Seminar
Computeralgebra
WS 2007/2008
Prof. Dr. Sebastian Iwanowski

Langzahlarithmetik
Addition und Multiplikation

Jörg Fitzner
minf2913

Inhalt

1 Einleitung

Die Computeralgebra befasst sich schwerpunktmäßig mit der Umsetzung symbolischen Rechnens in Computersystemen. Zu einem leistungsfähigen und mathematisch korrekten Umgang mit den zu verarbeitenden Formeln und Ausdrücken gehört dabei, neben Regeln zur strukturellen Umformung der Formeln, auch der exakte Umgang mit Zahlen. Schließlich stellen Zahlen, neben Symbolen und Operatoren, einen inhärenten Bestandteil mathematischer Formeln dar, beispielsweise in Form von Faktoren und Potenzen.

1.1 Langzahlarithmetik

Im Unterschied zu numerischen Systemen, muss die Behandlung von Zahlen in Computeralgebrasystemen stets exakt erfolgen. Es darf also kein Verlust an Genauigkeit durch Näherungen oder Rundungen auftreten. Andernfalls wäre eine strukturelle Weiterverarbeitung der Formeln nicht mehr sinnvoll.
Durch Umformungen und gegenseitiges Einsetzen von Formeln ineinander, können zudem sehr schnell äußerst große Zahlen auftreten, mit denen das System ebenso umgehen können muss.

Umsetzungen ganzer Zahlen in Computern, welche den Kriterien

- Darstellung beliebig langer Zahlen (Zahlenraum $\mathbb{Z}$)
- exakte Darstellung aller Werte des Zahlenraums

genügen, werden als *Langzahlen*, beziehungsweise entsprechende Rechenoperationen damit als *Langzahlarithmetik*, bezeichnet.

Grundlage für weitere Zahlenräume

Auf Grundlage von Langzahlen lassen sich in Computersystemen wiederum weitere Zahlenräume darstellen.

- Rationale Zahlen ($\mathbb{Q}$)
- bestimmte irrationale reelle Zahlen ($\subset\mathbb{R}$)
- Komplexe Zahlen ($\mathbb{C}$)

Die Darstellung geschieht dabei indirekt, durch Kombination von ganzen Zahlen (in Form von Langzahlen) mit entsprechenden Kombinationssymbolen. Rationale Zahlen können beispielsweise durch Angabe eines Bruchstrich-Symbols zusammen mit zwei Langzahlen (Zähler und Nenner) dargestellt werden. Bestimmte irrationale Zahlen durch Verwendung eines Wurzelzeichens, usw.
Bei der Durchführung von Berechnungen des Computeralgebrasystems wird dann im Raum der symbolischen Darstellung dieser Zahlenräume weitergerechnet.

1.2 Themeneingrenzung

Diese Ausarbeitung soll, basierend auf dem Seminarvortrag zum selben Thema, die notwendigen Konzepte zur Implementierung einer Langzahlarithmetik mit den grundlegenden Operationen Addition/Subtraktion und Multiplikation vorstellen.
Vorausgesetzt werden grundlegende Kenntnisse aus Mathematik und Informatik.

2 Zahlendarstellungen

2.1 Darstellungsformen natürlicher Zahlen

Die Menschheitsgeschichte hat eine Vielfalt an Darstellungsformen für (zunächst natürliche) Zahlen hervorgebracht. Im alltäglichen Umgang hat sich das Dezimalsystem mit den arabischen Ziffern durchgesetzt. Abstrahiert man davon, entsteht die Klasse der Positionssysteme, die sich gegenüber manchem traditionellem Zahlsystem (wie beispielsweise dem der römischen Zahlen) durch einfachere Algorithmen zur Berechnung mathematischer Operationen auszeichnet.

2.1.1 Positionssystem

Zahlen in einem Positionssystem werden in Form einer Ziffernfolge dargestellt. Die einzelnen Ziffern der Folge werden hierbei als Koeffizienten der Linearkombination von Potenzen einer vorgegebenen Basis $B \in \mathbb{N}_{\geq 2}$ interpretiert. Die Ziffern selbst sind dabei Symbole für die Zahlen $Null$ bis $B-1$.

$$z = \sum_{k=0}^{\#z-1} z_k B^k \quad \text{mit } z_k \in [0,1,...,B-1]$$

Die Anzahl der Ziffern einer Zahl z wird Länge der Zahl genannt, repräsentiert durch die symbolische Schreibweise $\#z$.
Die Position einer Ziffer z_k innerhalb der Ziffernfolge einer Zahl wird auch als Stelle bezeichnet (hier k); sie entspricht gleichzeitig dem Exponenten der Basis für die Wertigkeit der Ziffernposition (die Zählung beginnt also bei Null).

In kanonischer Schreibweise wird die niederwertigste Stelle immer ganz Links geschrieben. Diese Stelle wird daher auch als die letzte Stelle bezeichnet.

Beispiele

Zur Basis Zehn ergibt sich somit genau das arabische Dezimalsystem. Zur Basis Zwei das üblicherweise in Computersystemen auf Hardwareebene verwendete Binärsystem (auch Dualsystem genannt).

Geht die verwendete Basis nicht aus dem Kontext hervor, ist deren explizite Angabe als tiefgestelltes Suffix[1] sinnvoll.

- 440_{10}
- 110111000_2

Eigenschaften der Darstellung

Zahlendarstellungen in Positionssystemen sind, sofern ohne führende Nullen angegeben, immer eindeutig und existent (s. Abschnitt 2.1.3).

