Computergestütztes Repetitorium der Elementarmathematik

Jacobs, Jürgen; Riebesehl, Dieter

University of Applied Sciences

Fachhochschule Nordostniedersachsen

Lüneburg
Buxtehude
Suderburg

Arbeitsberichte aus der Informatik

Incl. CD

FINAL

üneburg

rbeitsberichte

formatik

achhochschule Nordostniedersachsen

Computergestütztes Repetitorium der Elementarmathematik

Jürgen Jacobs, Dieter Riebesehl

12. Jahrgang, Heft 4, September 2002, ISSN 0939-8821
Technical Reports and Working Papers
Hrsg: Hinrich E. G. Bonin
Volgershall 1, D-21339 Lüneburg
Phone: xx49.4131.677175 Fax: xx49.4131.677140

Vorwort

Mit diesem Heft wird eine zweite Version des Repetitoriums vorgelegt, die die dringend benötigte graphische Benutzeroberfläche bietet und die es gestattet, Formeln im Formelsatz einzugeben. Die erste Version litt sehr darunter, dass die Eingabe nur zeilenweise im Stil von z.B. `FORTRAN` erfolgen konnte. Nicht nur war die Eingabe sehr langsam und fehleranfällig, auch die Ausgaben des Programms waren schwer lesbar.

Die Bedienung der neuen Version ist sehr viel flexibler. Durch die Verwendung von Notebooks kann der Benutzer gleichzeitig auf verschiedene Weise an eine gestellte Aufgabe herangehen: er kann parallel zur Prüfung seines Ergebnisses sich die Lösung vorführen lassen und eigene Zwischenschritte zur Kontrolle eingeben. Neben den vom Repetitorium gestellten Aufgaben können auch eigene Aufgabe eingegeben werden.

Der Aufbau dieses Begleitheftes zum Repetitorium ist gleichgeblieben. Kapitel 1-7 des vorliegenden Begleitheftes geben eine knappe Zusammenfassung der behandelten Themengebiete und stellen die zugehörigen Übungsaufgaben vor. Im Anhang wird die Bedienung der neuen Oberfläche erklärt. Teil B des Anhangs erläutert etwas detaillierter, wie das Repetitorium realisiert wurde. Dabei wird auch auf die Programmierung von Dialogen in Notebooks unter Mathematica eingegangen[1]. Allerdings setzt dieser Teil eine gewisse Vertrautheit auch mit tiefergehenden Eigenschaften von Mathematica voraus.

Insgesamt sind die Möglichkeiten des Repetitoriums ganz erheblich erweitert worden. Ich wünsche ihm eine gute Akzeptanz.

Lüneburg, im November 2001 D. Riebesehl

[1] Mathematica ® ist ein Produkt der Wolfram Research, Inc.

Vorwort zur 1. Version

Das computergestützte Repetitorium der Elementarmathematik wendet sich an Studierende, die ihre Kenntnisse in der Schulmathematik auffrischen und überprüfen wollen.

Kapitel 1-7 des vorliegenden Begleitheftes geben eine knappe Zusammenfassung der behandelten Themengebiete und stellen die zugehörigen Übungsaufgaben vor. Anhang A erklärt die Bedienung des Lernprogrammes. Interessierte finden darüber hinaus in Anhang B Bemerkungen zum Aufbau des Programms.

Die Entwicklung des Repetitoriums geschah in der Absicht, Schwächen handelsüblicher Lernprogramme zu überwinden. Die uns bekannten Lernprogramme

- behandeln lediglich eine kleine Zahl feststehender Aufgaben,
- erlauben nur eine starre Führung des Benutzers entlang eines vorgegebenen Lösungsweges,
- aktzeptieren nicht jede richtige, sondern nur vorgesehene Eingaben und
- geben keine Fehlerhinweise.

Aus Zeitgründen musste dagegen leider auf die Berücksichtigung lernpsychologischer Elemente und die Bereitstellung einer komfortablen Benutzerschnittstelle verzichtet werden. Wir hoffen, dass dies der Verwendung des Repetitoriums keinen Abbruch tut.

Lüneburg, im Januar 1996 J. Jacobs, D. Riebesehl

Inhaltsverzeichnis

1

Zahlbereiche

1.1 Die natürlichen Zahlen

Die Menge der natürlichen Zahlen wird mit dem Symbol $\mathbb{N}$ bezeichnet. Sie besteht aus den Zahlen 0, 1, 2, 3 usw. Man schreibt dafür

$$\mathbb{N} = \{0,\ 1,\ 2,\ 3,\ \ldots\}.$$

Die Addition, d.h. die Operation $+$, und die Multiplikation, d.h. die Operation $\cdot$, sind in $\mathbb{N}$ unbeschränkt durchführbar. Das Ergebnis der Addition oder Multiplikation beliebiger natürlicher Zahlen ist wieder eine natürliche Zahl.

Dies gilt jedoch nicht für die Subtraktion $(-)$ und die Division $(:)$.
Beispiele:

$5 - 2$ ist eine natürliche Zahl, nämlich 3, nicht aber $5 - 7$.
$8 : 2$ ist eine natürliche Zahl, nämlich 4, nicht aber $8 : 5$.

1.2 Die ganzen Zahlen

Die Menge der ganzen Zahlen wird mit dem Symbol $\mathbb{Z}$ bezeichnet. Sie besteht aus den natürlichen Zahlen und den negativen ganzen Zahlen -1, -2, -3 usw. Man schreibt dafür

$$\mathbb{Z} = \{\ldots,\ -3,\ -2,\ -1,\ 0,\ 1,\ 2,\ 3,\ \ldots\}.$$

Die natürlichen Zahlen außer 0 werden auch positive ganze Zahlen genannt, und man schreibt zuweilen $+1$ anstelle von 1, $+2$ anstelle 2 usw.

Innerhalb der ganzen Zahlen ist nun neben der Addition und der Multiplikation auch die Subtraktion unbeschränkt durchführbar. Z.B. kann nun die Differenz $5 - 7$ zu -2 berechnet werden. Die Division ist weiterhin nicht unbeschränkt durchführbar.

1.3 Die rationalen Zahlen

Die Menge der rationalen Zahlen wird mit dem Symbol $\mathbb{Q}$ bezeichnet. Sie besteht aus Brüchen $\frac{p}{q}$, wobei p eine ganze Zahl und q eine ganze Zahl außer 0 ist. Formal:

$$\mathbb{Q} = \left\{ \frac{p}{q} \mid p \in \mathbb{Z} \text{ und } q \in \mathbb{Z} \setminus \{0\} \right\}.$$

p heißt Zähler des Bruches $\frac{p}{q}$, und q heißt Nenner.

Wegen $\frac{p}{1} = p$ sind die ganzen Zahlen in $\mathbb{Q}$ enthalten. Innerhalb von $\mathbb{Q}$ ist nun jede der Rechenoperationen $+$, $-$, $\cdot$, $:$ unbeschränkt durchführbar. Z.B. kann $8 : 5$ zu der rationalen Zahl $\frac{8}{5}$ berechnet werden.

Rationale Zahlen können durch Dezimalbrüche dargestellt werden.
Beispiel:

$$\frac{8}{5} = 8 : 5 = 1.6$$

In manchen Fällen entsteht eine nicht abbrechende Dezimalzahl.
Beispiele:

$$\begin{aligned} \frac{209}{90} &= 209 : 90 = 2.3222\ldots \\ \frac{7}{11} &= 7 : 11 = 0.636363\ldots \end{aligned}$$

In den letzten beiden Beispielen wiederholt sich ab einer bestimmten Stelle nach dem Dezimalpunkt eine bestimmte Ziffer oder Ziffernkombination unendlich oft. Man nennt eine solche Dezimalzahl periodisch. Die sich wiederholende Ziffernkombination heißt Periode und wird durch Überstreichung gekennzeichnet, man schreibt

$$\begin{aligned} 2.3222\ldots &= 2.3\overline{2} \\ 0.636363\ldots &= 0.\overline{63} \end{aligned}$$

Es gilt allgemein: Eine rationale Zahl läßt sich als abbrechende Dezimalzahl oder als periodische, nicht abbrechende Dezimalzahl darstellen.

Obwohl alle Rechenoperationen innerhalb der rationalen Zahlen unbeschränkt durchführbar sind, läßt sich nicht jede Rechenaufgabe lösen. Z.B. gibt es keine rationale Zahl x, für die gilt $x \cdot x = 2$.

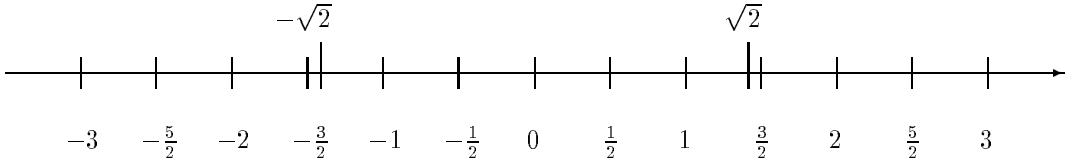
1.4 Die reellen Zahlen

Die Menge der reellen Zahlen wird mit dem Symbol $\mathbb{R}$ bezeichnet. Sie umfasst die rationalen Zahlen, sowie nicht abbrechende Dezimalzahlen, die nicht periodisch sind. Die nicht periodischen, nicht abbrechenden Dezimalzahlen heißen irrationale Zahlen.

Mit den reellen Zahlen läßt sich im Gegensatz zu den rationalen Zahlen die Aufgabe $x \cdot x = 2$ lösen. Genauer: Es gibt zwei Lösungen, die mit $\sqrt{2}$ (bzw. $2^{\frac{1}{2}}$) und $-\sqrt{2}$ (bzw. $-2^{\frac{1}{2}}$) bezeichnet werden.

Dagegen ist auch mit den reellen Zahlen die Aufgabe $x \cdot x = -1$ nicht lösbar. Eine entsprechende Zahlbereichserweiterung führt zu den komplexen Zahlen, die hier jedoch nicht besprochen werden sollen.

Die Menge der reellen Zahlen läßt sich alternativ als Menge aller Punkte auf der Zahlengeraden auffassen:



Zahlen, die durch Punkte rechts bzw. links vom Nullpunkt repräsentiert werden, heißen positiv bzw. negativ. Die reelle Zahl $-a$ entsteht durch Spiegelung der Zahl a am Nullpunkt und umgekehrt.

Ordnungsstruktur

Die reelle Zahl a heißt kleiner (bzw. größer) als die reelle Zahl b, falls a auf der Zahlengeraden links (bzw. rechts) von b angeordnet ist. Man schreibt:

$a < b$, falls a kleiner als b ist. Z.B. $-\sqrt{2} < -1$.
$a > b$, falls a größer als b ist. Z.B. $\frac{1}{2} > -2$.
$a \leq b$, falls $a < b$ oder $a = b$ ist.
$a \geq b$, falls $a > b$ oder $a = b$ ist.

Betrag

Der Betrag einer reellen Zahl a ist der Abstand des Punktes a vom Nullpunkt, d.h.

$$|a| = \begin{cases} a, & \text{falls } a \text{ positiv ist,} \\ 0, & \text{falls } a = 0 \text{ ist,} \\ -a, & \text{falls } a \text{ negativ ist.} \end{cases}$$

Praktisch erhält man den Betrag einer reellen Zahl, indem man ein eventuell vorhandenes negatives Vorzeichen wegläßt.
Beispiele:

$$\left|-\frac{1}{2}\right| = -\left(-\frac{1}{2}\right) = \frac{1}{2}$$
$$\left|\frac{1}{2}\right| = \frac{1}{2}$$

2

Grundrechenarten

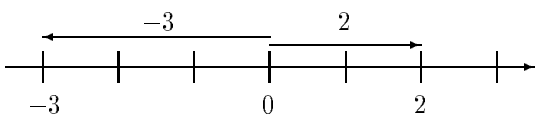
In dem Ausdruck $-7-3$ haben die auftretenden Symbole „$-$“ unterschiedliche Bedeutungen. Das erste Auftreten dient als Vorzeichen zur Kennzeichnung der negativen ganzen Zahl -7, und das zweite dient zur Kennzeichnung der Rechenoperation Subtraktion. In dem Ausdruck $-(3-5)$ schließlich dient das erste Auftreten von „$-$“ zur Kennzeichnung der Negation als Rechenoperation, die positive Zahlen in negative überführt und umgekehrt.

Zur deutlichen Unterscheidung zwischen Rechenzeichen und Vorzeichen innerhalb arithmetischer Ausdrücke werden vorerst alle reellen Zahlen – auch die positiven – mit Vorzeichen versehen und samt Vorzeichen in Klammern eingeschlossen. Wir schreiben also statt $-7-3$

$$\begin{array}{ccc} & \text{Rechenzeichen} & \\ & \downarrow & \\ (-7) & - & (+3) \\ \uparrow & & \uparrow \\ \multicolumn{3}{c}{\text{Vorzeichen}} \end{array}$$

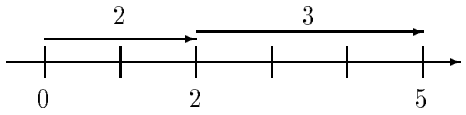
2.1 Addition und Multiplikation

Die Addition soll mit Hilfe der Zahlengeraden veranschaulicht werden. Eine reelle Zahl wird zu diesem Zweck abweichend von Abschnitt 1 nicht als ein bestimmter Punkt auf der Zahlengeraden, sondern als Pfeil vom Nullpunkt bis zu diesem Punkt dargestellt. Positive Zahlen werden also durch einen nach rechts gerichteten Pfeil, negative durch einen nach links gerichteten gekennzeichnet.
Beispiel:

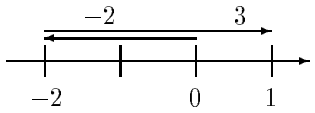


Zur Bestimmung von a+b wird der Anfang von dem zu b gehörigen Pfeil an die Spitze des zu a gehörigen Pfeils gesetzt. Die Spitze von b zeigt danach auf das Ergebnis.
Beispiele:

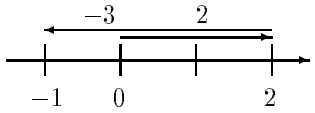
$(+2) + (+3) = +5$



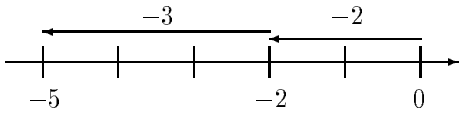
$(-2) + (+3) = +1$



$(+2) + (-3) = -1$



$(-2) + (-3) = -5$



Obige Ergebnisse sind Beispiele für folgende allgemeine Regeln, in denen a und b für positive ganze Zahlen stehen:

$$
\begin{array}{lllll}
\text{R1)} & (+a)+(+b) & = & +(a+b), & \\
\text{R2)} & (-a)+(+b) & = & -(a-b) & \text{für} \quad a \geq b, \\
 & & = & +(b-a) & \text{für} \quad a \leq b, \\
\text{R3)} & (+a)+(-b) & = & +(a-b) & \text{für} \quad a \geq b, \\
 & & = & -(b-a) & \text{für} \quad a \leq b, \\
\text{R4)} & (-a)+(-b) & = & -(a+b). &
\end{array}
$$

Mit Hilfe dieser Regeln kann die Addition von reellen Zahlen immer auf die Addition oder Subtraktion positiver reeller Zahlen mit positivem Ergebnis zurückgeführt werden.
Beispiel: Die Berechnung von $(+2) + (-3)$ erfolgt mit $a = 2$ und $b = 3$ nach Regel R3), wegen $b \geq a$ erhält man

$$(+2) + (-3) = -(b-a) = -(3-2) = -1$$

Die Subtraktion läßt sich auf die Addition zurückführen, wenn man beachtet, dass für jede reelle Zahl a gilt

$$-(-b) = +b \quad \text{und} \quad -(+b) = -b$$

Damit wird z.B.

$$
\begin{aligned}
(+a)-(+b) &= (+a)+(-b) = \begin{cases} +(a-b) & \text{für } a \geq b \\ -(b-a) & \text{für } a \leq b \end{cases} \\
(-a)-(-b) &= (-a)+(+b) = \begin{cases} -(a-b) & \text{für } a \geq b \\ +(b-a) & \text{für } a \leq b \end{cases} \\
&\text{usw.}
\end{aligned}
$$

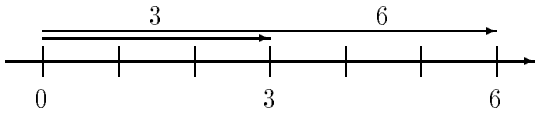
Beispiele:

$$
\begin{aligned}
(+2)-(+3) &= -(3-2) = -1 \\
(-2)-(+3) &= -(2+3) = -5 \\
(+2)-(-3) &= +(2+3) = +5 \\
(-2)-(-3) &= +(3-2) = +1
\end{aligned}
$$

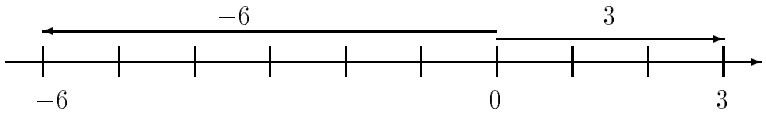
2.2 Multiplikation und Division

Die Multiplikation $a \cdot b$ kann durch eine Streckung des zu b gehörigen Pfeils auf das $|a|$-fache und anschließender Umkehrung des Pfeiles für negatives a veranschaulicht werden. Der Ergebnispfeil geht dabei immer vom Nullpunkt der Zahlengeraden aus.
Beispiele:

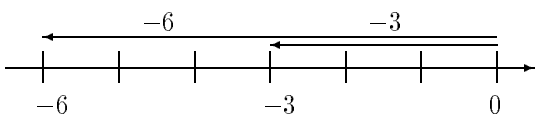
$$(+2) \cdot (+3) = +6$$



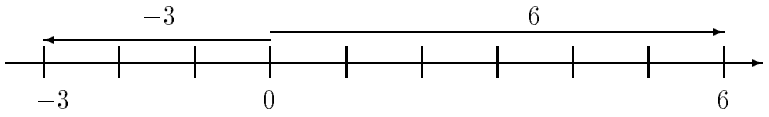
$$(-2) \cdot (+3) = -6$$



$$(+2) \cdot (-3) = -6$$



$$(-2)\cdot(-3) = +6$$



Obige Ergebnisse sind Beispiele für folgende Regeln:

$$\begin{array}{lrcl}
\text{R9)} & (+a)\cdot(+b) & = & +(a\cdot b)\\
\text{R10)} & (-a)\cdot(+b) & = & -(a\cdot b)\\
\text{R11)} & (+a)\cdot(-b) & = & -(a\cdot b)\\
\text{R12)} & (-a)\cdot(-b) & = & +(a\cdot b)
\end{array}$$

Mit Hilfe dieser Regeln läßt sich die Multiplikation beliebiger reeller Zahlen immer auf die Multiplikation positiver reeller Zahlen zurückführen.
Beispiel: Zur Berechnung von $(-2)\cdot(+3)$ berechnet man $2\cdot 3 = 6$ und fügt nach Regel R10) das Vorzeichen $-$ hinzu.

Für die Division existieren entsprechende Regeln. Man muss $b \neq 0$ voraussetzen, da die Division durch 0 nicht erlaubt ist:

$$\begin{array}{lrcl}
\text{R13)} & (+a):(+b) & = & +(a:b)\\
\text{R14)} & (-a):(+b) & = & -(a:b)\\
\text{R15)} & (+a):(-b) & = & -(a:b)\\
\text{R16)} & (-a):(-b) & = & +(a:b)
\end{array}$$

Beispiele:

$$\begin{array}{rcl}
(+3):(+2) & = & +(3:2) = 1.5\\
(-3):(+2) & = & -(3:2) = -1.5
\end{array}$$

Werden mehr als zwei Zahlen durch Rechenoperationen verknüpft, so wird die Ausführungsreihenfolge der einzelnen Rechenoperationen durch Klammerung gekennzeichnet.
Beispiel:

$$\underbrace{((+3)+\underbrace{((-2)-(-1))}_{a})}_{b}\cdot(-3)$$

Zunächst berechnet man

$$a = (-2)-(-1) = -1,$$

daraus erhält man

$$b = (+3)+\underbrace{(-1)}_{a} = +2$$

und schließlich das Endergebnis

$$\underbrace{(+2)}_{b} \cdot (-3) = -6$$

2.3 Übung

Führen Sie jetzt die Übung 2.3 zur Addition, Subtraktion, Multiplikation oder Division zweier ganzer Zahlen aus.
Beispiel:

`Bestimmen Sie` $(-8) : (-4)$.

Erwartetes Ergebnis: 2, *nicht* aber $\frac{4}{2}$

2.4 Übung

Führen Sie jetzt die Übung 2.4 zur kombinierten Addition, Subtraktion, Multiplikation oder Division mehrerer ganzer Zahlen aus.
Beispiel:

`Bestimmen Sie` $(-3) - (((+2) - (-2)) \cdot (-5))$.

Erwartetes Ergebnis: 23

3

Klammerrechnung

Arithmetische Ausdrücke, die Additionen, Subtraktionen und Multiplikationen enthalten, können mit Hilfe folgender Gesetze umgeformt werden[1]:

Seien $x, y, z \in \mathbb{R}$.

