg(q).

Nach dem obigen Lemma gibt es $\beta, z \in \mathbb{N}$, so dass es zu jedem $i \in \{0, ..., \beta-1\}$ ein Polynom $\mathrm{r_i} : \mathbb{Z} \to \mathbb{Q}$ mit rationalen Koeffizienten gibt, so dass $p(x)\,\mathrm{DIV}\,q(x) = r_{\mathrm{i}}(x)$ gilt. Es wird nun der DIV- Berechnungsknoten durch einen MB-Knoten vom Grad β ersetzt. Für $i \in \{0, ..., \beta - 1\}$ wird an den i-ten Zweig die Berechnung für das Polynom $\mathrm{r_i}$ angehängt. Eine Kopie des Teilbaumes von T unterhalb von υ wird an die Berechnung der $\mathrm{r_i}$ gehängt. In diesem Teilbaum wird jeweils $p(x)\,\mathrm{DIV}\,q(x)$ durch $r_{\mathrm{i}}(x)$ ersetzt.

Von der Wurzel zu den Blättern wird so schrittweise jeder DIV- Berechnungsknoten ersetzt. Der dadurch entstandene MBT erkennt eine Sprache L' mit $L' \cap \{x \in \mathbb{Z}, |x| \geq z\} = L \cap \{x \in \mathbb{Z}, |x| \geq z\}$ für ein genügend großes z.

Um L zu erkennen, wird zunächst nach $|x| \geq z$ verzweigt. Bei positiver Antwort wird der MBT durchlaufen, ansonsten wird durch Binärsuche die endliche Sprache $L \cap \{x \in \mathbb{Z}, |x| < z\}$ erkannt. □

Beweis. [von Lemma 1]

Aus der Algebra ist bekannt, dass man durch Polynomdivision Polynome r,s:$\mathbb{Z}\to\mathbb{Q}$ mit rationalen Koeffizienten und deg(s)<deg(q) mit p = r·q + s gibt. Wähle nun β, so dass es ein Polynom $\tilde{\mathrm{r}} : \mathbb{Z} \to \mathbb{Q}$ mit ganzzahligen Koeffizienten und $r = \frac{1}{\beta}\tilde{r}$ gibt. Dann erhält man für $p(x)\,\mathrm{DIV}\,q(x)$

$$p(x)\,\mathrm{DIV}\,q(x) = \left\lfloor r(x) + \frac{s(x)}{q(x)} \right\rfloor = \left\lfloor \frac{1}{\beta}\tilde{r}(x) + \frac{s(x)}{q(x)} \right\rfloor$$

Da deg(s) < deg(q) gilt, ist $\lim_{|x|\to\infty} \frac{s(x)}{q(x)} = 0$.

Das heißt, dass für genügend großes x $p(x)\,\mathrm{DIV}\,q(x) = \tilde{r}(x)\,\mathrm{DIV}\,\beta$ gilt.

Sei nun $i \in \{0, ..., \beta - 1\}$ fest und $x \bmod \beta = i$, d.h. $x = \lambda\beta + i$ für ein $\lambda \in \mathbb{Z}$ und $\tilde{r}(x) = \sum_{j=0}^{n} a_j x^j, a_j \in \mathbb{Z}$.

Wird i als Konstante betrachtet, kann $\tilde{r}(x) = \tilde{r}(\lambda\beta + i)$ als Polynom in λ geschrieben werden.

$$\tilde{r}(x) = \sum_{j=0}^{n} b_j(\lambda\beta)^j \qquad \text{mit } b_j \in \mathbb{Z}$$

(b_j hängt von i ab)

$$\tilde{r}(x) = \sum_{j=0}^{n} b_j(\lambda\beta)^j = b_0 + \beta \cdot g(\lambda) \text{ mit } g(\lambda) = \sum_{j=0}^{n} b_j \beta^{j-1} \lambda^j \in \mathbb{Z}$$

Daher erhält man für genügend großes x mit x mod β= i

$$p(x)\,\mathrm{DIV}\,q(x) = \tilde{r}(x)\,\mathrm{DIV}\,\beta = b_0\,\mathrm{DIV}\,\beta + g(\lambda)$$

$$= b_0 \mathrm{DIV}\,\beta + g((x-i)/\beta) = \text{Polynom in } x.$$

□

Beweis. [zu Satz 1]

Sei $L \subset \mathbb{Z}$ eine Sprache, die durch einen $(\{+, -, *, \mathrm{DIV}\}, \mathbb{Q})$-CT erkannt wird, dann ist nach Theorem 1 L eine AP-Sprache. Jede solche AP-Sprache hat folgende Darstellung

$$L(a, A, B) := B \cup \{d + \lambda a_1 | \lambda \in \mathbb{N}, d \in A_1\} \cup \{d - \lambda a_2 | \lambda \in \mathbb{N}, d \in A_2\}$$

Für eine Eingabe $x \in \mathbb{Z}$ entscheide, ob $x \in B$ gilt. Falls $x \in B$ akzeptiere, ansonsten berechne x mod a_1 = x - a_1(x div a_1) für x>0 und x mod a_2 = x - a_1(x div a_2) für x<0.

Ohne Einschränkung gelte für $d \in A_i$ $d \,\mathrm{div}\, a_i = k_i$. Entscheide, ob $x \bmod a_i + k_i \in A_i$. Falls ja, akzeptiere, ansonsten verwerfe. Da A_i und B endlich sind,

können mit dem folgenden Lemma die Abfragen $x \in B$ und $x \bmod a_i + k_i \in A_i$ in konstanter Zeit, unabhängig von A_i und B, entschieden werden, d.h L wird in konstanter Zeit, unabhängig von L, erkannt. □

Lemma 2. *Sei $A \subset \mathbb{Z}$ endlich. A kann durch einen $(\{+, -, *_c, \text{DIV}\}, \mathbb{Q})$-CT in 18 Schritten, unabhängig von A, erkannt werden.*

Beweis. [zu Lemma 2]

Da A endlich ist, kann A als Nullstellenmenge eines Polynoms p mit ganzzahligen Koeffizienten betrachtet werden. Sei $N := \max\{|x||x \in A\}$. Der folgende Berechnungsbaum erkennt A.

Für eine Eingabe $x \in \mathbb{Z}$ verwerfe, falls $|x| > N$ gilt. Ansonsten berechne $p(x)$ bzw. $p(-x)$ für $x < 0$ und akzeptiere, falls $p(x) = 0$ bzw. $p(-x) = 0$ für $x < 0$, ansonsten verwerfe. Nach Theorem 7 kann solch ein Polynom mit ganzzahligen Koeffizienten über einer endlichen Eingabemenge in 15 Schritten, unabhängig von p und N, in konstant vielen Schritten berechnet werden. □

Beweis. [zu Satz 2]

T sei ein $(\{+, -, \text{DIV}_c\}, \mathbb{Q})-$ CT der Tiefe D, der eine endliche Sprache L (#L=n), die keine arithmetische Progression der Länge k+1 enthält, erkennt. v sei ein Blatt von T und $v_1, ..., v_d = v$ sei der Pfad zu v, d$\leq D$. $f_i : \mathbb{Z} \to \mathbb{Q}$ seien die Funktionen, die an den Knoten v_i berechnet werden, und $c(v)$ die Menge der Eingaben, die zu v gelangt.

Mit dem folgenden Lemma wird $c(v)$ charakterisiert.

Lemma 3. *Es gibt ein konvexes Polytop P im $\mathbb{R}^{d+1}$ mit*

(i) $x \in c(v) \Leftrightarrow \exists (c_1, \ldots, c_d) \in \mathbb{Z}^d : (x, c_1, \ldots, c_d) \in P$.

(ii) Für jedes $x \in c(v)$ gibt es genau ein $(c_1, \ldots, c_d) \in \mathbb{Z}^d$ mit $(x, c_1, \ldots, c_d) \in P$.

Beweis. [zu Lemma 3]

Von der Wurzel zu den Blättern wird jede DIV_c- Operation durch eine neue Variable c_j ersetzt. Es werden höchstens d neue Variablen benötigt. Im Folgenden wird, falls das Ergebnis dieser DIV_c- Operation als Operand benutzt wird, statt des Ergebnisses dieser DIV_c- Operation diese neue Variable benutzt. An jedem Knoten v_i wird nun eine Funktion $g_\mathrm{i}(x, c_1, \ldots, c_d)$ berechnet. Da nur +,- als Operationen benutzt werden, sind die g_i linear. $I \subset 1, .., d$ sei die Indexmenge, für die an den Knoten v_i eine DIV_c- Operation benutzt wird.

Wenn an dem Knoten v_i, i ∈I, $a_\mathrm{i}(x, c_1, \ldots, c_d)\,\mathrm{DIV}_c\, b_\mathrm{i}$ berechnet wird, werden folgende Ungleichungen definiert.
(*)

$$b_\mathrm{i} \cdot c_\mathrm{i} \leq (\geq)\, a_\mathrm{i}(x, c_1, \ldots, c_d) < (>) b_\mathrm{i}(c_\mathrm{i} + 1)$$

Falls b_i positiv ist, steht dort $(\leq,<)$ ansonsten $(\geq>)$.
Es werden noch die Ungleichungen der Verzweigungsknoten v_i
(**)

$$g_\mathrm{i}(x, c_1, \ldots, c_d) > (\leq) 0$$

zu dem Ungleichungssystem aus (*) hinzugefügt. Ob dort $<$ oder $\geq$ steht, richtet sich danach, ob der rechte oder linke Zweig zum Pfad gehört.

Nach der Konstruktion dieses Systems linearer Ungleichungen aus (*) und (**) erfüllt dessen Lösungsmenge P gerade die Bedingungen (i) und (ii) und P ist als Lösungsmenge eines linearen Ungleichungssystems ein konvexes Polytop.

□

Sei P_v das zum Pfad v gehörende Polytop. Aus Lemma 2 folgt, dass $\#P_v \cap \mathbb{Z}^\mathrm{n} = \#c(v)$. Im Folgenden wird gezeigt, dass $\#P_v \cap \mathbb{Z}^\mathrm{n}$ klein ist, falls $c(v)$ keine lange arithmetische Progression enthält. Dazu benötigt man noch das folgende Lemma. Man spricht von einer Progression der Länge k im $\mathbb{Z}^\mathrm{n}$, falls es k Punkte gibt, die in jeder Koordinate zu einer arithmetischen Progression der Länge k gehören.

Lemma 4. *Sei B die Menge der ganzzahligen Punkte in einer konvexen Teilmenge des $\mathbb{R}^n$. Falls $|B| > k^n$ gilt, enthält B eine Progression der Länge k+1.*

Beweis. [zu Lemma4] [29]

Betrachte die Abbildung $\mathbb{Z}^n \rightarrow \mathbb{Z}_k^n$,

$$(x_1, x_2, \ldots, x_n) \rightarrow (x_1 \bmod k, x_2 \bmod k, \ldots, x_n \bmod k).$$

Da $|\mathbb{Z}_k^n| = \mathrm{k}^{\mathrm{n}}$ und $|\mathrm{B}| > \mathrm{k}^{\mathrm{n}}$ gilt, gibt es x, y ∈B, die denselben Bildpunkt haben. D.h. es gilt x - y = kv für einen von 0 verschiedenen Vektor v mit ganzzahligen Koordinaten. Da B konvex ist, gehören dann aber die k + 1 auf einer Gerade liegenden Punkte y, y + v, y + 2v, ..., y + kv = x ebenfalls zu B. □

Mit Hilfe dieses Lemmas kann nun Satz 2 bewiesen werden.

Sei v ein akzeptierendes Blatt. Da L keine Progression der Länge k+1 enthält, gibt es auch in $c(v)$ keine Progression der Länge k+1. Daher enthält auch $\#P_v \cap \mathbb{Z}^{\mathrm{d}+1}$ keine Progression der Länge k+1. Aus Lemma 4 und Lemma 7 folgt, dass $\#c(v) = \#P_v \cap \mathbb{Z}^{\mathrm{d}+1} \leq k^{\mathrm{d}+1}$ gilt. Da T höchstens 2^{D} akzeptierende Blätter hat, hat L höchstens $2^{\mathrm{D}} \cdot k^{\mathrm{D}+1}$ Elemente, d.h. $n \leq 2^{\mathrm{D}} \cdot k^{\mathrm{D}+1}$.

$$\Leftrightarrow D \geq \frac{\log n - \log k}{\log k + 1}$$

Hieraus folgt unmittelbar die angegebene untere Schranke. □

4 Beweise im Fall n> 1

Dieses Kapitel folgt den Ausführungen in [25]. Die dort für den Spezialfall n=2 durchgeführten Beweise werden verallgemeinert für den Fall n≥2.

4.1 Operationsmengen mit $\mathbf{DIV}_c$

Dieser Abschnitt enthält die Beweise zu Theorem 2 und die aus dem Theorem folgenden allgemeinen unteren Schranken.

Beweis. [von Theorem 2 a)]

"⇐" Sei $\mathbb{Z}^n$ die disjunkte Vereinigung endlich vieler a-Gitter $a\mathbb{Z}^n + \bar{b}, \bar{b} \in \{0, \ldots, a-1\}^n$. Auf jedem dieser Gitter kann L durch einen $\{+, -\}$ bzw.

$\{+,-,*\}$-CT erkannt werden. Der folgende Algorithmus mit $\{+,-,\mathrm{DIV}_c\}$ bzw. $\{+,-,*,\mathrm{DIV}_c\}$ als Operationsmenge erkennt L.

Entscheide, in welchem Gitter $a\mathbb{Z}^n+\bar{b}$ die Eingabe $\overline{x}$ liegt. Der Startwert $\bar{b}$ des Gitters wird durch $b_i = x_i - (x_i \,\mathrm{DIV}_c a)\cdot a$ für $i = 1,\ldots,n$ berechnet. Dies geht ohne Multiplikation, da a eine Konstante ist.

Für $\bar{x} \in a\mathbb{Z}^n + \bar{b}$ entscheide mit dem zugehörigen $\{+,-\}$-CT bzw. $\{+,-,*\}$-CT, ob $\bar{x} \in L \cap (a\mathbb{Z}^n + \bar{b})$ gilt.

"$\Rightarrow$" Ich führe den Beweis für $S = \{+,-,*,\mathrm{DIV}_c\}$. Der Beweis für $S = \{+,-,\mathrm{DIV}_c\}$ ist analog. Sei D ein $\{+,-,*,\mathrm{DIV}_c\}$-CT, der L erkennt. Ziel ist es, von der Wurzel zu den Blättern jeden Knoten v mit einer DIV_c Operation durch einen MB-Knoten zu ersetzen. Verzweigt wird an diesen Knoten nach allen möglichen Ergebnissen $(x_1 \bmod a,\ldots,x_n \bmod a) \in \{0,\ldots,a-1\}^n$ für eine Konstante $a \in \mathbb{N}$. Die Ersetzung des Berechnungsbaumes mit der Operation DIV_c durch einen MB- CT wurde im eindimensionalen Fall in [22] eingeführt.

Sei v der erste Berechnungsknoten von der Wurzel zu einem Blatt, an dem die Funktion DIV_c benutzt wird. Angenommen, es wird $f(\overline{x})\,\mathrm{DIV}_c\frac{a}{b}$ berechnet, wobei f auf dem Weg zu v berechnet wurde. Da v der erste Knoten auf dem Weg mit DIV_c ist, ist f ein Polynom mit rationalen Koeffizienten. $k \in \mathbb{N}$ sei so gewählt, dass $k \cdot f$ ganzzahlige Koeffizienten hat. Ersetze v durch einen MB-Knoten, bei dem nach den Ergebnissen von $(x_1 \bmod ka,\ldots x_n \bmod ka)$ verzweigt wird.

Lemma 5. *Für Eingaben auf dem Gitter $ka\mathbb{Z}^n + \bar{c}$ mit $\bar{c} \in \{0,\ldots,ka-1\}^n$, kann man $f(\overline{x})\,\mathrm{DIV}_c\frac{a}{b}$ durch $\frac{b}{a}f(\overline{x}) - \frac{l}{ka}$, mit einer von $\bar{c}$ abhängigen Konstanten l ersetzen.*

Beweis. [von Lemma 5]

Für lineare Funktionen und Polynome über $\mathbb{Z}$ ist $b\cdot f(\overline{x}) \bmod a$ für $x_i \bmod a = i_j$, $i = 1,...,n$, $i_j \in \{0,...,a-1\}$ eine Konstante $c_{i_1,\ldots,i_n}$ für jedes $(i_1,...,i_n) \in \{0,...,a-1\}^n$.

Da $f(\overline{x})\,\mathrm{DIV}_c\frac{a}{b} = b{\cdot}f(\overline{x})\,\mathrm{DIV}_c a = \frac{b}{a}f(\overline{x}) - \frac{b\cdot f(\overline{x}) \bmod a}{a}$ an dem mit $x_i \bmod a = i_j$, $i = 1, ..., n$, $i_j \in \{0, ..., a-1\}$ beschrifteten Zweig gilt, kann nach diesem Zweig $f(\overline{x})\,\mathrm{DIV}_c\frac{a}{b}$ durch $\frac{b}{a}f(\overline{x}) - \frac{c_{i_1,...,i_n}}{ka}$ ersetzt werden.

Für $b \cdot f(\overline{x}) \in \mathbb{Q}[\overline{\mathrm{x}}] \setminus \mathbb{Z}[\overline{\mathrm{x}}]$ sei k der Hauptnenner der Koeffizienten von $b \cdot f(\overline{x})$, dann ist $k \cdot b \cdot f(\overline{x}) \in \mathbb{Z}[\overline{\mathrm{x}}]$ und damit ist wie oben $f(\overline{x})\,\mathrm{DIV}_c\frac{a}{b}$ durch $\frac{b}{a}f(\overline{x}) - \frac{c_{i_1,...,i_n}}{ka}$ nach dem mit $x_i \bmod ka = i_j$, $i = 1, ..., n$, $i_j \in \{0, ..., ka-1\}$ beschrifteten Zweig substituierbar. □

Im Folgenden wird $f(\overline{x})\,\mathrm{DIV}_c\frac{a}{b}$ durch einen MB-Knoten vom Grad $(ka)^n$ ersetzt. Die Zweige sind Kopien des Teilbaumes von D mit der Wurzel v, bei dem DIV_c am Knoten v durch $\frac{b}{a}f(\overline{x}) - \frac{l}{ka}$ wie im obigen Lemma ersetzt wird. Wenn diese Ersetzungen von der Wurzel zu den Blättern für alle DIV_c- Operationen in dem Baum durchgeführt werden, erhält man einen Berechnungsbaum ohne DIV_c, aber mit MB- Knoten, der L erkennt.