[1] gemäß Konvention wird die Basis selbst immer zur Basis Zehn angegeben

2.1.2 Indirekte Darstellungen

Zahlen lassen sich auch indirekt, durch wohldefinierte mathematische Kombination anderer Zahlen ausdrücken. Die einzelnen Bestandteile einer solchen Darstellung sind also nicht Ziffern, sondern Zahlen – die selbst wiederum in einem beliebigen System dargestellt sein können.

Ganzzahliger Quotient mit modalem Rest

Hier wird die Zahl als ganzzahliger Quotient (Operation DIV) und Rest der ganzzahligen Division (Operation MOD) zu einem bestimmten Divisor $D \in \mathbb{Z}_{\neq 0}$ angegeben.

Beispiel (mit Divisor 440): $1999 = (4, 239)_{440} = 4 \cdot 440 + 239$

Primfaktorpotenzen

Die Zahl wird als Tupel der Potenzen aller aufeinander folgenden Primzahlen ihrer Primfaktorzerlegung dargestellt (vollständig von 2 bis zur größten in der Zerlegung auftretenden Primzahl).

Beispiel: $440 = {}_{\mathrm{PRIME}}(3, 0, 1, 0, 1) = 2^3 \cdot 3^0 \cdot 5^1 \cdot 7^0 \cdot 11^1$

Die beiden genannten indirekten Darstellungen sind ebenfalls eindeutig und existent (s. Abschnitt 2.1.3).

Weitere indirekte Darstellungen

Es lassen sich beliebige weitere indirekte Darstellungen denken. So ist zum Beispiel die Darstellung rationaler Zahlen durch ganzzahligen Zähler und Nenner mit einem Bruchstrich in diesem Sinne auch eine indirekte Zahlendarstellung. Jedoch sind, wie dieses Beispiel zeigt, nicht alle indirekten Darstellungen auch automatisch eindeutig und existent.

2.1.3 Eigenschaften von Zahlendarstellungsformen

Existenz

Existenz bedeutet in diesem Zusammenhang, dass für jede Zahl des Zahlenbereichs mindestens eine Darstellung existiert.
Dies ist für Computeralgebrasysteme wichtig, um jede mathematische Zahl tatsächlich repräsentieren zu können.

Eindeutigkeit

Eindeutigkeit bedeutet hier, dass für jede Zahl höchstens eine Repräsentation existiert.
Dies ist wichtig, um die Algorithmen für die mathematischen Operationen möglichst einfach zu halten. Insbesondere ist die Gleichheit (bzw. Ungleichheit) zweier Zahlen somit effizient zu erkennen.

Gelten für eine gegebene Darstellungsform Existenz und Eindeutigkeit zusammen, bedeutet dies, es gibt für jede Zahl *genau eine* Repräsentation in diesem System.

2.2 Kurzzahldarstellung im Computer

Computer bringen von sich aus ein natives System zur Darstellung und Verarbeitung von ganzen Zahlen mit sich. Dabei handelt es sich in aller Regel um eine Darstellung im Binärsystem (also ein Positionssystem zur Basis $B=2$, vgl. Abschnitt 2.1.1), welches in der Hardware des Systems umgesetzt ist.
Jedoch ist bei diesen nativen Computer-Zahlendarstellungen die Länge der Binärzahlen implizit auf die Wortlänge des Systems[2] beschränkt (bei derzeit gängigen Systemen üblicherweise 32). Deshalb spricht man hierbei von *Kurzzahlen*, beziehungsweise von *Kurzzahlarithmetik*.

Negative ganze Zahlen werden in der Kurzzahlarithmetik üblicherweise durch das Basiskomplement ihres Betrags dargestellt[3] (implizites Vorzeichen).

Insgesamt ergibt sich somit für den Wertebereich einer ganzen Zahl z in der Kurzzahldarstellung grundsätzlich folgende Einschränkung: $z \in \mathbb{Z},\ \text{mit}\ -2^{\text{Wortlänge}-1} \le z < 2^{\text{Wortlänge}-1}$

2.2.1 Kurzzahlarithmetik im Computer

Zur Verarbeitung der Kurzzahlen bringt die Hardware für grundlegende Operationen bereits eine entsprechende Kurzzahlarithmetik mit. Um zu gewährleisten, dass die Resultate von Operationen dieser Arithmetik immer im Zahlenbereich der Kurzzahldarstellung des Systems bleiben, müssen die Operandenbereiche zusätzlich eingeschränkt werden[4]:

- Vergleich $-2^{\text{Wortlänge}-1} \le z < 2^{\text{Wortlänge}-1}$
- Addition $-2^{\text{Wortlänge}-2} \le z < 2^{\text{Wortlänge}-2}$
- Subtraktion $-2^{\text{Wortlänge}-1} < z < 2^{\text{Wortlänge}-1}$
- Multiplikation $-2^{\lfloor(\text{Wortlänge}-1)/2\rfloor} \le z < 2^{\lfloor(\text{Wortlänge}-1)/2\rfloor}$
- Division $-2^{\text{Wortlänge}-1} < z < 2^{\text{Wortlänge}-1}$

Der Aufwand für die Ausführungsdauer einer Operation der Kurzzahlarithmetik lässt sich für die aufgeführten Operationen mit einer Konstanten abschätzen, hat also eine Komplexität von $O(1)$.

[2] präzise ausgedrückt: auf die Wortlänge des Systems gemessen in Bit
[3] Dies hat den Vorteil die Addition negativer Zahlen nicht gesondert behandeln zu müssen (vgl. Addition ganzer Zahlen in der Langzahlarithmetik, Abschnitt 3.2.3).
[4] N.B.: Bei der Addition zweier gleich langer Zahlen kann sich die Länge des Resultats maximal um eine Stelle erhöhen; bei der Multiplikation verdoppeln. Der Anteil des Resultats, der über den zulässigen Bereich der Operanden hinausgeht, wird als Übertrag bezeichnet.