$R1)$	$x + y$	$=$	$y + x$	Kommutativgesetz der Addition
$R2)$	$x \cdot y$	$=$	$y \cdot x$	Kommutativgesetz der Multiplikation
$R3)$	$x + (y + z)$	$=$	$(x + y) + z$	Assoziativgesetz der Addition
$R4)$	$x \cdot (y \cdot z)$	$=$	$(x \cdot y) \cdot z$	Assoziativgesetz der Multiplikation
$R5)$	$x \cdot (y + z)$	$=$	$(x \cdot y) + (x \cdot z)$	Distributivgesetze
	$(x + y) \cdot z$	$=$	$(x \cdot z) + (y \cdot z)$	

Die Assoziativgesetze besagen, dass bei der Addition oder Multiplikation mehrerer reeller Zahlen die Reihenfolge, in der die Zahlen addiert oder multipliziert werden, keinen Einfluss auf das Ergebnis hat.
Beispiele:[2]

$$\begin{aligned} \text{a)} \quad (-3) + (1 + (-4)) &= (-3) + (-3) = -6 \\ ((-3) + 1) + (-4) &= (-2) + (-4) = -6 \\ \text{b)} \quad (-3) \cdot ((-2) \cdot 4) &= (-3) \cdot (-8) = 24 \\ ((-3) \cdot (-2)) \cdot 4 &= 6 \cdot 4 = 24 \end{aligned}$$

Daher wird die Klammerung bei der Addition und Multiplikation häufig nicht angegeben. Man schreibt also

$$(-3) + 1 + (-4)$$

[1] Die Division wird erst in Abschnitt 4 behandelt

[2] Ab nun wird das „+“-Vorzeichen bei positiven Zahlen weggelassen.

anstelle von sowohl

$$((-3)+1)+(-4) \quad \text{als auch} \quad (-3)+(1+(-4))$$

bzw.

$$(-3)\cdot(-2)\cdot 4$$

anstelle von sowohl

$$((-3)\cdot(-2))\cdot 4 \quad \text{als auch} \quad (-3)\cdot((-2)\cdot 4)$$

Achtung:

Es gilt *nicht* $x-(y-z)=(x-y)-z$ und auch *nicht* $x:(y:z)=(x:y):z$.
Beispiele:

$$\begin{aligned} 8-(4-2) &= 8-2=6 \\ (8-4)-2 &= 4-2=2 \end{aligned}$$

$$\begin{aligned} 8:(4:2) &= 8:2=4 \\ (8:4):2 &= 2:2=1 \end{aligned}$$

Um auch in diesen Fällen Klammern einsparen zu können, werden folgende Kurzschreibweisen vereinbart:

$$\begin{aligned} x-y-z &:= (x-y)-z, && \text{verschieden von } x-(y-z) \\ x+y-z &:= (x+y)-z, && \text{aber gleich mit } x+(y-z) \\ x-y+z &:= (x-y)+z, && \text{verschieden von } x-(y+z) \end{aligned}$$

Hier und entsprechend bei mehr als drei Operanden ist also die Klammerung der Rechenzeichen von links nach rechts durchzuführen, wenn explizit keine Klammern gesetzt sind:

$$a-b+c-d=(a-b)+c-d=((a-b)+c)-d$$

Mittels dieser Regeln spart man soweit als möglich Klammern ein. Üblicherweise werden positive Vorzeichen gar nicht geschrieben. Negative Vorzeichen werden nur dann mit Klammern geschrieben, wenn sonst Zeichenfolgen der Form „$+-$“, „$--$“ oder „$\cdot-$“ entstehen würden.
Beispiele[3]:

Niemals $1+-4$, sondern stets $1+(-4)$;
niemals $1\cdot(+4)$, sondern stets $1\cdot 4$;
niemals $1\cdot -4$, sondern stets $1\cdot(-4)$;

Durch die Konvention, dass die Operation „$\cdot$“ stärker „bindet“ als die Operationen „$+$“ und „$-$“ („*Punktrechnung geht vor Strichrechnung*“), lassen sich ebenfalls Klammern in arithmetischen Ausdrücken einsparen.
Beispiele:

$$\begin{aligned} 3\cdot 2+4\cdot 5 &:= (3\cdot 2)+(4\cdot 5), \\ -3\cdot 5+2\cdot 4+6\cdot(-2) &:= ((-3\cdot 5)+(2\cdot 4))+(6\cdot(-2)), \\ (-3\cdot 5+2\cdot 4)\cdot 6 &:= ((-3\cdot 5)+(2\cdot 4))\cdot 6 \end{aligned}$$

Mit Hilfe des Distributivgesetzes läßt sich der letzte Ausdruck der vorigen Beispiele in einen Ausdruck ohne Klammern umformen:

$$(-3 \cdot 5 + 2 \cdot 4) \cdot 6 = -3 \cdot 5 \cdot 6 + 2 \cdot 4 \cdot 6$$

Die Rechenzeichen „+“ und „·“ in den Distributivgesetzen können i.a. nicht durch andere ersetzt werden, d.h.

es gilt *nicht* $\quad x + (y \cdot z) = (x + y) \cdot (x + z)$

Beispiel:

$$\begin{aligned} 1 + (2 \cdot 3) &= 1 + 6 = 7 \\ \text{aber } (1 + 2) \cdot (1 + 3) &= 3 \cdot 4 = 12, \end{aligned}$$

es gilt *nicht* $\quad x + (y - z) = (x + y) - (x + z)$

Beispiel:

$$\begin{aligned} 1 + (2 - 3) &= 1 + (-1) = 0 \\ \text{aber } (1 + 2) - (1 + 3) &= 3 - 4 = -1, \end{aligned}$$

es gilt *nicht* $\quad x : (y + z) = (x : y) + (x : z)$

Beispiel:

$$\begin{aligned} 12 : (2 + 4) &= 12 : 6 = 2 \\ \text{aber } (12 : 2) + (12 : 4) &= 6 + 3 = 9. \end{aligned}$$

Weitere Regeln zur Elimination von Klammern sind

$$\begin{array}{lrcl} R6) & -(-a) & = & +a \\ R7) & -(+a) & = & -a \\ R8) & -(a + b) & = & -a - b \\ R9) & -(a - b) & = & -a + b \\ R10) & a + (-b) & = & a - b \\ R11) & a - (-b) & = & a + b \\ R12) & (-a) \cdot (-b) & = & a \cdot b \end{array}$$

Beispiele:

$$3 + (-(-(-1))) \overset{\text{R6}}{=} 3 + (-(+1)) \overset{\text{R7}}{=} 3 + (-1) \overset{\text{R10}}{=} 3 - 1 = 2,$$

$$\begin{aligned} a + (b - c) &\overset{\text{R10}}{=} a + (b + (-c)) \\ &\overset{\text{R3}}{=} (a + b) + (-c) \\ &\overset{\text{R10}}{=} (a + b) - c \\ &= a + b - c, \end{aligned}$$

$$\begin{aligned} a - (b - c) &\overset{\text{R10}}{=} a + (-(b - c)) \\ &\overset{\text{R9}}{=} a + (-b + c) \\ &\overset{\text{R3}}{=} (a + (-b)) + c \\ &\overset{\text{R10}}{=} (a - b) + c \\ &= a - b + c, \end{aligned}$$

$$-(a + 5 \cdot b - 3 \cdot c + d) = -a - 5 \cdot b + 3 \cdot c - d,$$

wobei das letzte Ergebnis vollständig geklammert so aussieht:

$$(((-a - 5 \cdot b) + 3 \cdot c) - d).$$

Als Anwendungsbeispiel sollen die binomischen Formeln hergeleitet werden[4]:

$$\begin{array}{lrcl} R13) & (a+b)^2 & = & a^2 + 2ab + b^2 \\ R14) & (a-b)^2 & = & a^2 - 2ab + b^2 \\ R15) & (a+b)(a-b) & = & a^2 - b^2 \end{array}$$

Bemerkung: Die Schreibweise x^2 bedeutet $x \cdot x$, ebenso $x^3 := x \cdot x \cdot x$ usw.

Beweis zu R12):

$$\begin{array}{rcl} (a-b)^2 & = & (a-b) \cdot (a-b) \\ & = & (a + (-b)) \cdot (a + (-b)) \\ & = & a \cdot a + a \cdot (-b) + (-b) \cdot a + (-b) \cdot (-b) \\ & = & a^2 - a \cdot b - b \cdot a + b \cdot b \\ & = & a^2 - 2ab + b^2 \end{array}$$

Beweis zu R13):

$$\begin{array}{rcl} (a+b)(a-b) & = & (a+b) \cdot (a + (-b)) \\ & = & a \cdot a + a \cdot (-b) + b \cdot a + b \cdot (-b) \\ & = & a^2 - a \cdot b + b \cdot a - b \cdot b \\ & = & a^2 - b^2 \end{array}$$

3.1 Übung

Führen Sie jetzt die Übung 3.1 zum Entfernen von Klammern aus arithmetischen Ausdrücken aus.
Beispiel:

`Entfernen Sie die Klammern:` $(x + 2)(2x - 9)^2 - ((3 - x) + y)(y - 1)$

Erwartetes Ergebnis: $4x^3 - 28x^2 + xy - y^2 + 8x - 2y + 165$, oder die gleichen Summanden in anderer Reihenfolge.

3.2 Übung

Führen Sie jetzt die Übung 3.2 zum Ausklammern von Faktoren aus arithmetischen Ausdrücken aus.
Beispiel:

`Klammern Sie Faktoren aus:` $4x^2 - 24x + 36$

Erwartetes Ergebnis: $4(x-3)^2$, oder $4(x^2-6x+9)$, nicht aber $2(2x^2-12x+18)$.

[4] Üblicherweise schreibt man das Rechenzeichen „·“ nur zwischen Zahlen, nicht zwischen Variablen oder Variablen und Zahlen

4

Bruchrechnung

Seien $a, b \in \mathbb{R}$ und $b \neq 0$. Dann wird mit $a : b$ und mit $\frac{a}{b}$ die Zahl bezeichnet, die mit b multipliziert a ergibt,

$$b \cdot \frac{a}{b} = b \cdot (a : b) = a.$$

In $\frac{a}{b}$ heißt a Zähler und b Nenner des Bruches. Für Zahlen mit Vorzeichen gelten folgende Regeln:

$$R1) \qquad \frac{+a}{+b} = \frac{-a}{-b} = \frac{a}{b}$$

$$R2) \qquad \frac{-a}{+b} = \frac{+a}{-b} = -\frac{a}{b}$$

Beispiele:

$$-\frac{3}{2-x} = \frac{-3}{2-x} = \frac{-3}{-(-2+x)} = \frac{3}{-2+x} = \frac{3}{x-2}$$

4.1 Addition und Subtraktion

Seien $a, b, c \in \mathbb{R}$, $c \neq 0$. Brüche mit gleichem Nenner heißen gleichnamig. Für sie gelten die Regeln

$$R3) \qquad \frac{a}{c} + \frac{b}{c} = \frac{a+b}{c}$$

$$R4) \qquad \frac{a}{c} - \frac{b}{c} = \frac{a-b}{c}$$

Beispiele:

$$\frac{5}{9} + \frac{-7}{9} = \frac{5+(-7)}{9} = \frac{5-7}{9} = \frac{-2}{9} = -\frac{2}{9},$$

oder

$$\frac{5}{9} + \frac{-7}{9} = \frac{5}{9} + (-\frac{7}{9}) = \frac{5}{9} - \frac{7}{9} = \frac{5-7}{9},$$

und weiter wie zuvor.

Nicht gleichnamige Brüche werden durch Erweitern gleichnamig gemacht:

$$R5) \qquad \frac{a}{b} = \frac{a \cdot k}{b \cdot k}$$

für jede von Null verschiedene Zahl $k \in \mathbb{R}$.
Beispiele:

$$-\frac{1}{4} + \frac{2}{6} = \frac{-1}{4} + \frac{2}{6} \overset{R5}{=} \frac{-1 \cdot 3}{4 \cdot 3} + \frac{1 \cdot 2}{6 \cdot 2} = \frac{-3}{12} + \frac{2}{12} = \frac{-3+2}{12} = -\frac{1}{12}$$

$$\begin{aligned} \frac{a}{b} - \frac{c+d}{-e} &= \frac{a}{b} + \frac{c+d}{e} \\ &= \frac{a \cdot e}{b \cdot e} + \frac{(c+d) \cdot b}{e \cdot b} \\ &= \frac{ae + (c+d)b}{be} \\ &= \frac{ae + cb + db}{be} \end{aligned}$$

Die Erweiterungsregel R5) kann auch „von hinten nach vorn“ gelesen werden. Man spricht dann von Kürzung.
Beispiele:

$$\frac{6}{8} = \frac{3 \cdot 2}{4 \cdot 2} = \frac{3}{4}$$

$$\frac{ab + bc}{bd} = \frac{b(a+c)}{b \cdot d} = \frac{a+c}{d}$$

Achtung:

Es gilt *nicht* $\frac{a+b}{c+d} = \frac{a}{c} + \frac{b}{d}$.
Beispiel:[1]

$$\begin{aligned} \frac{8+4}{4+2} &= \frac{12}{6} = 2, \text{ aber} \\ \frac{8}{4} + \frac{4}{2} &= 2 + 2 = 4. \end{aligned}$$

4.2 Multiplikation und Division

Seien $a, b, c, d \in \mathbb{R}$, $b \neq 0$, $d \neq 0$. Dann gilt

$$R3) \qquad \frac{a}{b} \cdot \frac{c}{d} = \frac{a \cdot c}{b \cdot d}$$

Ist zusätzlich $c \neq 0$, dann gilt

$$R4) \qquad \frac{\frac{a}{b}}{\frac{c}{d}} = \frac{a}{b} : \frac{c}{d} = \frac{a}{b} \cdot \frac{d}{c} = \frac{a \cdot d}{b \cdot c}$$

[1] Aber (kleiner Scherz): $\frac{9+(-4)}{3+2} = \frac{9}{3} + \frac{-4}{2} = 3 - 2 = 1.$

Beispiele:

$$-\frac{1}{3}\cdot\frac{2}{3} = -\left(\frac{1}{3}\cdot\frac{2}{3}\right) = -\frac{1\cdot 2}{3\cdot 3} = -\frac{2}{9}$$

$$8\cdot\frac{3}{4} = \frac{8}{1}\cdot\frac{3}{4} = \frac{24}{4} = 6$$

$$\frac{\frac{3}{2}}{\frac{1}{7}} = \frac{3\cdot 7}{2\cdot 1} = \frac{21}{2}$$

$$\frac{2}{\frac{1}{2}} = 2\cdot\frac{2}{1} = 2\cdot 2 = 4$$

4.3 Übung

Führen Sie jetzt die Übung 4.3 zum Umwandeln einer Summe oder Differenz von Brüchen in einen einzigen Bruch aus.
Beispiel:

`Schreiben Sie als einen Bruch:` $\frac{7}{2} - 3\left(\frac{3}{8} + \frac{2}{7}\right)$

Erwartetes Ergebnis: $\frac{85}{56}$.

4.4 Übung

Führen Sie jetzt die Übung 4.4 zum Umwandeln einer Summe oder Differenz von Brüchen in einen einzigen Bruch aus. Die Ausdrücke enthalten auch Variable. Das Ergebnis soll ohne Klammern geschrieben sein und soweit wie möglich gekürzt werden.
Beispiel:

`Schreiben Sie als einen Bruch:` $\frac{2x-y}{2x-2y} - \frac{x-y}{3x+3y} + \frac{y(3y-x)}{3x^2-3y^2}$

Erwartetes Ergebnis: $\frac{4x+y}{6x-6y}$ oder $\frac{4x^2+5xy+y^2}{6x^2-6y^2}$.

4.5 Übung

Führen Sie jetzt die Übung 4.5 zum Umwandeln eines Produkts oder Quotienten von Brüchen in einen einzigen Bruch aus. Das Ergebnis soll ohne Klammern geschrieben sein und soweit wie möglich gekürzt werden.
Beispiel:

`Schreiben Sie als einen Bruch:` $\dfrac{\dfrac{1}{x}+\dfrac{1}{y}}{\dfrac{1}{x}-\dfrac{1}{y}}$

Erwartetes Ergebnis: $\dfrac{x+y}{y-x}$.

5

Potenzrechnung

Seien $a, b, x, y \in \mathbb{R}$ und $a > 0,\ b > 0$. Dann gilt

$$R1) \qquad a^x \cdot a^y = a^{x+y}$$

$$R2) \qquad \frac{a^x}{a^y} = a^{x-y}$$

$$R3) \qquad (a^x)^y = (a^y)^x = a^{x \cdot y}$$

$$R4) \qquad a^x \cdot b^x = (a \cdot b)^x$$

$$R5) \qquad \frac{a^x}{b^x} = \left(\frac{a}{b}\right)^x$$

Beispiele:

$$2^3 \cdot 2^2 = 2^{3+2} = 2^5$$

$$2^{\frac{3}{2}} \cdot 2^{\frac{1}{2}} = 2^{\frac{3}{2}+\frac{1}{2}} = 2^2$$

$$\frac{2^3}{2^2} = 2^{3-2} = 2^1 = 2$$

$$\frac{2^{\frac{1}{2}}}{2^2} = 2^{\frac{1}{2}-2} = 2^{-\frac{3}{2}} = \frac{1}{2^{\frac{3}{2}}}$$

$$(2^3)^2 = 2^{3 \cdot 2} = 2^6$$

$$2^2 \cdot 3^2 = (2 \cdot 3)^2 = 6^2$$

$$\left(\frac{1}{2}\right)^2 \cdot 4^2 = \left(\frac{1}{2} \cdot 4\right)^2 = 2^2$$

$$\frac{2^2}{3^2} = \left(\frac{2}{3}\right)^2$$

$$\frac{\left(\frac{1}{2}\right)^2}{4^2} = \left(\frac{\frac{1}{2}}{4}\right)^2 = \left(\frac{1}{8}\right)^2$$

Bemerkungen:

1. In dem Ausdruck a^x heißt a Basis und x Exponent. a^x ist i.a. nur für eine positive Basis $a > 0$ definiert.

   Ausnahmen:

   (a) Ist der Exponent positiv ($x > 0$), so darf die Basis Null sein. Es gilt

   $$0^x = 0 \text{ für } x > 0$$

   (b) Ist der Exponent eine ganze Zahl, so darf die Basis eine beliebige reelle Zahl außer Null sein. Es gilt für $a \neq 0$:

   $$\begin{aligned} a^0 &= 1 \\ a^1 &= 1 \\ a^n &= \underbrace{a \cdot a \cdot \ldots \cdot a}_{n \text{ Faktoren}} \text{ für } n \in \mathbb{N},\ n > 1 \\ a^{-n} &= \frac{1}{a^n} \text{ für } n \in \mathbb{N} \end{aligned}$$

   Beispiele: (*Man beachte die Wirkung der Klammern!*)

   $$\begin{aligned} (-2)^3 &= -8 \\ (-2)^4 &= 16 \\ -2^4 &= -16 \\ (-2)^{-2} &= \frac{1}{(-2)^2} = \frac{1}{4} \\ -2^{-2} &= -\frac{1}{2^2} = -\frac{1}{4} \end{aligned}$$

   (c) Ist der Exponent von der Form $\frac{1}{n}$ mit $n \in \mathbb{N}$ und ungerade, so darf die Basis eine beliebige reelle Zahl außer Null sein. Es gilt für $a < 0$ und ungerades $n \in \mathbb{N}$

   $$a^{\frac{1}{n}} = -(|a|^{\frac{1}{n}}).$$

   Beispiel:

   $$(-8)^{\frac{1}{3}} = -(8^{\frac{1}{3}}) = -2$$

   Achtung: Die Potenzgesetze gelten u.U. nicht für negative Basen und gebrochene Exponenten!
   Beispiel:

   $$-1 = (-1)^3 = (-1)^{\frac{6}{2}} \overset{R3?}{\neq} \left((-1)^6\right)^{\frac{1}{2}} = 1^{\frac{1}{2}} = 1$$

2. Die Potenz $a^{\frac{1}{n}}$ für $a \in \mathbb{R},\ a > 0$, und $n \in \mathbb{N} \setminus \{0\}$ ist diejenige *positive* Zahl, welche zur n-ten Potenz erhoben die Zahl a ergibt.
   Beispiel:

   $$4^{\frac{1}{2}} = 2 \text{ wegen } 2^2 = 4$$

Am Beispiel erkennt man, dass es u.U. auch negative Zahlen gibt, welche zur n-ten Potenz erhoben wieder a liefern:

$$(-2)^2 = 2^2 = 4$$

Man schreibt anstelle von $a^{\frac{1}{n}}$ für $n > 2$ auch $\sqrt[n]{a}$, und statt $a^{\frac{1}{2}}$ einfach $\sqrt{a}$.

Potenzen der Gestalt $a^{\frac{m}{n}}$ werden durch Anwendung der Regel R3) berechnet:

$$8^{\frac{2}{3}} = 8^{\frac{1}{3}\cdot 2} = (8^{\frac{1}{3}})^2 = \left(\sqrt[3]{8}\right)^2 = 2^2 = 4$$

3. Es gilt nicht

$$a^{(b^c)} = \left(a^b\right)^c .$$

Beispiel:[1]

$$\begin{aligned} 2^{3^2} &= 2^9 = 512 \\ \left(2^3\right)^2 &= 8^2 = 64 \end{aligned}$$

5.1 Übung

Führen Sie jetzt die Übung 5.1 zum Berechnen von Ausdrücken mit Potenzen aus.
Beispiel:

`Berechnen Sie:` $\left(\frac{4}{5}\right)^{-5} : \left(\frac{4}{5}\right)^{-3}$

Erwartetes Ergebnis: $\frac{25}{16}$, nicht aber $\left(\frac{5}{4}\right)^2$

Das Programm erwartet die Eingabe `25/16`.