Um zur gewünschten Charakterisierung zu gelangen, kann hier festgestellt werden, dass man einen $\{+, -, *\}$-CT erhält, wenn man an den MB- Knoten jeweils nur einen Ast wählt, d.h. wenn die Eingabemenge auf den Durchschnitt der zugehörigen Gitter beschränkt wird. Da dieser Durchschnitt wiederum ein Gitter ist und man diese erhaltenen Gitter als ein weiteres Gitter mit gemeinsamer Schrittweite angeben kann, folgt Theorem 2 a).

Theorem 2 b) folgt unmittelbar, denn die Tiefe ändert sich nicht, da jeder DIV- Berechnungsknoten durch einen MB- Knoten ersetzt wird und in dem neu entstandenen $S - \{DIV_c\} - CT$ jeweils nur ein Ast ausgewählt wird.

Um Theorem 2 c) zu beweisen, muss zusätzlich noch die Schrittweite a der Gitter analysiert werden. Da nur die Operationen $\{+, -, *\}$ zur Verfügung stehen, sind die Konstanten ganzzahlig. In der Tiefe t sind die durch $\{0, 1\}$ erzeugten Konstanten kleiner gleich 2^{2^t}, somit ist auch die Schrittweite in der Tiefe t höchstens 2^{2^t}. Um den Durchschnitt aller Gitter entlang aller Pfade zu bilden, wird höchstens das Produkt all dieser Schrittweiten berechnet. Die Schrittweite für den Durchschnitt der Gitter entlang eines Pfades ist kleiner

gleich $\prod_{t=0}^{T-1} 2^{2^t} \leq 2^{2^T}$ und da es höchsten 2^T Blätter gibt, ist die Schrittweite höchstens $\left(2^{2^T}\right)^{2^T} = 2^{2^{2T}}$. □

Beweis. [von Korollar 1]

Das Korollar folgt aus Theorem 2 b) und einem Ergebnis in [42] von Andrew YAO. Er zeigt, dass Michael BEN-ORs untere Schranke über die Anzahl der Zusammenhangskomponenten in [12] über $\{+, -, *, /\}$-CTs auch für einige Sprachen gilt, die nur ganzzahlige Eingaben erlaubt. Der dort geführte Beweis gilt analog für Eingaben aus einem Gitter $a\mathbb{Z}^n$ mit einem beliebigen $a \in \mathbb{N}$. So erhält man die unteren Schranken aus Korollar 1 durch Anwendung von Theorem 2 b). □

4.2 Operationsmenge $\{+, -, *_c, \mathbf{DIV}\}$

Dieser Abschnitt enthält den Beweis zu Theorem 3 und Korollar 2.

Beweis. [von Theorem 3 a)]

Sei β irrational, v der erste Knoten mit einer DIV-Operation auf dem Weg von der Wurzel zu einem Blatt in einem $\{+, -, *_c, DIV\}$-CT D. Da nur die Multiplikation mit Konstanten und nicht die allgemeine Multiplikation zur Operationsmenge gehört, sind die auf dem Weg berechneten Funktionen lineare Funktionen. An dem Knoten v wird $f(\bar{x})\,\mathrm{DIV}\,g(\bar{x})$ berechnet mit $f(\bar{x}) = a_1x_1 + ... + a_nx_n + a_{n+1}$, $g(\bar{x}) = b_1x_1 + ... + b_nx_n + b_{n+1}$, $a_i, b_i \in \mathbb{Q}$, $i = 1, ..., n+1$.

Falls $b_1 = 0, ..., b_n = 0$, wird $f(\bar{x})\,\mathrm{DIV}\,b_{n+1}$ berechnet, d.h. es wird nur DIV_c benötigt.

In den anderen Fällen soll wie im folgenden Lemma die Berechnung an dem Knoten v durch endlich viele Verzweigungen ersetzt werden.

Lemma 6. *Sei β irrational, $f, g : \mathbb{Z}^n \to \mathbb{Z}$ seien lineare Funktionen mit rationalen Koeffizienten, g nicht konstant, dann gibt es $c_1 < \beta < c_2$, so dass $f(\bar{x})\,\mathrm{DIV}\,g(\bar{x})$ nur endlich viele Werte für Eingaben aus $P = \{(\bar{x}) \in \mathbb{Z}^n, c_1 \leq \frac{x_{i+1}}{x_i} \leq c_2,\ i = 1, ..., n-1\}$ annimmt.*

Beweis. [von Lemma 6]

Da g nicht konstant ist, gibt es ein $j \in \{1, ..., n\}$, so dass $b_i = 0$ für $i = 1, .., j-1$ und $b_j \neq 0$. Seien $c_1, c_2 \in \mathbb{Q}, c_1 < \beta < c_2$ so gewählt, dass $b_j + b_{j+1}\gamma_{j+1} ... + b_n\gamma_n \neq 0$ für jedes $\gamma_k \in [c_1, c_2]$. Dies ist möglich, da β irrational ist. Wenn für Eingaben aus P die Variablen x_k durch $\gamma_k x_j$, $\gamma_k \in [c_1, c_2]$ ersetzt werden, erhält man

$$f(\bar{x}) \,\mathrm{DIV}\, g(\bar{x})$$

$$= ((a_1x_1 + ... + a_{j-1}x_{j-1}) + (a_j + a_{j+1}\gamma_{j+1} + ... + a_n\gamma_n)x_j + a_{n+1})$$

$$\mathrm{DIV}\,((b_j + b_{j+1}\gamma_{j+1} ... + b_n\gamma_n)x_j + b_{n+1}) =$$

Werden nun für Eingaben aus P die Variablen x_k durch $\gamma'_k x_1$, $\gamma'_k \in [c_1, c_2]$ ersetzt, erhält man

$$((a_1\gamma'_1 + ... + a_n\gamma'_n)x_1 + a_{n+1})\mathrm{DIV}(b_j + b_{j+1}\gamma'_{j+1} ... + b_n\gamma'_n)x_1 + b_{n+1})$$

Durch die Wahl von c_1, c_2, ist der obige Divisor von 0 verschieden, und sowohl Dividend als auch Divisor sind auf P beschränkt. Daher nimmt $f(\bar{x}) \,\mathrm{DIV}\, g(\bar{x})$ nur endlich viele Werte für Eingaben aus P an. □

Man kann nun nach Lemma 6 von oben nach unten jede DIV-Operation und nach Lemma 5 jede DIV_c -Operation ersetzen. Damit ist Theorem 3 a) bewiesen.

Da die Konstanten $c_1, c_2 \in \mathbb{Q}, c_1 < \beta < c_2$ so gewählt werden können, dass $f(\bar{x}) \,\mathrm{DIV}\, g(\bar{x})$ nur konstant viele Werte annimmt, ist die Tiefe des $\{+, -, *_c\}$-CT $\mathcal{O}(T)$, wenn T die Tiefe des $\{+, -, *_c, \mathrm{DIV}\}$-CT. Damit folgt Theorem 3 b).

Um Theorem 3 c) zu beweisen, werden die Konstanten genauer abgeschätzt. In der Tiefe t sind die Konstanten a_j und b_j kleiner gleich 2^{2^t} und damit sind die Konstanten, die $f(\bar{x}) \,\mathrm{DIV}\, g(\bar{x})$ ersetzen, kleiner gleich $5 \cdot 2^{2^t}$. Daher sind auch alle Konstanten in dem $\{+, -, *_c, \mathrm{DIV}_c\}$-CT nach Substitution der DIV-Knoten kleiner gleich $2^{2^{2T}}$. Die Schrittweite in dem $\{+, -, *_c\}$-CT ist somit kleiner gleich $2^{2^{3T}}$.

Wird wie in Teil a) für Eingaben aus P die Variable y durch $\gamma x, \gamma \in [c_1, c_2] \subset \left[\frac{1}{2}, 2\right]$ ersetzt, gilt $d\gamma + e \neq 0$ und $f(\bar{x})\,\mathrm{DIV}\,g(\bar{x}) = ((a\gamma + b)y + c)\,\mathrm{DIV}\,((d\gamma + e)y + h)$. Die Differenz zwischen den Polstellen zweier solcher Funktionen f und f' mit Koeffizienten $\leq 2^{2^{2T}}$ ist $\geq \left(2^{2^{2T}}\right)^2 = 2^{2^{2T+1}}$ (*). Wird γ so eingeschränkt, dass $c_2 - c_1 \geq 1/2^{2^{2T+4}}$ (**) gilt, dann nimmt $f(\bar{x})\,\mathrm{DIV}\,g(\bar{x})$ nur 2 Werte an. Da dieser $\{+, -, *_c, \mathrm{DIV}_c\}$-CT nun $2^{2^{T+1}}$ (***) Blätter hat, folgt aus (*), (**), (***) die Abschätzung für

$$c_2 - c_1 \geq \frac{1}{2 \cdot 2^{2^{T+1}} \cdot 2^{2^{2T+4}} \cdot 2^{2^{2T+1}}} \geq \frac{1}{2^{2^{3T}}}.$$

□

Beweis. [von Korollar 2]

Sei $P = \{(x, y) \in \mathbb{N}^2,\ k_1 y < x < k_2 y,\ k_i \in \mathbb{Q},\ \text{Nenner der}\ k_i \leq 2^{2^{2T}}\}$, p sei eine Primzahl $p \leq 2^{2^{2T}} + 1$ und p teilt nicht die Schrittweite a des Gitters.

Betrachte das Gitter, das den Punkt (p,0) enthält. Dann gehören alle Punkte $(k_1 al + p, al)$, bei denen l Vielfache des Nenners von k_1 sind, ebenfalls zum Gitter, d.h. $l = k_1 \cdot l'$.

Wird l' von p geteilt, teilt p auch den $\mathrm{ggT}(k_1 al + p, al)$ und wird l' nicht von p geteilt, gilt $\mathrm{ggT}(k_1 al + p, al) = 1$.

Für $n = 2^{2^{2T+3}}$ sind auf einem Geradenabschnitt in $P \cap \{(0, p) + (a\mathbb{Z})^2\}$ $2^{2^{2T}}$ Punkte (x,y), bei denen abwechselnd $\mathrm{ggT}(k_1 al + p, al) = 1$ und $\mathrm{ggT}(k_1 al + p, al) \neq 1$ gilt. Daher ist die Tiefe eines $\{+, -\}$-CT auf diesem Gitter mindestens $\log\left(2^{2^{2T}}\right) = \frac{1}{8}\log\left(2^{2^{2T+3}}\right) = \frac{1}{8}\log(n)$. Da nach Theorem 3 b) die Tiefe des $\{+, -\}$-CT $\mathcal{O}(T)$ ist, wenn T die Tiefe des $\{+, -, DIV\}$-CT ist, ist T ebenfalls $\Omega(\log n)$. □

4.3 Operationsmenge $\{+, -, *, \mathrm{DIV}\}$

Dieser Abschnitt enthält die Beweise zu Theorem 4 und die daraus folgende spezielle untere Schranke in Korollar 2.

Folgende Bezeichnungen aus [27] für Polynome in 2 Variablen werden erweitert auf Polynome in n Variablen.

Der *Grad* $\deg_{x_i} P$, eines Polynoms $P(x_1, \ldots, x_n)$ in einer Variablen x_i ist der maximale Exponent von x_i, der in einem Monom von $P(x_1, \ldots, x_n)$ vorkommt.

Der *Maximalgrad* maxdeg P ist das Maximum der Grade $\deg_{x_i} P$, nicht zu verwechseln mit der klassischen Definition des Grades (des Totalgrades) eines multivariaten Polynoms.

Die *Höhe* ht(P) von P ist das Maximum der Absolutbeträge der Koeffizienten von P und das *Gewicht* wt(P) von P ist die Summe der Absolutbeträge von P.

Es wird noch die Definition einer lexikographischen Ordnung auf der Menge der Polynome benötigt.

Definition 2. *Für zwei Monome* $cx_1^{i_1} \cdots x_n^{i_n}$ *und* $dx_1^{j_1} \cdots x_n^{j_n}$ *gilt* $cx_1^{i_1} \cdots x_n^{i_n} \succ dx_1^{j_1} \cdots x_n^{j_n}$, *falls*

1) $i_1 > j_1$ oder
2) es ein $k \in \{1, \ldots, n\}$ gibt mit $i_l = j_l$ für l $<$ k und $i_k > j_k$ oder
3) $i_l = j_l$ für alle $l \in \{1, \ldots, n\}$ und $\mid c \mid > \mid d \mid$

Ein Polynom ist in *Normalform*, falls die Darstellung minimal bezüglich der Zahl der Monome ist und diese Monome bezüglich obiger Ordnung absteigend sortiert sind.

Das erste Monom in der Normalform eines Polynoms $P(x_1, \ldots, x_n)$ heißt *Leitmonom* von $P(x_1, \ldots, x_n)$ und der Koeffizient dieses Monoms heißt *Leitkoeffizient* von $P(x_1, \ldots, x_n)$.

Für zwei Polynome $P(x_1, ..., x_n)$ und $Q(x_1, ..., x_n)$ gilt $P(x_1, ..., x_n) \succ Q(x_1, ..., x_n)$, falls in den Normalformen dieser beiden Polynome für ein i $>$ 1 das (i-te Monom von P)$\succ$(i-te Monom von Q) ist und alle vorhergehenden Monome gleich sind.

Lemma 7. *Zu jedem Polynom* $P(x_1, \ldots, x_n)$ *gibt es positive ganze Zahlen* $\pi_1(P)$*und* $\pi_2(P)$, *so dass für alle* $\bar{x} \in \mathbb{N}^n$ *mit* $x_1 > x_i^{\pi_1(P)}$ *und* $x_i > \pi_2(P)$ *das Vorzeichen von* $P(x_1, \ldots, x_n)$ *mit dem Vorzeichen des Leitkoeffizienten von* $P(x_1, \ldots, x_n)$ *übereinstimmt. Für* $\pi_1(P)$ *und* $\pi_2(P)$ *gilt* $\pi_1(P) \leq \sum_{i=2}^{n} d_i + 1$ *und* $\pi_2(P) \leq 3\sqrt[n-1]{\frac{M}{L}}$, *wobei L der Leitkoeffizient von P, M die Höhe von P und* d_i *der Grad von P in* x_i *ist.*

Beweis. Sei $P(x_1,\ldots,x_n) = \Sigma_{i=0}^{d_1} f_i(x_2,\ldots,x_n)x_1^i$ mit $d_i = \deg_{x_i} P$

Behauptung 1 $|P(x_1,\ldots,x_n) - Lx_1^{i_1}\cdots x_n^{i_n}| < |L|x_1^{i_1}\cdots x_n^{i_n}$ *für das Leitmonom* $Lx_1^{i_1}\cdots x_n^{i_n}$

Beweis. Sei M die Höhe von P. Es gilt

$$|P(x_1,\ldots,x_n) - L\,x_1^{i_1}\cdots x_n^{i_n}| <$$

$$M\,x_1^{i_1}\Sigma_{j=0}^{i_2-1}x_2^j\cdots\Sigma_{j=0}^{i_n-1}x_n^j + M\,\Sigma_{j=0}^{i_1-1}x_1^j\Sigma_{j=0}^{d_2}x_2^j\cdots\Sigma_{j=0}^{d_n}x_n^j \leq$$

$$2\max\left\{\mathrm{M\,x_1^{i_1}\Sigma_{j=0}^{i_2-1}x_2^j\cdots\Sigma_{j=0}^{i_n-1}x_n^j}, \mathrm{M\,\Sigma_{j=0}^{i_1-1}x_1^j\Sigma_{j=0}^{d_2}x_2^j\cdots\Sigma_{j=0}^{d_n}x_n^j}\right\}$$

.

Der erste Term beschränkt die Summe über alle Monome, deren Grad in x_1 gleich i_1 und der zweite Term die mit einem Grad in x_1 kleiner i_1.
Da $|L|x_1^{i_1}\cdots x_n^{i_n} > 2M\,x_1^{i_1}\Sigma_{j=0}^{i_2-1}x_2^j\cdots\Sigma_{j=0}^{i_n-1}x_n^j$ für $x_i > 3\sqrt[n-1]{\frac{M}{L}}$ und
$|L|x_1^{i_1}\cdots x_n^{i_n} > 2M\Sigma_{j=0}^{i_1-1}x_1^j\Sigma_{j=0}^{d_2}x_2^j\cdots\Sigma_{j=0}^{d_n}x_n^j$ für $x_1 > x_i^{\sum_{i=2}^n d_i+1}$und $x_i > 3\sqrt[n-1]{\frac{M}{L}}$
gilt, folgt daraus

$$|P(x_1,\ldots,x_n) - L\,x_1^{i_1}\cdots x_n^{i_n}| < |L|x_1^{i_1}\cdots x_n^{i_n}$$

□

Korollar 4 *Zu zwei Polynomen* $P(x_1,\ldots,x_n)$ *und* $Q(x_1,\ldots,x_n)$ *gibt es positive ganze Zahlen* $\pi_1(P-Q)$*und* $\pi_2(P-Q)$*, so dass für alle* $\bar{x} \in \mathbb{N}^n$ *mit* $x_1 > x_i^{\pi_1(P-Q)}$*und* $x_i > \pi_2(P-Q)$ *entweder* $P(x_1,\ldots,x_n) < Q(x_1,\ldots,x_n)$ *oder immer* $P(x_1,\ldots,x_n) \geq Q(x_1,\ldots,x_n)$ *gilt.*

Beweis. [von Theorem 4 a)]

Sei D ein $\{+,-,*,\mathrm{DIV}\}$-CT, der $L \subseteq \mathbb{Z}^n$ erkennt.

Um Theorem 4 zu beweisen, sollen wiederum alle Berechnungsknoten v mit DIV von oben nach unten durch Berechnungsknoten ohne DIV ersetzt werden, in diesem Fall durch rationale Funktionen.

Die folgenden 3 Lemmata erläutern den Ersetzungsvorgang.

Lemma 8. *Seien $P, Q \in \mathbb{Z}[x_1, \ldots, x_n]$ Polynome. Dann gibt es Polynome $A, R \in \mathbb{Z}[x_1, \ldots, x_n]$ mit*

$$P(x_1, \ldots, x_n) = \frac{A(x_1, \ldots, x_n) \cdot Q(x_1, \ldots, x_n) + R(x_1, \ldots, x_n)}{p(x_2, \ldots, x_n)},$$

wobei $Q(x_1, \ldots, x_n) = \Sigma_{j=1}^{d_1} f_j(x_2, \ldots, x_n) x_1^j$, $\delta = max\{$-1,$\deg_{x_1} P - \deg_{x_1} Q\}$, $\deg_{x_1} R < \deg_{x_1} Q$, $\deg_{x_i} R < \deg_{x_i} P + (\delta + 1) \deg_{x_i} Q$ $\deg_{x_1} A \leq max\{0, \delta\}$, $\deg_{x_i} A \leq \deg_{x_i} P + \delta \deg_{x_i} Q$, $\deg_{x_i} p \leq (\delta + 1) \deg_{x_i} Q$, $i > 1$.