2.3 Langzahldarstellung im Computer

Zur Darstellung der Langzahlen im Computer wird ebenfalls ein Positionssystem gewählt. Allerdings wird hier nicht lediglich ein Bit pro Ziffernstelle aufgewendet, wie bei den Kurzzahlen, sondern ein ganzes Wort für jede Ziffer. Die Basis beträgt hier also nicht 2, sondern maximal die Wortlänge des Systems.
Da in den Algorithmen der Langzahlarithmetik die Kurzzahladdition und Kurzzahlmultiplikation auf den einzelnen Ziffern der Langzahlen verwendet werden können soll, muss die Basis des Positionssystems zur Langzahldarstellung auf den Operandenbereich der Kurzzahlmultiplikation eingeschränkt werden $\Rightarrow B=2^{\lfloor(\text{Wortlänge}-1)/2\rfloor}$.

Die komplette Ziffernfolge wird nun in einer dynamischen Datenstruktur abgelegt. Um ganze Zahlen darstellen zu können, wird zusätzlich ein explizites Vorzeichen pro Langzahl gespeichert[5]. Alle Ziffern werden dabei als positive Kurzzahlen dargestellt, unabhängig vom Vorzeichen der Langzahl. (Wohlgemerkt: Bei einer Wortlänge von 32 Bit kommen die einzelnen *Ziffern* dann aus einem Bereich von 0 bis 32767.)

Als dynamische Datenstrukturen zur Implementation bieten sich eine lineare Liste oder ein variables Array an. In der Praxis bleibt die maximale Länge der darstellbaren Langzahlen allerdings immer durch die Menge des verfügbaren Speichers beschränkt.

2.3.1 Hinweise zu den Datenstrukturen

Zur Implementation der Datenstruktur seien hier einige Hinweise gegeben, auch wenn sich deren Zusammenhänge teilweise erst bei der Betrachtung der Algorithmen zur Langzahlarithmetik erschließen.

Lineare Liste

- Falls einfach-verkettet, muss die niederwertigste Stelle zuerst verkettet sein, damit die Übertragspropagation und das Hinzufügen abschließender Nullen effizient durchgeführt werden kann. (Besser: ring-verkettet → günstiges Hinzfügen auch von führenden Nullen möglich.)
- Für eine effiziente Bisektionierung muss die Länge der Langzahlen auf den Wertebereich der Kurzzahlen beschränkt sein[6]. (Sonst erhöht sich Komplexität der Bisektionierungsalgorithmen um den Summanden $\#z^2 \cdot \log_2(\#z)$.)
- Speicherverwaltung
  (+) effiziente Anpassung der Speichermenge
  (−) ca. doppelter Speicherbedarf (durch Zeiger auf jeweils nächste Stelle)

Variables Array

- Die Länge der darstellbaren Zahlen ist implizit auf den Wertebereich des Indexdatentyps des Arrays beschränkt (i.d.R. $2^{\text{Wortlänge}}$), da als Index keine Langzahlen verwendet werden können (Henne/Ei-Problem)[6].
- Der Indexdatentyp muss über eine Arithmetik mit $O(1)$ verfügen (sonst ist die Komplexitätsbetrachtung für die Langzahlalgorithmen hinfällig).
- Das explizite Hinzufügen von führenden oder abschließenden Nullen ist aufwändiger (im Allgemeinen Kopieren aller bisherigen Ziffern notwendig).
- Speicherverwaltung
  (+) geringerer Speicherbedarf
  (−) aufwändigere Anpassung der Speichermenge

[5] Dies zieht die Notwendigkeit einer Normierung der Darstellung der Null nach sich, um die Eindeutigkeit zu wahren.
[6] Die Beschränkung der *Länge* der Langzahlen auf den Wertebereich der Kurzzahlen ist in der Praxis i.d.R. folgenlos, da die Menge der tatsächlich adressierbaren Speicherworte üblicherweise ohnehin auf den Wertebereich der Kurzzahlen eingeschränkt ist.

3 Algorithmen der Langzahlarithmetik

Die hier vorgestellten Algorithmen der Arithmetik für Langzahlen haben eine Gemeinsamkeit: Sie führen Operationen der Langzahlarithmetik letztlich auf Operationen der Kurzzahlarithmetik zurück.

Für die Vergegenwärtigung der Funktionsweise der einzelnen Algorithmen kann es hilfreich sein die Basis der Langzahlen einfach als 10 anzunehmen, anstatt z.B. 32768 bei einem 32 Bit System. Man stellt sich die Algorithmen dann also auf das Dezimalsystem angewendet vor. (Die schriftlichen Verfahren der Dezimalzahlarithmetik stellen im Übrigen tatsächlich den Ausgangspunkt der meisten dieser Algorithmen dar.)

Für die Komplexitätsbetrachtungen wird im Folgenden als Kostenmaß die jeweils notwendige Anzahl an Kurzzahloperationen zugrunde gelegt[7], wobei alle mathematischen Operationen für genau zwei Operanden betrachtet werden.

Implizite Operandenpräfixerweiterung

In vielen Berechnungen der Langzahlalgorithmen werden Verarbeitungsschritte mit Ziffern der gleichen Position beider Operanden durchgeführt. Dabei kann der Fall auftreten, dass ein Operand (oder gar beide) an der betreffenden Position über gar keine Ziffern verfügt, weil dessen Länge kürzer ist. Effektiv hat eine Zahl an einer solchen Stelle mathematisch immer den zur Ziffer *Null* äquivalenten Wert[8].