5.2 Übung

Führen Sie jetzt die Übung 5.2 zum Zerlegen eines Ausdrucks mit Potenzen in Faktoren aus.
Beispiel:

`Berechnen Sie:` $a^7 - 4a^6 + 4a^5$

Erwartetes Ergebnis: $a^5(a-2)^2$ oder $a^5(a^2 - 4a + 4)$.

[1] Anstelle von $a^{(b^c)}$ schreibt man oft a^{b^c}.

5.3 Übung

Führen Sie jetzt die Übung 5.3 zum Kürzen eines Bruchs mit Potenzen aus.
Beispiel:

`Berechnen Sie:` $\dfrac{z^{q-1} - 4z^{q-2}}{z^{q+1} - 8z^q + 16z^{q-1}}$

Erwartetes Ergebnis: $\dfrac{z-4}{z^3 - 8z^2 + 16z}$ oder $\dfrac{1}{z(z-4)}$.

5.4 Übung

Führen Sie jetzt die Übung 5.4 zum Zusammenfassen und Kürzen mehrerer Brüche mit Potenzen aus.
Beispiel:

`Berechnen Sie:`

$$\frac{2x^3 - x^2 + 2}{x^{n+1}} - \frac{x^5 - x}{x^{n+3}} + \frac{2-x}{x^{n-1}}$$

Erwartetes Ergebnis: $\dfrac{x^3 + 2x + 1}{x^{n+2}}$, nicht aber $\dfrac{x^4 + 2x^2 + x}{x^{n+3}}$.

6

Gleichungen

Im folgenden werden nur Gleichungen mit einer Lösungsvariablen x betrachtet.
Beispiele:

1. Die Gleichung
$$x + 3 = 4$$
hat die eindeutige Lösung $x = 1$.

2. Die Gleichung
$$x^2 + 1 = 10$$
hat die Lösungen $x = 1$ und $x = 3$.

3. Die Gleichung
$$\frac{2}{x} = 0$$
besitzt keine Lösung.

6.1 Allgemeines Lösungsverfahren

Gleichungen werden i.a. gelöst, indem man die linke und die rechte Seite so umformt, dass eine neue Gleichung mit denselben Lösungen entsteht. Man führt mehrere Umformungen so durch, dass am Schluss die Lösungsvariable nur auf einer Seite der Gleichung auftritt.
Beispiel:

$$\begin{array}{rcll} x - 7 & = & 2x + 3 & | \quad +7 \\ x & = & 2x + 10 & | \quad -2x \\ -x & = & 10 & | \quad \cdot(-1) \\ x & = & 10 & \end{array}$$

Es gelten folgende Regeln für die Umformung von Gleichungen:

Die Lösungsmenge einer Gleichung ändert sich nicht, wenn

R1) auf beiden Seiten der Gleichung dieselbe Zahl oder derselbe arithmetische Ausdruck addiert oder subtrahiert wird,

R2) beide Seiten der Gleichung mit derselben Zahl oder demselben arithmetischen Ausdruck multipliziert oder dadurch dividiert werden, sofern diese Zahl/dieser Ausdruck nicht Null ist.

Achtung: Die Missachtung der Bedingung in R2) kann unauffällig sein:

Beispiele:

1.

$$\begin{array}{rcll} x(x-1) & = & 2(x-1) & \mid \ :(x-1) \\ x & = & 2 & \end{array}$$

Es ist aber auch $x = 1$ eine Lösung der Gleichung. Sie ging deshalb verloren, weil für $x = 1$ die Umformung nicht mehr erlaubt ist, weshalb die entstandene Gleichung nicht mehr die gleiche Lösungsmenge haben muss wie die ursprüngliche.

2.

$$\begin{array}{rcll} \dfrac{1}{x+3} + \dfrac{1}{x-3} & = & \dfrac{6}{x^2-9} & \mid \ \cdot(x+3) \\ 1 + \dfrac{x+3}{x-3} & = & \dfrac{6}{x-3} & \mid \ \cdot(x-3) \\ (x-3)+(x+3) & = & 6 & \\ 2x & = & 6 & \mid \ :2 \\ x & = & 3 & \end{array}$$

Hier wurde bei der Umformung nicht dividiert, dennoch ist $x = 3$ keine Lösung der Gleichung, weil $\frac{1}{x-3}$ für $x = 3$ nicht definiert ist.

In Beispiel 2 muss durch Einsetzen geprüft werden, ob $x = 3$ eine Lösung ist. In Beispiel 1 muss die Lösung der Gleichung $x - 1 = 0$ der Lösungsmenge eventuell hinzugefügt werden. Dies ist durch Einsetzen zu überprüfen. Es gilt generell:

R3) Wird eine Gleichung durch Division oder durch Multiplikation mit einem arithmetischen Ausdruck A umgeformt, welcher den Wert Null annehmen kann, so sind

1. die Lösungen der Gleichung $A = 0$ der Lösungsmenge hinzuzufügen und
2. alle Lösungen durch Einsetzen zu überprüfen.

Weitere Regeln zur Umformung:

R4) Für ungerades $n \in \mathbb{N}$ ändert sich die Lösungsmenge einer Gleichung nicht, wenn beide Seiten mit n oder $\frac{1}{n}$ potenziert werden.

R5) Für gerades $n \in \mathbb{N}$ vergrößert sich i.a. die Lösungsmenge einer Gleichung, wenn beide Seiten mit n potenziert werden. Alle gefundenen Lösungen sind durch Einsetzen zu überprüfen.

R6) Für gerades $n \in \mathbb{N}$ verkleinert sich i.a. die Lösungsmenge einer Gleichung (*linke Seite*)=(*rechte Seite*), wenn beide Seiten mit $\frac{1}{n}$ potenziert werden. Die evtl. verlorengegangenen Lösungen sind aber stets Lösungen der Gleichung

$$(\textit{linke Seite})^{\frac{1}{n}} = -(\textit{rechte Seite})^{\frac{1}{n}}$$

Beispiele:

1.

$$\begin{array}{rcll} (x+1)^{\frac{1}{3}} & = & -2 & \mid \ (\,)^3 \\ x+1 & = & -8 & \mid \ -1 \\ x & = & -9 & \end{array}$$

2.

$$\begin{array}{rcll} (x+1)^3 & = & 8 & \mid \ (\,)^{\frac{1}{3}} \\ x+1 & = & 2 & \mid \ -1 \\ x & = & 1 & \end{array}$$

3.

$$\begin{array}{rcll} (x+1)^2 & = & 4 & \mid \ (\,)^{\frac{1}{2}} \\ x+1 & = & 2 & \mid \ -1 \\ x & = & 1 & \end{array}$$

Zusätzlich ist wegen des Potenzierens mit $\frac{1}{2}$ auch die Gleichung

$$x + 1 = -2$$

zu betrachten. Man erhält die Lösungen $x = 1$ und $x = -3$.

4.

$$\begin{array}{rcll} (x^2+1)^2 & = & 4 & \mid \ (\,)^{\frac{1}{2}} \\ x^2+1 & = & 2 & \mid \ -1 \\ x^2 & = & 1 & \mid \ (\,)^{\frac{1}{2}} \\ x & = & 1 & \end{array}$$

Wegen des Potenzierens mit $\frac{1}{2}$ ist auch die Gleichung

$$x = -1$$

zu betrachten.

Zusätzlich ist die Gleichung

$$x^2 + 1 = -2$$

zu betrachten. Diese hat aber keine Lösung, da stets $x^2 \geq 0$ ist. Man erhält die Lösungen $x = 1$ und $x = -1$

5.

$$\begin{array}{rcl|r} x+1 & = & \sqrt{2x+10} & | \quad (\,)^2 \\ (x+1)^2 & = & 2x+10 & | \quad \text{Ausmultiplizieren} \\ x^2+2x+1 & = & 2x+10 & | \quad -2x \\ x^2+1 & = & 10 & | \quad -1 \\ x^2 & = & 9 & | \quad (\,)^{\frac{1}{2}} \\ x & = & 3 & \end{array}$$

Zusätzlich ist wegen des Potenzierens mit $\frac{1}{2}$ auch die Gleichung

$$x = -3$$

zu betrachten. Da vorher mit 2 potenziert wurde, sind die beiden Werte $x = 3$ und $x = -3$ durch Einsetzen zu prüfen:

$$\begin{array}{rcl} (3+1) & = & \sqrt{2 \cdot 3 + 10} \\ (-3+1) & \neq & \sqrt{2 \cdot (-3) + 10} \\ & & \text{wegen } \sqrt{2 \cdot (-3) + 10} = \sqrt{4} = 2. \end{array}$$

6.

$$\begin{array}{rcl|r} x+1 & = & \sqrt{2x+2} & | \quad (\,)^2 \\ (x+1)^2 & = & 2x+2 & \\ x^2+2x+1 & = & 2x+2 & | \quad -2x \\ x^2+1 & = & 2 & | \quad -1 \\ x^2 & = & 1 & | \quad (\,)^{\frac{1}{2}} \\ x & = & 1 & \end{array}$$

Zusätzlich ist wegen des Potenzierens mit $\frac{1}{2}$ auch die Gleichung

$$x = -1$$

zu betrachten. Da voher mit 2 potenziert wurde, sind die beiden Werte $x = 1$ und $x = -1$ durch Einsetzen zu prüfen:

$$\begin{array}{rcl} (1+1) & = & \sqrt{2 \cdot 1 + 2} \\ (-1+1) & = & \sqrt{2 \cdot (-1) + 2} \end{array}$$

6.2 Lineare und quadratische Gleichungen

Wir wollen im folgenden Gleichungen betrachten, die sich durch Umformungen auf folgende Normalformen bringen lassen:

$$\begin{array}{rcll} ax+b & = & 0 \text{ mit } a, b \in \mathbb{R} & \text{(lineare Gleichung)} \\ x^2+px+q & = & 0 \text{ mit } p, q \in \mathbb{R} & \text{(quadratische Gleichung)} \end{array}$$

Die lineare Gleichung hat

- die eindeutige Lösung $x = -\dfrac{b}{a}$ für $a \neq 0$,
- keine Lösung für $a = 0$ und $b \neq 0$,
- alle reellen Zahlen als Lösung für $a = b = 0$:

	$b \neq 0$	$b = 0$
$a = 0$	keine Lös.	alle $x \in \mathbb{R}$
$a \neq 0$	$x = -\frac{b}{a}$	

Die quadratische Gleichung hat

- die Lösungen

$$x = \frac{1}{2}\left(-p + \sqrt{p^2 - 4q}\right) \text{ und } x = \frac{1}{2}\left(-p - \sqrt{p^2 - 4q}\right),$$

  falls $p^2 > 4q$ ist,
- die Lösung $x = -\dfrac{p}{2}$ für $p^2 = 4q$,
- keine Lösung für $p^2 < 4q$.

Beispiele:

1.

$$\begin{array}{rcll} (x-4)(2x-9) & = & (x-6)^2 & \mid \text{ Klammern auflösen} \\ 2x^2 - 17x + 36 & = & x^2 - 12x + 36 & \mid -36 \\ 2x^2 - 17x & = & x^2 - 12x & \mid +12x \\ 2x^2 - 5x & = & x^2 & \mid -x^2 \\ x^2 - 5x & = & 0 & \mid \text{ quadratische Gleichung mit } p = -5, q = 0 \end{array}$$

Lösungen:

$$x = \frac{1}{2}\left(-(-5) + \sqrt{25}\right) = \frac{5+5}{2} = 5$$

und

$$x = \frac{1}{2}\left(-(-5) - \sqrt{25}\right) = \frac{5-5}{2} = 0.$$

2.

$$\begin{array}{rcll} \dfrac{7x}{10} - \dfrac{2}{5} & = & \dfrac{x}{2} & \mid -\dfrac{x}{2} \\ \dfrac{7x}{10} - \dfrac{x}{2} - \dfrac{2}{5} & = & 0 & \mid \cdot 10 \\ 7x - 5x - 4 & = & 0 & \\ 2x - 4 & = & 0 & \mid \text{ lineare Gleichung mit } a = 2, b = -4 \end{array}$$

Lösung: $x = 2$.

3.

$$\begin{array}{rcll} 3+\sqrt{2x-3} & = & x & | \quad -3 \\ \sqrt{2x-3} & = & x-3 & | \quad (\,)^2 \\ 2x-3 & = & x^2-6x+9 & | \quad -2x \\ -3 & = & x^2-8x+9 & | \quad +3 \\ 0 & = & x^2-8x+12 & | \quad p=-8, q=12 \end{array}$$

mögliche Lösungen:

$$x = \frac{1}{2}\left(-(-8)+\sqrt{64-48}\right) = \frac{8+4}{2} = 6$$

und

$$x = \frac{8-4}{2} = 2.$$

Es ist eine Probe erforderlich:

$$\begin{array}{lcll} x=6 & \longrightarrow & 3+\sqrt{2\cdot 6-3} & = \quad 6, \\ x=2 & \longrightarrow & 3+\sqrt{2\cdot 2-3} & \neq \quad 2. \end{array}$$

Nur $x = 6$ ist Lösung.

6.3 Übung

Führen Sie jetzt die Übung 6.3 aus, in der Sie lineare Gleichungen lösen sollen.
Beispiel:

`Lösen Sie:` $3(5x-4)+2 = 4(3x-2)-6x$

Erwartetes Ergebnis: $x = 2$.

Das Programm erwartet die Eingabe `{2}`.

6.4 Übung

Führen Sie jetzt die Übung 6.4 aus, in der Sie quadratische Gleichungen lösen sollen.
Beispiel:

`Lösen Sie:` $4x^2+7 = 3(4x+1)$

Erwartetes Ergebnis: $x \in \left\{\frac{3+\sqrt{5}}{2}, \frac{3-\sqrt{5}}{2}\right\}$.

Das Programm erwartet die Eingabe $\left\{\frac{3+\sqrt{5}}{2}, \frac{3-\sqrt{5}}{2}\right\}$.

6.5 Übung

Führen Sie jetzt die Übung 6.5 aus, in der Sie Gleichungen mit Variablen im Nenner lösen sollen.
Beispiel:

`Lösen Sie:`

$$\frac{9x+1}{8x-24}+2+\frac{x+5}{x-3}=\frac{9x-7}{2x-6}$$

Erwartetes Ergebnis: $x=7$.

Das Programm erwartet die Eingabe `{7}`.

6.6 Übung

Führen Sie jetzt die Übung 6.6 aus, in der Sie Gleichungen mit Wurzeln lösen sollen.
Beispiel:

`Lösen Sie:` $\sqrt{5+x}-\sqrt{5-x}=2$

Erwartetes Ergebnis: $x=4$.

Das Programm erwartet die Eingabe `{4}`.

7

Rechnen mit Beträgen

Der Betrag einer reellen Zahl a ist durch $|a| = \begin{cases} a & \text{falls } a \geq 0 \\ -a & \text{falls } a < 0 \end{cases}$ definiert.

Beispiele:

$$\begin{aligned} \left|-\frac{1}{2}\right| &= -\left(-\frac{1}{2}\right) = \frac{1}{2} \\ |2.5| &= 2.5 \\ |0| &= 0 \end{aligned}$$

Es gelten folgende Regeln:

$R1)$ $\quad |a| = |-a|$

$R2)$ $\quad |a| \geq 0$

$R3)$ $\quad a \leq |a| \quad \text{und} \quad -a \leq |a|$

$R4)$ $\quad |x| \leq a$ für $a > 0$ genau dann, wenn $-a \leq x \leq a$

$R5)$ $\quad |a \cdot b| = |a| \cdot |b|$

$R6)$ $\quad |a + b| \leq |a| + |b|$

Beispiele:

$$\begin{aligned} |-2 \cdot 3| &= |-6| = 6, \\ &= |-2| \cdot |3| = 2 \cdot 3 = 6; \\ |1-2| &= |-1| = 1, \\ &= |1 + (-2)| \leq |1| + |-2| = 1 + 2 = 3; \\ |-1-2| &= |-3| = 3, \\ &= |(-1) + (-2)| \leq |-1| + |-2| = 1 + 2 = 3. \end{aligned}$$

Gleichungen mit Beträgen

Treten in einer Bestimmungsgleichung für eine Unbekannte x Beträge auf, so müssen Fallunterscheidungen getroffen werden. Dabei führt jeder Teilterm, der von Betragsstrichen umschlossen ist, zu zwei zu unterscheidenden Fällen, die vom Vorzeichen dieses Terms abhängen. Bei einem Betragsterm sind deshalb 2 Fälle, bei zwei Betragstermen 4 Fälle, bei drei Betragstermen 8 Fälle, allgemein bei n Betragstermen 2^n Fälle zu unterscheiden.

Beispiel 1:

$$|x+6| = 3+2x$$

Diese Gleichung enthält den Betragsterm $|x+6|$. Man erhält die beiden Fälle

(I.) $x+6 \geq 0$:

Dieser Fall tritt genau dann ein, wenn die Bedingung $x \geq -6$ erfüllt ist.

Jetzt ist $|x+6| = x+6$ und die Gleichung wird zu

$$x+6 = 3+2x$$

mit der Lösungsmenge $x \in \{3\}$. Wegen $3 \geq -6$ ist die Bedingung für diesen Fall erfüllt und 3 ist eine Lösung der Gleichung.

(II.) $x+6 < 0$:

Dieser Fall tritt genau dann ein, wenn die Bedingung $x < -6$ erfüllt ist.

Jetzt ist $|x+6| = -x-6$ und die Gleichung wird zu

$$-x-6 = 3+2x$$

mit der Lösungsmenge $x \in \{-3\}$. Diesmal ist die Bedingung $-3 < -6$ für diesen Fall *nicht* erfüllt und -3 ist *keine* Lösung der Gleichung.

Insgesamt ist die Lösungsmenge der Gleichung $|x+6| = 3+2x$ gegeben durch $\{2\}$.

Bemerkung: Man kann auch statt des Falles $x+6 \geq 0$ den Fall $x+6 > 0$ analog zu (II.) behandeln. Man erhält $x = 3$ als Lösung. Dann bleibt noch der Sonderfall $x+6 = 0$ zu untersuchen. Die Bedingung lautet nun $x = -6$, und die Gleichung wird zu $0 = 3 + 2 \cdot (-6)$. Diese Gleichung ist nicht erfüllt, $x = -6$ ist keine Lösung.[1]

Beispiel 2:

$$3 + |x| = x + 2 + |x-1|$$

Diese Gleichung enthält die beiden Betragsterme $|x|$ und $|x-1|$, so dass 4 Fälle zu unterscheiden sind:

[1] Diese Vorgehensweise wurde im Repetitorum gewählt, weil dann einheitlich nur Ungleichungen mit $<$ oder $>$ vorkommen.

	$x \geq 0$	
	$x-1 \geq 0$	$x-1 < 0$
Fall	(I.)	(II.)
Bedingung an x	$x \geq 1$	$0 \leq x < 1$
Gleichung ohne $\lvert\ \rvert$	$3+x = x+2+x-1$	$3+x = x+2-x+1$
vereinfacht	$3+x = 2x+1$	$3+x = 3$
Lösungsmenge	$\{2\}$	$\{0\}$
Bedingung prüfen	ja	ja

	$x < 0$	
	$x-1 \geq 0$	$x-1 < 0$
Fall	(III.)	(IV.)
Bedingung an x	unerfüllbar	$x < 0$
Gleichung ohne $\lvert\ \rvert$	entf.	$3-x = x+2-x+1$
vereinfacht		$3-x = 3$
Lösungsmenge		$\{0\}$
Bedingung prüfen		nein

Damit ergibt sich $\{0, 2\}$ als Lösungsmenge.

Ungleichungen

Ungleichungen als Bestimmungsgleichungen für eine Unbekannte x können auf Gleichungen zurückgeführt werden. Dies geschieht in drei Schritten:

1. Die Ungleichung wird so umgeformt, dass auf der rechten Seite Null steht. Dazu ist nur eine Subtraktion der gesamten rechten Seite von beiden Seiten der Gleichung nötig.
2. Das Ungleichheitszeichen wird durch ein Gleichheitszeichen ersetzt und die entstandene Gleichung gelöst.
3. Die Lösungen der Gleichung zerlegen die reelle Achse in mehrere Intervalle, von denen das linkeste und das rechteste jeweils unendlich lang ist. Für jedes Intervall wird durch Einsetzen eines Testwertes aus der ungefähren Mitte des Intervalls und der beiden Intervallgrenzen (bei unendlich langen Intervallen nur der einen Intervallgrenze) geprüft, ob die ursprüngliche Ungleichung erfüllt ist oder nicht. Der Test liefert als Ergebnis jene Intervalle, auf denen die Ungleichung erfüllt ist. Das können ausnahmsweise auch einzelne Punkte sein.