Beweis. des Lemmas über Induktion nach δ

Der Induktionsanfang gilt für $\delta = -1$ und $A(\bar{x}) = 0$ und $R(\bar{x}) = Q(\bar{x})$

Für den Induktionsschritt wird angenommen, dass die Induktionsannahme für alle $\delta < k$ für ein $k > -1$ gilt und es wird bewiesen, dass sie dann auch für k gilt.

Seien $P(\bar{x}) = \Sigma_{j=1}^{d} g_j(x_2, \ldots, x_n) x_1^j$ und $Q(\bar{x}) = \Sigma_{j=1}^{m} f_j(x_2, \ldots, x_n) x_1^j$. Dann ist $\delta = d - m$

Betrachte das Polynom

$$S(\bar{x}) = f_m(x_2, \ldots, x_n) P(\bar{x}) - x_1^k g_d(x_2, \ldots, x_n) Q(\bar{x})$$

Da $\deg_{x_1} S < \deg_{x_1} P$, gilt die Induktionsannahme für die beiden Polynome $S(\bar{x})$ und $Q(\bar{x})$. Daher gilt nach Induktionsannahme für geeignete $\tilde{A}, \tilde{R}$ mit ganzzahligen Koeffizienten, $\deg_{x_1}(\tilde{R}) < \deg_{x_1}(Q)$

$$S(\bar{x}) = \frac{\tilde{A}(\bar{x}) \cdot Q(\bar{x}) + \tilde{R}(\bar{x})}{(f_m(x_2, \ldots, x_n))^{\delta}}.$$

Ersetzt man nun für $S(\bar{x})$ obigen Quotienten

$$P(\bar{x}) = \frac{\tilde{A}(\bar{x}) \cdot Q(\bar{x}) + (f_m(x_2, \ldots, x_n))^{\delta} x_1^k g_d(x_2, \ldots, x_n Q(\bar{x}) + \tilde{R}(\bar{x})}{(f_m(x_2, \ldots, x_n))^{\delta+1}}$$

$$= \frac{A(\bar{x}) \cdot Q(\bar{x}) + \tilde{R}(\bar{x})}{(f_m(x_2, \ldots, x_n))^{\delta+1}}$$

Für die Grade gelten, da $\deg_{x_i} S \leq \deg_{x_i} P + \deg_{x_i} Q$ und $\deg_{x_i} \tilde{A} \leq \deg_{x_i} S + (\delta - 1)\deg_{x_i} Q$,

$$\deg_{x_i} A = \max\{\deg_{x_i} \tilde{A}, \delta \deg_{x_i} Q + \deg_{x_i} P\} \leq \deg_{x_i} P + \delta \deg_{x_i} Q,$$

$$\deg_{x_1} A = \max\{\deg_{x_1} \tilde{A}, \deg_{x_1} P - \deg_{x_1} Q\} \leq \deg_{x_1} P - \deg_{x_1} Q.$$

□

Lemma 9. *Seien $P, Q \in \mathbb{Z}[x_1, \ldots, x_n]$ Polynome. Dann gibt es Polynome $A_1 \in \mathbb{Z}[x_1, \ldots, x_n]$ und $A_2, p \in \mathbb{Z}[x_2, \ldots, x_n]$ und Konstanten π_1 und π_2, so dass für alle $\bar{x} \in \mathbb{N}^n$ mit $x_1 > x_i^{\pi_1}$, $x_1 \equiv c \bmod p(x_2, \ldots, x_n)$ und $x_i > \pi_2$*

$$P(\bar{x}) \text{ DIV } Q(\bar{x}) = \frac{A_1(x_1, \ldots, x_n)}{p(x_2, \ldots, x_n)} + \left\lfloor \frac{A_2(x_2, \ldots, x_n)}{p(x_2, \ldots, x_n)} \right\rfloor$$

gilt.

Beweis. [von Lemma 9]

Nach Lemma 8 gilt

$$P(x_1, \ldots, x_n) \text{ DIV } Q(x_1, \ldots, x_n) =$$

$$\left\lfloor \frac{A(x_1, \ldots, x_n)}{p(x_2, \ldots, x_n)} + \frac{R(x_1, \ldots, x_n)}{p(x_2, \ldots, x_n) \cdot Q(x_1, \ldots, x_n)} \right\rfloor .$$

Nun kann gefolgert werden

(i) Sei $k_1 = \sum_{i=2}^{n} d_i + 1$ mit $d_i = deg_{x_i}(Q - R)$.
Für $x_1 > x_j^{k_1}$, $j = 2, ..., n$, konvergiert $\frac{R(x_1,\ldots,x_n)}{(p(x_2,\ldots,x_n)\cdot Q(x_1,\ldots,x_n)}$ gegen 0, wenn $\| (x_1, ..., x_n) \|_1$ gegen ∞ strebt.

(ii) Sei $c \in \mathbb{Z}$ fest. Für $x \equiv c \bmod p(x_2, \ldots, x_n)$ gilt $\frac{A(x_1,\ldots,x_n)}{p(x_2,\ldots,x_n)} = \frac{A_1(x_1,\ldots,x_n)}{p(x_2,\ldots,x_n)} + \frac{A_2(x_2,\ldots,x_n)}{p(x_2,\ldots,x_n)}$ und $\frac{A_1(x_1,\ldots,x_n)}{p(x_2,\ldots,x_n)} \in \mathbb{Z}$.

(i) folgt direkt aus den Definitionen von R, p, und k_1.

(ii) kann folgendermaßen festgestellt werden:

Sei $x_1 \equiv c \bmod p(x_2, \ldots, x_n)$, d.h. $x_1 = l \cdot p(x_2, \ldots, x_n) + c$ für ein $l \in \mathbb{N}$. Dann gilt

$$\frac{A(x_1, \ldots, x_n)}{p(x_2, \ldots, x_n)} = \frac{\sum_{k_1,\ldots,k_n=0,}^{d_1,\ldots,d_n} a'_{k_1,\ldots,k_n}(x_1 - c)^{k_1} x_2^{k_2 \cdots} x_n^{k_n}}{p(x_2, \ldots, x_n)}$$

$$= \sum_{k_1,\ldots,k_n=0,}^{d_1,\ldots,d_n} a'_{k_1,\ldots,k_n}(l^{k_1}(p(x_2, \ldots, x_n))^{k_1-1} \cdot x_2^{k_2 \cdots} x_n^{k_n}$$

$$+ \frac{\sum_{k_2,\ldots,k_n=0,}^{d_2,\ldots,d_n} a'_{0,k_2,\ldots,k_n} x_2^{k_2 \cdots} x_n^{k_n}}{p(x_2, \ldots, x_n)}$$

$$=: \frac{A_1(x_1, \ldots, x_n)}{p(x_2, \ldots, x_n)} + \frac{A_2(x_2, \ldots, x_n)}{p(x_2, \ldots, x_n)}$$

wobei die $a'_{k_1,\ldots,k_n}$ von c abhängige Koeffizienten sind. □

Mit Lemma 9 ist zwar immer noch die ganzahlige Division in dem Ausdruck, aber nur noch als Quotient zweier Polynome mit n-1 Variablen, d.h. mit einer Variablen weniger. Wird dieses Lemma 9 nun (n-1)-mal angewendet und in jedem Durchgang mit einer Variablen weniger im Nennerpolynom, so erhält man folgendes Lemma

Lemma 10. *Seien $P, Q \in \mathbb{Z}[x_1, \ldots, x_n]$ Polynome, $c_1, ..., c_n \in \mathbb{N}$ Konstanten. Dann gibt es Konstanten $k_i, K \in \mathbb{N}$ und Polynome $A_i \in \mathbb{Z}[x_{1i}, \ldots, x_n]$, $P_i \in \mathbb{Z}[x_{i+1}, \ldots, x_n]$, i=1,...,n-1 ,$P_n \in \mathbb{Z}$, so dass für alle $\bar{x} \in \mathbb{N}^n$ mit $x_i > x_{i+1}^{k_i}$, $x_n > K$, $x_i \equiv c_i \bmod P_{i+1}(x_{i+1}, \ldots, x_n)$*

$$P(\bar{x}) \text{ DIV } Q(\bar{x}) = \sum_{i=1}^{n-1} \frac{A_i(x_i, \ldots, x_n)}{P_i(x_{i+1}, \ldots, x_n)} + A_n(x_n)$$

gilt.

Beweis. [von Lemma 10]

Nach dem (n-1). Durchgang ist bei dem letzten Summanden noch die DIV-Operation

$$\sum_{i=1}^{n-1} \frac{A_i(x_i, \ldots, x_n)}{P_i(x_{i+1}, \ldots, x_n)} + \left\lfloor \frac{A'_n(x_n)}{P_{n-1}(x_n)} \right\rfloor .$$

Wie in Lemma 1 gezeigt, kann $A'_n(x) \,\mathrm{DIV}\, P_{n-1}(x_n)$ als Polynom $A_n(x_n)$ für Eingaben $x_n \equiv c_n \bmod P_n$, $x_n > K$, für genügend großes K geschrieben werden. □

Jede DIV-Operation in dem CT D kann nun durch die Berechnung einer rationalen Funktion ersetzt werden, wie in Lemma 10 gezeigt.

Wird nun von der Wurzel zu den Blättern jeder DIV- Berechnungsknoten durch die rationale Funktion als Quotient zweier Polynome mit ganzzahligen Koeffizienten ersetzt, bei der Addition und Subtraktion zweier rationaler Funktionen nach der Rechenregel $\frac{P}{Q} + (-)\frac{S}{T} = \frac{PT+(-)SQ}{QT}$ und bei der Multiplikation nach der Rechenregel $\frac{P}{Q} \cdot \frac{S}{T} = \frac{P \cdot S}{QT}$ wiederum als Quotient zweier Polynome mit ganzzahligen Koeffizienten umgeformt, so werden nur Polynome mit ganzzahligen Koeffizienten berechnet. Wenn nun an einem Verzweigungsknoten nach $\frac{P}{Q} > 0$ verzweigt wird, wird für $P > 0$ und $Q > 0$ oder aber $P < 0$ und $Q < 0$ der Zweig für „$\frac{P}{Q} > 0$ ist erfüllt" und ansonsten der für „$\frac{P}{Q} > 0$ ist nicht erfüllt" gewählt.

Dies ergibt einen $\{+, -, *\}$-CT, der für eine wie in Theorem 4 eingeschränkte Eingabemenge L erkennt. Die Polynome $p_i(x_{i+1}, \ldots, x_n)$ sind die Produkte aller Polynome P_i, die als Divisor bei der Ersetzung der DIV-Operationen wie in Lemma 10 vorkommen.

□

Beweis. [von Theorem 2 b)]

Durch Induktion über die Tiefe des Berechnungsbaumes werden die Grade der rationalen Funktionen $\sum_{i=1}^{n-1} \frac{A_i(x_i,\ldots,x_n)}{P_i(x_{i+1},\ldots,x_n)} + A_n(x_n) = \frac{S(\bar{x})}{T(\bar{x})}$ und die Größe der k_i abgeschätzt.

Behauptung 2 *Nach der Substitution der DIV-Operationen in den oberen t Levels von D sind die Grade der Zähler- und Nennerpolynome der in Lemma 10 definierten rationalen Funktionen* $\leq 2^{2^{(n-1)(t-1)}}$.

Beweis. der Behauptung

Der Induktionsanfang für $t = 1$ ist offensichtlich.

Die Induktionsannahme gelte für alle $k < t$.

Bei der Umformung der Berechnungsknoten, die mit $\frac{P(\bar{x})}{Q(\bar{x}))}$ + (-) $\frac{S(\bar{x})}{T(\bar{x}))}$ beschriftet sind, zu einem Quotienten zweier Polynome verdoppelt sich der Grad der Polynome höchstens.

Daher müssen nur noch Berechnungsknoten in der Tiefe t, die mit $P(\bar{x})\mathrm{DIV}Q(\bar{x}) = \sum_{i=1}^{n-1} \frac{A_i(x_i,\ldots,x_n)}{P_i(x_{i+1},\ldots,x_n)} + A_n(x_n) = \frac{S(\bar{x})}{T(\bar{x})}$ beschriftet sind, genauer betrachtet werden.

Nach Induktionsannahme gelten dort folgende Ungleichungen:
$\deg P \leq 2^{2^{(n-1)(t-2)}}$, $\deg Q \leq 2^{2^{(n-1)(t-2)}}$ und $\deg P_i \leq 2^{2^{(n-1)(t-2)}}$.

Nach Lemma 8 gilt für $\delta = \max\{\text{-}1, \deg_{x_1} P - \deg_{x_1} Q\}$

$\deg_{x_1} A_i \leq \max\{0, \delta\} \leq 2^{2^{(n-1)(t-2)}}$ und

$\deg_{x_i} A_j \leq \deg_{x_i} P + (\delta + 1)^j \deg_{x_i} Q$,

$\deg_{x_i} P_{j+1} \leq (\delta + 1)^j \deg_{x_i} Q \leq (2\delta)^j 2^{2^{(n-1)(t-2)}} \qquad j \geq i$

und $\deg_{x_1} S \leq \deg_{x_1} Q \leq 2^{2^{(n-1)(t-2)}}$.

Daher gilt

$$\deg_{x_i} T \leq \sum_{j=0}^{n-1} \deg_{x_i} P_{j+1} \leq 2^{2^{(n-1)(t-2)}} \sum_{j=1}^{n-1} (2\delta)^j$$

$$\leq 2^{2^{(n-1)(t-2)}} (2 \cdot 2^{2^{(n-1)(t-2)}})^n \leq 2^{2^{(n-1)(t-1)}}$$

und ebenso $\deg_{x_i} S \leq 2^{2^{(n-1)(t-1)}}$ und $\deg_{x_i} S \leq 2^{2^{(n-1)(t-1)}}$. □

Da es höchstens 2^t Knoten gibt, ist der Grad des Produkts dieser Polynome höchstens $2^{2^{(n-1)t}}$.

Aus der obigen Induktion und von Lemma 7 und Korollar 4 folgt unmittelbar $k_i \leq 2^{2^{(n-1)t}}$. □

Das Theorem 4 b) im 2-dimensionalen Fall wird nun benutzt, um die erste untere Schranke für die mächtige Operationsmenge $\{+, -, *, \text{DIV}\}$ und Konstanten aus $\mathbb{Q}$ zu beweisen.

Beweis. [von Korollar 3]

Angenommen ein $(\{+, -, *, \text{DIV}\}, \mathbb{Q})$-CT M der Tiefe T erkennt L_r.
Theorem 2 b) besagt, dass es k_2 und ein Polynom $p(y)$ vom Grad $\leq 2^{2^T}$ gibt, so dass es zu $y \in k_2\mathbb{Z}$, $x_1(y); x_2(y)$, $x_1(y) < x_2(y), x_2(y) - x_1(y) \leq p(y), x_1(y) < r(y) < x_2(y)$ gibt, so dass M $A = \{x_1(y), y \in k_2\mathbb{Z}\}$ akzeptiert und $B = \{x_2(y), y \in k_2\mathbb{Z}\}$ verwirft, d.h. $r(y)$ trennt A von B.

Für $d \leq \max\{k_1, \deg(p(y))\}$ folgt die untere Schranke direkt aus Theorem 4 b).

Für $d > \max\{k_1, \deg(p(y))\}$ erkennt der $(\{+, -, *\}, \mathbb{Q})$-CT D' aus Theorem 4 b) L auch für Eingaben aus $A \cup B$. Daher berechnet D' ein Polynom $s(y)$, das A von B trennt. Da $s(y) \in \{x_1(y), \ldots, x_2(y)\} \subseteq \{r(y) - p(y), \ldots, r(y) + p(y)\}$ und $d > \deg(p(y))$, hat $s(y)$ Grad d. Da D' nur Polynome vom Grad $\mathcal{O}(2^{2^T})$ berechnet, gilt $d = \mathcal{O}(2^{2^T})$. □

4.4 Separationsresultate

Dieser Abschnitt enthält die Beweise und die fehlenden Sprachen, um die Sprachklassen zu unterscheiden.

Beweis. [von Theorem 5]

$CC_2\{+, -\} \subsetneq CC_2\{+, -, *\}$ gilt offensichtlich, da bekannterweise $L = \{(x, y), x^2 > y\} \in CC_2\{+, -, *\} - CC_2\{+, -\}$.

Die folgenden Unterscheidungen der Sprachklassen folgen bereits aus [22], sie gelten schon im Fall $n = 1$.

$CC_n(\{+,-\}) \subsetneqq CC_n\{+,-,\mathrm{DIV}_c\}$

$CC_n\{+,-,\mathrm{DIV}_c\} - CC_n\{+,-,*\} \neq \emptyset$

$CC_n\{+,-,*\} \subsetneqq CC_n\{+,-,*,\mathrm{DIV}_c\}$

Aus Theorem 2 folgt:

$CC_n\{+,-,DIV_c\} \subsetneqq CC_n\{+,-,*,\mathrm{DIV}_c\}$.

Das folgende Korollar der Theoreme 4 und 2 beweist die fehlenden Beziehungen.

Korollar 5

a) $\{(x,y) \in \mathbb{Z}^2, x^2 > 2y^2\} \in CC_2(\{+,-,*\}) - CC_2(\{+,-,\mathrm{DIV}\})$

b) $\{(x,y) \in \mathbb{Z}^2, (y-1)\mathrm{DIV}\, x - y\,\mathrm{DIV}\, x < 0\} \in CC_2(\{+,-,\mathrm{DIV}\}) - CC_2(\{+,-,*,$ $\mathrm{DIV}_c\})$ ″

Beweis. zu a) durch Widerspruch

Nach der Definition von L gilt $L \in CC_2(\{+,-,*\}$.

Es wird gezeigt, dass unter der Annahme, dass ein $\{+,-,\mathrm{DIV}\}$-CT diese Sprache erkennt, sie bereits von einem $\{+,-\}$-CT erkannt wird.

Nach Theorem 4 gibt es k_1, $k_2 \in \mathbb{Q}$, so dass L auf $\{(x,y) \in \mathbb{N}^2 \mid k_1 y < x < k_2 y\} \cap (a\mathbb{Z})^2$, dem Gitter, das den Nullpunkt enthält, bereits durch einen $\{+,-\}$-CT erkannt wird. Da $\sqrt{2} \notin \mathbb{Q}$ ist, können k_1, $k_2 \in \mathbb{Q}$ so gewählt werden, dass $k_1 < \sqrt{2} < k_2$ gilt. Folgendermaßen wird dann L durch einen $\{+,-\}$-CT erkannt: akzeptiere, falls $|x| \leq k_1 |y|$, und verwerfe, falls $|x| \geq k_2 |y|$.

Für $k_1 |y| < |x| < k_2 |y|$ entscheide mit dem obigen $\{+,-\}$-CT, ob $(a|x|, a|y|) \in L$ gilt. Falls ja, akzeptiere, ansonsten verwerfe. Dies ist im Widerspruch zu L$\notin CC_2(\{+,-\})$.

zu b) durch Widerspruch

Nach der Definition von L gilt $L \in CC_2(\{+,-,\mathrm{DIV}\}$.