3.1 Relationale Operationen

3.1.1 Allgemeine Vergleichsoperation

Ein Algorithmus zur Implementierung einer Funktion, die ermittelt in welcher Reihenfolge zwei Langzahlen bezüglich ihrer natürlichen Ordnung zueinander stehen, kann wie folgt ablaufen.

- Ziffern jeweils gleicher Stellen werden, beginnend bei der letzten, miteinander verglichen
- Gesamtergebnis steht fest, sobald Ziffern einer Stelle unterschiedlich, oder die vorderste Stelle der kürzeren Zahl verglichen wurde
- falls Gesamtergebnis noch nicht fest steht, nächste Stelle vergleichen
- Gesamtvergleichsergebnis ist Ergebnis des letzten Ziffernvergleichs

Zusätzlich ist ein Vergleich des Vorzeichens notwendig, was effizienter Weise als erstes geschieht.

Komplexitätsbetrachtung

Jeder Ziffernvergleich, sowie der Vorzeichenvergleich, kann durch 1 Kurzzahlvergleich durchgeführt werden. Somit ergeben sich insgesamt maximal $\min(\#a,\#b)+1$ Kurzzahlvergleiche $\Rightarrow \text{compare}(a,b) \in O(\min(\#a,\#b))$.

3.1.2 Prädikative Vergleiche

Spezifische Vergleichsoperationen wie $=$, $\neq$, $<$, $\leq$, $>$ und $\geq$ können unmittelbar auf Basis der allgemeinen Vergleichsoperation implementiert werden. Dementsprechend gehören sie der selben Komplexitätsklasse an $\Rightarrow a \leq b \in O(\min(\#a,\#b))$.

[7] Grundsätzlich sollte angemerkt werden, dass bei der Verarbeitung algorithmischer Kontrollstrukturen in einer konkreten Umsetzung als Programm implizit ggf. weitere Lang- oder Kurzzahloperationen verwendet werden (z.B. Inkrementierung von Schleifenvariablen, Vergleiche oder Längenermittlung). Diese lassen sich hier allerdings so integrieren, dass sie die Komplexitätsklasse nicht beeinflussen.

[8] Man mache sich diesen Sachverhalt anhand der schriftlichen Schuladdition zweier natürlicher Zahlen unterschiedlicher Länge bewusst (siehe auch Abschnitt 3.2.1).

3.2 Addition und Subtraktion

3.2.1 Addition natürlicher Zahlen

Der Algorithmus zur Addition entspricht genau dem Verfahren der Schulmethode, also der schriftlichen Addition. (Hierzu sollte man sich bewusst machen, dass bei diesem Verfahren ausschließlich natürliche Zahlen addiert werden.)

- Ziffern an den gleichen Stellen werden jeweils addiert (beginnend bei der letzten Stelle)
- eventuell entstehender Übertrag[9] wird jeweils bei der nächst höherwertigen Stelle mit hinzuaddiert

Man beginnt bei der letzten Stelle, damit die Überträge immer direkt bei der unmittelbar folgenden Ziffernaddition mit verarbeitet werden können. (In umgekehrter Richtung müsste man sich alle Stellenüberträge merken und würde zwei Iterationen benötigen.)

Komplexitätsbetrachtung

Jede Addition von zwei Ziffern, beziehungsweise von Ziffer und Übertrag, kann durch eine Kurzzahladdition durchgeführt werden. Somit ergeben sich insgesamt maximal $2 \cdot \max(\#a,\#b)$ Kurzzahladditionen $\Rightarrow |a|+|b| \in O(\max(\#a,\#b))$.

Zudem bleibt festzustellen, dass die Länge des Resultats maximal um eine Stelle größer werden kann, als die des längsten Operanden.

3.2.2 Subtraktion natürlicher Zahlen

Analog zur Addition, verfährt man bei der Subtraktion gemäß der Schulmethode zur schriftlichen Subtraktion. Dabei wird immer der kleinere Betrag von dem größeren abgezogen.

- Voraussetzung: Minuend ≥ Subtrahend
- Ziffern an den gleichen Stellen werden jeweils subtrahiert (beginnend bei der letzten Stelle)
  - ⇨ ergibt sich dabei eine negative Zahl, entsteht ein Übertrag[9] und als Stellenresultat wird das Basiskomplement des Betrags dieser Stellensubtraktion gebildet
- eventuell entstandener Übertrag wird jeweils bei der nächst höherwertigen Stelle zusätzlich abgezogen

Komplexitätsbetrachtung

Jede Subtraktion von zwei Ziffern, beziehungsweise Ziffer und Übertrag, kann durch eine Kurzzahlsubtraktion durchgeführt werden. Jedes Basiskomplement kann durch zwei Kurzzahloperationen gebildet werden[10]. Insgesamt ergeben sich damit also maximal $4 \cdot \max(\#a,\#b)-3$ Kurzzahloperationen $\Rightarrow |a|-|b| \in O(\max(\#a,\#b))$.

Die Länge des Resultats bleibt im Übrigen maximal genau so lang, wie die des längsten Operanden.

[9] der Stellenübertrag bei der Addition und Subtraktion ist immer genau 0 oder 1

[10] Das Basiskomplement kann nicht mit einer Kurzzahlsubtraktion (als Basis–Ziffer) berechnet werden, da die Basis außerhalb des Wertebereichs der Operanden liegt. Somit muss zunächst das Stellenkomplement (als (Basis-1)–Ziffer) und dann das Basiskomplement (als Stellenkomplement+1) gebildet werden.