Beispiel 1:

$$x^2 > x + 2$$

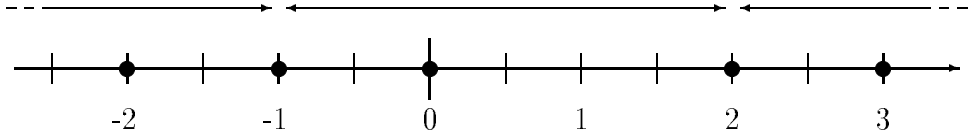
Die drei Schritte werden so durchgeführt:

1.

$$\begin{aligned} x^2 &> x+2 && \mid\ -(x+2) \\ x^2 - x - 2 &> 0 \end{aligned}$$

2. Es ist die Gleichung
$$x^2 - x - 2 = 0$$
zu lösen. Die Lösungsmenge lautet $\{-1, 2\}$.

3. Die beiden Lösungen zerlegen die reelle Achse in die drei Intervalle $\{-\infty < x \leq -1\}$, $\{-1 \leq x \leq 2\}$ und $\{2 \leq x < -\infty\}$. Als Testwerte zum Einsetzen in $x^2 > x + 2$ werden gewählt $x = -2, -1, 0, 2$ und 3. Dabei können statt -2, 0 und 3 auch andere Werte aus dem Inneren der Intervalle gewählt werden.



Der Test zeigt, dass -2 und 3 die einzigen Werte sind, die die Ungleichung erfüllen. Damit sind die Lösungsintervalle gegeben durch
$$\{-\infty < x < -1, \quad 2 < x < \infty\}.$$

Beispiel 2:
$$x^2 \leq 2x - 1$$

1.
$$\begin{array}{rcll} x^2 & \leq & 2x - 1 & | \quad -(2x-1) \\ x^2 - 2x + 1 & \leq & 0 & \end{array}$$

2. $x^2 - 2x + 1 = 0$ hat als einzige Lösung $x = 1$.

3. Die Intervalle lauten $\{-\infty < x \leq 1\}$ und $\{1 \leq x < \infty\}$, Testwerte sind (z.B.) 0, 1 und 2. Von diesen erfüllt nur $x = 1$ die Ungleichung $x^2 \leq 2x - 1$, er ist Randpunkt der beiden möglichen Intervalle. Daher lautet die Lösungsmenge $\{1\}$.

Ungleichungen mit Beträgen

Enthält eine Bestimmungsgleichung neben Ungleichungen auch Beträge, so ist das vorstehend Gesagte zu kombinieren. Dazu ein

Beispiel:
$$x^2 > |x - 1| + 1$$

Zu unterscheiden sind die Fälle (I.) $x - 1 \geq 0$ und (II.) $x < 0$.

(I.) Unter der Bedingung $x \geq 1$ lautet die Ungleichung
$$x^2 > x - 1 + 1 = x$$
oder
$$x^2 - x > 0.$$

Die zugehörige Gleichung $x^2 - x = 0$ hat die Lösungen $\{0, 1\}$, die Intervalle sind $\{-\infty < x \leq 0\}$, $\{0 \leq x \leq 1\}$ und $\{1 \leq x < \infty\}$, die Testwerte -1, 0, $\frac{1}{2}$, 1 und 2 zeigen, dass die Intervalle $\{-\infty < x < 0\}$ und $\{1 < x < \infty\}$ die Ungleichung lösen. Es muss aber noch die Bedingung $x \geq 1$ erfüllt sein, und daher ist

$$\{1 < x < \infty\}$$

die erste Teillösung.

Bemerkung: Der Sonderfall $x - 1 = 0$ kann wieder extra behandelt werden. Unter der Bedingung $x = 1$ lautet die Ungleichung $1 < 1$. Sie ist nicht erfüllt, $x = 1$ ist also keine Lösung.

(II.) Unter der Bedingung $x < 1$ lautet die Ungleichung

$$x^2 > -x + 2$$

oder

$$x^2 + x - 2 > 0.$$

Die zugehörige Gleichung $x^2 + x - 2 = 0$ hat die Lösungen $\{-2, 1\}$, die Intervalle sind $\{-\infty < x \leq -2\}$, $\{-2 \leq x \leq 1\}$ und $\{1 \leq x < \infty\}$, die Testwerte -3, -2, 0, 1 und 2 zeigen, dass die Intervalle $\{-\infty < x < -2\}$ und $\{1 < x < \infty\}$ die Ungleichung lösen. Es muss aber noch die Bedingung $x < 1$ erfüllt sein, und daher ist

$$\{-\infty < x < -2\}$$

die zweite Teillösung.

Damit ist die Lösungsmenge $\{1 < x < \infty, \quad -\infty < x < -2\}$.

Alternative Vorgehensweise

Man kann Ungleichungen auch analog zu Gleichungen durch äquivalente Umformungen auf Normalformen bringen, aus denen die Lösung direkt abzulesen ist. Dabei ist zu beachten, dass für Ungleichungen die Menge der äquivalenten Umformungen kleiner ist als bei Gleichungen:

- Die Multiplikation beider Seiten einer Ungleichung mit einem Term ist nur dann eine Äquivalenzumformung, wenn der Term positiv ist. Enthält der Term die Unbekannte x, so ist das überhaupt nicht im voraus feststellbar! Es müssen dann wieder Fallunterscheidungen vorgenommen werden.
- Das Potenzieren beider Seiten einer Ungleichung ist nur für ungerade Exponenten eine Äquivalenzumformung. Bei geraden Potenzen hat die Lösungsmenge der neuen Ungleichung im allgemeinen überhaupt keinen Zusammenhang mehr mit der ursprünglichen Ungleichung.

  Beispiel:

$$\begin{array}{rcll} x & < & -1 & \quad | \quad (\ldots)^2 \\ x^2 & < & 1 & \end{array}$$

  Die erste Ungleichung hat als Lösungsmenge das Intervall $\{-\infty < x < -1\}$, die zweite aber $\{-1 < x < 1\}$. Beide Intervalle haben keinen einzigen Punkt gemeinsam!

7.1 Übung

Führen Sie jetzt die Übung 7.1 aus, in der Sie unbekannte Werte x aus Gleichungen mit Beträgen bestimmen sollen.
Beispiel:

`Finden Sie alle` x`, die die folgende Gleichung erfüllen:`
$|2 - x| = 3$

Erwartetes Ergebnis: $x \in \{-1, 5\}$.

Das Programm erwartet die Eingabe `{-1,5}`.

7.2 Übung

Führen Sie jetzt die Übung 7.2 aus, in der Sie unbekannte Werte x aus Ungleichungen mit Beträgen bestimmen sollen.
Beispiel:

`Finden Sie alle` x`, die die folgende Ungleichung erfüllen:`
$|x - 48| < \frac{x}{5}$

Erwartetes Ergebnis: $40 < x < 60$.

Das Programm erwartet die Eingabe $\{40 < x < 60\}$.

Anhang A

Bedienung des Repetitoriums

Das Repetitorium nutzt die Möglichkeiten der graphischen Oberfläche von Mathematica 4.0 aus und bietet eine menügesteuerte Bedienung. Die Programmsteuerung erfolgt über Dialoge, Ergebnisse werden in Notebooks dargestellt, und die Eingabe von Formeln in üblicher mathematischer Schreibweise ist möglich.

A.1 Starten und Beenden

Zum Start des Programmes sind folgende Schritte auszuführen:

1. Starten des Mathematica-Programmes unter einem geeigneten Betriebssystem (z.B. Windows, Linux oder auf einem Macintosh-Rechner).

2. Öffnen des Starter-Notebooks `Starter.nb` über das Datei-Menü von Mathematica, siehe Abb. A.1.

Abbildung A.1: *Starter-Notebook*

3. Starten des Repetitoriums durch Klick auf den Button *Starten*.

Daraufhin öffnet sich ein Notebook[1], welches im oberen Teil ein Auswahlmenü mit den Übungen zum Repetitorium enthält (Abb. A.2).

Abbildung A.2: *Notebook mit Auswahlmenü*

Klicken Sie wahlweise auf den *Auswahl*-Button oder auf das Dreieck links davon. Es erscheint eine Liste der Übungsteile (Abb. A.3).

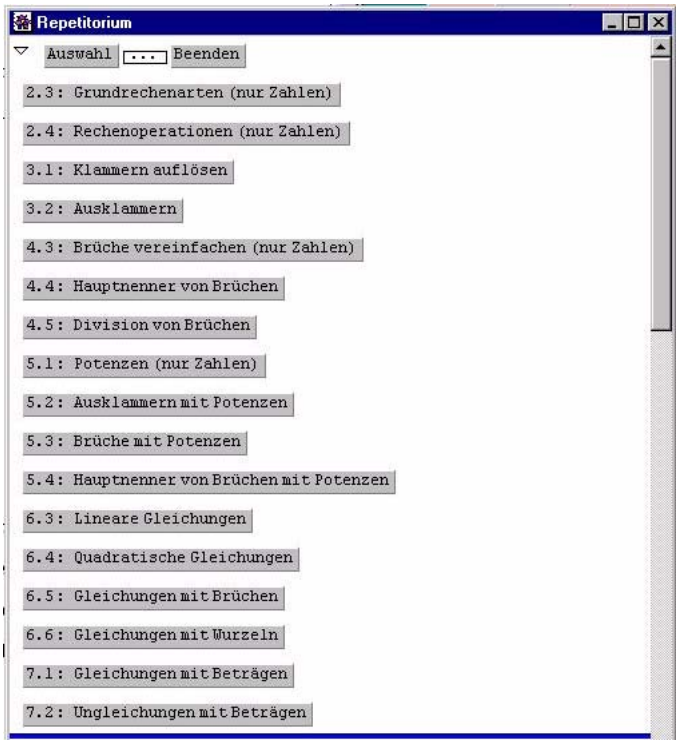


Abbildung A.3: *Auswahlmenü mit Liste der Übungen*

Wählen Sie die gewünschte Übung aus. Es erscheint im unteren Teil des Notebooks ein Dialog, der Ihnen eine zufällig gewählte Übungsaufgabe zur Lösung anbietet.

Nach Auswahl von z.B. *2.3: Grundrechenarten (nur Zahlen)* finden Sie den Dialog wie in Abb. A.4 vor.

Sie sollten die Aufgabe mit Papier und Bleistift lösen, ohne Mathematica zu Hilfe zu nehmen. Ihre Antwort tragen Sie in das Antwortfeld ein und klicken dann auf

[1] Beim ersten Starten des Repetitoriums erscheint ein Dialog, in welchem Mathematica fragt, ob alle Initialisierungszellen evaluiert werden sollen. Beantworten Sie diese Frage mit „ja“.

Abbildung A.4: *Dialog für Übung 2.3*

den Button *Prüfen*. Falls Ihre Antwort nicht oder nicht vollständig richtig ist, bekommen Sie die richtige Antwort angezeigt. Dazu wird Ihnen, wenn möglich, eine Hilfestellung angeboten, die einen Hinweis auf die Ursache des Fehlers gibt. Ein Beispiel zeigt Abb. A.5.

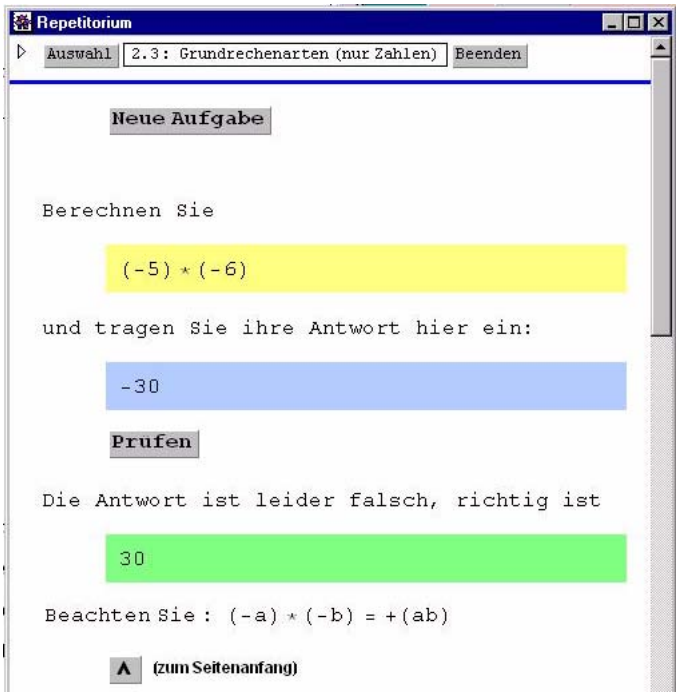


Abbildung A.5: *Ergebnis der Prüfung für Übung 2.3*

Abhängig vom jeweiligen Übungsteil werden Ihnen weitere Möglichkeiten angeboten:

Vorführen: Bei vielen Übungen wird Ihnen eine Vorführung des Lösungsweges angeboten. Klicken Sie einfach auf den Button *Vorführung*, und es öffnet sich ein weiteres Notebook mit einem kommentierten Lösungsweg. Die Abbildungen A.8 bis A.10, ab Seite 46, zeigen ein Beispiel für die Übung 4.3, „Brüche vereinfachen (nur Zahlen)“.

Kontrolle: Bei Klick auf diesen Button öffnet sich ein Notebook mit einem Dialog, der es Ihnen ermöglicht, Ihre Zwischenergebnisse in beliebiger Reihenfolge einzugeben und auf Richtigkeit prüfen zu lassen. Bei Fehlern wird Ihnen möglichst wieder eine Hilfestellung gegeben. Die Abbildungen A.11 und A.12, ab Seite 47, zeigen ein Beispiel für die Übung 5.4, „Hauptnenner von Brüchen mit Potenzen“.

Beim Lösen von Gleichungen bietet dieser Dialog noch weitere Möglichkeiten:

- Auswahl einer Termumformung und Prüfung des eigenen Umformungsergebnisses auf Korrektheit,
- Übernahme der korrekt umgeformten Gleichung als neue Gleichung für die nächste Umformung
- Prüfung einer beliebigen, nicht im Dialog auswählbaren Umformung,
- Das Repetitorium kann eine Umformung vorschlagen.

Die Abbildungen A.13 bis A.15, ab Seite 48, zeigen ein Beispiel.

Eigene Aufgabe: Sie haben hier Gelegenheit, eine Aufgabe eigener Wahl einzugeben. Damit die Aufgabe akzeptiert wird, muss sie natürlich zum gewählten Übungsteil passen. Sie können dann mit dieser Aufgabe so verfahren wie mit jeder anderen angebotenen Aufgabe auch. Diese Option ist insbesondere dann interessant, wenn man sich Aufgaben aus anderen Quellen vorführen lassen möchte.

Für Gleichungen und Ungleichungen mit Beträgen (Übungen 7.1 und 7.2) erscheint ein Button, der die bequeme Eingabe von Absolutbetragsstrichen ermöglicht. Er wird wie jeder andere Button aus den Paletten von Mathematica verwendet. Eine Vorführung ist nur möglich, wenn die eingegebene Aufgabe höchstens einen Betragsterm enthält. Die Abbildungen A.16 bis A.18, ab Seite 51, zeigen ein Beispiel.

> *Bemerkung:* Der Button *Eigene Aufgabe* gibt das Aufgabenfeld zur Eingabe frei. Nach Auswahl einer anderen Funktion (*Prüfen, Vorführen, ...* ist das Aufgabenfeld wieder gesperrt. Der Button muss vor jedem neuen Editieren erneut gedrückt werden.

Alle sich öffnenden Notebooks besitzen einen Button zum Schließen oder Beenden und können jederzeit verlassen werden. Der *Schließen*-Button des Starter-Notebooks schließt das Starter-Fenster und beendet die Anwendung.

A.2 Paletten und Tastaturkürzel in Mathematica

Vorbemerkung: Zur deutlichen Unterscheidung sind in diesem Abschnitt Terme in mathematischer Schreibweise *kursiv* gesetzt. Eingaben an Mathematica

werden hingegen in `Schreibmaschinenschrift` gesetzt. Das Zeichen ␣ steht dabei für ein Leerzeichen (Leerschritt).

Die Benutzereingaben müssen in einer Weise erfolgen, die Mathematica versteht, d.h. sie müssen der Syntax von Mathematica genügen. Die Eingabe von Termen in Mathematica ist weitgehend der mathematischen Schreibweise angeglichen. Sie weicht aber in einigen Punkten davon ab. Es stehen für das Repetitorium alle Möglichkeiten der graphischen und menügesteuerten Eingabe von Formeln im Formelsatz zur Verfügung.

Zur Eingabe von Brüchen, Potenzen und Wurzeln gibt es in Mathematica u.a. die folgenden zwei Möglichkeiten:

1. Eingabe per Palette.

   Standardmäßig öffnet Mathematica eine Palette, auf der Symbole für Brüche, Potenzen und Wurzeln zu sehen sind. Termteile werden durch Platzhalter dargestellt. Will man einen Palettenbutton nutzen, so markiere man zuerst den gewünschten Teilterm mit der Maus und klicke dann auf den Button. Der markierte Teilterm wird dann jeweils zum Zähler des Bruches, zur Basis der Potenz oder zum Radikanden. Per Tabulator ⇥ erreicht man einen Platzhalter, in den man den fehlenden Nenner bzw. den Exponenten eintragen kann. Siehe die Abb. A.6.

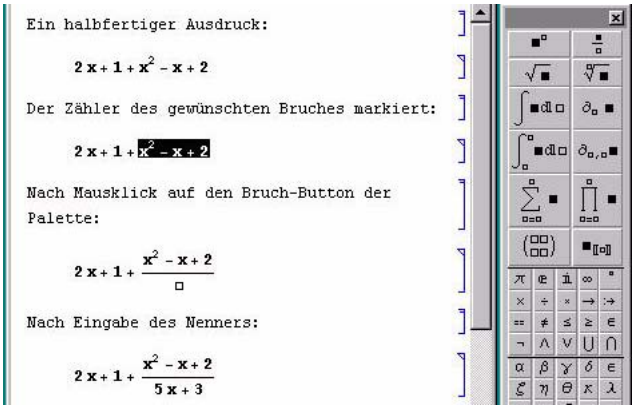


Abbildung A.6: *Benutzung eines Palettenbuttons*

2. Eingabe per Tastaturkürzel.

   Zuerst markiert man den Teilterm, der Zähler eines Bruches, Basis einer Potenz oder Radikand einer Wurzel werden soll. Dies kann auch per Tastatur über „*Ctrl-.*“ geschehen. Bei jedem Tastendruck auf diese Kombination wird der nächstgrößere Teilterm an der Mausposition markiert. Anschließend erzeugt man

   - einen Bruch mit „*Ctrl-Shift-7*“,
   - eine Potenz mit „^“,
   - eine Quadratwurzel mit „*Ctrl-2*“.

Danach kann der Nenner bzw. der Exponent direkt eingegeben werden. Man bewegt den Cursor rechts neben den Formelteil mit der Pfeiltaste $\boxed{\rightarrow}$. Siehe Abb. A.7.

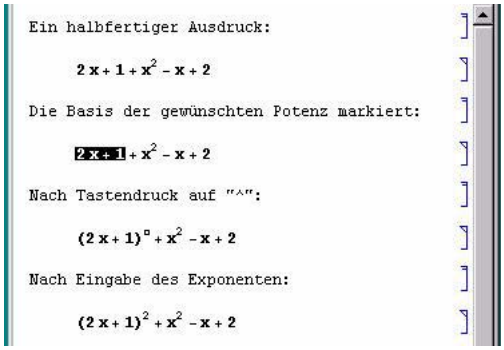


Abbildung A.7: *Benutzung eines Tastaturkürzels*

A.3 Formeln im Repetitorium

Auf einige Besonderheiten soll nun eingegangen werden:

Vorzeichen von Zahlen und Variablen werden in Mathematica nicht notwendig mit der Zahl zusammen in einer Klammer geschrieben. Statt

$$5 + (-2)$$

darf also `5+-2` eingegeben werden. Es schadet aber nichts, wenn bei der Eingabe die Klammern wie in `5+(-2)` gesetzt werden. In der Ausgabe werden die Klammern allerdings so gut wie immer weggelassen.

Allgemein gilt, dass nur unbedingt notwendige Klammern geschrieben werden müssen, und dass in der Ausgabe nur unumgängliche Klammern geschrieben werden. Mathematica versteht aber Ausdrücke mit zusätzlichen Klammern richtig.

Multiplikation wird in Mathematica außer durch das Zeichen „*“ auch durch ein – oder mehrere aufeinanderfolgende – Leerzeichen dargestellt, sofern dieses Leerzeichen nicht links oder rechts an ein anderes Operationszeichen anschließt. $a \cdot b$ darf also als `a*b` oder als `a␣b` eingegeben werden. `ab` hingegen meint eine Variable mit Namen ab.

`a+␣b` meint hingegen nicht $a + \cdot b$, sondern $a + b$, das Leerzeichen wirkt hier nur als Leerzeichen, nicht als Multiplikation.

Zwischen einer Zahl und einer folgenden Variablen oder zwischen einer Zahl und einer folgenden öffnenden Klammer kann sogar noch das Leerzeichen weggelassen werden: $2ab$ darf als `2a␣b` eingegeben werden, $2(a+b)$ als `2(a+b)`.

Dadurch wird eine enge Anlehnung an die in der Mathematik übliche Schreibweise ab für das Produkt von a und b möglich.

Ungewöhnlich ist, dass auch zwischen Zahlen die Multiplikation durch eine Leerstelle angegeben werden kann. `2␣3` steht also auch für `2*3`. Im Repetitorium ist allerdings dafür gesorgt worden, dass diese Schreibweise bei Ausgaben nicht vorkommt. Auch wird meistens das Multiplikationszeichen „*“ durch einen Punkt „·“ ersetzt. Solche Ausgaben lassen sich aber nicht per *cut-and-paste* als Eingaben nutzen.