Annahme L $= \{(x,y) \in \mathbb{Z}^2, (y-1)DIV\, x - y\,\mathrm{DIV}\, x < 0\} \in CC_2(\{+,-,*,$ $\mathrm{DIV}_c\})$

Nach Theorem 2 b) ist der $\mathbb{Z}^2$ endliche disjunkte Vereinigung affiner Gitter und L kann auf jedem dieser Gitter ohne DIV_c entschieden werden. Dies gilt insbesondere für das Gitter S, das den Nullpunkt enthält. Nach Theorem 2 b) gibt es einen $\{+,-,*\}$-CT, der L auf $S \cap \mathbb{Z}^2$ entscheidet.

Da S den Nullpunkt enthält, ist $S = (a,0)\mathbb{Z}+(0,\mathrm{a})\mathbb{Z}$.

Sei nun (x,y)$\in S$, d.h. (x,y)=(ax',ay'). Da $(y-1)\mathrm{DIV}\, x - y\,\mathrm{DIV}\, x < 0 \Leftrightarrow y \bmod x = 0$, wird $ay' \bmod ax' = 0 \Leftrightarrow y' \bmod x' = 0$ ohne DIV entschieden. Dies ist im Widerspruch zu L$\notin CC_2(\{+,-,*\})$.

□

5 Polynomauswertung

Diese beiden folgenden Kapitel entsprechen weitestgehend dem Aufsatz von Martin ZIEGLER und mir in [26]. Anders als in den Kapiteln zu Berechenbarkeits - und Komplexitätsbetrachtungen von S-CTs werden in den Kapiteln 5 und 6 die einzelnen Ergebnisse direkt mit den dazugehörigen Beweisen vorgestellt. Es wird in diesen beiden Kapiteln vorausgesetzt, dass außer, wenn ein Algorithmus explizit als SLP angegeben wird, bei den vorgestellten Algorithmen „$\leq$" zur Operationsmenge gehört.

Aus der unteren Schranke über $\{+,-,*,\mathrm{DIV}\}$ ergibt sich eine doppelexponentielle Lücke zwischen oberer Schranke zur Polynomauswertung über $\{+,*\}$ und unterer Schranke zur Polynomauswertung über $\{+,-,*,\mathrm{DIV}\}$. Daher stellt sich in diesem Kapitel die Frage, ob die Polynomauswertung mit zusätzlichen nicht klassischen Grundoperationen beschleunigt werden kann, ob z.B. Polynome vom Grad d über $\{+,-,*,\mathrm{DIV}\}$ in $o(d)$ berechnet werden können. Da gerade die Polynomauswertung in den Computerwissenschaften an vielen Stellen benutzt wird, z.B. bei Splines oder Reed-Solomon-Codes, ist deren Beschleunigung wünschenswert.

5.1 Polynome mit einer Variablen

Das klassische Verfahren zur Polynomauswertung bei univariaten Polynomen ist das Hornerschema und benötigt $\mathcal{O}$(d) Schritte, wenn d den Grad des Po-

lynoms bezeichnet. Dieses Verfahren ist in vielen Fällen mit den Operationen $\{+,-,*\}$ optimal [8, Theorem 6.5]. Dies gilt nicht für ganzzahlige Polynome mit im Verhältnis zum Grad kleinen Koeffizienten.

Theorem 6. *Gegeben sei ein Polynom* $P(x) = \Sigma_{n=0}^{\mathrm{d}-1} a_n x^n \in \mathbb{Z}[\mathrm{x}]$ *mit* $a_0, ..., a_{d-1} \in \mathbb{Z}$ *und* $|a_n| < P$. *Dann kann P(x) in* $\mathcal{O}(\mathrm{d}/\log_{\mathrm{p}}\mathrm{d})$ *Schritten über* $\{+,*,\}$ *berechnet werden.*

Beweis. Da P(x) als Differenz zweier Polynome mit positiven Koeffizienten betrachtet werden kann, sei o.B.d.A. $P(x) \in \mathbb{N}[\mathrm{x}]$. Zerlege P in $\lceil d/k \rceil$ Polynome q_i vom Grad kleiner gleich k. Wie dieses k in Abhängigkeit vom Grad d und der maximalen Größe der Koeffizienten P zu wählen ist, wird später gezeigt. Es gibt höchstens P^{k} verschiedene Polynome vom Grad kleiner k und mit Koeffizienten aus {0,1,..,P-1}. Mit dem Hornerschema wird jedes dieser P^{k} verschiedenen Polynome an der Stelle $x \in \mathbb{Z}$ in $\mathcal{O}(\mathrm{k})$ Schritten berechnet, d.h. es werden für alle P^{k} verschiedenen Polynome insgesamt $\mathcal{O}(\mathrm{k} \cdot \mathrm{P}^{\mathrm{k}})$ Schritte zur Auswertung benötigt.

Danach wird das Polynom $\Sigma_{i=0}^{\lceil \mathrm{d}/\mathrm{k} \rceil} q_i(x) \cdot Y^i$ an der Stelle $Y = x^k$ mit dem Hornerschema in $\mathcal{O}(\mathrm{d/k})$ Schritten berechnet und erhält so P(x).

Es werden zur Berechnung also $\mathcal{O}(\mathrm{d/k} + \mathrm{k} \cdot \mathrm{P}^{\mathrm{k}})$ Schritte benötigt.
Für k:= $\log_p$d-2 $\log_p\log_p$d ist

$$\mathcal{O}(\mathrm{d/k} + \mathrm{k} \cdot \mathrm{P}^{\mathrm{k}}) =$$

$$\mathcal{O}(\mathrm{d}/(\log_p \mathrm{d} - 2\log_p\log_p \mathrm{d}) + (\log_p \mathrm{d} - 2\log_p\log_p \mathrm{d}) \cdot \mathrm{P}^{\log_p \mathrm{d} - 2\log_p\log_p \mathrm{d}})$$

$$= \mathcal{O}(\mathrm{d}/\log_{\mathrm{p}}\mathrm{d})$$

□

Wird als weitere Operation die ganzzahlige Division hinzugenommen, so kann die Berechnung eines Polynoms über einem endlichen Bereich beschleunigt werden.

Satz 3 *Zu* $X \in \mathbb{N}$ *und* $Z > \max_{0 \le x \le X} P(x)$ *speichere* $Y := \sum_{x=0}^{X} Z^x \cdot P(x)$. *Folgendermaßen wird für* $0 \le x \le X$, *aus den gespeicherten Werten* $P(x)$

berechnet: durch wiederholtes Quadrieren in $\mathcal{O}(\log x)$ *Schritten berechne* Z^x. *Es gilt* $P(x) = (Y \,\mathrm{DIV}\, Z^x) \,mod\, Z$.

Die Laufzeit ist unabhängig von $\deg(P)$ und logarithmisch in x.

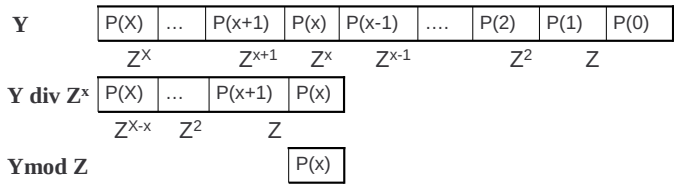

Abbildung 6. Berechnungen im Beweis von Satz 3

Aber auch die Abhängigkeit von der Eingabe ist zur Berechnung nicht notwendig [6].

Theorem 7. *Ein Polynom* $P(x) \in \mathbb{Z}\,[\mathrm{X}]$ *kann über einer endlichen Menge* $\mathrm{D} \subset \mathbb{Z}$ *in 15 Schritten (unabhängig von P und D) berechnet werden.*

Beweis. Für negative Argumente $x \in \mathbb{Z}$ berechne P(-x). Da jedes Polynom mit ganzzahligen Koeffizienten als Differenz zweier Polynome mit natürlichen Koeffizienten dargestellt werden kann, wird o.B.d.A. der Beweis für Polynome mit natürlichen Koeffizienten und $\mathrm{D} \subset \mathbb{N}$ geführt.

Sei $P(x) = \sum_{i=0}^{d} p_i x^i$, $p_i \in \mathbb{N}$, $\deg p = d$ und $\mathrm{wt}(P) \leq P$.

Behauptung 3 *Für* $Z > max\{X^d \cdot P, (X^{d+1}+1) \cdot X\}$ *berechnet der folgende Algorithmus p(x) für* $|x| \leq X = max\{|x|\,, x \in D\}$.

Z, Z^d *und p(Z) sind Konstanten und werden vorher berechnet und gespeichert.*

$g = Z^{d+1}\mathrm{DIV}\,(Z - x)$	*2 Operationen*
$h = p(Z) \cdot g$	*1 Operation*
$a = h \,\mathrm{DIV}\, Z^d$	*1 Operation*
$b = a \bmod Z = a - (a \,\mathrm{DIV}\, Z) \cdot Z$	*3 Operationen*

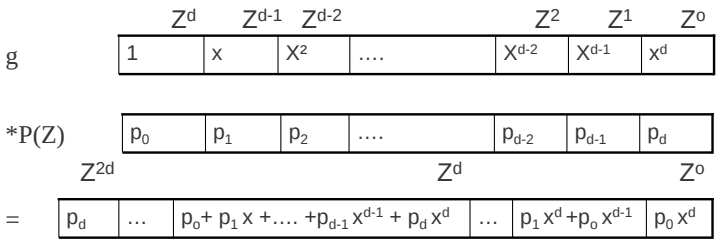

Abbildung 7. Berechnungen in Bshoutys Algorithmus

Beweis. Um die Korrektheit des Algorithmus zu beweisen, folge ich dem Beweis von Nader BSHOUTY [6].

Für x ∈{1,...,X} gilt b=p(x)

(*)

$$g(x) = \left\lfloor \frac{Z^{\mathrm{d}+1}}{Z-x} \right\rfloor = \left\lfloor \frac{Z^{\mathrm{d}}}{1-\frac{x}{Z}} \right\rfloor = \left\lfloor Z^d \cdot \sum_{i=0}^{\infty} \left(\tfrac{x}{Z}\right)^i \right\rfloor =$$

$$\left\lfloor \underbrace{\sum_{i=0}^{d} Z^{d-i} \cdot x^i}_{\in \mathbb{N}} + \underbrace{\sum_{i=d+1}^{\infty} Z^d \cdot \left(\frac{x}{Z}\right)^i}_{= \frac{x^{\mathrm{d}+1}}{Z-x} < 1} \right\rfloor = \sum_{i=0}^{d} Z^{d-i} \cdot x^i,$$

da laut Voraussetzung $Z > (X^{d+1}+1) \cdot X$ gilt.

$$h(x) = p(Z) \cdot \sum_{i=0}^{d} Z^{d-i} \cdot x^i =$$

$$\sum_{j=0}^{d} p_j \cdot Z^j \cdot \sum_{i=0}^{d} Z^{d-i} \cdot x^i = \sum_{i=0}^{d} (\sum_{j=0}^{d} p_j \cdot Z^{d-i+j}) \cdot x^i$$

$$a(x) = h(x) \,\mathrm{DIV}\, Z^d = \sum_{i=0}^{d} (\sum_{j=0}^{d} p_j Z^{d-i+j}) \cdot x^i \,\mathrm{DIV}\, Z^d =$$

$$\left\lfloor \sum_{i=0}^{d} (\sum_{j=0}^{d} p_j Z^{-i+j}) \cdot x^i \right\rfloor = \sum_{i=0}^{d} (\sum_{0 \le i \le j}^{d} p_j \cdot Z^{-i+j}) \cdot x^i.$$

Da für $i \neq j$ die Summanden ganzzahlig sind und laut Voraussetzung $Z > X^d \cdot \mathrm{wt}(P)$ ist, gilt

$$b(x) = a(x) \bmod Z = \sum_{j=0}^{d} p_j \cdot x^j = p(x)$$

□

Korollar 6 *Mit der Operationsmenge* $\{+, -, *_c, \mathrm{DIV}\}$ *kann jede endliche Folge* $y_0, y_1, \ldots, y_N$ *(oder formaler die Abbildung* $\{0, 1, \ldots, N\} \ni n \mapsto y_n$*) in konstanter Zeit* unabhängig *von (der Länge N) der Folge erkannt werden.*

Beweis. Betrachte das Interpolationspolynom $p \in \mathbb{Q}[X]$ vom Grad $\leq N + 1$, so dass $p(n) = y_n$, $n \in \{0, \ldots, N\}$ gilt. Wähle $M \in \mathbb{N}$, so dass $M \cdot p \in \mathbb{Z}[X]$. Nach Theorem 7 kann $n \mapsto M \cdot p(n)$ in konstanter Zeit berechnet werden. Nach ganzzahliger Division durch M erhält man $p(n)$. □

Aus diesem Korollar folgt für eine Sprache $L \subseteq \mathbb{Z}$ ebenfalls Bemerkung 2, wenn es auf die charakteristische Folge $(y_0, \ldots, y_N)$ von L, die durch $y_n := 1$ für $n \in L$ und $y_n := 0$ für $n \notin L$ definiert ist, angewendet wird.

Wie der nächste Abschnitt zeigt, gelten diese Aussagen auch für Folgen $(\bar{y_0}, \ldots, \bar{y_N})$ in $\mathbb{Z}^d$ und für endliche Sprachen $L \subseteq \mathbb{Z}^d$ für ein festes d.

5.2 Polynome mit mehreren Variablen

Dieser Algorithmus von Nader BSHOUTY wird erweitert zu einem Algorithmus für multivariate Polynome.

Theorem 8. *Ein Polynom* $P(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ *kann über einer endlichen Menge* $D \subset \mathbb{Z}^n$ *in* $\mathcal{O}(n)$ *Schritten unabhängig von p und D mit der Operationsmenge* $\{+, -, *, \mathrm{DIV}\}$ *berechnet werden.*

Beweis. Um nur Polynome für nicht-negative Argumente $\bar{x} = (x_1, \ldots, x_n) \in \mathbb{N}^n$ zu berechnen, wird zu einer Eingabe $\bar{x} \in \mathbb{Z}^n$ zunächst

in $\mathcal{O}(\mathrm{n})$ Schritten entschieden, welches der 2^{n} im Allgemeinen verschiedenen Polynome $P(\pm x_1, \pm x_2, \ldots, \pm x_n)$ an der Stelle $(|x_1|, |x_2|, \ldots, |x_n|)$ berechnet werden muss, um den Wert von $P(\bar{x})$ zu berechnen.
Sei o.B.d.A. $P(\bar{x}) = \Sigma_{i_1,\ldots,i_n=0}^{\mathsf{d}_1,\ldots,\mathsf{d}_n} p_{i_1,\ldots,i_n} x_1^{i_1} \cdots x_n^{i_n} \in \mathbb{N}[\mathrm{x}_1, \ldots, \mathrm{x}_n]$, andernfalls betrachte wie im univariaten Fall , falls $P(\bar{x}) \in \mathbb{Z}[\mathrm{x}_1, \ldots, \mathrm{x}_n]$, $P(\bar{x})$ wiederum als Differenz zweier Polynome mit natürlichen Koeffizienten.

Behauptung 4 *Der folgende Algorithmus berechnet* $f(\bar{\mathrm{x}}) = P(\bar{x})$ *für* $\bar{x} \in D$, *falls* $Z > d^n \cdot \mathrm{wt}\,(P) \cdot max\,\{x_i; i = 1, ..., n\}^{nd}$

mit $d := max\,\{d_i; i = 1, ..., n\} + 1$, $d_i = \deg_{x_i} P(\bar{x})$ *gilt.*

$g(\bar{\mathrm{x}}) = \prod_{i=1}^{n} Z^{d^i} \,\mathrm{DIV}\, (Z^{d^{i-1}} - x_i)$ *3n Operationen*

$h(\bar{\mathrm{x}}) = P(Z, Z^{d^i}, ..., Z^{d^{n-1}}) \cdot g(\bar{\mathrm{x}})$ *1 Operation*

$k(\bar{\mathrm{x}}) = h(\bar{\mathrm{x}}) \,\mathrm{DIV}\, Z^{d^n - 1}$ *1 Operation*

$f(\bar{\mathrm{x}}) = k(\bar{\mathrm{x}}) \bmod Z = k(\bar{\mathrm{x}}) - (k(\bar{\mathrm{x}}) \,\mathrm{DIV}\, Z) \cdot Z$ *3 Operationen*

Beweis. der Behauptung

Es gilt die Gleichung (*) auf Seite 42 $\left\lfloor \frac{Z^{\mathrm{d}}}{Z - x} \right\rfloor = \sum_{i=0}^{d-1} Z^{d-1-i} \cdot x^i$ im univariaten Fall für alle $Z > \Omega(x^d)$.