3.2.3 Addition ganzer Zahlen

Die Addition ganzer Zahlen wird nun auf die Verwendung der Addition und Subtraktion natürlicher Zahlen zurückgeführt. Dies entspricht im Prinzip genau der Vorgehensweise die üblicherweise beim Kopfrechnen anwendet wird.

- falls $\text{sign}(a) \equiv \text{sign}(b) \quad \Rightarrow \quad |a+b| = |a|+|b| \,, \quad \text{sign}(a+b) = \text{sign}(a)$
- falls $\text{sign}(a) \neq \text{sign}(b)$
  - ⇨ falls $|a| \geq |b| \quad \Rightarrow \quad |a+b| = |a|-|b| \,, \quad \text{sign}(a+b) = \text{sign}(a)$
  - ⇨ falls $|a| < |b| \quad \Rightarrow \quad |a+b| = |b|-|a| \,, \quad \text{sign}(a+b) = \text{sign}(b)$

Dabei ist es wichtig, für die interne Subtraktionsoperation den Algorithmus für natürliche Zahlen mit Minuend größer oder gleich Subtrahend (s. Abschnitt 3.2.2) zu verwenden und nicht die Subtraktion für ganze Zahlen, um nicht in eine Endlosrekursion zu geraten.

Komplexitätsbetrachtung

Aufgrund der getrennten Darstellung von Betrag und Vorzeichen in den Langzahlen, stellen die Zugriffe auf die Vorzeichen sowie die Ermittlung der Beträge zur Verarbeitung keinen weiteren Aufwand dar.

Bei der Berechnung ist maximal 1 Langzahlvergleich, sowie entweder eine Langzahladdition oder -Subtraktion natürlicher Zahlen notwendig. Die maximale Gesamtanzahl von Kurzzahloperationen beträgt daher $4 \cdot \max(\#a,\#b)+\min(\#a,\#b)-2$ Kurzzahloperationen $\Rightarrow a+b \in O(\max(\#a,\#b))$.

3.2.4 Subtraktion ganzer Zahlen

Die Subtraktion ganzer Zahlen lässt sich durch Umkehrung eines Vorzeichens auf eine Addition von ganzen Zahlen zurückführen.

- es gilt $a - b = a + (-b)$

Komplexitätsbetrachtung

Aufgrund der getrennten Darstellung von Vorzeichen und Betrag, lässt sich die Invertierung des Vorzeichens mit 1 Kurzzahloperation bewerkstelligen. Daraus resultieren maximal $4 \cdot \max(\#a,\#b)+\min(\#a,\#b)-1$ Kurzzahloperationen $\Rightarrow a-b \in O(\max(\#a,\#b))$.

3.3 Multiplikation

Zur Implementierung der Multiplikation genügt zunächst die Betrachtung von natürlichen Zahlen. Da $|a \cdot b| = |a| \cdot |b|$ und $\text{sign}(a \cdot b) = \text{sign}(a) \cdot \text{sign}(b)$, kann jeglicher Algorithmus zur Multiplikation natürlicher Zahlen leicht auf ganze Zahlen erweitert werden, indem zunächst des Produkt der Beträge, und anschließend separat das Vorzeichen des ganzen Resultats ermittelt wird.
Weil sich Vorzeichen und Beträge dank expliziter Darstellung unmittelbar zugreifen lassen, und sich das Vorzeichen des Resultats mittels einer einzigen zusätzlichen Kurzzahloperation berechnen lässt, ändert sich an der Einstufung der Komplexitätsklasse nichts.

Grundsätzlich kann bei einer Multiplikation ganzer Zahlen das Resultat maximal so lang werden wie die Summe der Längen aller Operanden.
Die Multiplikation einer Zahl mit einer Potenz der Basis kann auch durch direktes Anhängen der entsprechenden Anzahl von Stellen mit der Ziffer 0 bewerkstelligt werden[11]. (Wie man es auch vom alltäglichen Umgang mit den Dezimalzahlen kennt.)

Da sich die Komplexitätsbetrachtungen ohnehin immer auf den ungünstigsten Fall beziehen, werden diese, der Übersichtlichkeit halber, im Folgenden ohne Beschränkung der Allgemeingültigkeit zumeist einfach für die Multiplikation zweier gleich langer Operanden betrachtet. (Für die Länge der Operanden wird in diesem Fall immer das Symbol $\#z$ verwendet, gleich welche Symbole konkret für die Operanden gewählt wurden.)

3.3.1 Inkrementelle Methode

Der erste Ansatz zur Umsetzung der Multiplikation entspricht genau der mathematischen Definition der Multiplikation und führt sie direkt auf eine mehrfache Anwendung der Langzahladdition zurück.

$$a \cdot b = \sum_{i=1}^{b} a \quad \text{mit } a,b \in \mathbb{N} \text{ (und hier } a \geq b)$$

- die größere Zahl wird zu sich selbst addiert (Langzahladdition)
- dies wird so lange kumulierend wiederholt, bis die Anzahl der addierten Summanden den Wert der kleineren Zahl erreicht hat

Komplexitätsbetrachtung

Da jede einzelne Langzahladdition hier mit $O(\max(\#a,\#b))$ zu Buche schlägt, beträgt die Gesamtkomplexität an Kurzzahloperationen $\Rightarrow a \cdot b \in O(\max(\#a,\#b) \cdot \min(a,b)) \subseteq O(\#z \cdot B^{\#z})$.

Angesichts des für gleich lange Zahlen äußerst ungünstigen exponentiellen Wachstumsverhaltens, besteht dringender Bedarf an einem effizienteren Algorithmus.