Im Repetitorium ist die Anpassung an die mathematisch üblichen Schreibweisen von Produkten so weit gegangen, dass auch das Leerzeichen zwischen Variablen (und Zahlen!) zur Darstellung der Multiplikation nicht angezeigt wird. `a␣b` erscheint also als `ab`. Die Leerstelle *muss* aber auf jeden Fall eingegeben werden und ist in der internen Darstellung auch vorhanden. Der Benutzer darf sich nicht durch den Augenschein täuschen lassen, was nicht leicht ist, da optisch zwischen der Variablen mit Namen `ab` und dem Produkt `a*b` kein Unterschied zu sehen ist. Es ist daher vorzuziehen, in der Eingabe den Asterisk („*“) als Multiplikationszeichen zu wählen.

Brüche können sowohl mit Bruchstrich als auch im Formelsatz eingegeben werden. Statt

$$\frac{a+b}{c+d}$$

ist die gleichbedeutende Schreibweise

$$(a+b)/(c+d)$$

möglich und als `(a+b)/(c+d)` einzugeben. Die Klammern müssen dabei unbedingt gesetzt werden, denn `a+b/c+d` bedeutet

$$a+\frac{b}{c}+d.$$

Absolutbeträge können nicht mit dem üblichen senkrechten Strich („|“) geschrieben werden, sondern es müssen zwei spezielle Zeichen verwendet werden, die Mathematica zur Verfügung stellt. Zwei Zeichen sind insbesondere deshalb nötig, damit der „linke“ Strich vom „rechten“ unterschieden werden kann. Da sie über die Tastatur nur umständlich einzugeben sind, wird ein Button bereitgestellt, mit dessen Hilfe ein Term leicht mit Absolutbeträgen versehen werden kann. Dazu muss wie bei einem Palettenbutton nur der (Teil-)term mit der Maus markiert und anschließend der Button gedrückt werden.

A.4 Lösen von Gleichungen

Besonders umfangreiche Möglichkeiten bieten die Abschnitte 6.3 bis 6.6 mit Aufgaben zu verschiedenen Typen von Gleichungen. Bei der Eingabe eigener Aufgaben ist der Benutzer frei, beliebig komplizierte Gleichungen einzugeben, sofern er im Bereich der rationalen Terme mit evtl. Wurzelfunktionen verbleibt. Bei Klick auf den Button *Prüfen* wird er fast immer die Lösungsmenge angezeigt bekommen, auch für Gleichungen höheren als 2. Grades. Eine Vorführung

ist aber nur vorgesehen und sinnvoll, wenn die resultierende algebraische Gleichung nicht vom 3. oder höherem Grad ist.

Bei der Kontrolle eigener Rechnungen ist, wie aus Abb. A.15, Seite 50, ersichtlich, gerade jener Vorrat an Termumformungen in Gestalt von *Radio-buttons* vorgegeben, der benötigt wird, um Gleichungen, die auf algebraische Gleichungen 2. Grades führen, bearbeiten zu können. Mit dem Button *Freie Umformung prüfen* können aber auch andere Umformungen, z.B. das Ausziehen dritter Wurzeln, kontrolliert werden. Allerdings kann dabei nicht festgestellt werden, ob beide Seiten der Gleichung äquivalent umgeformt wurden, denn dazu muss das System Hinweise über die vorgesehene Umformung haben. Es wird stattdessen überprüft, ob die Lösungsmenge durch die Umformung verändert wurde.

Der Vorschlag zur Termumformung ist technisch einfach so ausgeführt, dass der nächste Schritt, den das Repetitorium vornehmen würde, angezeigt wird. Dabei wird dieser Schritt unabhängig von vorhergehenden Umformungen ausgewählt, was dazu führen kann, dass abwechselnd Multiplizieren und Dividieren mit demselben Term vorgeschlagen wird. Beim Vorführen einer Gleichung durch das Repetitorium wird solch zyklisches Verhalten natürlich verhindert. Sicherlich ließe sich das auch für die Vorschläge tun, aber so ist der Benutzer gehalten, die Vorschläge selber auf ihren Sinn zu prüfen, er kann ihnen nicht blindlings folgen.

Während der Kontrolle werden fortlaufend Kommentare über ausgeschlossene Werte der Unbekannten x, Veränderungen der Lösungsmenge und zusätzlich zu betrachtende Gleichungen gegeben. Ihre Beachtung ist Sache des Benutzers.

A.5 Grundsätzliche Hinweise

Das Repetitorium ist weit davon entfernt, fehlerfrei zu sein, trotz vieler Anstrengungen. Insbesondere bei der Behandlung eigener Aufgaben muss man auf Überraschungen gefasst sein. Fehler, die Mathematica während der Durchführung des Repetitoriums feststellt, werden unter Umständen mitten in die Dialoge der Notebooks hineingeschrieben und stören dann deren Aufbau. Man verfahre dann wie folgt:

- Zellen mit Fehlermeldungen (kenntlich an der blauen Schrift, dem englischen, meist etwas kryptischen Text) lassen sich markieren[2] und dann löschen.
- Fehlermeldungen verschwinden meist, wenn eine neue Aufgabe angefordert wird.
- Neuauswahl eines Übungsteils liefert immer einen sauberen neuen Dialog.
- Wenn, in sehr seltenen Fällen, das Repetitorium in eine Endlosschleife gerät, kann es mit den für Mathematica üblichen Methoden, d.h. durch Drücken von „*Alt-.*“, unterbrochen werden.

[2] am rechten Notebookrand, da, wo normalerweise die Zellenklammern stehen. Diese sind im Repetitorium verdeckt worden, aber mit ihrer Funktionalität noch vorhanden.

Mathematica ist, wenigstens auf manchen Rechnern, durchaus im Stande, urplötzlich und sang- und klanglos abzustürzen. Wer die Ergebnisse von Vorführungen länger aufheben möchte, kann die generierten Notebooks ganz normal speichern, und er sollte dies rechtzeitig tun. Die Zahl der möglichen generierten Aufgaben ist so groß, das die Wahrscheinlichkeit gering ist, dieselbe Aufgabe zweimal zu sehen.[3]

[3] Eine Ausnahme bildet der Abschnitt 5.1, wo die Auswahl an sinnvollen Aufgaben klein ist, sollen die vorkommenden Zahlen nicht unsinnig groß werden.

A.6 Beispieldialoge

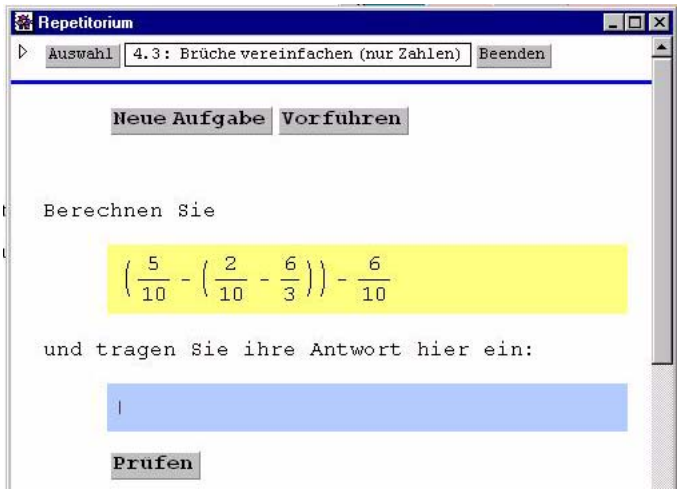


Abbildung A.8: *Dialog für Übung 4.3*

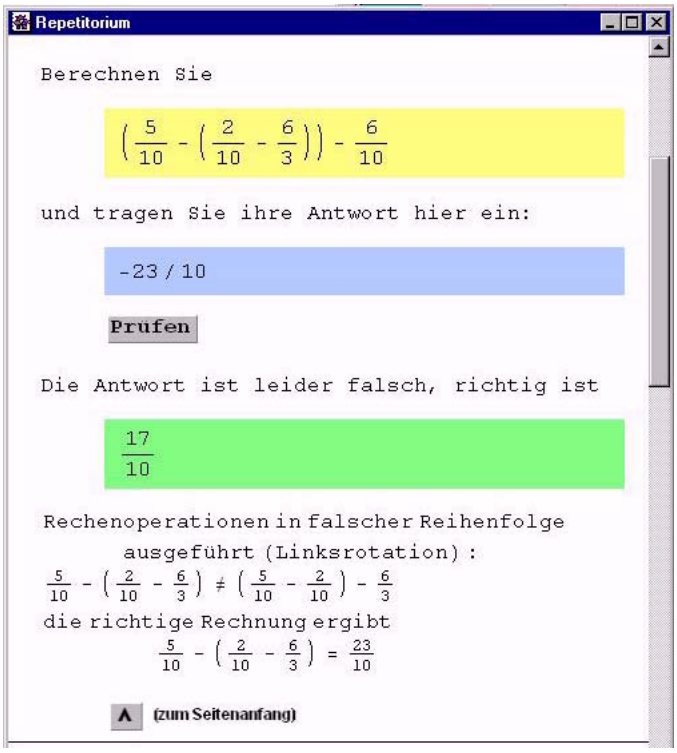


Abbildung A.9: *Ergebnis der Prüfung für Übung 4.3*

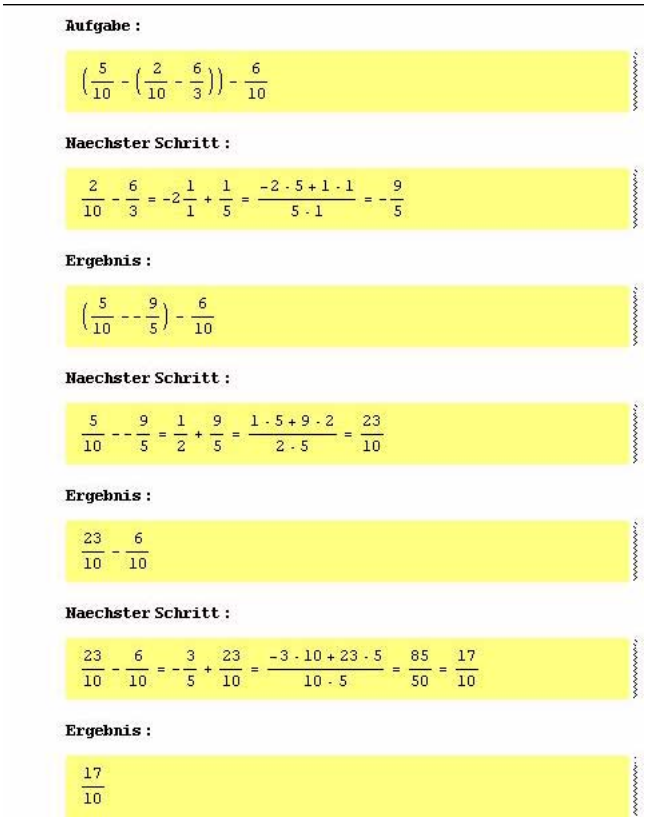


Abbildung A.10: *Vorführung für Übung 4.3*

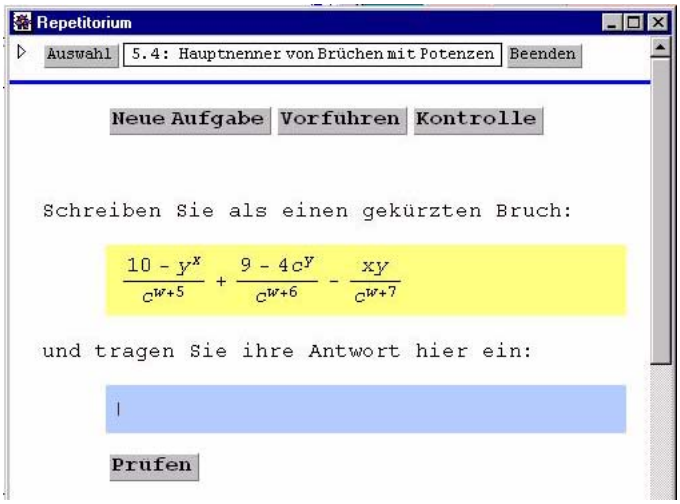


Abbildung A.11: *Dialog für Übung 5.4*

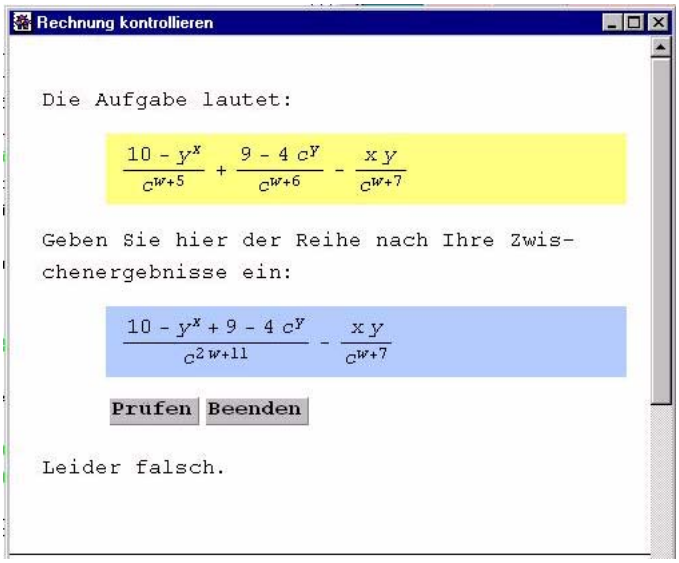


Abbildung A.12: *Kontrolle für Übung 5.4*

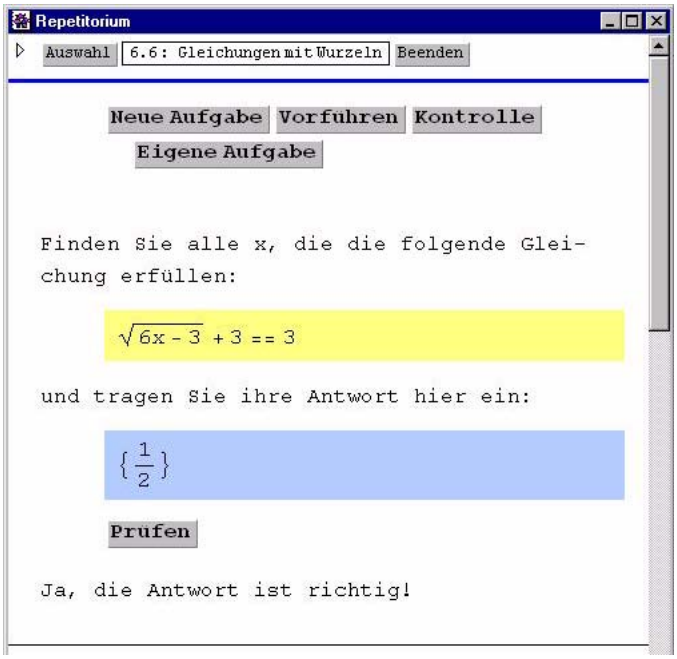


Abbildung A.13: *Dialog für Übung 6.6*

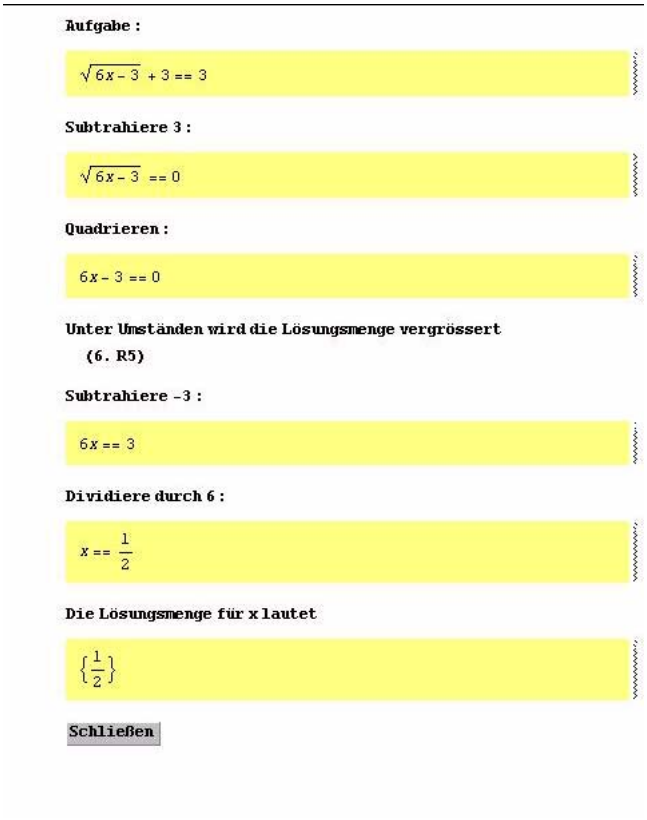


Abbildung A.14: *Vorführung für Übung 6.6*

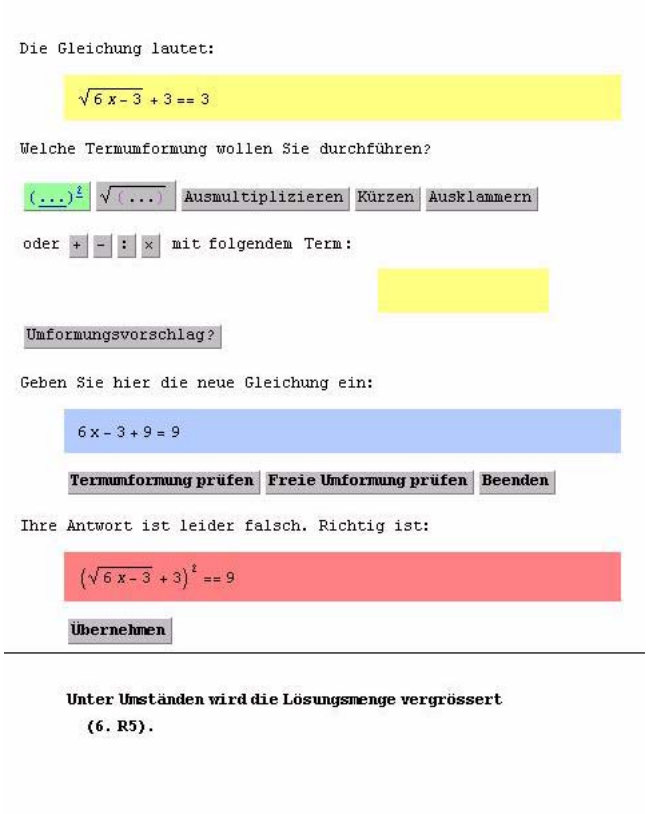


Abbildung A.15: *Kontrolle für Übung 6.6*

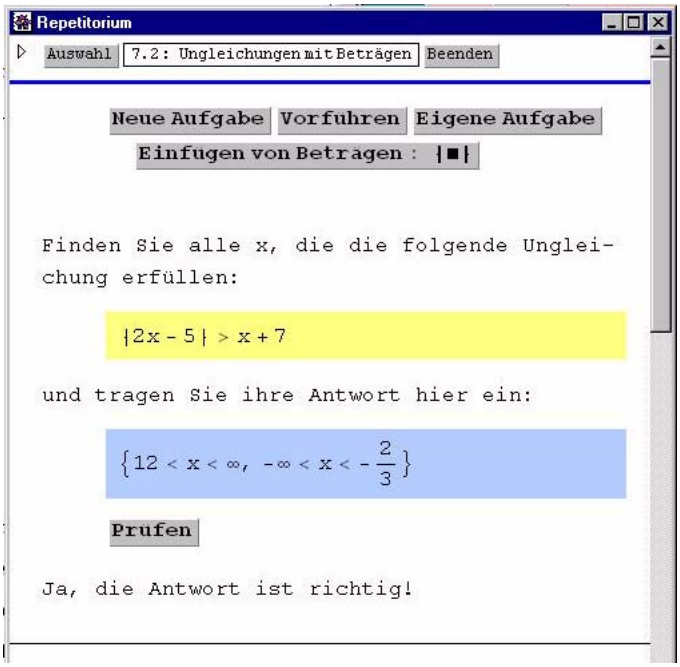


Abbildung A.16: *Dialog für Übung 7.2*

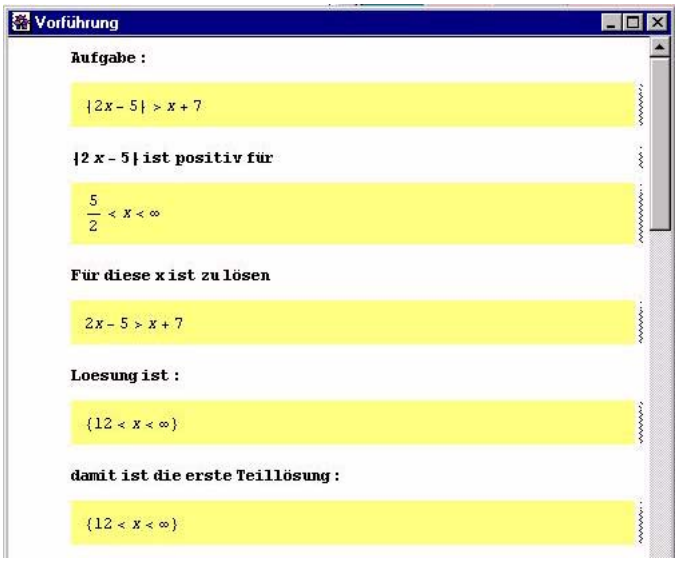


Abbildung A.17: *Vorführung für Übung 7.2*

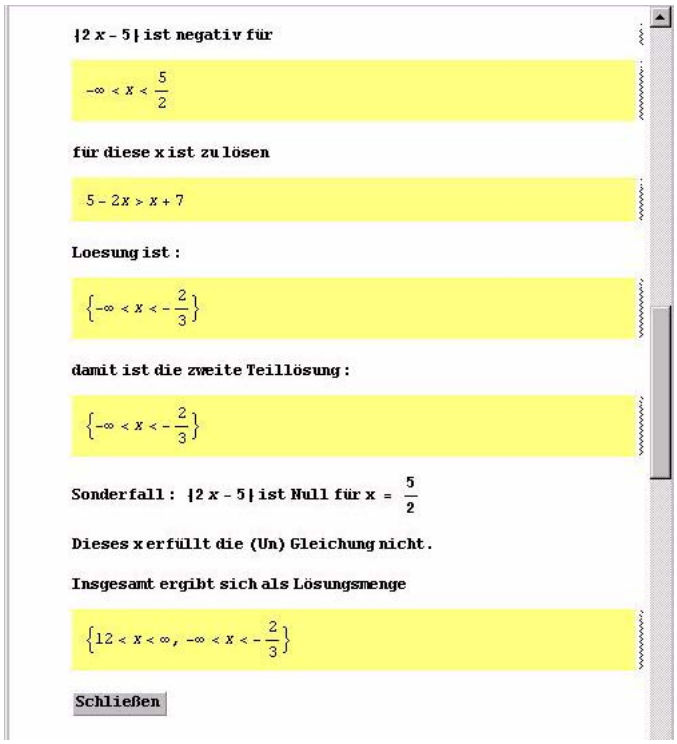


Abbildung A.18: *Vorführung für Übung 7.2 (Forts.)*

Anhang B

Die Struktur des Repetitorium-Programmes

Das Programm des Repetitoriums liegt nun in der zweiten Version vor, stellt aber immer noch einen Prototyp dar. Zu seiner Realisierung wurde das Softwarprodukt Mathematica in der Version 4.0 benutzt. Seit Version 3.0 ist Mathematica mit einer graphischen Oberfläche ausgestattet und unterstützt eine besondere Dokumentform, in der (interaktive) Formeln und Textsatz vereint sind, die sogenannten *Notebooks*. Diese Notebooks sind aus Mathematica heraus programmierbar, woraus sich die realisierten neuen Möglichkeiten der Benutzerführung ergeben haben.