Wird diese Gleichung auch auf x_2 und $Z_2 := Z^d$ angewendet, führt dies zur folgenden Gleichung

$$\left(Z^{d^2} \,\mathrm{DIV}\, (Z^d - x_2)\right) \cdot \left(Z^d \,\mathrm{DIV}\, (Z - x_1)\right) \quad =$$

$$\sum_{i_2=0}^{d-1} Z^{d(d-1-i_2)} \cdot x_2^{i_2} \cdot \sum_{i_1=0}^{d-1} Z^{d-1-i_1} \cdot x_1^{i_1} =$$

$$\sum\nolimits_{i_1,i_2=0}^{d-1} Z^{d^2-1-(di_2+i_1)} \cdot x_2^{i_2} \cdot x_1^{i_1}.$$

Induktiv für $i = 1, ..., n$ wird diese Gleichung auf x_i und $Z_i := Z^{d^{i-1}}$ angewendet und man erhält in $\mathcal{O}(\mathrm{n})$ Schritten über $\{+, -, *, \mathrm{DIV}\}$

$$g(\bar{\mathrm{x}}) = \prod_{i=1}^{n} Z^{d^i} \,\mathrm{DIV}\, (Z^{d^{i-1}} - x_i) =$$

$$\sum_{i_1,\ldots,i_n=0}^{d-1} Z^{d^n-1-(d^{n-1}i_n+\cdots+di_2+i_1)} \cdot x_n^{i_n} \cdots x_2^{i_2} \cdot x_1^{i_1}$$

Diese Summe wird nun mit der Konstanten

$$P(Z, Z^d, Z^{d^2}, \ldots, Z^{d^{n-1}}) \quad = \quad \sum_{i_1,\ldots,i_n=0}^{d-1} p_{\bar{i}} \cdot Z^{i_1+di_2+d^2i_3+\cdots+d^{n-1}i_n}$$

multipliziert.

$$h(\bar{\mathrm{x}}) = P(Z, Z^{d^i}, ..., Z^{d^{n-1}}) \cdot g(\bar{\mathrm{x}})$$

$$\sum_{k_1,\ldots,k_n=0}^{d-1} p_{\bar{k}} Z^{k_1+dk_2+\cdots\cdots+d^{n-1}k_n} \cdot \sum_{i_1,\ldots,i_n=0}^{d-1} Z^{d^n-1} \cdot Z^{-i_1} \cdots Z^{-d^{n-1}i_n} \cdot x_1^{i_1} \cdots x_n^{i_n}$$

$$\sum_{k_1,\ldots,k_n,i_1,\ldots,i_n=0}^{d-1} p_{\bar{k}}\, Z^{d^n-1} \cdot Z^{(k_1-i_1)} \cdots Z^{(d^{n-1}k_n-d^{n-1}i_n)} \cdot x_1^{i_1} \cdots x_n^{i_n}$$

Wie im Fall einer Variablen wird wiederum durch Anwendung der ganzzahligen Division und anschließender Restbildung $\mathrm{P}(\mathrm{x})$ extrahiert.

$$k(\bar{\mathrm{x}}) = h(\bar{\mathrm{x}})\,\mathrm{DIV}\, Z^{d^n-1} =$$

$$\Big(\sum_{k_1,\ldots,k_n,i_1,\ldots,i_n=0}^{d-1} p_{\bar{k}}\, Z^{d^n-1} \cdot Z^{(k_1-i_1)} \cdots Z^{d^{n-1}(k_n-i_n)} \cdot x_1^{i_1} \cdots x_n^{i_n}\Big)\mathrm{DIV}\, Z^{d^n-1} =$$

$$\left\lfloor \sum_{k_1,\ldots,k_n,i_1,\ldots,i_n=0}^{d-1} p_{\bar{k}} \cdot Z^{(k_1-i_1)} \cdots Z^{d^{n-1}(k_n-i_n)} \cdot x_1^{i_1} \cdots x_n^{i_n} \right\rfloor =$$

$$\left\lfloor \sum_{k_1,\ldots,k_n,i_1,\ldots,i_n=0}^{d-1} p_{\bar{k}} \cdot Z^{\Sigma_{i=0}^{n}(k_j-i_j)d^{j-1}} \cdot x_1^{i_1} \cdots x_n^{i_n} \right\rfloor =$$

Für $\Sigma_{\mathrm{i}=0}^{\mathrm{n}}(\mathrm{k_j}-\mathrm{i_j})\mathrm{d}^{\mathrm{j}-1}{\geq}0$ sind die Summanden ganzzahlig.

Für $\Sigma_{\mathrm{i}=0}^{\mathrm{n}}(\mathrm{k_j}-\mathrm{i_j})\mathrm{d}^{\mathrm{j}-1}{<}0$ ist die Summe dieser Summanden kleiner 1, da $Z > d^n{\cdot}\mathrm{wt}(P){\cdot}\max\{x_i; i = 1, ..., n\}^{nd}$ gilt.

$$= \sum_{\substack{k_1,...,k_n,i_1,...,i_n=0 \\ \Sigma_{i=0}^{n}(k_j-i_j)d^{j-1}\geq 0}}^{d-1} p_{\bar{k}} \cdot Z^{\Sigma_{i=0}^{n}(k_j-i_j)d^{j-1}} \cdot x_1^{i_1} \cdots x_n^{i_n}$$

$$f(\bar{x}) = k(\bar{x}) \bmod Z =$$

$$\left(\sum_{k_1,...,k_n=0,}^{d_1,...,d_n} \sum_{\substack{i_1,...,i_n=0 \\ \Sigma_{i=0}^{n}(k_j-i_j)d^{j-1}\geq 0}}^{d^n,...,d} p_{\bar{k}} \cdot Z^{\Sigma_{i=0}^{n}(k_j-i_j)d^{j-1}} \cdot x_1^{i_1} \cdots x_n^{i_n} \right) \bmod Z$$

Falls $k_n < i_n$ ist, gilt $\Sigma_{i=0}^{n}(k_j - i_j)d^{j-1} < 0$ für $j = 1, ..., n$, da $d > k_j$ gilt. Dies ist im Widerspruch zu $\Sigma_{i=0}^{n}(k_j - i_j)d^{j-1} \geq 0$.

Falls $k_n > i_n$ ist, teilt Z die Summanden

$p_{\bar{k}} \cdot Z^{\Sigma_{i=0}^{n}(k_j-i_j)d^{j-1}} \cdot x_1^{i_1} \cdots x_n^{i_n}$

Daraus kann gefolgert werden, dass

$$\left(p_{\bar{k}} \cdot Z^{\Sigma_{i=0}^{n}(k_j-i_j)d^{j-1}} \cdot x_1^{i_1} \cdots x_n^{i_n} \right) \bmod Z \neq 0$$

nur für $k_n = i_n$ gilt.

In der gleichen Weise wird $k_j = i_j$ induktiv für $j = n - 1$ bis $j = 1$ gezeigt.

Nach Definition von Z gilt nun

$$\sum_{k_1,...,k_n=0,}^{d_1,...,d_n} p_{\bar{k}} x_1^{k_1} \cdots x_n^{k_n} = P(\bar{x}).$$

□

5.3 Polynomauswertung mit bitweiser Konjunktion

Im Gegensatz zum Hornerschema erfolgt die Polynomauswertung mit dem Algorithmus von Nader BSHOUTY und dessen Erweiterung auf multivariate

Poynome nur über *endliche* Eingabebereiche. Für ein im Verhältnis zur Eingabe genügend großes Z, so dass es bei dem Produkt von $Z^{d+1}DIV\,(Z-x)$ und P(Z) nicht zu unerwünschten Überlappungen kommt, wurde P(Z) als Konstante vorher gespeichert. Wird die bitweise Konjunktion als weitere Operation hinzugenommen und das Z im Verhältnis zur Eingabe geeignet gewählt, lässt sich die Polynomauswertung im Vergleich zum Hornerschema deutlich beschleunigen.

Satz 4 *Sei $p \in \mathbb{N}[x]$ ein Polynom vom Grad d, dann kann $\mathbb{N} \ni x \mapsto p(x)$ in $\mathcal{O}(\log d)$ Schritten über $\{+,-,*,\mathrm{DIV},\&\}$ berechnet werden.*

Beweis. Sei $P(x) = \Sigma_{i=0}^{\mathrm{d}} p_i x^i \in \mathbb{N}[\mathrm{x}]$ mit $p_i \in \mathbb{N}$ und $p_i < 2^l$ für $i = 0, ..., d$.

Zu gegebenem $x \in \mathbb{N}$ berechne durch wiederholtes Quadrieren in $\mathcal{O}(\log d)$ Schritten $Z' := x^{\mathrm{d}+2}$.

$Z := Z' \cdot Y$ mit $Y := 2^l$ erfüllt die Bedingungen zu Satz 3.

$P(Y) = \sum_{i=0}^{d} p_i 2^{li} = P(2^l)$ ist für ein festes Polynom eine Konstante und kann vorab gespeichert werden.

$W := \sum_{i=0}^{d} Z'^i$ und ebenso $V := \sum_{i=0}^{d} (Z' \cdot Y)^i$ werden in $\mathcal{O}(\log d)$ Schritten berechnet, indem zunächst wiederum durch sukzessives Quadrieren Z'^{d+1} bzw. $(Z' \cdot Y)^{d+1}$ berechnet wird und V bzw. W werden nach der folgenden Gleichung $W := \sum_{i=0}^{d} Z'^i = Z'^{d+1}\mathrm{DIV}\,(Z'-1)$ bzw. $V := \sum_{i=0}^{d} (Z' \cdot Y)^i = (Z' \cdot Y)^{d+1}\mathrm{DIV}\,(Z' \cdot Y - 1)$ berechnet.

In der folgenden Abbildung ist zu sehen , wie P(Z) aus U, V, W und Y mit der bitweisen Konjunktion als weiterer Operation nach den obigen Abschätzungen in $\mathcal{O}(\log d)$ Schritten als

$$P(Z) = (P(Y) \cdot W)\&((Y-1) \cdot V)$$

berechnet wird.

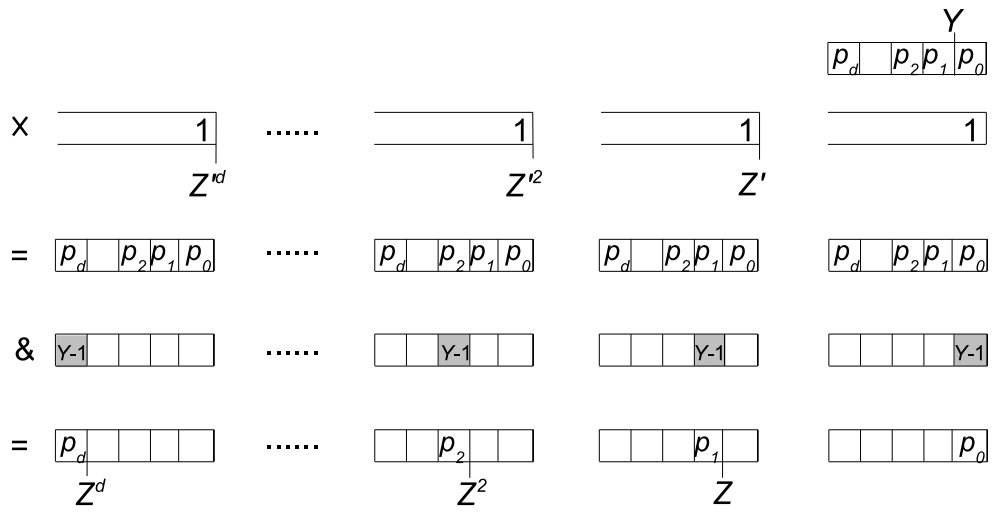

Abbildung 8. Berechnungen im Beweis von Satz 4

$$P(Y) \cdot W = \sum_{i=0}^{d}\sum_{j=0}^{d} p_i 2^{li} Z'^{j} = \sum_{i=0}^{d}\sum_{j=0}^{d} p_i 2^{li} Z'^{j}$$

$$(Y-1) \cdot V = \sum_{i=0}^{d}\sum_{i=0}^{l-1} (Z' \cdot Y)^i 2^j = \sum_{i=0}^{d}\sum_{j=0}^{l-1} 2^{li+j} Z'^{j}$$

Da $(Y-1) \cdot V$ nur Koeffizienten $\neq 0$ bei 2^{li+j} für $i = 0, ..., d$ und $j = 0, ..., l-1$ hat, sind es genau die Koeffizienten p_i in binärer Darstellung für $i = 0, ..., d$, $P(Z) = (P(Y) \cdot W)\&((Y-1) \cdot V)$. □

Nach dem obigen Beweis werden die $\mathcal{O}(\log d)$ Schritte benötigt, um durch wiederholtes Quadrieren $Z' := x^{\mathrm{d}+2}$ und Z^{d} zu berechnen. Alle anderen Berechnungen gehen in konstanter Zeit, wenn die Konstanten wie Y^{d} oder P(Y) vorher berechnet werden. Ist x$<\mathcal{O}(2^d)$ können $Z' := x^{\mathrm{d}+2}$ und Z^{d} schneller in $\mathcal{O}(\log\log |\mathrm{x}|)$ Schritten berechnet werden, wenn nicht x, sondern 2^d und 2^{d^2}sukzessive quadriert werden. In diesem Fall wird auch nicht die Multiplikation, sondern nur die Multiplikation mit Konstanten benötigt.

Korollar 7 *Sei $p \in \mathbb{N}[x]$ ein Polynom vom Grad d, dann kann $\mathbb{N} \ni x \mapsto p(x)$ in $\mathcal{O}(\log log|x|)$ Schritten über $\{+, -, *_c, \mathrm{DIV}, \&\}$ berechnet werden.*

Ohne Einschränkung kann der Grad des Polynoms als Zweierpotenz vorausgesetzt werden, da ansonsten das Polynom durch Hinzufügen von Summanden mit den Koeffizienten Null erweitert wird. Die Laufzeit in der $\mathcal{O}$-Notation ändert sich nicht dabei. Wird zusätzlich noch das schnellere Quadrieren mit der ganzzahligen Division aus dem Aufsatz von Yishay MANSOUR, Baruch SCHIEBER, PrasoonTIWARI [28] verallgemeinert für beliebige positive ganze Zahlen, erhält man die Zeitabschätzung in Theorem 9 zur Polynomauswertung.

Satz 5 *Gegeben seien $a, k \in \mathbb{N}$ und eine beliebige ganze Zahl $b \geq a^{2^k}$, dann lässt sich a^{2^k} über $(+, -, *, \mathrm{DIV})$ in $\mathcal{O}(\sqrt{k})$ Schritten berechnen.*

Beweis. Zunächst wird m mit $k = m^2 + b$ mit $0 \leq b < 2m + 1$ gesucht. Dies geht in $\mathcal{O}(\sqrt{k})$ Schritten, indem nacheinander i für $i = 1, ..., m + 1$ quadriert wird, bis $i^2 > k$ gilt. Da $m = \left\lfloor \sqrt{k} \right\rfloor$ gilt, kann durch sukzessives Quadrieren von $a^{2^{m^2}}$ in $\mathcal{O}(\sqrt{k})$ Schritten a^{2^k} berechnet werden. Es bleibt zu zeigen, dass $a^{2^{m^2}}$ in $\mathcal{O}(m) = \mathcal{O}(\sqrt{k})$ Schritten berechnet werden kann.

Sei $b_{1,0} = ab$. Durch m-faches Quadrieren von $b_{1,0} = ab$ erhält man für $i = 1, ..., m$

$$b_{1,i} = {b_{1,i-1}}^2 = (ab)^{2^i}$$

Sei $b_{0,0} = a$ und berechne

$$b_{0,i} = b_{1,m} \bmod (\mathrm{b_{1,0}} - \mathrm{b_{0,i-1}}) \text{ für } i = 1, ..., m.$$

Es wird gezeigt, dass $b_{0,i} = a^{2^{mi}}$ und daher $b_{0,m} = a^{2^{m^2}}$ gilt.

Es wird ausgenutzt, dass für $y^r < (x - y)$

$$x^r \bmod (x - y) = ((x - y) + y)^r \bmod (x - y) = y^r$$

gilt. Aus $b_{1,m} = (ab)^{2^m}$ und $b_{1,0} - b_{0,i-1} > a^{2^{m^2}}$ folgt daher $b_{0,i} = b_{0,i-1}^{2^m} = a^{2^{mi}}$.

□

Ohne ganzzahlige Division benötigt man für die 2^{k}. Potenz einer natürlichen Zahl durch wiederholtes Quadrieren $\mathcal{O}(k)$ Schritte. Die zusätzliche Eingabe von dem b kann gewissermaßen als 'Katalysator' zur Beschleunigung betrachtet werden, vergleiche Abschnitt 6.3.

Theorem 9. *Ein Polynom $P \in \mathbb{Z}[x]$ vom Grad d kann an der Stelle $x \in \mathbb{Z}$ in $\mathcal{O}(min\{log\,\mathrm{d}, log\,log\,|\mathrm{x}|\})$ Schritten über $\{+, -, *, \mathrm{DIV}, \&\ \}$ berechnet werden.*

Ist zusätzlich eine natürliche Zahl $y \geq |x|^{d^2}$ gegeben, kann $P(x)$ über $\{+, -, *,$ $\mathrm{DIV}, \&\}$ in $\mathcal{O}(\sqrt{min\{log\,d, log\,log\,|x|\}})$ berechnet werden.

Dieses Theorem kann wie im endlichen Fall von Poynomen mit einer Variablen auf solche mit mehreren Variablen übertragen werden.

Theorem 10. *Ein Polynom $P(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ mit maximalem Grad kleiner d kann über $\mathbb{Z}^{\mathrm{n}}$ in $\mathcal{O}(\mathrm{n} \cdot min\{log\,d, log\,log\,max\,\{|x_i|, i = 1, ..., n\})$ Schritten mit der Operationsmenge $\{+, -, *, \&, \mathrm{DIV}\}$ berechnet werden.*

Ist zusätzlich eine natürliche Zahl $y \geq max\{|x_i|, i = 1, ..., n\}^{d^{n+1}}$ gegeben, kann $P(x_1, \ldots, x_n)$ in $\mathcal{O}(\mathrm{n} \cdot \sqrt{min\{log\,d, log\,log\,max\,\{|x_i|, i = 1, ..., n\}})$ über $\{+, -, *,$ $\&, \mathrm{DIV}\}$ berechnet werden.

Beweis. Sei $P(\bar{x}) \in \mathbb{Z}[\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}]$.
Wie im endlichen Fall wird in $\mathcal{O}(n)$ Schritten entschieden, welches der 2^{n} Polynome $P_k(|x_1|, \ldots, |x_n|) \in \mathbb{Z}[|x_1|, \ldots, |x_n|]$ mit $P(x_1, \ldots, x_n) = P_k(|x_1|, \ldots, |x_n|)$ für eine Eingabe $\bar{\mathrm{x}} \in \mathbb{Z}^{\mathrm{n}}$ an der Stelle $(|x_1|, |x_2|, \ldots, |x_n|)$ berechnet werden muss, um den Wert von $P(\bar{x})$ zu berechnen.

Sei o.B.d.A. $P(\bar{x}) = \Sigma_{i_1,\ldots,i_n=0}^{\mathsf{d}_1,\ldots,\mathsf{d}_\mathsf{n}} a_{i_1,\ldots,i_n} x_1^{i_1} \cdots x_n^{i_n} \in \mathbb{N}[\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}]$. Andernfalls, falls $P(\bar{x}) \in \mathbb{Z}[\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}]$, unterteile P wiederum in die Differenz zweier Polynome mit natürlichen Koeffizienten.

Um Theorem 8 auf eine Eingabe $\bar{x} \in \mathbb{Z}^{\mathrm{n}}$ anwenden zu können, muss wie in Theorem 8 $(Z, Z^d, ..., Z^{d^{n-1}})$ für ein $Z > \Omega(\max\{x_i;\ i = 1, ..., n\}^{d^n})$ mit $d := \mathsf{max}\,\{2, \mathsf{d}_1 + 1, ..., \mathsf{d}_\mathsf{n} + 1\}$; $d_i = \deg_{x_i} P(\bar{x})$ gefunden werden. Dies geht durch wiederholtes Quadrieren von $\max\{|x_i|, i = 1, ..., n\}$ oder von $(2^d, 2^{d^2}, ..., 2^{d^n})$ in $\mathcal{O}(\mathrm{n} \cdot \min\{\log \mathrm{d}, \log\log\ \max\{|x_i|, i = 1, ..., n\})$. Bei zusätzlicher Eingabe von $y \geq \max\{|x_i|, i = 1, ..., n\}^{d^{n+1}}$ kann wiederum wie in [28] das Quadrieren zu $\mathcal{O}(\mathrm{n} \cdot \sqrt{\min\{\log d, \log\log\max\{|x_i|, i = 1, ..., n\}})$ beschleunigt werden.