[11] Der Aufwand für das Hinzufügen einer abschließenden Ziffer zu einer Zahl z ist von der Implementation der dynamischen Datenstruktur zur Darstellung der Langzahlen abhängig. Bei einer linearen Liste beträgt die Komplexität $O(1)$, beim variablen Array $O(\#z)$.
Das Hinzufügen von n abschließenden Ziffern in einem gebündelten Schritt lässt sich so umsetzen, dass in beiden Fällen eine Komplexität von höchstens $O(\#z+n)$ resultiert.

3.3.2 Fortgeschrittene Schulmethode

Der nächste Ansatz besteht darin, das Verfahren der schriftlichen Multiplikation, der so genannten Schulmethode, zur Multiplikation von Langzahlen zu verwenden.

Einfache Schulmethode

Die Vorgehensweise dieser Methode besteht darin, zunächst sukzessive die Produkte des ersten Operanden mit jeder einzelnen Ziffer des zweiten Operanden zu bilden und entsprechend der Wertigkeit der Ziffernstelle des zweiten Operanden zu notieren. Die Summe dieser Teilprodukte ergibt dann das Gesamtresultat der Multiplikation.
Um die einzelnen Teilprodukte zu berechnen wird man, spätestens bei wirklich großen Zahlen, jeweils eine Nebenrechnung durchführen, in der immer die Summe der Produkte jeder einzelnen Ziffer des ersten Operanden mit der aktuell betrachteten Ziffer des zweiten Operanden (unter Beachtung der Wertigkeit der Ziffernstelle) gebildet wird.

Auf unterster Ebene dieser Methode werden also nur noch Ziffern miteinander multipliziert – also *Kurzzahlmultiplikationen*. Und zwar genau die Produkte aller paarweisen Kombinationen von Ziffern beider Operanden – das sind $\#z^2$ Produkte.

Andererseits ist zu beachten, dass es sich in der so beschriebenen einfachen Variante der Schulmethode bei allen Additionen zunächst um *Langzahladditionen* handelt. Bei $O(\#z)$ Kurzzahladditionen pro Langzahladdition bedeutet dies einen Gesamtaufwand von $O(\#z^3)$ allein an Kurzzahladditionen.
Allerdings werden dabei, aufgrund der Verschiebung der berechneten Teilprodukte entsprechend ihrer Wertigkeit[12], effektiv sehr viele Null-Ziffern aufaddiert.

Fortgeschrittene Schulmethode

Durch günstige Wahl der Reihenfolge von Berechnung und Akkumulation aller Ziffernpaarprodukte, lassen sich deren Überträge sowie die Überträge aller Ziffern-Additionen en passant durch direkte *Kurzzahladditionen* verrechnen.

- alle Kombinationen von Ziffern beider Zahlen werden paarweise miteinander multipliziert (Kurzzahlmultiplikation)
- Kombinationsresultate werden kumulativ zu den entsprechenden Stellen des Gesamtresultats addiert (Kurzzahladdition)
  - ⇨ dabei entstehende Überträge werden unmittelbar in die Akkumulation der nachfolgenden Stelle einbezogen (Kurzzahladdition)

Um die richtige Reihenfolge zu erhalten, muss die Kombination verschachtelt, bei den letzten Stellen beginnend, jeweils bei stufenweise festgehaltener Ziffer der einen Zahl, über alle Ziffern der anderen Zahl durchlaufen werden.

Summa summarum handelt es sich also im Prinzip um die Berechnung der vollständig ausmultiplizierten Darstellung des Produkts beider Operanden entsprechend ihrer Definition durch das Positionssystem. Allerdings in einer bestimmten, wenngleich nahe liegenden, Reihenfolge.

$$a \cdot b = \sum_{k=0}^{\#a-1} a_k B^k \cdot \sum_{l=0}^{\#b-1} b_l B^l = \sum_{l=0}^{\#b-1} \left(\sum_{k=0}^{\#a-1} a_k \cdot b_l \cdot B^{k+l} \right)$$

Komplexitätsbetrachtung

Jede Ziffernmultiplikation kann mit 1 Kurzzahlmultiplikation berechnet werden. Jeder Akkumulationsschritt von Teilprodukt und Übertrag[13] zu einer Resultatstelle kann mit 2 Kurzzahladditionen durchgeführt werden. Dies ergibt insgesamt maximal $3 \cdot \#z^2$ Kurzzahloperationen $\Rightarrow a \cdot b \in O(\#z^2)$.

[12] Bei der Berechnung des Produkts zweier Ziffern a_i und b_j ergibt sich die effektive Wertigkeit als Faktor B^{i+j}. Dies ist gleichbedeutend mit dem Anhängen von $i+j$ abschließenden Nullen bzw. einer Verschiebung im Rechentableau um $i+j$ Stellen nach rechts.

[13] Die Länge der jeweiligen Ziffernproduktresultate kann maximal 2 werden. Daher bestehen die entstehenden Überträge nicht nur aus Additionsüberträgen durch die Akkumulation in der betreffenden Stelle, sondern vor allem auch aus dem höherwertigen Teil des jeweiligen Kombinationsresultates.