B.1 Realisierung der Oberfläche des Repetitoriums

Der Aufbau von Mathematica-Notebooks aus einzelnen Zellen, die je nach Typ Formeln, Text, Buttons oder anderes enthalten können, wird im Handbuch zu Mathematica detailliert erklärt. Damit sind Notebooks selber Ausdrücke, die mit den üblichen Mitteln von Mathematica manipuliert und verändert werden können. Benötigt werden nur noch eine Handvoll Funktionen, die Zellen, Zellinhalte oder auch ganze Notebooks in Variablen einlesen bzw. Notebooks oder Zellen schreiben können. Diese Funktionen sind (in Auswahl)

SelectionMove zum Navigieren innerhalb der Notebooks,

NotebookRead zum Einlesen von Notebooks oder Zellen,

NotebookWrite zum Schreiben und

NotebookPut zum Öffnen eines neuen Notebooks.

Diese Funktionen sind im Handbuch zwar beschrieben, ihr Einsatz zur Gestaltung einer Benutzeroberfläche wird aber nicht ausgeführt. Die folgenden Abschnitte sollen erläutern, wie man dabei vorgehen kann.

B.1.1 Auswahlmenü

Für die Gestaltung eines Auswahlmenüs findet sich im Mathematica-Handbuch ein Beispiel in Gestalt einer Palette, die ein Pulldown-Menü bereitstellt. Erläuterungen zur Arbeitsweise werden nicht gegeben, der Code ist unübersichtlich. Obwohl man denselben sicher hätte analysieren können, ist für das Auswahlmenü eine einfachere und direktere Konstruktion verwendet worden.

Die Grundidee besteht darin, einfach eine Zellengruppe zu verwenden, deren oberste Zelle den gewählten Menüpunkt anzeigt, während die anderen Zellen Buttons enthalten, welche bei Mausklick die Auswahl tätigen. Durch Öffnen bzw. Schließen der Zellgruppe werden die Auswahlbuttons sichtbar gemacht bzw. wieder verdeckt. Auf diese Weise muß nur die oberste Zelle, in der die Auswahl angezeigt wird, schreibend verändert werden, alle anderen Zellen sind statisch. Das Auswahlmenü in geschlossenem bzw. geöffnetem Zustand zeigen die Abbildungen A.2 bzw. A.3, ab Seite 38.

Interessant sind die Funktionen, die sich hinter den Auswahlbuttons verbergen. Die Funktion des Buttons *Auswahl* ist:

```
openChoices:=(Module[{nb,c,t},
        nb=ButtonNotebook[];
        SelectionMove[nb,All,CellGroup];
        c=NotebookRead[nb];
        c[[1,-1]]=Open;
        NotebookWrite[nb,c];
        ])&
```

Diese Funktion öffnet die Auswahlliste. Dies geschieht in der Zeile

```
  c[[1,-1]]=Open.
```

Verständlich wird dieser Schritt, wenn man sich die Zellgruppe, die das Auswahlmenü enthält, ansieht:

```
Cell[
  CellGroupData[..., Closed]
]
```

Die Details der Zellgruppe, durch „...“ angedeutet, sind hier nicht ausgeführt, wichtig ist, dass der letzte Parameter über „geöffnet“ oder „geschlossen“ entscheidet und durch `openChoices` umgeändert wird.

Die einzelnen Einträge der Auswahlbuttons sehen so aus (dargestellt am Beispiel der Übung 2.3):

```
Cell[
  BoxData[
    ButtonBox["Grundrechenarten (nur Zahlen)",
              Active->True,
              ButtonData->f2$3,
              ButtonSource->ButtonContents,
```

```
                ButtonEvaluator->Automatic,
                ButtonFunction->makeChoice
        ]
      ]
    ]
```

Dabei ist `f2$3` eine Funktion, die den Dialog für Übung 2.3 erstellt. Dazu wertet sie eine Liste mit den Beschreibungen der Buttons, die zum Dialog für Übung 2.3 gehören, aus. Die Funktion `makechoice` klappt die Auswahlliste wieder zu und erstellt dann mittels `f2$3` den Dialog:

```
     makeChoice:=(Module[{nb,c,t},
        nb=ButtonNotebook[];
        SelectionMove[nb,All,CellGroup];
        c=NotebookRead[nb];
(1)     t=#1[[1]];
(2)     c[[1,1,1]]=c[[1,1,1]]/.FrameBox[_]->FrameBox[t];
(3)     c[[1,-1]]=Closed;
        NotebookWrite[nb,c];
        ...
(4)     #2[];
        ])&
```

Hier wird in Zeile (1) der erste Parameter der Buttonfunktion in die Variable `t` gestellt. Dies ist[1] der Buttontext „Grundrechenarten (nur Zahlen)“. Zeile (2) schreibt diesen Text in das Textfeld des Auswahlmenüs. In Zeile (3) wird die Zellengruppe geschlossen. Zeile (4) ruft den zweiten Parameter der Buttonfunktion als Funktion auf, also die Funktion `f2$3`, die nun den Dialog für Übung 2.3 zeichnet.

Die Verwendung von `f2$3` mutet vielleicht etwas umständlich an, schließlich hätte man hier in Zeile (4) einfach mittels `NotebookWrite` jene Zellen explizit schreiben können, die den Dialog ausmachen. Ein kurzer Ausschnitt aus den Definitionen der Funktionen `fn$m`, nämlich

```
f2$3=(makeDialog[{
            {"Neue Aufgabe",DialogX["2.3"]&}
            }]&);

f2$4=(makeDialog[{
            {"Neue Aufgabe",DialogX["2.4"]&},
            {"Vorführen",
              VorfuehrungRechenaufgabe["2.4",aufgBaum]&}
            }]&);

f3$1=(makeDialog[{
            {"Neue Aufgabe",DialogX["3.1"]&},
            {"Vorführen",VorfuehrungFaktorAufgabe&},
            {"Kontrolle",KontrolleFaktorAufgabe&},
```

[1] siehe Mathematica-Handbuch, `ButtonFunction`, [WO99]

```
        {"Eigene Aufgabe",eigeneAufgabe&}
        }]&);
```

zeigt aber, dass auf die hier gewählte Weise die Übungsteile leicht und übersichtlich erweiterbar sind. Die Liste hinter `makeDialog` enthält Paare von Buttontexten und Buttonfunktionen. Man sieht auch die einfache Syntax: irgendein Symbol wird zu einer Funktion durch Anhängen von „&“.

B.1.2 Dialogsteuerung

Die Erstellung der Dialoge zu den Übungen ist kein besonderes Problem. Man kann sie fast wie mit einem GUI-Werkzeug entwerfen, indem man einfach in einem Hilfs-Notebook mit den üblichen Menüfunktionen von Mathematica den Dialog aus Texten und Buttons zusammenstellt, mit Hintergrundfarben, Schriftarten, etc. versieht und dann mittels `NotebookRead` in eine Variable einliest. Dann hat man kein Problem mehr, mit Hilfe dieser Variablen diesen Dialog dynamisch zu erzeugen. Man sehe sich noch einmal Abb. A.4 an: der Button *Neue Aufgabe* wird durch `makeDialog` geschrieben, der eigentliche Ein-/Ausgabebereich hingegen ist eine vorgefertigte Zellgruppe:

```
einAusCellGroup =Cell[CellGroupData[{
        Cell[BoxData[""],
          "Text",
          CellTags->"anfangsdummy",
          Editable->False,
          CellOpen->False],
        Cell["",
          "Text",
          FormatType->TextForm,
          Editable->False,
          CellTags->"aufgabentext"],
        Cell[BoxData[""],
          "Print",
          Background->RGBColor[1,1,0.5],
          ZeroWidthTimes->True,
          Editable->False,
          CellTags->"aufgabenfeld"],
        Cell["und tragen Sie ihre Antwort hier ein:",
          "Text",
          Editable->False,
          FormatType->TextForm],
        Cell[BoxData[""],
          "Print",
          Background->RGBColor[0.7,0.8,1],
          ZeroWidthTimes->True,
          Editable->True,
          CellTags->"eingabefeld"],
        Cell[BoxData[ButtonBox[
              "Prüfen",
```

```
            Active->True,
            ButtonStyle->"Evaluate",
            ButtonEvaluator->Automatic,
            ButtonFunction->Null
            ]],
        "Input",
        Editable->False,
        CellTags->"fertigbutton"],
      Cell[""",
        "Text",
        FormatType->TextForm,
        CellTags->"bemerkung",
        Editable->False,
        CellOpen->True],
      Cell[BoxData[""],
        "Print",
        Background->RGBColor[0.5,1,0.5],
        ZeroWidthTimes->True,
        CellTags->"antwortfeld",
        Editable->False,
        CellOpen->False],
      Cell[BoxData[""],
        "Text",
        CellTags->"analysefeld",
        Editable->False,
        CellOpen->True],
      Cell[BoxData[ButtonBox[
            "^",
            Active->True,
            ButtonStyle->"Evaluate",
            ButtonEvaluator->Automatic,
            ButtonFunction:>
              (SelectionMove[ButtonNotebook[],Before,Notebook]&)
            ]],
        "Input",
        Editable->False,
        CellOpen->False,
        CellTags->"topbutton"],
      Cell[BoxData[""],
        "Text",
        CellTags->"abschlussdummy",
        Editable->False,
        CellOpen->False]
      },
    Open]];
```

Hierzu einige Kommentare:

- Jede Zelle hat ein `CellTag` bekommen. Damit sind diese Zellen leicht und eindeutig mit `SelectionMove` ansteuerbar, können gelesen, verändert und

wieder neu geschrieben werden.

- Fast alle Zellen tragen das Attribut `Editable->False`. Dadurch wird der Dialog vor versehentlicher Veränderung geschützt. Eingabefelder (`CellTag` „eingabefeld“) sind natürlich ausgenommen. Das Beschreiben der ebenfalls schreibgeschützten Ausgabefelder geschieht durch einfache Funktionen wie z.B.

```
printToCell[inhalt_,tag_]:=
  Module[{nb},
    nb=SelectedNotebook[];
    cellEditable[tag];
    If[NotebookFind[nb,tag,All,CellTags]===$Failed,
      SelectionMove[nb,After,Notebook]];
    SelectionMove[nb,All,CellContents];
    NotebookWrite[nb,ToBoxes[inhalt,TraditionalForm]];
    cellNotEditable[tag];
    showCell[tag];
    ]

cellEditable[tag_]:=
  Module[{nb,cc},
    nb=SelectedNotebook[];
    If[NotebookFind[nb,tag,All,CellTags]===$Failed,Return[]];
    SelectionMove[nb,All,Cell];
    cc=NotebookRead[nb];
    cc=cc/.Rule[Editable,_]->Rule[Editable,True];
    NotebookWrite[nb,cc];
    ]

cellNotEditable[tag_]:=
  Module[{nb,cc},
    nb=SelectedNotebook[];
    If[NotebookFind[nb,tag,All,CellTags]===$Failed,Return[]];
    SelectionMove[nb,All,Cell];
    cc=NotebookRead[nb];
    cc=cc/.Rule[Editable,_]->Rule[Editable,False];
    NotebookWrite[nb,cc];
    ]
```

- Das Attribut `CellOpen` entscheidet über die Sichtbarkeit der Zelle. Durch Ändern von `False` auf `True` und umgekehrt kann die Zelle sichtbar gemacht bzw. verdeckt werden.

- Der Button *Prüfen* hat `Null` als Buttonfunktion. Diese wird beim Schreiben der Zellgruppe durch die richtige Funktion ersetzt:

```
c=c/.Null->((hideCell["antwortfeld"];
             hideCell["analysefeld"];
             hideCell["topbutton"];
             setFocus["eingabefeld"];
```

```
DialogIn[typ])&);
```

Funktionen wie `hideCell` haben einen `CellTag` als Parameter und tun genau das, was ihr Name sagt. Die eigentliche Prüfung der Eingabe übernimmt `DialogIn[typ]`, `typ` enthält die Übungsnummer.

Ein wenig komplexer ist es, Radiobuttons zu erstellen, wie sie im Dialog zur Kontrolle der Termumformungen bei Gleichungen verwendet wurden, siehe Abb. A.15. Hier bilden mehrere Buttons, verteilt auf mehrere Zellen, eine Gruppe von Radiobuttons. Zur Realisierung wurden die Zellen zu einer Zellgruppe zusammengefaßt. Nun können alle Radiobuttons in eine Variable eingelesen werden, indem diese Zellgruppe als Ganzes gelesen wird. Die Buttonfunktion eines Radiobuttons heißt `buttonSelected` und bekommt als Parameter den Buttontext:

```
ButtonBox["+",
  ButtonFunction:>(buttonSelected["+"]&),
  ButtonEvaluator->Automatic,
  Active->True,
  ButtonStyle->"Evaluate",
  Background->GrayLevel[0.75]],
```

Die Buttonfunktion selber sieht so aus:

```
buttonSelected[caption_]:=Module[{nb,cb,cg,mb},
    nb=ButtonNotebook[];
    SelectionMove[nb,Before,Notebook];
    SelectionMove[nb,Next,CellGroup];
    cg=NotebookRead[nb];
    cg=cg/.ButtonBox[b___,Rule[ButtonStyle,_],c___]->
          ButtonBox[b,Rule[ButtonStyle,"Evaluate"],c];
    cg=cg/.ButtonBox[b___,Rule[Background,_],c___]->
          ButtonBox[b,Rule[Background,GrayLevel[0.75]],c];
    cg=cg/.ButtonBox[caption,b___,Rule[ButtonStyle,_],c___]->
          ButtonBox[caption,b,Rule[ButtonStyle,"Hyperlink"],c];
    cg=cg/.ButtonBox[caption,b___,Rule[Background,_],c___]->
          ButtonBox[caption,b,Rule[Background,RGBColor[0.6,1.,0.6]],c];
    NotebookWrite[nb,cg];
    radio$Selection=caption;
    ]
```

Der Ablauf:

1. Einlesen aller Buttons in die Variable `cg`,
2. durch mehrfache Regelanwendung
   `cg=cg/.ButtonBox[b___,...]->ButtonBox[b___,...]`
   werden alle Buttons deselektiert (kenntlich durch `Style->"Evaluate"`, Farbe=Grau),
3. durch mehrfache Regelanwendung
   `cg=cg/.ButtonBox[caption,...]->ButtonBox[caption,...]`
   wird der ausgewählte Button selektiert (`Style->"Hyperlink"`, Farbe=Grün),

4. die Selektion wird in der globalen Variablen `radio$Selection` festgehalten.

B.1.3 Generierung von Notebooks

Die verschiedenen Notebooks für Vorführung, Kontrolle, Fehlermeldung etc. lassen sich genauso einfach erstellen und benutzen wie die Dialoge. Das Notebook *Vorführung* sei als Beispiel dargestellt:

```
vorfuehrNb=Notebook[{Cell[BoxData[
    ButtonBox["Schließen",
      Active->True,
      ButtonStyle->"Evaluate",
      ButtonEvaluator->Automatic,
      ButtonFunction:>(NotebookClose[ButtonNotebook[]]&)]
    ],
    "Input",
    CellTags->"closebutton"]},
    CellGrouping->Manual,
    Visible->False,
    ShowCellBracket->True,
    CellBracketOptions->{"Color"->GrayLevel[1]},
    ScreenStyleEnvironment->Working,
    WindowTitle->"Vorführung"];
```

Hier ist die Liste der Optionen interessant. Da solche Notebooks mehrfach geöffnet werden und der Benutzer gerne vergisst, sie wieder zu schließen, wurde dafür gesorgt, dass offene Notebooks wiederverwendet werden. Dazu wurde die Funktion `NotebookPut` umgeschrieben:

```
notebookRePut[hNb_,notebook_]:=
  Module[{hMargins,nNb,eOpt},
    hMargins=$Failed;
    If[Head[hNb]===NotebookObject,
      hMargins=Options[hNb,WindowMargins];
      NotebookClose[hNb];
    ];
    nNb=NotebookPut[notebook];
    eOpt=Options[nNb,Editable];
    SetOptions[nNb,Editable->True];
    If[hMargins=!=$Failed,
      SetOptions[nNb,#]&/@hMargins
    ];
    SetOptions[nNb,Visible->True];
    SetOptions[nNb,#]&/@eOpt;
    Return[nNb];
    ]
```

Die Variable `hNb` ist ein Handle, das beim vorigen Öffnen eines Notebooks mit `notebookRePut` zurückgegeben wird. Anhand dieses Handles kann die Position des Windows bestimmt und in `hMargins` abgelegt werden. Falls das Fenster

noch nie geöffnet war oder wieder geschlossen wurde – das ist feststellbar anhand des Inhalts von `hNb` bzw. an der Rückgabe `$Failed` für `hMargins` – wird das Notebook einfach neu geöffnet, sonst aber vorher geschlossen. Wenn vorhanden wird die alte Position aus `hMargins` wieder eingenommen. Man beachte den kleinen Trick mit der Option `Visible`: jedes Notebook ist zunächst unsichtbar und wird erst kurz vor Schluss der Funktion sichtbar gemacht, damit der Benutzer eine evtl. Positionsänderung nicht wahrnimmt. Ebenfalls wurde sorgfältig darauf geachtet, dass die Option `Editable` genau den Wert behält, den es vor dem Aufruf dieser Funktion hatte (das ist derjenige, der im vorbereiteten Notebook `notebook` eingetragen ist).

B.2 Realisierung der Übungsteile

In diesem Abschnitt soll nicht im Detail erklärt werden, wie die Übungen, deren Vorführung und Kontrolle realisiert wurden. Dazu muss auf den Quellcode verwiesen werden. Stattdessen soll hier auf spezielle Probleme und deren Lösung eingegangen werden.

Mathematica wendet sich ja an den erfahrenen Anwender, der Ergebnisse in Standardnotation und übersichtlicher Form, möglichst in irgendeiner einfachen Normalform, sehen möchte. Deshalb führt Mathematica viele Umformungen und Vereinfachungen im Hintergrund durch. Im Repetitorium sollen aber ja gerade Zwischenschritte sichtbar werden bzw. Eingaben auf ihre genaue Syntax hin untersucht werden. Die an sich auch im Repetitorium erwünschten Fähigkeiten von Mathematica müssen also manchmal umgangen oder ausgeschaltet werden.

Insbesondere ist es schwierig,

1. die Eingabestruktur von Termen mit ausschließlich Zahlen als Operanden zu erhalten,
2. solche Terme als Ausgabe zu generieren,
3. die genaue Struktur der Klammerung einer Eingabe zu erhalten,
4. Ausgaben mit einer definierten Klammerung zu generieren,

Verschiedene Lösungsmöglichkeiten sind denkbar:

- Die automatische Umformung von Eingaben dadurch zu unterdrücken, dass Funktionen wie `Times` und `Plus` alle Attribute wie `Flatten` und `Listable` entzogen werden.

  Allerdings bleibt dann auch von den erwünschten Fähigkeiten von Mathematica nicht viel übrig.

- Terme als Zeichenketten (*strings*) zu generieren und zu verarbeiten.