Als nächstes muss $P(Z, Z^d, ..., Z^{d^{n-1}})$ berechnet werden.

$$P(Z, Z^d, ..., Z^{d^{n-1}}) = \sum_{k_1,...,k_n=0,}^{d_1,...,d_n} p_{k_1,...,k_n}(Z)^{k_1} \cdots (Z^{d^{n-1}})^{k_n} =$$

$$\sum_{k_1,...,k_n=0,}^{d_1,...,d_n} p_{k_1,...,k_n}(Z)^{k_1+...+d^{n-1}k_n}$$

Da $k_1 + ... + d^{n-1}k_n$ für $k_1, ..., k_n \in \{0, ..., d-1\}$ verschiedene Zahlen in d-adischer Darstellung sind, kann $P(Z, Z^d, ..., Z^{d^{n-1}})$ als ein univariates Polynom $P'(Z)$ mit $P'(x) \in \mathbb{N}[\mathrm{x}]$ und $\deg P' \leq d^n$ betrachtet werden.

□

Für Eingaben aus den natürlichen Zahlen ist der Algorithmus ein SLP, bei Eingaben aus den ganzen Zahlen werden als zusätzliche Operation noch Vergleiche „$\leq$" benötigt.

5.4 Speichern und Extrahieren algebraischer Zahlen

Wenn die bitweise Konjunktion nicht hinzugenommen wird, bleibt das Hornerschema das schnellste bekannte Verfahren, um ein beliebiges, aber festes Polynom über ganz $\mathbb{N}$ mit der Operationsmenge $\{+, -, *, DIV\}$ auszuwerten. In [25] ist eine untere Schranke von $\Omega(log\, log\, d)$ für einige Polynome zur Polynomauswertung bewiesen worden, wenn d den Grad des Polynoms bezeichnet. Es entsteht also eine doppelexponentielle Lücke zur oberen Schranke von $\mathcal{O}$(d) des Hornerschemas über $\{+, -, *\}$. Damit stellt sich folgende

Frage 1 *Kann jedes beliebige, aber feste Polynom $p \in \mathbb{N}[\mathbb{X}]$ an jeder Stelle $x \in \mathbb{N}$ in $o(deg\, p)$ über $\{+, -, *, DIV\}$ ausgewertet werden.*

Nach den Überlegungen des letzten Abschnitts kann diese Frage positiv beantwortet werden, falls es möglich ist, zu $x \in \mathbb{N}$ die Zahl $p(Z)$ für irgendein $Z > \Omega(x^d)$ mit $d > \deg p$ zu berechnen. Dazu wähle $Z_n := Y \cdot 2^n$ mit $Y = 2^k > \mathrm{wt}(p)$ und stelle die binäre Erweiterung der Folge $p'(Z_n) = 2p(Z_n) <$

$Z_n^d \cdot \mathrm{wt}(p) \leq 2^{K+dn+1}$ mit $n \in \mathbb{N}$ und $K := k \cdot (d+1)$, wie in Satz 3 folgendermaßen als reelle Zahl dar:

$$\rho_p := \sum_{n=0}^{\infty} p'(Z_n) \cdot 2^{-n \cdot (K+dn+1)}.$$

Um zu gegebenem $x \in \mathbb{N}$ aus dieser reellen Zahl $p'(Z_n)$ zu extrahieren, genügt es, ρ_p bis auf eine Abweichung $\varepsilon < 2^{-n \cdot (K+dn+1)}$ für ein $n > \Omega(d \cdot log\, x)$ zu approximieren.

Es wird $2p(Z_n)$ extrahiert, da das kleinste Bit bei $2p(Z_n)$ eine 0 ist und sollte es bei der Approximation durch wiederholte Überträge kleinerer Bits zu einem falschen Ergebnis des kleinsten Bits von $2p(Z_n)$ kommen, liefert die ganzzahlige Division durch 2 das korrekte Ergebnis $p(Z_n)$.

Lemma 11. *Sei $\alpha \in \mathbb{R}$ algebraisch vom Grad $< d$. Dann können zu gegebenem $n \in \mathbb{N}$ über $\{+, -, *\}$ in $\mathcal{O}(\delta \cdot log\, \mathrm{n})$ $u, v \in \mathbb{N}$ berechnet werden, so dass $|\alpha - u/v| \leq 2^{-n}$ gilt.*

Für einige transzendente Zahlen wurden ähnliche Ergebnisse, obwohl mit unterschiedlichen Methoden, erzielt [7].

Beweis. Sei $q \in \mathbb{Z}[\mathbb{X}]$ das Minimalpolynom von α. Mit dem Newtonschen Nährungsverfahren lassen sich mit quadratischer Konvergenzgeschwindigkeit die Nullstellen eines Polynoms approximieren. Zu einem festen $\alpha \in \mathbb{R}$ können das Minimalpolynom $q \in \mathbb{Z}[\mathbb{X}]$, seine Ableitung $q' \in \mathbb{Z}[\mathbb{X}]$ und ein geeigneter Ausgangspunkt vorab als Konstanten gespeichert werden. □

Aus diesem Lemma folgt für Polynome mit einer leichten Abhängigkeit von der Eingabe x die Antwort auf die eingangs in diesem Abschnitt gestellte Frage.

Theorem 11. *Sei $p \in \mathbb{N}[\mathbb{X}]$ ein Polynom vom Grad $<d$ und angenommen $\sum_{n=0}^{\infty} 2^{-dn^2}$ ist algebraisch vom Grad $<\delta$. Dann kann p(x) für $x \in \mathbb{N}$ über $\{+, -, *, DIV\}$ in $\mathcal{O}(\delta \cdot \log\log \mathrm{x})$ Schritten berechnet werden.*

Leider ist es weder bekannt, ob die Reihe $\sum_{n=0}^{\infty} 2^{-dn^2}$ algebraisch ist, noch, falls sie algebraisch ist, von welchem Grad. Dies stellt ein ungelöstes Problem in der Zahlentheorie dar [38, Abschnitt 10.7.B, Beispiel 1, S.314,].

Um die zu Beginn dieses Abschnittes gestellte Frage zu beantworten, könnte man, statt alle $p(Z_n)$ in einer Reihe zu speichern, eventuell eine linear rekurrente Folge von Polynomen $p(Z_n)$ suchen, die mit einer leichten Abhängigkeit von x schnell berechnet werden kann. Ein Ansatz hierzu könnte durch die in [17] beschriebenen Algorithmen bei Hinzunahme der ganzzahligen Division und in Verbindung mit der folgenden Bemerkung gegeben sein.

Bemerkung 5 *Sei $p \in \mathbb{Z}[x]$ vom Grad $< d$ und $c \in \mathbb{N}$. Dann ist die ganzzahlige Folge $p(1), p(c), p(c^2), \ldots, p(c^n), \ldots$ linear rekurrent vom Grad d, d.h. es gibt $a_0, \ldots, a_{d-1}, a_d \in \mathbb{Z}$, so dass $p(c^{n+1}) = \big(a_1 \cdot p(c^n) + \cdots + a_d \cdot p(c^{n-d+1})\big)/a_0$ für alle $n \in \mathbb{N}$ gilt.*

Beweis. Für $k = d-1$ haben die $(d+1)$ Polynome $p(cx), p(x), p(x/c), \ldots, p(xc^{-k})$ alle einen Grad $< d$. Daher sind sie linear abhängig über $\mathbb{Q}$. $q_0 \cdot p(cx) + q_1 \cdot p(x) + \cdots + q_{k+1} \cdot p(xc^{-k}) \equiv 0$; o.B.d.A. $q_i \in \mathbb{Z}$. Falls k minimal ist, folgt $q_0 \neq 0$. □

6 Anwendungen in der Linearen Algebra

Wie eingangs erwähnt und im letzten Kapitel für die Polynomauswertung gezeigt, hängt die Berechnungsmächtigkeit und die Komplexität einer Registermaschine RAM stark von der zugrundeliegenden Operationsmenge ab. Im folgenden Kapitel geht es um ausgewählte Probleme, deren Laufzeiten durch diese zusätzlichen Operationen wie ganzzahlige Division usw. linear oder sogar sublinear sind, wie auch in den folgenden Beispielen.

Beispiel 2

a) Über $\{+, -, *, \mathrm{DIV})\}$ ist nicht nur der Primzahltest, sondern sogar die Faktorisierung einer ganzen Zahl x in $\mathcal{O}(\log x)$ Schritten möglich, d.h. linear in der Binärlänge von x .

b) Über $\{+, -, *, \mathrm{DIV})\}$ und mit indirekter Adressierung lässt sich der größte gemeinsame Teiler $\mathrm{ggT}(x, y)$ zweier Zahlen x,y mit $N = \max\{x, y\}$ in $\mathcal{O}(\log N / \log\log N)$ Schritten berechnen.

[0] Ich danke Riko Jacob für den Hinweis auf die Punkte d) und e) in diesem Beispiel.

c) Über $\{+,-,\&,\leftarrow,\rightarrow\}$ *(aber ohne indirekte Adressierung wie bei Bucket Sort) können* n *ganze Zahlen* $x_1,\ldots,x_n$ *in* $\mathcal{O}(n)$ *Schritten sortiert werden und über* $\{+,-,*,\mathrm{DIV}\}$ *in* $\mathcal{O}(n\cdot\log\log\max_i x_i)$ *Schritten.*

d) 3SUM, die Frage, ob es zu $x_1,\ldots,x_n,y_1,\ldots,y_n,\ z_1,\ldots,z_n$ i,j,k *mit* $x_i+y_j=z_k$ *gibt, kann in* $\mathcal{O}(n)$ *Schritten über* $\{+,-,*,\&\}$ *entschieden werden.*

e) Die Permanente $Perm(A)=\sum_{\pi\in\mathcal{S}_n} a_{1,\pi(1)}\cdots a_{n,\pi(n)}$ *einer* $n\times n$ *Matrix* A *kann in* $\mathcal{O}(n^2)$ *Schritten über* $\{+,-,*,\mathrm{DIV}\}$ *berechnet werden, d.h. in quasioptimaler Zeit.*

b) Die Laufzeit des Euklidischen Algorithmus benötigt $\Theta(\log N)$ für Fibonaccizahlen $x=F_n=N$, $y=F_{n-1}$.

c) Um die Permutation von der Eingabe zur sortierten Ausgabe zu beschreiben erfordert bereits $\Omega(n\cdot\log n)$ Bits.

e) Ohne ganzzahlige Division ist Perm *VALIANT-NP*-vollständig [8, Theorem 21.17], wenn das Einheitskostenmodell vorausgesetzt wird, im Bitmodell ist Perm sogar #P-vollständig.

d) Auch 3SUM wird ohne DIV als 'n^2-vollständig' betrachtet, vergl. [19].

Beweis. a) Siehe [40]; b) siehe [5]; c) siehe [24], für aktuellere Ergebnisse zu den Aufwandsabschätzungen beim Sortieren mit verschiedenen Operationsmengen siehe [20];

e) siehe [1, Proposition 2.4] und den Beweis zu Satz 6.

Die Behauptung aus d) kann von den allgemeineren Betrachtungen in [9] abgeleitet werden. Auf dieses Beispiel angewendet führt dies zu:
Für $0\le a_0,\ldots,a_{N-1},b_0,\ldots,b_{N-1}<2^{t-1}$, sei $A:=\sum_{i=0}^{N-1} a_i\cdot 2^{ti}$, $B:=\sum_i b_i\cdot 2^{ti}$, und $C:=\sum_i 2^{t-1}\cdot 2^{ti}$. Dann gilt

$$\forall i=0,\ldots,N-1:\ a_i\ge b_i \quad\Leftrightarrow\quad (A+C-B)\&C\ =\ C\ .$$

Aufgrund der obigen Kodierung kann die folgende Aussage „$\exists i:\ a_i=b_i$" in konstanter Zeit über $\{+,-,\&\}$ überprüft werden. Diese Kodierung erhält man für die doppelte Folge $(x_i+y_j)_{i+nj}$ in Linearzeit $\mathcal{O}(n)$ über $\{+,-,*\}$, vergl. dazu den Beweis von Satz 12. □

6.1 Matrixmultiplikation

Der derzeitige Rekord bei der Matrixmultiplikation ohne ganzzahlige Division liegt bei $\mathcal{O}(n^{\omega})$ mit $\omega < 2{,}38$, aufgestellt von Don COPPERSMITH und Shmuel WINOGRAD *[15], [8, Chapter 15].* Es wird in diesem Abschnitt gezeigt, dass bei Hinzunahme der ganzzahligen Operation als weitere Operation die Matrizen in quadratischer Zeit, d.h. quasi-optimaler Zeit, berechnet werden können.

Theorem 12. *Seien* $A \in \mathbb{Z}^{kxn}$ *und* $B \in \mathbb{Z}^{nxm}$. $C := A \cdot B \in \mathbb{Z}^{kxm}$ *kann über* $\{+, -, *, \mathrm{DIV}\}$ *in* $\mathcal{O}((k+m)n + km)$ *Schritten berechnet werden.*

Beweis. Für $i = 1, ..., k$ und $j = 1, .., m$ soll $c_{i,j} = \Sigma_{l=1}^{n} a_{i,l} \cdot b_{l,l}$ berechnet werden. O.B.d.A. seien $a_{i,l}, b_{l,l} \geq 0$, ansonsten zerlege die Matrizen und multipliziere nach dem nicht-kommutativen Distributivgesetz für Matrizen.

Z sei eine Zweierpotenz und so gewählt, dass $Z > (\max_{i,l} a_{i,l}) \cdot (\max_{l,j} b_{l,j}) \cdot n$.

Die Matrizen A und B werden folgendermaßen als ganze Zahlen in $\mathcal{O}(kn)$ und $\mathcal{O}(nm)$ Schritten kodiert.

$$\alpha := \Sigma_{i=1}^{k} \Sigma_{l=1}^{n} a_{i,l} \cdot Z^{(l-1)+2nm(i-1)} \text{ und } \quad \beta := \Sigma_{i=1}^{k} \Sigma_{l=1}^{n} b_{l,i} \cdot Z^{(n-l)+2n(j-1)}$$

Vorab werden alle zur Kodierung und späteren Dekodierung benötigten Potenzen von Z in $\mathcal{O}(\log(knm))$ berechnet.

Wie in der folgenden Abbildung dargestellt, stehen die gesuchten Zahlen $c_{i,j}$ bei dem Produkt $\gamma := \alpha \cdot \beta$ in Z-adischer Darstellung genau an den Positionen $Z^{2n(j-1)+(n-1)+2nm(i-1)}$. Mittels ganzzahliger Division werden die $c_{i,j}$ in $\mathcal{O}(km)$ Schritten aus dem Produkt γ extrahiert.

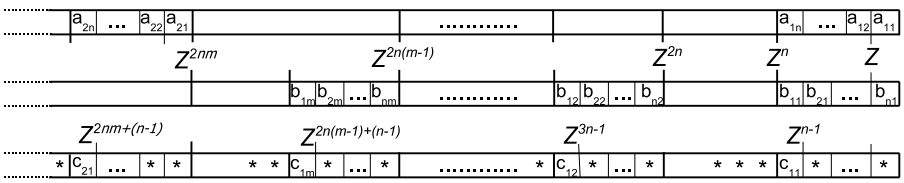

Abbildung 9. Kodierung der Matrizen A und B und Dekodierung der Matrix C

Die Berechnungszeit wird zum Kodieren und Dekodieren der Matrizen benötigt und die eigentliche Multiplikation erfolgt in konstanter Zeit. □

6.2 Permanente und Determinante

Dass bei Hinzunahme der ganzzahligen Division die Permanente einer Matrix in quadratischer Zeit berechnet werden kann, wurde für Matrizen mit Einträgen aus $\{0,1\}$ von Eric ALLENDER, Peter BÜRGISSER et al. in [1] gezeigt. Dieser Beweis wird auf Matrizen mit Einträgen aus $\mathbb{N}$ übertragen. In Verbindung mit der Polynomberechnung bei multivariaten Polynomen über einem endlichen Bereich lässt sich diese Laufzeit auf die Berechnung der Determinante einer Matrix mit ganzzahligen Einträgen übertragen.

Satz 6 *Man kann*

$$\mathbb{N}^{n\times n} \ni A \mapsto Perm(A) = \sum_{\pi\in\mathcal{S}_n} a_{1,\pi(1)}\cdots a_{n,\pi(n)}$$

über $\{+,-,*,DIV\}$ *in* $\mathcal{O}(n^2)$ *Schritten berechnen.*

Beweis. Betrachte das multivariate Polynom f_n.

$$f_n := \sum_{i=1}^{n2^{n-1}} f_{n,i}Y^i = \prod_{i=1}^{n}(\sum_{j=1}^{n} X_{i,j}Y^{2^{j-1}})$$

Die $f_{n,i}$ sind Polynome mit natürlichen Koeffizienten in n^2 Variablen.

Alle $Y^{2^{j-1}}$ für j=1,...,n und damit auch jede Summe für i=1,...,n lassen sich in $\mathcal{O}(n)$ berechnen. Da es n Summen sind und das Produkt über diese Summe wiederum in $\mathcal{O}(n)$ berechnet werden kann, ist dieses Polynom in $\mathcal{O}(n^2)$ Schritten über $\{+,-,*\}$ berechenbar.

Zu einer Matrix A mit Einträgen $a_{i,j} \in \mathbb{N}$ wird dieses Polynom f_n für $X_{i,j} = a_{i,j}$ und für ein genügend großes $Y=Z>(\max_{i,j=1}^{n} a_{i,j})^{n^3} > n^n \cdot (\max_{i,j=1}^{n} a_{i,j})^n$, ebenfalls in $\mathcal{O}(n^2)$ berechenbar, ausgewertet. Nach Wahl von Z ist $f_{n,2^n-1}$ an der Stelle $X_{i,j} = a_{i,j}$ gerade die Permanente von A. Durch die ganzzahlige Division und Restbildung lässt sich $f_{n,2^n-1}$ aus $f_n(Z)$ extrahieren. □

Im Folgenden wird gezeigt, dass diese quasi-optimale Polynomialzeit $\mathcal{O}(n^2)$ nicht nur für die Berechnung der Permanente, sondern auch für die Berechnung der Determinante gilt.

Theorem 13. *Sei* $A \in \mathbb{Z}^{n\times n}$, $\mathrm{Det}(A)$ *kann in* $\mathcal{O}(n^2)$ *Schritten über* $\{+, -, *,$ *DIV}* *berechnet werden.*

Im Gegensatz zu Theorem 10 wird die bitweise Konjunktion „&" hier nicht als weitere Operation benötigt.

Beweis. Zunächst wird die Matrix $A \in \mathbb{Z}^{n\times n}$ zu einer Matrix $A' \in \mathbb{N}^{n\times n}$, deren Determinante sich höchstens um das Vorzeichen von der Determinante von A unterscheidet, in $\mathcal{O}(n^2)$ Schritten umgeformt.