3.3.3 Rekursive Bisektionierung

Bei dem nächsten Verfahren werden beide Operanden einfach der Länge nach in jeweils zwei Hälften aufgeteilt (dieses wird als Bisektionierung bezeichnet). Dazu muss die Länge der Operanden allerdings eine gerade Zahl sein.

$$a = \overleftarrow{a} \cdot B^{\#z/2} + \vec{a} \qquad \text{und}^{14} \qquad b = \overleftarrow{b} \cdot B^{\#z/2} + \vec{b}$$

Für das Produkt erhält man dann durch ausmultiplizieren:

$$a \cdot b = \overleftarrow{a} \cdot \overleftarrow{b} \cdot B^{\#z} + (\overleftarrow{a} \cdot \vec{b} + \vec{a} \cdot \overleftarrow{b}) \cdot B^{\#z/2} + \vec{a} \cdot \vec{b} \qquad \textit{Bisektionsgleichung}$$

Sofern es sich bei a und b um zwei gleich lange Operanden handelt, deren Länge genau eine Potenz von 2 ist ($\#z=2^n$, $n \in \mathbb{N}$), lassen sich nun die einzelnen Kombinationsprodukte der Hälften wiederum durch Bisektionierung bilden. Führt man dieses Verfahren rekursiv weiter, erhält man auf unterster Ebene ausschließlich Produkte von Zahlen der Länge 1, also Ziffern, die sich dann jeweils durch eine *Kurzzahlmultiplikation* berechnen lassen.

Um nach diesem Schema ebenso Operanden anderer Längen zu verarbeiten, können diese temporär mit führenden Nullen auf die nächste 2er-Potenz-Länge aufgefüllt werden[15].

Man beachte, dass zur Umsetzung der Bisektionierung eine gewisse Arithmetik auf der Länge der Zahlen notwendig ist. Die Länge der Langzahlen muss ermittelt werden können, und mit den Längen muss, zumindest in einem gewissen Rahmen, gerechnet werden können (Halbierung des Wertes). Siehe hierzu auch Abschnitt 2.3.1, „Hinweise zu den Datenstrukturen".

3.3.3.1 Symmetrische Komposition

Führt man die Multiplikation zweier Zahlen vollständig rekursiv nach obiger Formel durch, entsteht folgender Algorithmus.

- jeder Operand wird der Länge nach in zwei Hälften aufgeteilt
- alle paarweisen Kombinationen der entstandenen vier Teile werden miteinander multipliziert (ergibt vier Kombinationen)
- Teilprodukte werden entsprechend ihrer resultierenden Wertigkeit summiert
- Teilprodukte werden rekursiv durch erneute Halbierung berechnet
  - ⇨ Produkte zweier Ziffern werden mittels Kurzzahlmultiplikation berechnet

Komplexitätsbetrachtung

Bei diesem Ablauf vervierfacht sich die Anzahl der Multiplikationen pro Rekursionsschritt. Auf unterster Ebene der Rekursion ergeben sich daher $4^{\log_2(\#z)} = \#z^2$ *Kurzzahlmultiplikationen* – also $\#z^2$ paarweise Multiplikationen von Ziffern beider Operanden. Dies sind genau dieselben Ziffernmultiplikationen wie bei der Schulmethode, nämlich wieder schlichtweg alle paarweisen Ziffernkombinationen.

Allerdings sind hier zur Komposition des Resultats *mehr Kurzzahladditionen* notwendig, als bei der fortgeschrittenen Schulmethode, nämlich $12 \cdot \#z^2 - 12 \cdot \#z$ (gegenüber $2 \cdot \#z^2$)[16].

Wenngleich sich insgesamt mit $a \cdot b \in O(\#z^2)$ dieselbe Komplexitätsklasse ergibt, hat man gegenüber der fortgeschrittenen Schulmethode zunächst nichts gewonnen.
Zudem ist zu bedenken, dass durch die rekursive Umsetzung des Berechnungsablaufs absolut gesehen ein Vielfaches an Speicher benötigt wird (die Speicherkomplexität beträgt jedoch weiterhin $O(\#z)$).

---

[14] Die Pfeile sollen hier keine Vektoren darstellen, sondern den vorderen bzw. hinteren Teil einer Zahl markieren. Es gilt $\overleftarrow{a} = a \operatorname{DIV} B^{\#z/2}$ und $\vec{a} = a \operatorname{MOD} B^{\#z/2}$.

[15] Eine Operation zum Auffüllen der Länge einer Zahl mit führenden Nullen in einem gebündelten Schritt auf insgesamt $\#z$ Stellen, lässt sich sowohl bei der linearen Liste als auch beim dynamischen Array mit einer Komplexität von $O(\#z)$ realisieren.

[16] Die zusätzlichen Kurzzahladditionen im Vergleich zur fortgeschrittenen Schulmethode entstehen dadurch, dass bei der Komposition der Zwischenresultate Langzahlen addiert werden müssen.

3.3.3.2 Karatsuba-Algorithmus

Macht man sich folgende Gleichheit zunutze,

$$\overleftarrow{a} \cdot \vec{b} + \vec{a} \cdot \overleftarrow{b} = \overleftarrow{a} \cdot \overleftarrow{b} + \vec{a} \cdot \vec{b} + (\overleftarrow{a} - \vec{a}) \cdot (\vec{b} - \overleftarrow{b})$$

erhält man durch Einsetzen in obige Bisektionsgleichung:

$$a \cdot b = \overleftarrow{a} \cdot \overleftarrow{b} \cdot B^{\#z} + (\overleftarrow{a} \cdot \overleftarrow{b} + \vec{a} \cdot \vec{b} + (\overleftarrow{a} - \vec{a}) \cdot (\vec{b} - \overleftarrow{b})) \cdot B^{\#z/2} + \vec{a} \cdot \vec{b}$$

Auf Basis dieser Gleichung lässt sich das Produkt zweier Zahlen, analog zur symmetrischen Komposition, ebenso rekursiv durch vollständige Bisektionierung berechnen.