  So wurde in den Übungen vorgegangen, in denen mit Zahlen gerechnet wird.

- Verzicht auf die Kenntnis der genauen Termstruktur.

  So wurde in den späteren Übungen, in denen einfachere Umformungen bereits beherrscht werden sollten, vorgegangen.

- Verwendung von `HoldForm` und Varianten davon. Die Funktion `HoldForm` verhindert die Auswertung von Ausdrücken.

Darüber hinaus erwies sich die Überprüfung der Richtigkeit von Benutzereingaben als aufwendig. So erkennt Mathematica z.B. nicht ohne weiteres, dass die Ausdrücke

$$\frac{1}{2}(1+\sqrt{3}) \text{ und } \frac{1}{2}+\frac{1}{2}\sqrt{3}$$

äquivalent sind. Genauer: Es gilt nicht

```
1/2(1+Sqrt[3])===1/2+1/2 Sqrt[3].
```

Ebenso gilt nicht

```
MemberQ[{1/2(1+Sqrt[3])},1/2+1/2 Sqrt[3]]==True.
```

Daher konnten an vielen Stellen die von Mathematica bereitgestellten Funktionen nicht unmittelbar benutzt werden.

B.2.1 Grundrechenarten

Das Hauptproblem bei Übungen mit Zahlenrechnungen besteht darin, dass Mathematica jeden Ausdruck, der nur Zahlen enthält, sofort „ausrechnet“. Deshalb werden alle Rechenaufgaben als Baumstruktur generiert. Die folgende kleine Grammatik zeigt den einfachen Aufbau:

```
aufgBaum := List[operation, operand, operand]

operation := "+" | "-" | "*"

operand := aufgBaum | integer | rational

integer := digits | "(-" <> digits <> ")"

rational := digits <> "/" <> digits

digits := digit digits | (leeres Wort)

digit := 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Ein typisches Beispiel wäre

```
aBaum = {"*", "(-23)", {"-", "2/3", "7"}},
```

welches natürlich für

$$(-23)\cdot(\frac{2}{3}-7)$$

stehen soll.

Ein solcher Aufgabenbaum kann in einen Zahlausdruck umgewandelt werden:

```
BaumZuString[x_String]:=x

BaumZuString[{p_,a_,b_}]:=If[p==="*",
    BaumZuString[a]<>p<>BaumZuString[b],
    "("<>BaumZuString[a]<>p<>BaumZuString[b]<>")"
    ]
```

Im Beispiel wäre

```
BaumZuString[aBaum] = "(-23)*(2/3-7)"
```

Der Aufruf `ToExpression[BaumZuString[aBaum]]` liefert dann die Zahl, die sich bei der Auswertung des Zahlausdrucks ergibt.

Zur Anzeige von Aufgaben für den Benutzer muss der Aufgabenbaum mittels `HoldForm` aufbereitet werden, damit Auswertungen verhindert werden. Gleichzeitig wird eine möglichst korrekte Klammerung angestrebt:

```
BaumZuStringHold[x_]:=
  If[Head[x]===String,
(* then *)
    "HoldForm[" <> x <> "]",
(* else *)
    "HoldForm[(" <>
      BaumZuStringHold[x[[2]]] <>
      x[[1]]<>"HoldForm[" <>
      BaumZuStringHold[x[[3]]] <>
      "])]"]
```

Wieder muss `ToExpression` auf das Resultat angewendet werden, um einen Ausdruck zu erhalten, der angezeigt werden kann. Durch mehrfaches Anwenden von `ReleaseHold` kann der Ausdruck auch ausgewertet werden.

Die Schachtelung von `HoldForm` ist nötig, weil ein Ausdruck wie

```
HoldForm[2*3-4*5+(-3)*(-3-6)]
```

zwar nicht ausgewertet wird, der Rechenausdruck innerhalb des `HoldForm`-Aufrufs aber umgeordnet oder anders geklammert werden kann. Will man solche Details kontrollieren, so muss man innere Teile ebenfalls mit `HoldForm` schützen.

Besonders sorgfältig muss man vorgehen, wenn Aufgaben mit geschachtelten Brüchen wie in Übung 4.3 vorgeführt werden sollen. Dabei müssen außerdem Zähler und Nenner eines Bruches aus den Strings isoliert werden.

B.2.2 Termumformungen

Die Behandlung algebraischer Terme ist weniger problematisch. Mathematica wandelt Terme nicht automatisch in Normalformen um. So werden z.B. Klammerausdrücke nicht ausmultipliziert. Erst die Anwendung von Funktionen wie

`Expand` führt Termumformungen durch. Daher ist die Generierung von Aufgaben unproblematisch und kann durch explizite Vorgabe von Mathematica-Ausdrücken erfolgen.

Größere Sorgfalt erfordert die Prüfung der Eingaben und die Vorführung von Aufgaben. Wenn die genaue Gestalt der Eingabe wichtig ist, wird die Eingabe beim Einlesen mit einer `HoldForm`-Anweisung umhüllt.

Bei der Vorführung von Aufgaben soll es möglich sein, Zwischenschritte sichtbar zu machen. Deshalb sollte eine `Expand`-Anweisung nicht sofort vollständig ausgeführt werden. Oft hilft ebenfalls `HoldForm`, in manchen Fällen muss anders vorgegangen werden. Als Beispiel sei der Code für die schrittweise Faktorisierung eines Terms („Ausklammern“, Übung 2.3) gezeigt:

```
VorfuehrungExpandAufgabe0[aufg_]:=
  Module[{fakts,n,l,letzter,m},
    fakts=FactorList[aufg];
    l=Length[fakts];
    fakts=Apply[Power,fakts,{1}];
    fakts=Sort[fakts,Depth[#1]<Depth[#2]&];
    If[fakts[[-1,0]]===Power&&
        fakts[[-1,2]]===2,n=l,
      n=l-1];
    printLineNb[vNb,"Fortschreitendes Ausklammern
      ergibt nacheinander"];
    Do[printLineNb[
        vNb,(Times@@Take[fakts,{1,m}])(Expand[
              Times@@Take[fakts,{m+1,l}]])],{m,1,n}]]
```

Bei den Vorführungen von Umformungen ergab sich generell das Problem, dass die Terme einer längeren Formel nach internen Sortierkriterien von Mathematica angeordnet werden. Nach einem Umformungsschritt ist dann irritierenderweise die Reihenfolge der Terme anders als erwartet. Beispielsweise wird aus

$$(a+x)(x+y)$$

nach dem Ausmultiplizieren mit `Expand`

$$ax+ay+xy+x^2,$$

anders, als dies ein menschlicher Schreiber tun würde. Auf die Sortierreihenfolge von Mathematica kann man keinen Einfluss nehmen. Die gewünschte Reihenfolge wäre nur durch massiven Einsatz von `HoldForm` zu erzwingen. Da damit behandelte Terme aber nur schwer weiterverarbeitet werden können, wurde davon Abstand genommen. Dieser didaktisch unschöne Umstand bleibt also unbefriedigend.

B.3 Fehlererkennung und -behandlung

Erklärtes Ziel des Repetitoriums ist es, die Antworten des Benutzers nicht nur mit „richtig“ oder „falsch“ zu bewerten, sondern auch Hinweise auf den

gemachten Fehler zu geben. Diesem Ziel sind dadurch Grenzen gesetzt, dass die meisten Fehler schlicht Rechenfehler sein dürften und nur die wenigsten auf einem expliziten Verstoß gegen Rechenregeln beruhen werden. Reine Rechenfehler wie $2 + 2 = 5$ oder Ansammlungen von Unachtsamkeiten wie in $(2a+7b)(6a^2+9b) = 12a^3+42ab+27ab+63b^2$ ergeben so viele Möglichkeiten von falschen Ergebnissen, dass eine Analyse des Fehlers in vielen Fällen nicht möglich ist. Es wird deshalb vor allem nach Fehlern durch Regelverstöße ohne zusätzliche Flüchtigkeitsfehler gesucht. Weiterhin sollte das Programm möglichst tolerant gegenüber sytaktischen Fehlern bei der Eingabe sein.

B.3.1 Fehler bei der Eingabe

Das Repetitorium, welches ja vor allem für Benutzer gedacht ist, welche nicht im Umgang mit Mathematica geübt sind, muss Fehler bei der Eingabe möglichst gut tolerieren. Der Benutzer wird ja im Dialog geführt, die Zellen der Notebooks sind schreibgeschützt, so dass Fehler nur bei der Eingabe von Antworten in die vorgesehenen Eingabefelder auftreten können. Man kann zwei Arten falscher Eingaben unterscheiden:

- Eingaben, die nicht zur aktuellen Übung passen, ansonsten aber korrekt sind, und
- Eingaben, die syntaktische Fehler enthalten und keine gültigen Mathematica-Ausdrücke darstellen.

Die erste Art von falschen Eingaben wird durch die jeweiligen Übungsteile abgefangen. Im allgemeinen erhält der Benutzer einen Hinweis darauf, welche Antworten erwartet werden, und wenn möglich, wird ein direkter Hinweis darauf gegeben, was nicht passend ist. Für den Zweck der Untersuchung der Eingaben stehen innerhalb Mathematica genügend viele Funktionen bereit, z.B.

NumberQ zum Test, ob eine Eingabe numerisch ist;

PolynomialQ zum Test, ob ein Polynom eingeben wurde;

UserSymbols – eine einfache Funktion, die nicht zum Standardvorrat von Mathematica gehört – liefert alle Variablen eines Ausdrucks[2];

Cases mit geeigneten Mustern kann z.B. alle Wurzelexponenten eines Ausdrucks liefern und so die Eingabe von dritten Wurzeln erkennen und ablehnen;

...

Mit ihrer Hilfe sind die nötigen Prüfungen relativ leicht zu erledigen und beliebig sorgfältig durchzuführen.

Schwieriger zu behandeln sind Syntaxfehler. Mathematica reagiert nämlich auf den Versuch, einen fehlerhaften Ausdruck einzulesen, immer mit einer Fehlermeldung, die das optische Bild eines Dialogs stört. Es wurde folgendes Verfahren angewendet:

[2] siehe [MA01]

1. Eingabezellen werden zunächst per `NotebookRead` eingelesen. Man erhält eine Zeichenkette, die die Boxstruktur der eingelesenen Zelle enthält. Diese Boxstruktur an sich ist stets syntaktisch korrekt, ist aber kein auswertbarer Ausdruck.

2. Die Funktion `ToExpression` versucht, eine Zeichenkette in einen Mathematica-Ausdruck umzuwandeln. Genau hier treten unvermeidbar Fehlermeldungen auf. Deshalb werden unmittelbar vor dem Aufruf von `ToExpression` alle Fehlermeldungen aus-, und danach wieder eingeschaltet.

3. Da im Fehlerfalle der Rückgabewert auf das spezielle Symbol `$Failed` gesetzt ist, läßt sich eine syntaktisch falsche Eingabe danach zweifelsfrei erkennen und eine entsprechende Meldung ausgeben.

Die Funktion `InputExpressionHold` führt diesen Ablauf durch:

```
InputExpressionHold[tag_]:=
  Module[{input},
    input=readFromCell[tag];
    messagesOff;
    input=ToExpression[input,TraditionalForm,HoldForm];
    messagesOn;
    If[input===$Failed,
      printMessage["Syntaxfehler, bitte Eingabe korrigieren!"];
      Return[False]];
    If[input=!={}&&input[[1,0]]==Set,input[[1,0]]=Equal];
    expr=ReleaseHold[input];
    If[expr==ComplexInfinity||expr==Indeterminate,expr=$Failed;
      printMessage["Division durch Null ist nicht erlaubt!"];
      Return[False]];
    Return[input]
    ]
```

In dieser Funktion sind noch weitere Dinge miterledigt worden:

- Durch den dritten Parameter `HoldForm` im Aufruf von `ToExpression` wird erreicht, dass der erhaltene Ausdruck nicht sofort ausgewertet wird. Dadurch wird eine genaue Untersuchung des eingegebenen Terms möglich. Andernfalls erhielte man eine Normalform desselben, die manche Prüfung unmöglich macht.

- Die Zeile

  ```
  If[input=!={}&&input[[1,0]]==Set,input[[1,0]]=Equal];
  ```

  ermöglicht es, einen typischen Eingabefehler, nämlich die Eingabe einer Bestimmungsgleichung mit dem einfachen Gleichheitszeichen „=“ statt korrekt mit dem doppelten „==“, für den Benutzer folgenlos zu lassen. Eine Eingabe der Form `a+x=0` wird nämlich zunächst als `HoldForm[Set[a+x,0]]` eingelesen. Durch die obige Zeile wird die falsche Zuweisung `Set` durch den richtigen Vergleich auf Gleichheit `Equal` ersetzt, und es wird `a+x==0` weiterverarbeitet, ohne dass der Benutzer irgendetwas davon bemerkt.

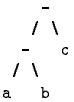
- Eingaben, die bei ihrer Auswertung auf eine Division durch Null – und damit zu unschönen Fehlermeldungen – führen würden, werden ebenfalls abgefangen.

In ähnlicher Weise wurde der Fall behandelt, dass die Lösungsmenge einer Gleichung, insbesondere wenn es nur eine Lösung gibt, ohne Mengenklammern eingegeben wurde. Dies dürfte auch ein typischer Eingabefehler sein. Hier werden die Klammern automatisch ergänzt und es wird ein Hinweis darauf gegeben. Leider können die Klammern nicht automatisch ergänzt werden, wenn es mehrere Lösungen gibt und der Benutzer z.B. statt „$\{2,3\}$“ nur „$2,3$“ eingibt. Die Eingabe „$2,3$“ wird nämlich von Mathematica als unvollständige Eingabe angesehen und wird auf keine Weise von `ToExpression` akzeptiert. Auf die mögliche Untersuchung der Boxstruktur daraufhin, ob sie eine Liste von durch Kommas getrennten Zahlen enthält, wurde verzichtet, da Boxstrukturen sehr unübersichtlich sein können.

B.3.2 Fehlerhafte Antworten

Ein von Anfängern häufig gemachter Fehler ist die Umformung

$$a - (b - c) = a - b - c.$$

Stellt man Terme als Bäume dar, so sieht man, dass hier der Baum  fälschlich durch den Baum  ersetzt wird. Diese beiden Bäume gehen durch eine sog. *Linksrotation* auseinander hervor. Viele typische Fehler lassen sich auf eine Linksrotation oder die analoge Rechtsrotation des Termbaumes zurückführen.

Eine Linksrotation ist in Mathematica leicht direkt zu programmieren:

```
LinksRotation[t_]:=Apply[t[[2,0]],
       {Apply[t[[0]],{t[[1]],t[[2,1]]}],
        t[[2,2]]}]
```

Besser und übersichtlicher ist es, die Linksrotation als Ersetzungsregel zu formulieren:

```
LinksRotationsRegel=p_[a_,q_[b_,c_]]->q[p[a,b],c]
```

Man wendet diese beiden Möglichkeiten wie folgt an:

```
In[]  = LinksRotation[a-(b-c)]
Out[] = a-b-c
```

bzw.

```
In[]  = a-(b-c) /. LinksRotationsregel
Out[] = a-b-c
```

Die beiden Varianten wirken in gleicher Weise auf *alle* Ausdrücke, die entweder entsprechend viele Teile haben oder auf das Muster passen. Das ist aber manchmal nicht der Fall:

```
In[]  = a-(b-c)+d /. LinksRotationsregel
Out[] = a-(b-c)+d
```

Die Ursache liegt in der Eigenschaft *Listable* von Symbolen wie *Plus*, die dazu führen, dass `a-(b-c)+d` die interne Darstellung

```
Plus[a,Times[-1,Plus[b,Times[-1,c]]],d]
```

hat, auf welche das Muster `p_[a_,_]` nicht passt. Eine gewisse Abhilfe bietet

```
LinksRotationsRegel=p_[a_,q_[b_,c_,r___],s___]->q[p[a,b,r],c,s]
```

mit den Platzhaltern `r___` und `s___` für beliebig viele oder auch keine weiteren Parameter.

Alternativ läßt sich der obige Fehler durch eine spezielle Regel für genau diesen Fall darstellen:

```
a_-(b_-c_) -> a-b-c
```

Wenn ein Benutzer den Term `a-(b-c)` fälschlich in `a-b-c` umgeformt hat, so kann man entweder die Linksrotation oder die Regel auf den Term `a-(b-c)` anwenden, das Ergebnis evaluieren und mit der Eingabe `a-b-c` vergleichen. Bei Übereinstimmung ist der Fehler erkannt und ein entsprechender Hinweis kann ausgegeben werden.

Die Rotationsregeln sind, da sie auf alle möglichen Ausdrücke passen, in manchen Fällen zu allgemein und ergeben „Fehler"-Terme, die weit jenseits dessen liegen, was ein Benutzer an falschen Eingaben produzieren würde:

```
In[] =  a-b /. LinksRotationsRegel
Out[] = (a-1)b
```

Zur Erklärung bedenke man, dass $a - b$ intern als `Plus[a,Times[-1,b]]` dargestellt wird.

In dieser Arbeit wurden beide Ansätze weiterverfolgt. Für das Ausmultiplizieren von Termen (Übung 3.1) wurden z.B. folgende Regeln verwendet:

```
x_ (y_+z_)          ->    x y - x z,
x_ (y_+z_)          ->   -x y - x z,
(a_+b_) (c_+d_) e_. ->  ( a c + b d)e,
(a_+b_) (c_+d_) e_. ->  ( a d + b c + b d)e,
(a_+b_) (c_+d_) e_. ->  (-a c + a d + b c + b d)e,
(a_+b_) (c_+d_) e_. ->  (-a c - a d + b c + b d)e,
(a_+b_) (c_+d_) e_. ->  (-a c - a d - b c + b d)e,
(a_+b_) (c_+d_) e_. ->  (-a c + a d + b c - b d)e,
a_ (b_+c_)          ->    a b + c
```

Die linken Seiten der Regeln sind stets für die Addition formuliert und nicht für die Subtraktion, was ja eigentlich näher liegen würde. Weil Mathematica grundsätzlich die Subtraktion auf die Multiplikation mit -1 und anschließende Addition zurückführt, decken die Regeln so alle gewünschten und noch weitere Fälle zusätzlich ab.

Die Regeln decken auch unvollständiges Ausmultiplizieren ab (3., 4. und letzte Regel).

Für die Rechenoperationen mit Zahlen (Übung 2.4) wurden hingegen Rotationsregeln eingesetzt:

```
{q_,a_,{p_,b_,c_}}          ->  {p,{q,a,b},c}
{p_,{q_,a_,b_},c_}          ->  {q,a,{p,b,c}}
{r_,a_,{q_,b_,{p_,c_,d_}}}  ->  {p,{q,{r,a,b},c},d}
{{p_,{q_,{r_,a_,b_},c_},d_} ->  {r,a,{q,b,{p,c,d}}}
```

Diese Regeln beschreiben sukzessive eine Links-, eine Rechts-, eine doppelte Links- und eine doppelte Rechtsrotation. Dass hier die Ausdrücke als geschachtelte Listen formuliert sind liegt daran, dass für die Rechenoperationen mit Zahlen – wie weiter oben beschrieben – Ausdrücke mit Zahlen explizit als Termbäume modelliert wurden, um deren Evaluation durch Mathematica kontrollieren zu können.

Sämtliche obigen Regeln scheinen das vorhin angesprochene Problem mit der *Listable*-Eigenschaft nicht zu beachten, da es keine Platzhalter wie `r___` gibt. In Wirklichkeit tritt dieses Problem deshalb nicht auf, weil die Regeln stets auf Teilterme angewendet werden, die eben nicht zu viele Argumente enthalten. Solche Teilterme können mit der Funktion `Position` gefunden werden:

```
In[] =  Position[a(b+c)+d(e+f),x_+y_]
Out[] = {{},{1,2},{2,2}}
```

Anschaulicher ist die Funktion `Cases`, die die gefundenen Teilterme zurückgibt:

```
In[] =  Cases[a(b+c)+d(e+f),x_+y_]
Out[] = {a(b+c)+d(e+f),b+c,e+f}
```

Seit der Version 3.0 von Mathematica ist es auch möglich, nicht nur die erste mögliche, sondern alle Belegungen eines Musters zu finden, die zu einem Term passen:

```
In[] =  (a+b)(c+d) /. x_(y_+z_)->{x,y,z}
Out[] = {a+b,c,d}

In[] =  ReplaceList[(a+b)(c+d), x_(y_+z_)->{x,y,z}]
Out[] = {{a+b,c,d},
         {a+b,d,c},
         {c+d,a,b},
         {c+d,b,a}}
```

Erst `ReplaceList` in Verbindung mit `Cases` und `Position` macht eine vollständige Suche nach falschen Termumformungen mit Regeln möglich.

Rechenfehler, die sich nur in falschen Koeffizienten äußern, lassen sich kaum sinnvoll durch Regeln abdecken, da man sich ja auf alle erdenklichen Arten „verrechnen“ kann. Es ist aber über Funktionen wie `CoefficientList`, die die Koeffizienten eines Polynoms – auch in mehreren Veränderlichen – zurückgibt, wenigstens zu prüfen, ob alle erwarteten Terme überhaupt vorhanden sind. Dann kann immerhin noch der allgemeine Hinweis auf einen Rechenfehler gegeben werden.

B.4 Generierung von zufälligen Aufgaben

Es wurde besonderer Wert darauf gelegt, den Benutzer mit einer reichlichen und vielfältigen Menge von Aufgaben zu versorgen, die sich nur sehr selten wiederholen und alle denkbaren Sonderfälle mit enthalten sollen.

B.4.1 Termskelette

Da sich jeder algebraische Term – auch Gleichungen – als (binärer) Baum mit Operationen in den Knoten und Operanden in den Blättern darstellen läßt, ist die naheliegendste Idee, einen Aufgabentyp durch Bedingungen an den Baum zu charakterisieren und dann zufällig Bäume, die diesen Bedingungen genügen, zu generieren. Die Programmiersprache von Mathematica bietet gute Möglichkeiten zur Listenverarbeitung – ähnlich wie in LISP. Es ist daher kein Problem, Terme als Baum zu generieren und unmittelbar in Mathematica weiter zu verwenden.

Für diesen Prototyp wurde ein anderer Weg eingeschlagen: Für jeden Aufgabentyp wurden mehrere Untertypen als Termskelett in einer Liste abgelegt. Die Operanden und evtl. auch die Operationen in diesem Termskelett sind durch Variable belegt, hinter denen sich eine Funktion zur zufälligen Generierung eines Operanden oder einer Operation verbirgt. Typische Vertreter sind etwa

```
RandomSign:=2Random[Integer]-1
ZahlOderNull:=RandomSign Random[Integer,10]
positiveZahl:=Random[Integer,{1,10}]
Zahl:=RandomSign positiveZahl
linearerTerm:=Zahl x+ZahlOderNull
positiverLinearerTerm:=positiveZahl x + ZahlOderNull
Wurzelterm:=Sqrt[positiverLinearerTerm]
```

Diese Variablen können z.B. als Termskelett

```
Wurzelterm+linearerTerm==linearerTerm,
```

für Wurzelgleichungen verwendet werden.

Diese Methode eignet sich auch für jene Übungsteile, die Aufgaben mit Zahlen enthalten. Die dort verwendeten Listen mit Baumstrukturen für Ausdrücke

lassen sich auf die gleiche Weise mit zufälligen Inhalten bei vorgegebener Termstruktur füllen.

Die Erzeugung von zufälligen Termen über Zufallsbäume von Ausdrücken wurde nicht weiter verfolgt. Die gestellten Aufgaben dürfen nämlich nicht zu umfangreich sein, wenn der Benutzer sich nicht in zu langwierigen Rechnungen verstricken soll. Für Aufgaben, die eine gewisse Länge nicht überschreiten, lassen sich aber die möglichen Terme relativ rasch vollständig aufzählen und in einer Liste von Termskeletten ablegen. Man gewinnt dadurch die Möglichkeit, jeden Term auf eine auf ihn speziell zugeschnittene Weise zu behandeln. Dieser Weg der individuellen Behandlung wurde z.B. für Potenzaufgaben mit Zahlen gewählt. Damit die Potenzgesetze sinnvoll anwendbar sind, muss nicht nur der Term die richtige Struktur haben, er muss auch mit passenden Zahlen gefüllt werden, damit interessante Aufgaben entstehen. Jeder Termtyp hat außerdem seinen eigenen spezifischen Lösungsweg. Dies alles würde bei zufällig gewählten Termstrukturen erheblich aufwendiger zu realisieren sein.

B.4.2 Glatte Lösungen

Manche Aufgabentypen können bei der Lösung schnell zu großen Zahlen und komplizierten Lösungen führen. Dies gilt besonders für Gleichungen 2. Grades, vor allem dann, wenn sie noch Wurzelterme enthalten. Der Lerneffekt ist nicht besonders hoch, wenn der Benutzer sich mit Lösungsmengen wie

$$x \in \left\{ \frac{29}{33} + \sqrt{\frac{1258}{363}}, \frac{29}{33} - \sqrt{\frac{1258}{363}} \right\}$$

herumschlagen muss. Es ist deshalb sinnvoll, die Gleichungen so auszuwählen, dass glatte, also ganzzahlige Lösungen, oder zumindest nur Brüche mit kleinen Nennern vorkommen. Eine Gleichung auf eine bestimmte Lösung hin zu konstruieren ist nicht ganz einfach und macht eine Zufallsauswahl schwierig. Deshalb wurde ein anderer Ansatz implementiert, der darin besteht, eine zufällig ausgewählte Gleichung im Nachhinein so zu verändern, dass eine ganzzahlige Lösung herauskommt. Zufällige Gleichungen haben meist irrationale Lösungen. Eine davon kann man auf eine ganze Zahl runden und in die Gleichung einsetzen. Dann kommt nicht mehr Null heraus, sondern etwas anderes. Addiert man diese Zahl zur rechten Seite der Gleichung, so hat man was man will. Die Funktion `GlatteLoesung` leistet dies:

```
GlatteLoesung[aufg_]:=
  Module[{xs,korr,nenner},
    xs=Solve[aufg];
    If[xs===={},Return[aufg]];
    xs=x/.(xs//N);
    xs=xs[[1]];
    If[Head[xs]===Complex,Return[aufg]];
    If[Head[xs]===Rational,Return[aufg]];
    If[Head[xs]===Integer,Return[aufg]];
    xs=xs//Round;
    nenner=Denominator[Subtract@@aufg[[1]]//Together];
```

```
While[nenner/.x->xs==0,xs+=1];
korr=(Subtract@@aufg[[1]])/.x->xs;
ReplacePart[aufg,aufg[[1,2]]+korr,{1,2}]
]
```

Hierin ist `aufg` eine zuvor zufällig erzeugte Gleichung[3]. Diese wird numerisch gelöst, eine Lösung gerundet, der nötige Summand für die rechte Seite (in der Variablen `korr`) bestimmt und mit `ReplacePart` in die Gleichung eingefügt. Etliche Sonderfälle müssen beachtet werden: falls die Gleichung keine Lösung hat oder diese komplex ist, kann das Verfahren nicht angewendet werden. Das Verfahren ist andererseits nicht nötig, wenn eine Lösung bereits ganzzahlig oder rational ist. Die Polstellen bei Aufgaben mit rationalen Funktionen müssen besonders beachtet werden: die gerundete Lösung darf nicht auf einen Pol zu liegen kommen. Hier wird in der `While`-Schleife die gewünschte Lösung so lange um 1 erhöht, bis man nicht mehr auf einer Polstelle steht.

Gleichungen mit Wurzeltermen können leider nicht so behandelt werden, weil die Korrektursummanden im allgemeinen selbst irrational sind. Die geänderte Gleichung hat dann meist einen wesentlich komplizierteren Lösungsweg, der sich nicht mehr automatisch vorführen läßt. Deshalb wurde hier doch versucht, Aufgaben mit einer vorgegebenen Lösung zu konstruieren. Zu diesem Zweck wurde das weiter oben genannte einfache Termskelett für Wurzelterme anders gestaltet:

```
Wurzelterm[xs_] :=
  Module[{a, c},
    If[Head[xs] === Rational,
      a = 2 kleineZahl,
      a = positiveZahl;
      c = kleineZahl;
      Sqrt[a x + c^2 - a xs]
  ]
```

Hier ist `xs` die gewünschte Lösung, die auch halbzahlig sein kann. Es wird ein zufälliger Wurzelterm zurückgeliefert, der die kleine ganze Zahl `c` als Wert hat, wenn man $x = xs$ setzt. Die entstehende zufällige Gleichung muss am Ende noch korrigiert werden, damit auch `xs` Lösung wird, denn durch `Wurzelterm` sind nur ganzrationale Wurzeln garantiert.

Allerdings gestaltet sich die Korrektur der Gleichung etwas schwieriger, da dafür Sorge getragen werden muss, dass die neue Gleichung höchstens vom 2. Grade ist. Weiterhin stellt sich heraus, dass die erzeugte Gleichung zwar eine einfache Lösung besitzt (die vorgegebene), die zweite Lösung ist jedoch oft ein Bruch mit großem Nenner, und bei der Lösung der Gleichung treten große Zahlen auf.

[3] eingehüllt in eine Liste

Anhang C

Fazit

Die Arbeit am Repetitorium kann sicher noch beliebig weitergeführt werden. So könnten weitere Aufgabentypen aufgenommen werden. Eine Erweiterung des Stoffes auf Themen aus der Analysis wäre denkbar. In diesem Bereich gibt es allerdings schon ansprechende Tutorials, auch auf der Basis von Mathematica. Unter der Internetadresse [UIUC] findet sich ein Programm, welches schrittweise Funktionen differenziert. Auf ähnliche Weise wird dort die Berechnung unbestimmter Integrale vorgeführt, dieser Teil - und einige andere - ist allerdings gebührenpflichtig.

Die Verwendung von Mathematica für dieses Tutorial hat sich letztlich doch bewährt. Die Notebooks bieten genügend Komfort für den Benutzer, Dialoge lassen sich damit recht bequem erstellen, sobald die grundlegenden Aufgaben für Menüsteuerung und Dialoggestaltung gelöst sind. Positiv zu bemerken ist auch die Möglichkeit des schnellen Prototyping. Bequem ist das Testen von neuen Funktionen: bei laufendem Repetitorium können neue Funktionen interaktiv erstellt, getestet und ins Repetitoriumsprogramm eingestellt werden. Das reibungslose Zusammenspiel ist sofort erkennbar. Allerdings: die häufigen, aus heiterem Himmel stattfindenden Abstürze in der Testphase – meist im Zusammenhang mit Aktionen in den generierten Notebooks – konnten sehr lästig werden.

Die bereits an früherer Stelle erwähnten Schwierigkeiten mit der automatischen Evaluierung von Ausdrücken lassen sich letztlich doch gut in den Griff bekommen. Es ist nicht notwendig, auf direkte Programmierung in anderen Programmiersprachen zurückzugreifen.

Letztlich erweist sich der Funktionsumfang von Mathematica als so groß und reichhaltig, dass sich für jedes auftretende Problem eine Lösung finden ließ, und oft sogar eine recht elegante.

Literaturverzeichnis

[GU92] Charlie Gunn: Remarks on Mathematical Courseware. in: S. Cunningham, R. J. Hubbold [Eds.]: Interactive Learning Through Visualization. Berlin 1992, p. 115-127

[MA01] Emily Martin: Mathematica 4.1, Standard Add-on Packages. Cambridge 2001

[MA99] Kurt Marti, Detlef Groeger: Brückenkurs Mathematik. Wittenberg 1999

[TA92] Jonathan P. Taylor: Prospero, A System for Representing the Lazy Evaluation of Functions. in: S. Cunningham, R. J. Hubbold [Eds.]: Interactive Learning Through Visualization. Berlin 1992, p. 145-157

[WO99] Stephen Wolfram: The Mathematica Book. Cambridge 1999

[UIUC] http://mtl.math.uiuc.edu/classroom_resources.htm

In der Reihe FINAL sind bisher erschienen:

1. Jahrgang 1991:

1. Hinrich E. G. Bonin; Softwaretechnik, Heft 1, 1991 (ersetzt durch Heft 2, 1992).
2. Hinrich E. G. Bonin (Herausgeber); Konturen der Verwaltungsinformatik, Heft 2, 1991 (überarbeitet und erschienen im Wissenschaftsverlag, Bibliographisches Institut & F. A. Brockhaus AG, Mannheim 1992, ISBN 3-411-15671-6).

2. Jahrgang 1992:

1. Hinrich E. G. Bonin; Produktionshilfen zur Softwaretechnik --- Computer-Aided Software Engineering --- CASE, Materialien zum Seminar 1992, Heft 1, 1992.
2. Hinrich E. G. Bonin; Arbeitstechniken für die Softwareentwicklung, Heft 2, 1992 (3. überarbeitete Auflage Februar 1994), PDF-Format (Passwort: arbeiten).
3. Hinrich E. G. Bonin; Object-Orientedness --- a New Boxologie, Heft 3, 1992.
4. Hinrich E. G. Bonin; Objekt-orientierte Analyse, Entwurf und Programmierung, Materialien zum Seminar 1992, Heft 4, 1992.
5. Hinrich E. G. Bonin; Kooperative Produktion von Dokumenten, Materialien zum Seminar 1992, Heft 5, 1992.

3. Jahrgang 1993:

1. Hinrich E. G. Bonin; Systems Engineering in Public Administration, Proceedings IFIP TC8/ WG8.5: Governmental and Municipal Information Systems, March 3--5, 1993, Lüneburg, Heft 1, 1993 (überarbeitet und erschienen bei North-Holland, IFIP Transactions A-36, ISSN 0926-5473).
2. Antje Binder, Ralf Linhart, Jürgen Schultz, Frank Sperschneider, Thomas True, Bernd Willenbockel; COTEXT --- ein Prototyp für die kooperative Produktion von Dokumenten, 19. März 1993, Heft 2, 1993.
3. Gareth Harries; An Introduction to Artificial Intelligence, April 1993, Heft 3, 1993.
4. Jens Benecke, Jürgen Grothmann, Mark Hilmer, Manfred Hölzen, Heiko Köster, Peter Mattfeld, Andre Peters, Harald Weiss; ConFusion --- Das Produkt des AWÖ-Projektes 1992/93, 1. August 1993, Heft 4, 1993.
5. Hinrich E. G. Bonin; The Joy of Computer Science --- Skript zur Vorlesung EDV ---, September 1993, Heft 5, 1993 (4. ergänzte Auflage März 1995).
6. Hans-Joachim Blanke; UNIX to UNIX Copy --- Interactive application for installation and configuration of UUCP ---, Oktober 1993, Heft 6, 1993.

4. Jahrgang 1994:

1. Andre Peters, Harald Weiss; COMO 1.0 --- Programmierumgebung für die Sprache COBOL --- Benutzerhandbuch, Februar 1994, Heft 1, 1994.

2. Manfred Hölzen; UNIX-Mail --- Schnelleinstieg und Handbuch ---, März 1994, Heft 2, 1994.
3. Norbert Kröger, Roland Seen; EBrain --- Documentation of the 1994 AWÖ-Project Prototype ---, June 11, 1994, Heft 3, 1994.
4. Dirk Mayer, Rainer Saalfeld; ADLATUS --- Documentation of the 1994 AWÖ-Project Prototype -- -, July 26, 1994, Heft 4, 1994.
5. Ulrich Hoffmann; Datenverarbeitungssystem 1, September 1994, Heft 5, 1994. (2. überarbeitete Auflage Dezember 1994)
6. Karl Goede; EDV-gestützte Kommunikation und Hochschulorganisation, Oktober 1994, Heft 6 (Teil 1), 1994.
7. Ulrich Hoffmann; Zur Situation der Informatik, Oktober 1994, Heft 6 (Teil 2), 1994.

5. Jahrgang 1995:

1. Horst Meyer-Wachsmuth; Systemprogrammierung 1, Januar 1995, Heft 1, 1995.
2. Ulrich Hoffmann; Datenverarbeitungssystem 2, Februar 1995, Heft 2, 1995.
3. Michael Guder / Kersten Kalischefski / Jörg Meier / Ralf Stöver / Cheikh Zeine; OFFICE-LINK --- Das Produkt des AWÖ-Projektes 1994/95, März 1995, Heft 3, 1995.
4. Dieter Riebesehl; Lineare Optimierung und Operations Research, März 1995, Heft 4, 1995.
5. Jürgen Mattern / Mark Hilmer; Sicherheitsrahmen einer UTM-Anwendung, April 1995, Heft 5, 1995.
6. Hinrich E. G. Bonin; Publizieren im World-Wide Web --- HyperText Markup Language und die Kunst der Programmierung ---, Mai 1995, Heft 6, 1995
7. Dieter Riebesehl; Einführung in Grundlagen der theoretischen Informatik, Juli 1995, Heft 7, 1995
8. Jürgen Jacobs; Anwendungsprogrammierung mit Embedded-SQL, August 1995, Heft 8, 1995
9. Ulrich Hoffmann; Systemnahe Programmierung, September 1995, Heft 9, 1995 (ersetzt durch Heft 1, 1999).
10. Klaus Lindner; Neuere statistische Ergebnisse, Dezember 1995, Heft 10, 1995

6. Jahrgang 1996:

1. Jürgen Jacobs / Dieter Riebesehl; Computergestütztes Repetitorium der Elementarmathematik, Februar 1996, Heft 1, 1996
2. Hinrich E. G. Bonin; "Schlanker Staat" & Informatik, März 1996, Heft 2, 1996
3. Jürgen Jacobs; Datenmodellierung mit dem Entity-Relationship-Ansatz, Mai 1996, Heft 3, 1996
4. Ulrich Hoffmann; Systemnahe Programmierung, (2. überarbeitete Auflage von Heft 9, 1995), September 1996, Heft 4, 1996 (ersetzt durch Heft 1, 1999).
5. Dieter Riebesehl; Prolog und relationale Datenbanken als Grundlagen zur Implementierung einer NF2-Datenbank (Sommer 1995), November 1996, Heft 5, 1996

7. Jahrgang 1997:

1. Jan Binge, Hinrich E. G. Bonin, Volker Neumann, Ingo Stadtsholte, Jürgen Utz; Intranet-/Internet- Technologie für die Öffentliche Verwaltung --- Das AÖW-Projekt im WS96/97 --- (Anwendungen in der Öffentlichen Verwaltung), Februar 1997, Heft 1, 1997
2. Hinrich E. G. Bonin; Auswirkungen des Java-Konzeptes für Verwaltungen, FTVI'97, Oktober 1997, Heft 2, 1997

8. Jahrgang 1998:

1. Hinrich E. G. Bonin; Der Java-Coach, Oktober 1998, Heft 1, 1998 (CD-ROM, PDF-Format; aktuelle Fassung)
2. Hinrich E. G. Bonin (Hrsg.); Anwendungsentwicklung WS 1997/98 --- Programmierbeispiele in COBOL & Java mit Oracle, Dokumentation in HTML und tcl/tk, September 1998, Heft 2, 1998 (CD-ROM)
3. Hinrich E. G. Bonin (Hrsg); Anwendungsentwicklung SS 1998 --- Innovator, SNiFF+, Java, Tools, Oktober 1998, Heft 3, 1998 (CD-ROM)
4. Hinrich E. G. Bonin (Hrsg); Anwendungsentwicklung WS 1998 --- Innovator, SNiFF+, Java, Mail und andere Tools, November 1998, Heft 4, 1998 (CD-ROM)
5. Hinrich E. G. Bonin; Persistente Objekte --- Der Elchtest für ein Java-Programm, Dezember 1998, Heft 5, 1998 (CD-ROM)

9. Jahrgang 1999:

1. Ulrich Hoffmann; Systemnahe Programmierung (3. überarbeitete Auflage von Heft 9, 1995), Juli 1999, Heft 1, 1999 (CD-ROM und Papierform), Postscript-Format, zip-Postscript-Format, PDF- Format und zip-PDF-Format.

10. Jahrgang 2000:

1. Hinrich E. G. Bonin; Citizen Relationship Management, September 2000, Heft 1, 2000 (CD-ROM und Papierform), PDF-Format --- Password: arbeiten
2. Hinrich E. G. Bonin; WI>DATA --- Eine Einführung in die Wirtschaftsinformatik auf der Basis der Web_Technologie, September 2000, Heft 2, 2000 (CD-ROM und Papierform), PDF-Format --- Password: arbeiten
3. Ulrich Hoffmann; Angewandte Komplexitätstheorie, November 2000, Heft 3, 2000 (CD-ROM und Papierform), PDF-Format
4. Hinrich E. G. Bonin; Der kleine XMLer, Dezember 2000, Heft 4, 2000 (CD-ROM und Papierform), PDF-Format, aktuelle Fassung --- Password: arbeiten

11. Jahrgang 2001:

1. Hinrich E. G. Bonin (Hrsg.): 4. SAP-Anwenderforum der FHNON, März 2001, (CD-ROM und Papierform), Downloads & Videos.
2. J. Jacobs / G. Weinrich; Bonitätsklassifikation kleiner Unternehmen mit multivariater linear Diskriminanzanalyse und Neuronalen Netzen; Mai 2001, Heft 2, 2001, (CD-ROM und Papierform), PDF-Format und MS Word DOC-Format --- Password: arbeiten
3. K. Lindner; Simultanttestprozedur für globale Nullhypothesen bei beliebiger Abhängigkeitsstruktur der Einzeltests, September 2001, Heft 3, 2001 (CD-ROM und Papierform).

12. Jahrgang 2002:

1. Hinrich E. G. Bonin: Aspect-Oriented Software Development. März 2002, Heft 1, 2002 (CD-ROM und Papierform), PDF-Format --- Password: arbeiten.
2. Hinrich E. G. Bonin: WAP & WML --- Das Projekt Jagdzeit ---. April 2002, Heft 2, 2002 (CD-ROM und Papierform), PDF-Format --- Password: arbeiten.
3. Ulrich Hoffmann; Ausgewählte Kapitel der Theoretischen Informatik. August 2002, Heft 3, 2002 (CD-ROM und Papierform), PDF-Format ---

Herausgeber:

Prof. Dr. Dipl.-Ing. Dipl.-Wirtsch.-Ing. Hinrich E. G. Bonin Fachhochschule Nordostniedersachsen (FH NON), Volgershall 1, D-21339 Lüneburg, email: bonin@fhnon.de

Verlag:

Eigenverlag (Fotographische Vervielfältigung), FH NON

Erscheinungsweise:

ca. 4 Hefte pro Jahr Für unverlangt eingesendete Manuskripte wird nicht gehaftet. Sie sind aber willkommen.