Sei $a_{i_0,j_0} = max\{|a_{i,j}|, i, j = 1, ..., n\} \in \mathbb{N}$, ansonsten, falls a_{i_0,j_0} negativ ist, multipliziere die i_0. Zeile mit -1. Die beiden Determinanten unterscheiden sich dann um das Vorzeichen.

Addiere zu jeder Zeile das Doppelte der Zeile i_0. In der Spalte j_0 haben nun alle Einträge dasselbe Vorzeichen wie a_{i_0,j_0} und sind betragsmäßig größer als a_{i_0,j_0} und die Einträge in den anderen Spalten sind $\leq 3 \cdot a_{i_0,j_0}$.

Wird nun zu jeder Spalte das Vierfache der Spalte j_0 addiert, so sind alle Einträge positiv und haben sich betragsmäßig höchstens verachtfacht. Da sich die Determinante bei diesen Zeilen- und Spaltenumformungen nicht ändert, wird im Folgenden o.B.d.A. $A \in \mathbb{N}^{n\times n}$ angenommen.

Wird die Determinante für Matrizen mit natürlichen Einträgen in ihren positiven Teil und ihren negativen Teil aufgeteilt,

$$\mathrm{Det}_+(A) = \sum_{\substack{\pi\in\mathcal{S}_n \\ \mathrm{sgn}(\pi)=+}} a_{1,\pi(1)} \cdots a_{n,\pi(n)} \quad \text{und}$$

$$\mathrm{Det}_-(A) = \sum_{\substack{\pi\in\mathcal{S}_n \\ \mathrm{sgn}(\pi)=-}} a_{1,\pi(1)} \cdots a_{n,\pi(n)},$$

so lassen sich die Permanente und die Determinante folgendermaßen ausdrücken: $\mathrm{Perm}(A) = \mathrm{Det}_+(A) + \mathrm{Det}_-(A)$ und $\mathrm{Det}\,(A) = \mathrm{Det}_+(A) - \mathrm{Det}_-(A)$.

Da sowohl $\mathrm{Det}_+(A)$ als auch $\mathrm{Det}_-(A)$ Polynome in n^2 Variablen $x_{i+n(j-1)} := a_{i,j}$ mit Maximalgrad kleiner als $d := 2$ (*der totale* Grad ist n) und Koeffizienten $0, 1$ sind, genügt es wie in Abschnitt 5.2 und dem Beweis von Satz 8, die Werte von $\mathrm{Det}_+ = (\mathrm{Perm} + \mathrm{Det})/2$ und von $\mathrm{Det}_- = (\mathrm{Perm} - \mathrm{Det})/2$ an der Stelle $\overline{x} = (x_0, \ldots, x_{n^2-1}) := (Z', Z'^2, Z'^4, \ldots, Z'^{2^{n^2-1}})$ zu berechnen, wobei $Z' := Z \cdot Y$ für $Z := (\max_k |x_k|)^{2^{n^2}}$ und eine geeignete Konstante Y in $\mathcal{O}(n^2)$ Schritten berechnet werden. Nach Satz 6 kann die Permanente ebenfalls in $\mathcal{O}(n^2)$ Schritten berechnet werden.

Um die Determinante dieser Matrix zu berechnen, wird sie folgendermaßen umgeformt:

$$\begin{vmatrix} Z' & Z'^2 & Z'^4 & Z'^8 & \cdots & Z'^{2^{n-1}} \\ Z'^{2^n} & Z'^{2^{n+1}} & Z'^{2^{n+2}} & \cdots & & Z'^{2^{2n-1}} \\ Z'^{2^{2n}} & Z'^{2^{2n+1}} & \ddots & & & Z'^{2^{3n-1}} \\ Z'^{2^{3n}} & \ddots & & & & Z'^{2^{4n-1}} \\ \vdots & & & & & \vdots \\ Z'^{2^{(n-1)n}} & \cdots & & & \cdots & Z'^{2^{n^2-1}} \end{vmatrix} =$$

$$= \begin{vmatrix} Z' & Z'^2 & Z'^4 & Z'^8 & \cdots & Z'^{2^{n-1}} \\ Z'^{2^n} & \left(Z'^{2^n}\right)^2 & \left(Z'^{2^n}\right)^4 & \left(Z'^{2^n}\right)^8 & & \left(Z'^{2^n}\right)^{2^{n-1}} \\ Z'^{2^{2n}} & \left(Z'^{2^{2n}}\right)^2 & \left(Z'^{2^{2n}}\right)^4 & \left(Z'^{2^{2n}}\right)^8 & & \left(Z'^{2^{2n}}\right)^{2^{n-1}} \\ Z'^{2^{3n}} & \left(Z'^{2^{3n}}\right)^2 & \ddots & & & \left(Z'^{2^{3n}}\right)^{2^{n-1}} \\ \vdots & & & & & \vdots \\ Z'^{2^{(n-1)n}} & \left(Z'^{2^{(n-1)n}}\right)^2 & \cdots & & \cdots & \left(Z'^{2^{(n-1)n}}\right)^{2^{n-1}} \end{vmatrix} =$$

Da dies die Determinante einer Vandermondematrix ist, gilt für diese

$$= \quad Z' \cdot Z'^{2^n} \cdot Z'^{2^{2n}} \cdots Z'^{2^{(n-1)n}} \cdot \prod_{1 \le i < j \le n} \left(Z'^{2^{(j-1)n}} - Z'^{2^{(i-1)n}} \right).$$

Dieser Ausdruck lässt sich ebenfalls in $\mathcal{O}(n^2)$ Schritten berechnen. □

Frage 2 *Sei $\mathcal{P} \subseteq \mathcal{S}_n$ eine beliebige Familie von Permutationen von $[n]$. Kann auch $\sum_{\pi\in\mathcal{P}} \prod_{i=0}^{n-1} x_{i+n\pi'(i))}$ in $\mathcal{O}(n^2)$ Schritten über $\{+,-,*,DIV\}$ berechnet werden?*

6.3 Potenzierung ganzzahliger Matrizen

Bekannterweise kann a^{2^k} durch wiederholtes Quadrieren in k Schritten berechnet werden. In [11] wurde gezeigt, dass bei Hinzunahme der ganzzahligen Division als weitere Operation und einer ganzen Zahl b größer als a^{2^k} nur $\mathcal{O}(\sqrt{\mathrm{k}})$ Schritte benötigt werden. Dieses Verfahren wird auf die Potenzierung von Matrizen übertragen. Um eine Laufzeitverbesserung gegenüber der Laufzeit von $\mathcal{O}(\mathrm{k}\cdot\mathrm{d}^2)$ für die k-fache Matrixmultiplikation von d x d-Matrizen aus dem vorherigen Abschnitt zu erzielen, benötigt man als zusätzliche Operation auf den ganzen Zahlen die Bildung des größten gemeinsamen Teilers ggT.

Definition 3. *Seien $X, C \in \mathbb{Z}^{\mathrm{dxd}}$*

a) $\mathrm{ggT}(C) := \mathrm{ggT}(c_{ij} : 1 \leq i, j \leq d)$

b) X rem $C := (x_{ij} \bmod \mathrm{ggT}(C))$

c) $X \equiv Y \bmod C$, *falls der* $\mathrm{ggT}(C)$ *jeden Eintrag* x_{ij} - y_{ij} *von* X - Y *teilt.*

Dies ergibt für festes C eine Äquivalenzrelation auf $\mathbb{Z}^{\mathrm{dxd}}$, sogar eine zweiseitige Kongruenzrelation [1] wie das folgende Lemma zeigt.

Lemma 12. *a) Falls $X \equiv Y \bmod C$ ist, gilt $S\cdot X\cdot T \equiv S\cdot Y\cdot T \bmod C$.*

b) Für jedes $n \in \mathbb{N}$ gilt $X^n \equiv Y^n (\mathrm{mod}\,(X-Y))$.

c) X rem $C \equiv X \,(\mathrm{mod}\ C)$.

d) Falls $0 \leq x_{ij} < \mathrm{ggT}(C)$ ist, gilt X rem $C = X$.

Beweis. a) gilt nach dem Distributivgesetz für Matrizen.

b) folgt aus a) und dem nicht-kommutativen Binomialtheorem.

c) und d) folgen unmittelbar aus den Definitionen. □

[1] Umgekehrt hat jedes zweiseitige Ideal in $\mathbb{Z}^{d\times d}$ diese Form [21, Proposition III.2.1]

Theorem 14. *Seien* $k \in \mathbb{N}$, $A \in \mathbb{N}^{\mathrm{dxd}}$ *gegeben und* $\gamma := d^{2^k-1} \cdot (\max_{\mathrm{ij}} a_{ij})^{2^k}$. *Ferner sei noch ein* $B \in \mathbb{N}^{\mathrm{dxd}}$ *gegeben, so dass für alle* $C \in \{0, 1, ..., \gamma\}^{\mathrm{dxd}}$ *ggT(B - C)* $> \gamma$ *gilt, dann kann* A^{2^k} *in* $\mathcal{O}(\sqrt{\mathrm{k}} \cdot \mathrm{d}^2)$ *Schritten über* $\{+, -, *, \mathrm{DIV}, \mathrm{ggT}\}$ *berechnet werden.*

Bemerkung 6 *Nach der Wahl von B kann nicht nur* A^{2^k} *in* $\mathcal{O}(\sqrt{\mathrm{k}} \cdot \mathrm{d}^2)$ *Schritten über* $\{+, -, *, \mathrm{DIV}, \mathrm{ggT}\}$ *berechnet werden, sondern auch jedes* $A'^{2^{k'}}$ *für jedes* $0 \leq k' \leq k$ *und* $0 \leq \max_{\mathrm{ij}} |a'_{ij}| \leq \max_{\mathrm{ij}} |a_{ij}|$ *in* $\mathcal{O}(\sqrt{\mathrm{k}'} \cdot \mathrm{d}^2)$ *Schritten .*

Beweis. Es genügt den Fall k=l^2 zu betrachten. Nach Theorem 12 wird zunächst die Matrix B^{2^l} durch wiederholtes Quadrieren in $\mathcal{O}(\mathrm{l} \cdot \mathrm{d}^2)$ Schritten berechnet.

Durch Anwendung von Lemma 12 auf $n := 2^{\mathrm{l}}$, $X := A^{2^{l(j-1)}}$, $Y := B^{2^l}$ und $C := Y - X$ erhält man folgende Gleichung

(*)

$$A^{2^{lj}} = (A^{2^{l(j-1)}})^{2^l} = B^{2^l} \mathrm{rem}\, (B - A^{2^{l(j-1)}}),$$

falls der $\mathrm{ggT}(B - A^{2^{l(j-1)}})$ größer als die Einträge von $A^{2^{l^2}}$ ist.

Induktiv wird für $j = 1, ..., l$ nach Gleichung (*) $A^{2^{lj}}$ aus $A^{2^{l(j-1)}}$ jeweils in $\mathcal{O}(\mathrm{d}^2)$ Schritten berechnet.

Da die m-te Potenz einer d x d-Matrix A mit Einträgen aus der Menge $\{0, 1, ..., s\}$ Einträge aus der Menge $\{0, 1, ..., d^{\mathrm{m\text{-}1}} \cdot s^{\mathrm{m}}\}$ hat, gilt nach der Voraussetzung für B die obige Gleichung.

Die Operation ggT wird benötigt, um den ggT(C) in $\mathcal{O}(\mathrm{d}^2)$ Schritten und damit X rem C nach der obigen Definition zu berechnen. □

Erstaunlicherweise erhält man die Folge von Matrizen $A^{2^{lj}}$, $j = 1, ..., l$ nur durch Modulo-Berechnungen der Matrizeneinträge.

Es fehlt noch der Beweis, dass es solch ein B, bei dem der ggT(B-C)$>\gamma$ für alle Matrizen $C \in \{0, 1, ..., \gamma\}^{\mathrm{dxd}}$ ist, überhaupt gibt.

Im Fall d = 1 heißt das gerade, dass, wie in Satz 5, B=(b) größer als $a^{2^{l^2}}$ für A=(a) ist, vgl. [11].

Für den Fall d>1 wird im Lemma des nächsten Abschnitts gezeigt, dass es viele solcher B gibt, und es wird beschrieben, wie man sie erhält.

6.4 Lokale untere Schranke des ggT

Eine reelle Funktion $f : \mathbb{R}^{\mathrm{d}} \to \mathbb{R}$ heißt halbstetig nach oben an der Stelle $\overline{x}$, wenn für alle $\overline{u}$ in einer Umgebung von $\overline{x}$ die Funktionswerte von $\overline{x}$ und $\overline{u}$ nicht zu weit auseinander liegen, siehe [36]. Da der ggT eine diskrete Funktion ist, sind diese topologischen Kriterien streng genommen nicht auf diese Funktion anwendbar. Aber dennoch lassen sich auch für den größten gemeinsamen Teiler Punkte $\overline{x}$ angeben, die dem Begriff der oberen Halbstetigkeit ähnlich sind.

Lemma 13. *Für alle d, r, s* $\in \mathbb{N}$ *gibt es* $x_1, x_2, ..., x_d \in \mathbb{N}$*, so dass für alle* $v_1, v_2, ..., v_d \in \{0, 1, ...,\mathrm{s}-1\}$ $\mathrm{ggT}(x_1 + v_1, ..., x_d + v_d) \geq r$ *gilt.*

Beweis. Man nehme paarweise teilerfremde natürliche Zahlen $p_{\overline{v}} > r$, $\overline{v} \in \{0, 1, ...,\mathrm{s}-1\}^d$, man kann im Allgemeinen die $p_{\overline{v}}$ als Primzahlen wählen. Für $i = 1, ..., d$ und $j = 0, ..., d-1$ sei $u_{i,j} := \prod_{\overline{v}, v_i = j} p_{\overline{v}}$. Die Zahlen $u_{i,0}, u_{i,1}, ..., u_{i,s-1}$ sind bei festem i wiederum teilerfremd. Nach dem Chinesischen Restsatz gibt es ein $x_i \in \mathbb{N}$, so dass $u_{i,j}$ für alle $j = 0, 1, ..., s-1$ x_i+j teilt. Da insbesondere alle $p_{\overline{v}}$ in den u_{i,v_i} als Teiler vorkommen, teilen sie $x_i + v_i$ für jedes $i = 1, ..., d$ und damit auch den $\mathrm{ggT}(x_1+v_1, ..., x_d+v_d)$. Daher muss der $\mathrm{ggT}(x_1+v_1, ..., x_d+v_d)$ mindestens so groß sein wie $p_{\overline{v}} > r$. □

Bemerkung 7 *a) Die* $x_1, x_2, ..., x_d \in \mathbb{N}$ *aus dem vorherigen Lemma können so gewählt werden, dass sie zwischen 0 und* $(r \cdot S)^{\mathcal{O}(S)}$ *mit S:=*s^d *liegen.*

b) Über $\{+, -, *, \mathrm{DIV}, \mathrm{eggT}\}$ *können sie in* $\mathcal{O}(S)$ *Schritten gebildet werden (aber nicht notwendigerweise innerhalb obiger Schranke).*

„eggT" bezeichnet den erweiterten größten gemeinsamen Teiler, d.h. dass es zu gegebenen $x, y \in \mathbb{N}$ $s, t \in \mathbb{Z}$ (o.B.d.A. s, t teilerfremd) gibt mit ggT(x,y)= sx +ty.

Beweis. a) Nach dem Primzahltheorem hat die k-te Primzahl eine Größenordnung von $\mathcal{O}(k \cdot \log k)$ und es gibt höchstens $\pi(n) \leq \mathcal{O}(n/\log n)$ Primzahlen, die kleiner als n sind. Daher hat die erste Primzahl, die wenigstens so groß wie r ist, den Index $k_r := \pi(r) \leq \mathcal{O}(r/\log r)$.

Es soll das Produkt $N := p_{k_r} \cdot ... \cdot p_{k_r+S}$ beschränkt werden, das ist gerade der Quotient der Primfakultäten[2] (r + l)#/r# mit $r + l = p_{k_r+S} = r + (S \cdot \log S)$. In [32] ist gezeigt worden, dass $\pi(r + l) - \pi(r) \leq 2\pi(l)$ [3]gilt. Also gibt es zwischen r und r + l mindestens $\mathcal{O}(l/\log l) = \mathcal{O}(S)$ Primzahlen und jede dieser Primzahlen ist natürlich nicht größer als r + l. Daher gilt

$$N = (r + l)\,\#/r\# \leq (r + l)^{\mathcal{O}(S \cdot \log S)} \leq (r \cdot l)^{\mathcal{O}(S \cdot \log S)}$$

für $l = \mathcal{O}(S \cdot \log S)$.

b) Die paarweise teilerfremden ganzzahligen $p_i \geq r$ erhält man folgendermaßen: $p_1 := r,\ p_2 := p_1 + 1, p_3 := p_1 \cdot p_2 + 1, ...,\ p_{i+1} := p_1 \cdot ... \cdot p_i + 1$. Durch Anwendung des folgenden Lemmas, dem Chinesischen Restsatz mit Laufzeitabschätzungen, folgt der Beweis zu b).

Lemma 14. (Chinesischer Restsatz) *Zu gegebenen $a_1, a_2, ..., a_n \in \mathbb{Z}$ und teilerfremden $m_1, m_2, ..., m_n \in \mathbb{N}$ kann ein $x \in \mathbb{N}$ mit $x \equiv a_i \bmod m_i$ für $i = 1, ..., n$ in $\mathcal{O}(\log n \cdot \Sigma_{i=1}^{n} \log m_i)$ Schritten über $\{+, -, *, \mathrm{DIV}\}$ berechnet werden. Wird der erweiterte größte gemeinsame Teiler eggT als weitere Operation hinzugenommen, reduziert sich die Laufzeit auf $\mathcal{O}(n)$*[4].

Beweis. Berechne $N := m_1 \cdot m_2 \cdot ... \cdot m_n$ und mit der Operation eggT berechne $s_i, t_i \in \mathbb{Z}$ mit $1 = \mathrm{ggT}(m_i, N/m_i) = s_i m_i + t_i N/m_i$. Für $e_i := t_i N/m_i$, $i = 1, ..., n$ gilt $e_i \equiv 1 \bmod m_i$ und $e_i \equiv 0 \bmod m_j$ für $i \neq j$. Daraus folgt, dass $x := \Sigma_{i=1}^{n} a_i e_i$ die geforderten Kongruenzen erfüllt.