Komplexitätsbetrachtung

Wie man sieht, tauchen in dieser Formel die Produkte $\overleftarrow{a} \cdot \overleftarrow{b}$ und $\vec{a} \cdot \vec{b}$ jeweils an zwei Stellen auf, die Produkte von $\overleftarrow{a} \cdot \vec{b}$ und $\vec{a} \cdot \overleftarrow{b}$ hingegen gar nicht mehr. Auch wenn nun noch ein neues Produkt in der Formel auftaucht, reduziert sich die Anzahl der effektiv notwendigen rekursiv durchzuführenden Multiplikation doch auf 3 pro Rekursionsschritt, da sich die einmal ermittelten Teilprodukte für $\overleftarrow{a} \cdot \overleftarrow{b}$ und $\vec{a} \cdot \vec{b}$ innerhalb einer Summierung mehrfach verwenden lassen. Dabei verändert sich die Rekursionstiefe gegenüber der symmetrischen Komposition nicht, es ergeben sich also effektiv weniger *Kurzzahlmultiplikationen*, nämlich insgesamt nur noch $3^{\log_2(\#z)} = \#z^{\log_2(3)}$.

Die Anzahl der *Kurzzahladditionen und -Subtraktionen* verdoppelt sich zwar pro Schritt gegenüber der symmetrischen Komposition, lässt sich aber wieder mit dem gleichen asymptotischen Wachstum abschätzen wie die Anzahl der Kurzzahlmultiplikationen (wenngleich mit einem größeren konstanten Faktor).

Somit ergibt sich für den Gesamtaufwand an Kurzzahloperationen die Komplexitätsklasse $a \cdot b \in O(\#z^{\log_2(3)}) \subset O(\#z^{1,6})$.

Der Karatsuba-Algorithmus ist also, dank der niedrigeren Rekursionsbreite, für große Operanden signifikant besser, als die fortgeschrittene Schulmethode. Allerdings macht sich der Vorteil der geringeren Anzahl an Kurzzahlmultiplikationen erst bei wirklich großen Zahlen bemerkbar (in der Praxis etwa im Bereich ab 100 Dezimalstellen), da die Anzahl der Kurzzahladditionen nur asymptotisch kleiner ist.

4 Resümee

4.1 Fazit

Einen der Grundbausteine eines leistungsfähigen Computeralgebrasystems bildet die Implementierung langer Zahlen und einer zugehörigen Langzahlarithmetik. Dies ermöglicht dem System exaktes Rechnen mit (theoretisch) beliebig langen Zahlen.

Verfügt man als Ausgangspunkt über die Implementierung einer Kurzzahlarithmetik mit konstanter Laufzeitkomplexität, so lassen sich mit den beschriebenen Algorithmen Langzahloperationen mit folgenden Komplexitätscharakteristiken realisieren.

- Vergleich $a \leq b \in O(\#z)$
- Addition/Subtraktion $a \pm b \in O(\#z)$
- Multiplikation
  - ⇨ Schulalgorithmus $a \cdot b \in O(\#z^2)$
  - ⇨ Karatsuba-Algorithmus $a \cdot b \in O(\#z^{\log_2(3)})$

4.2 Ausblick

Schnellere Multiplikation

Ausgehend von der rekursiven Halbierung der Operanden im Karatsuba-Algorithmus zur Multiplikation, lässt sich mit dem *Toom-Cook-Algorithmus* durch feinere Stückelung der Operanden in mehr als zwei gleichlange Teile, der Exponent der Komplexität beliebig nahe an 1 heranbringen $\rightarrow a \cdot b \in O(\#z^{1+\varepsilon})$ $\text{mit } \varepsilon > 0$.
Der konstante Faktor des Aufwands steigt dabei jedoch weiter an, so dass diese Herangehensweise eher von theoretischem Wert ist.

Den bislang komplexitätseffizientesten Algorithmus zur Multiplikation stellt der *Schönhage-Strassen-Algorithmus* dar. Er schafft es die schnelle Fourier-Transformation (FFT) zur Multiplikation ganzer Zahlen zu nutzen und erreicht $\rightarrow a \cdot b \in O(\#z \cdot \log(\#z) \cdot \log(\log(\#z)))$.

Weitere Operationen

Auf Grundlage der Multiplikation sind weitere mathematische Operationen definiert. Dazu gehören insbesondere

- Division
- Potenzierung, Radizierung und Logarithmierung
- Fakultät, …

Es stellt sich die Frage, wie effiziente Algorithmen für diese Operationen funktionieren könnten.

5 Anhang

A Quellenangaben

Der grundlegende thematische Aufbau dieser Ausarbeitung basiert auf den einschlägigen Kapiteln zum Thema Langzahlarithmetik der Computeralgebra-Bücher von Michael Kaplan [Kap2005] und Wolfram Koepf [Koe2006]. Die konkrete Strukturierung und Themenauswahl wurde mit einer Reihe eigener Überlegungen angereichert und in konsistente Form gebracht.
Einige zahlentheoretische Überlegungen basieren auf der Vorlesung Diskrete Mathematik des Wintersemesters 2005 von Professor Sebastian Iwanowski an der Fachhochschule Wedel [Iwa2005].

Literatur

[Iwa2005] Sebastian Iwanowski:
Diskrete Mathematik – Handouts zur Vorlesung –,
FH Wedel, Wedel 2005

[Kap2005] Michael Kaplan:
Computeralgebra, 1. Auflage,
Springer-Verlag, Berlin 2004, ISBN 3-540-21379-1

[Koe2006] Wolfram Koepf:
Computeralgebra – Eine algorithmisch orientierte Einführung, 1. Auflage,
Springer-Verlag, Berlin 2006, ISBN 3-540-29894-0