Über $\{+, -, *, \mathrm{DIV}\}$ benötigt die Berechnung von $\mathrm{eggT}(m_i, N/m_i)$ $\mathcal{O}(\log N) = \mathcal{O}(\Sigma_{i=1}^{n} \log m_i)$ Schritte für jedes $i = 1, .., n$, also eine Gesamtlaufzeit von $\mathcal{O}(n \cdot \Sigma_{i=1}^{n} \log m_i) = \mathcal{O}(n \cdot \log N)$, falls diese n Berechnungen mit Hilfe des

[2] Eine Primfakultät (engl. primorial) p# ist das Produkt der ersten p Primzahlen.
[3] Ich danke Stephan Wehmeier für den Hinweis auf diese Schranke.
[4] Da $m_1, ..., m_n$ teilerfremd sind, gilt $\Sigma_{i=1}^{n} \log m_i > \Omega(n)$.

erweiterten größten gemeinsamen Teiler nacheinander durchgeführt werden. Über $\{+, -, *, \mathrm{DIV}, \mathrm{eggT}\}$ benötigt man $\mathcal{O}(n)$ Schritte.

Um den Faktor n in der obigen Berechnung nach log n zu verbessern, werden die simultanen Kongruenzen $x \equiv a_i \bmod m_i$ für $i = 1, .., n$ folgendermaßen in einem Binärbaum angeordnet:

Berechne zunächst die simultanen Kongruenzen y_j für $j = 1, ..., n/2$. Danach berechne folgende simultane Kongruenzen $x \equiv y_{2j} (\bmod m_{4j} \cdot m_{4j+1})$ und $x \equiv y_{2j+1} (\bmod m_{4j+2} \cdot m_{4j+3})$ für $j = 1, ..., n/4$. Auf der k-ten Ebene sind $n/2^\mathrm{k}$ simultane Kongruenzen, bei denen für die Berechnungen der Moduli als Produkte disjunkte k-tupel aus $\{m_1, m_2, ..., m_n\}$ benutzt werden. Auf jede dieser log n Ebenen werden $\mathcal{O}(\Sigma_{i=1}^\mathrm{n} \log m_i)$ Schritte unabhängig von k $= 1, ...,\mathcal{O}(\log n)$ benötigt, d.h. eine Gesamtlaufzeit von $\mathcal{O}(\log n \cdot \Sigma_{i=1}^\mathrm{n} \log m_i)$. □

6.5 Primzahlbildung mit Hilfe der ganzzahligen Division

Die größten der in dem Beweis zur Bermerkung 7 b) konstruierten Primzahlen und damit aber auch die x_i aus Lemma 13 sind von der Größenordnung $\Omega(r^{2^{s-1}})$ und damit erheblich größer als die möglichen in Teil a) als Primzahlen gewählten $\mathrm{p_j}$. Es stellt sich dann natürlich die Frage, ob die genannten nicht klassischen Operationen die Berechnung dieser Primzahlen ermöglicht, d.h. kann die Suche nach einer Funktion, die als Funktionswerte nur Primzahlen hat, mit diesen Operationen gelingen? Diese Frage wird am Anfang von Kapitel 3 in [37] gestellt und in Abschnitt II dieses Kapitels weiter erörtert.

Das Sieb des Eratosthenes findet alle Primzahlen kleiner N in $\mathcal{O}(N)$ Schritten über $\{+, -\}$. Dies kann zu $\mathcal{O}(N/\log\log N)$ beschleunigt werden [35]. Da nach dem Primzahltheorem es $\Theta(N/\log N)$ Primzahlen kleiner N gibt, ist diese Schranke im Vergleich zur Ausgabe optimal. Damit lassen sich in einem einfachen randomisierten Verfahren Primzahlen finden.

Bemerkung 8 *Zu $N \in \mathbb{N}$ rate irgendein $N \leq n < 2N$. Mit einer Wahrscheinlichkeit von $\Theta(1/\log N)$ ist n eine Primzahl. Nach $\mathcal{O}(\log N)$ unabhängigen Versuchen hat man mit konstanter Wahrscheinlichkeit eine Primzahl ge-*

funden. Mit dem Primzahltest aus [14] führt dies über $\{+, -, *, \text{DIV}\}$ *zu einer Laufzeit von* $\mathcal{O}(\log^2 N)$.

Da das Bertrand-Chebyshev Theorem besagt, dass es stets eine Primzahl zwischen N und 2 N gibt, lässt sich dieser einfache Algorithmus etwas verbessern.

Theorem 15. *Zu* $N \in \mathbb{N}$ *gibt es einen Algorithmus über* $\{+, -, *, \text{DIV}\}$*, mit dem man mit konstanter Wahrscheinlichkeit und in* $\mathcal{O}(\log^2 N / \log\log N)$ *Schritten eine Primzahl* $p \geq N$ *erhält.*

Beweis. Prüfe zunächst, ob N eine Primzahl ist, indem nach Wilson's Theorem geprüft wird, ob $(N-1)!$ von N geteilt wird. Das geht über $\{+, -, *, \text{DIV}\}$ in $\mathcal{O}(\log N)$ Schritten [40, Abschnitt 3].

Alle benachbarten Fakultäten $(N+k)!$, $k = 1, ..., K$ können in konstanter Zeit berechnet werden, d.h. nach dem Primzahltest für N erhöht sich in der $\mathcal{O}$-Notation für die Primzahltests von $N+1, N+2, ..., N+K$ für $K := \mathcal{O}(\log N)$ nichts.

Rate nun irgendeine $\mathcal{O}(\log N)$-Bit-Zahl $M \leq N$ und teste danach wie oben in $\mathcal{O}(\log N)$ Schritten, ob die Zahlen $N+M, N+M+1, ..., N+M+K$ Primzahlen sind.

Behauptung 5 *Mit einer Wahrscheinlichkeit von* $\Omega(\log\log N / \log N)$ *wird eine Primzahl gefunden.*

Nach $\mathcal{O}(\log N / \log\log N)$ unabhängigen Versuchen hat man mit konstanter Wahrscheinlichkeit eine Primzahl gefunden.

Beweis. der Behauptung

Nach dem Primzahltheorem liegen zwischen N und 2N $\Omega(N/\log N)$ Primzahlen, außerdem liegen in jedem Intervall der Länge K zwischen N und 2N höchstens $\pi(K) \leq \mathcal{O}(K/\log K)$ Primzahlen [32].
Daher besagt das Dirichletsche Schubfachprinzip, dass innerhalb dieser N/K Intervalle mindestens $\Omega(\log K / \log N)$ viele Intervalle wenigstens eine Primzahl enthalten, d.h. $\Omega(\log\log N / \log N)$ für K:=$\mathcal{O}(\log N)$. □

Einen Ansatz zu einem sogar noch schnelleren und deterministischen Weg zur Primzahlbildung erhält man folgendermaßen:

Bemerkung 9 *1947 bewies W.H.MILLS die Existenz einer reellen Zahl* $\theta \approx 1,3063789$ [14], *so dass* $p_n := \lfloor \theta^{3^n} \rfloor$ *eine Folge von Primzahlen bildet mit* $p_{n+1} > p_n^3$.

Nicht bekannt ist, ob θ rational ist. Falls θ rational ist, kann man in $\mathcal{O}(n) = \mathcal{O}(\log N)$ Schritten über $\{+, -, *, \mathrm{DIV}\}$ eine Primzahl $p_n > 3^n =: N$ finden.

Aber wir erhalten auch diese Schranke, selbst wenn θ irrational und *algebraisch* ist. Um $\lfloor \theta^N \rfloor$zu berechnen, betrachte

$$(\theta + \varepsilon)^N = \theta^N + \underbrace{N \cdot \varepsilon^{N-1} + \sum_{k=2}^{\mathsf{N}} \binom{N}{k} \cdot \varepsilon^k \cdot \theta^{N-k}}_{< 1}$$

Aus dieser Abschätzung folgt, dass es genügt eine rationale Approximation θ' von θ mit einer Abweichung von $\varepsilon \approx 2^{-N}/N$ zu berechnen.

Nach Lemma 11 geht das, falls θ algebraisch ist, in $\mathcal{O}(\log N)$ Schritten und dann kann man, da $\lfloor \theta^N \rfloor = \lfloor \theta'^N \rfloor$ gilt, in $\mathcal{O}(\log N)$ Schritten über $\{+, -, *, \mathrm{DIV}\}$ eine Primzahl >N finden

7 Rückblick und Ausblick

In dieser Arbeit wurden die Sprachklassen, die mit der ganzzahligen Division erkannt werden können, auch für den n-dimensionalen Fall vollständig unterschieden. Eine Charakterisierung dieser Sprachen gelang vollständig nur für die ganzzahlige Division mit Konstanten. Für die anderen Operationsmengen $S \subseteq \{+, -, *, *_c, \mathrm{DIV_c}, \mathrm{DIV}\}$ gelang eine Charakterisierung nur für sehr eingeschränkte Eingabemengen. Es könnte im Folgenden untersucht werden, für welche Sprachen sich weitere untere Schranken aus diesen Charakterisierungen von den unteren Schranken für diese Sprachen mit Operationsmengen ohne ganzzahlige Division ableiten lassen.

Die doppellogarithmische Lücke zwischen oberer Schranke $\mathcal{O}(\mathrm{d})$ und unterer Schranke $\Omega(\log\log(\mathrm{d}))$, falls d den Grad des Polynoms bezeichnet, zur Polynomauswertung über dieser mächtigen Operationsmenge $\{+,-,*,\mathrm{DIV}\}$ mit rationalen Konstanten konnte nicht geschlossen werden, sondern führte zu einem seit langem offenen Problem der Zahlentheorie, ob die Reihe $\sum_{n=0}^{\infty} 2^{-dn^2}$ algebraisch ist, und, wenn ja, von welchem Grad. Das Ziel ist weiterhin für eine unendliche Folge von Eingabewerten eine Möglichkeit zu finden, die zugehörigen Werte des Polynoms in $o(\deg p)$ zu berechnen, so dass in Verbindung mit dem Algorithmus von Bshouty insgesamt Polynome in $o(\deg p)$ ausgewertet werden können. Eine Beschleunigung zu $\mathcal{O}(\log \mathrm{d})$ gelang nur bei Hinzunahme der bitweisen Konjunktion als weiterer Operation. Da es aber nicht bekannt ist, ob die bitweise Konjunktion „&" in Polynomialzeit mit der Operationsmenge $\{+,-,*,\mathrm{DIV}\}$ simuliert werden kann, führt dieser Weg nicht zu einem Algorithmus zur Polynomauswertung in $o(\deg p)$ über $\{+,-,*,\mathrm{DIV}\}$, sondern zu der seit langem offenen Frage NP=PSPACE.

Ausgehend von diesem schnellen Algorithmus von Bshouty zur Polynomauswertung über einem endlichen Bereich stellt sich die Frage, ob dieser Algorithmus auf einem modernen Computer schneller ist als andere Algorithmen wie z.B. das Hornerschema. Statt der maximal d Multiplikationen, wenn d der Grad des Polynoms ist, werden 2 ganzzahlige Divisionen benutzt. Dies ist der entgegengesetzte Weg zur Vorgehensweise in z.B. [18], bei der Multiplikationen durch ganzzahlige Divisionen ersetzt werden. Ein Nachteil ist, dass bei diesem Algorithmus riesige Zahlen zur Berechnung benötigt werden und kein realer Rechner in der Lage ist, solch große Zahlen in konstanter Zeit zu verarbeiten. Heißt das von vornherein, dass dieser Algorithmus nicht praktikabel ist. In den letzten Jahrzehnten ist der technische Fortschritt dermaßen angewachsen, dass es in der Bandbreite der arithmetisch-logischen Einheiten (ALU=arithmetical-logical unit) von Prozessoren einen exponentiellen Zuwachs gab. Heutige Computer können auf 64 oder sogar 128 Bits in einer einzigen Anweisung operieren, d.h. das Einheitskostenmodell gilt bereits für sehr große Eingaben und dieser Bitbereich wird immer noch größer.

Nicht nur für die Polynomauswertung wurde ein Algorithmus vorgestellt, der die ganzzahlige Division und bitweise Konjunktion zur Beschleunigung benutzt, sondern es wurden auch Algorithmen entwickelt, die die ganzzahlige Division und andere Operationen wie den größten gemeinsamen Teiler nutzen, um die Matrixmultiplikation und -potenzierung und zahlentheoretische Rechnungen zu beschleunigen. Es wurden einige Lösungsansätze vorgestellt, die zu wirklich effektiven Lösungen führen, wenn gleichzeitig seit langem offene Fragestellungen der Zahlentheorie bewiesen werden könnten. Andere Probleme ließen sich dadurch beschleunigen, dass zusätzlich im Vergleich zur Eingabe sehr große Zahlen mit eingegeben werden. Die Laufzeit dabei ist sogar kürzer als die informationstheoretischen unteren Schranken.

Würde als weitere Operation mit Einheitskosten der Linksshift $\leftarrow: y \mapsto 2^y$ wie in [41] hinzugenommen oder sogar allgemein die Exponentiation $\mathbb{N} \times \mathbb{N} \ni (x, y) \mapsto x^y$ ließen sich auch diese Zahlen, da sie doppelexponentiell groß sind, schnell berechnen.

Abbildungsverzeichnis

Literatur

1. E.Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, On the complexity of numerical analysis, 21 IEEE CCC, 331-339, 2006.
2. L. Babai, B. Just, F. Meyer auf der Heide, On the limits of computations with the floor function, Information and Computation 78(2), 99–107, 1988.
3. A. Bertoni, G. Mauri, N. Sabadini, A characterization of the class of functions computable in polynomiel time on random access machines , STOC,168–176, (1981).
4. A. Bertoni, G. Mauri, N. Sabadini, Simulations among classes of Random Access Machines and equivalence among numbers succinctly represented , Ann. Discrete Math. vol.25,65–90, (1985).
5. N. Bshouty, Euclidean GCD algorithm is not optimal, preprint 1989.
6. N. Bshouty, private communication, 1992.
7. J. Borwein, P. Borwein, PI and the AG, Wiley (1987).
8. P.Bürgisser, M.Clausen, M.A. Shokrollah, Algebraic complexity theory, Springer 1997.
9. I. Baran, E.D. Demaine, M. Patrascu, Subquadratic algorithms for 3 SUM, 9th WADS, Springer LNCS vol. 3608, 409–421, 2005.
10. R. Brent, H. Kung, Systolic VLSI-arrays for linear time GCD-computation, Proc.Int. Conf. on Very Large Scale Integration. (VLSI 83, IFIP), F.Anceau and E. J. Aas(eds), 145-154, 1983.
11. N.Bshouty, Y. Mansour, B.Schieber, P.Tiwari, Fast exponentiation using the truncation operation, computational complexity vol.2, 244–255, 1992.
12. M. Ben-Or, Lower bounds for algebraic computation trees, 15th ACM-STOC, 80–86, 1983.
13. Q. Cheng, On the ultimate complexity of factorials,20 th STACS 2003, Springer LNCS vol.2607, 157–166, 2003.
14. C.K. Caldwell, Y. Cheng, Determing Mill's constant and a note on Honaker' problem, Journal of Integer Sequences vol. 8, article 05.4.1, 2005.
15. D. Coppersmith, S. Winograd, Matrix multiplication via arithmetic progressions, Journal of symbolic computation vol. 9, 251–280, 1990.
16. D. Dobkin, R. L. Lipton, A lower bound of $\frac{1}{2}n^2$ on linear search programs for the knapsack problem, JCSS 16, 417–421, 1975.
17. C.M. Fiduccia, An efficient formula for linear recurrences, SIAM J. Comput. vol.**14:1,** 106–112 , (1985).
18. T. Granlund, P.L. Montgomery, Division by invariant integers using multiplication, ACM SIGPLAN Notices, 61–72, June 1994.
19. A. Gajentaan, M.H. Overmars, On a class of $\mathcal{O}(\mathrm{n}^2)$problems in computational geometry, Computational Geometry: Theory and Applications vol.5, 165–185, 1995.
20. Y. Han, Deterministic sorting in $\mathcal{O}(\mathrm{n}\log\log\mathrm{n})$ time and linear space, Journal of algorithms vol. 50, 96–105, 2004.
21. N. Jacobson, Structure of Rings, American Mathematical Society Colloquium Publications vol.**37** (1964).
22. B. Just, F. Meyer auf der Heide, A. Wigderson, On computations with integer division, RAIRO Informatique Theoretique 23(1), 101–111, 1989.
23. P. Klein, F. Meyer auf der Heide, A lower bound for the knapsack problem on random access machines, Acta Informatica 19(3), 385–396, 1983.

24. D. Kirkpatrick, S. Reisch, Upper bounds on sorting integers on random access machines, Theoretical computer science 28 (3), 263–276, 1989.
25. K. Lürwer-Brüggemeier, F. Meyer auf der Heide, Capabilities and complexity of computations with integer division, 10th STACS, Springer LNCS vol 665, 463–472, 1993.
26. K. Lürwer-Brüggemeier, M. Ziegler, On faster integer calculations using non-arithmetic primitives, 7th UC, Springer LNCS vol 5204, 111–128, 2008.
27. Y. Mansur, B. Schieber, P. Tiwari, Lower bounds for integer greatest common divisor computation, 29th IEEE FOCS, 54–63, 1988.
28. Y. Mansour, B. Schieber, P. Tiwari, The complexity of approximating the square root, 30th IEEE FOCS, 325–330, 1989.
29. J. Meidanis, private communication, 1992.
30. F. Meyer auf der Heide, Lower bounds for solving linear Diophantine equations on random access machines, J. ACM 32(4), 929–937, 1985.
31. F. Meyer auf der Heide, On genuinely time bounded computations, 5th STACS, 1–16,1989Nr. 285,1988
32. H.L.Montgomery, R.C. Vaughan, The large sieve, Mathematika vol. 20, 119–134, 1973.
33. W. J. Paul, J. Simon, Decision trees and random access machines, Monographie 30, L'Enseignement Mathematique, Logique et Algorithmique, Univ. Geneva, Switzerland, 331–340, 1992.
34. V.R. Pratt, M.O. Rabin, L.J. Stockmeyer, A Characterization of the Power of Vector Machines, Proc. 6th Annual ACM Symposium on Theory of Computing, 122–134, (STOC 1974).
35. P. Pritchard, a sublinear additive sieve for finding prime numbers, Communications of the ACM, vol.24, 18–23, 1981.
36. J.F. Randolph, Basic real and abstract analysis, Academic Press, 1968.
37. P. Ribenboim, The new book of prime number records, 3rd edition springer, 1996.
38. P. Ribenboim, My numbers, my friends, Springer, 2000.
39. A.Schönhage, On the Power of Random Access Machines, Automata Languages and Programming, 6th Colloquium, Springer LNCS vol. 71,520-529, 1979
40. A.Shamir, Factoring numbers in $\mathcal{O}(\log n)$ arithmetic steps, Information Processing Letters vol. 8(1), 28-31, 1979.
41. J. Simon, Division is good, 20th IEEE FOCS, 411–420, 1979.
42. A. Yao, Lower bounds for algebraic computation trees with integer inputs, 30th IEEE FOCS, 308–313, 1989.