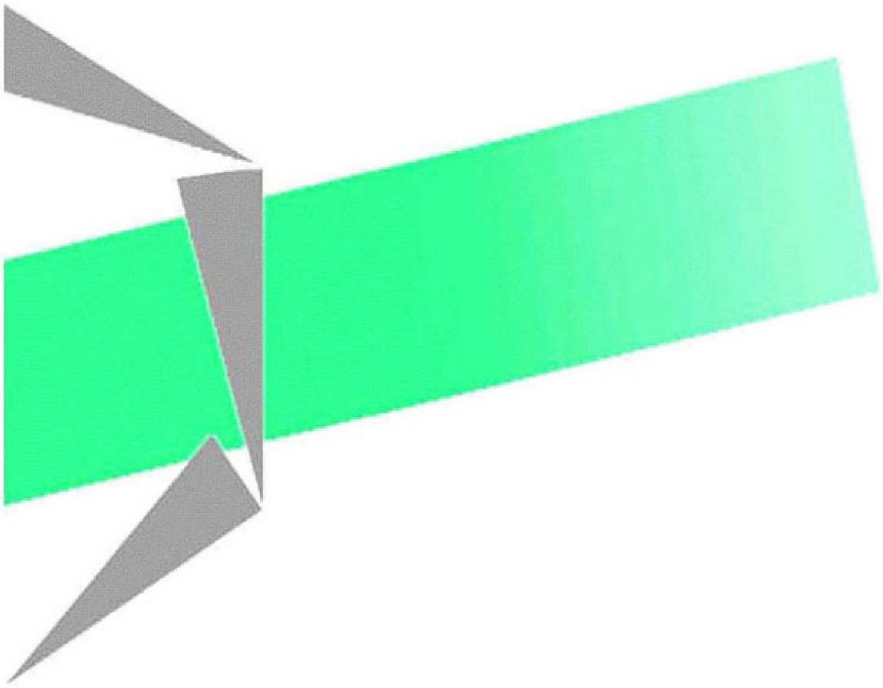
Natural Editing of Algebraic Expressions

Jean-François Nicaud, Denis Bouhineau

Les cahiers
Leibniz



Natural Editing of Algebraic Expressions

Jean-François Nicaud and Denis Bouhineau

Laboratoire G-SCOP
46 av. Félix Viallet, 38000 GRENOBLE, France
ISSN : 1298-020X

n° 169

Juin 2008

Site internet : http://www.g-scop.inpg.fr/CahiersLeibniz/

Natural Editing of Algebraic Expressions

Jean-François Nicaud and Denis Bouhineau
LIG, 46 avenue Félix-Viallet,
38031 Grenoble cedex France
Jean-Francois.Nicaud@imag.fr
Denis.Bouhineau@imag.fr

Abstract
We call "natural editing of algebraic expressions" the editing of algebraic expressions in their natural representation, the one that is used on paper and blackboard. This is an issue we have investigated in the Aplusix project, a project which develops a system aiming at helping students to learn algebra. The paper summarises first this project. Second it defines the notion of algebraic expression, and of representation of algebraic expression. After an exploration of 20 software systems which represent and edit algebraic expressions, the idea of natural editing of algebraic expressions (insertion, deletion, selection, cut, copy, paste, drag and drop) is presented, leading to the notion of "smart editor".

1. Introduction

In 2000, our team restarts from scratch the development of a new Interactive Learning Environment (ILE) with two main goals. The first main goal was to create an ILE for algebra allowing the student to freely build and transform algebraic expressions. The environment should provide epistemic feedback that can help in the first stages of the learning of algebra. At that time, there were no ILE for algebra that allowed students to freely build and transform algebraic expressions, as they do on paper, and to follow their own reasoning; the existing ILEs were, and most of them still are, command-based systems that did not allow the student to proceed without applying a command chosen from a menu. See MathXpert, [4] and Cognitive Tutor [13] as good examples of these command-based systems. So we decided to build a system to help students solve exercises in numerical calculations and formal algebra, where they would perform their own calculations by typing the expressions and making the steps they want, as on paper. The system, called Aplusix, would give feedbacks, mainly whether the calculations are correct or not, and whether the exercise is solved or not, those feedbacks depending on the semantics of algebraic expressions, and of the syntactic form of the expression.
The second main goal was to create an ILE that would be used for real. This goal requires building a usable and useful system. We considered two main points for usefulness: Encompassing a large mathematical domain and having several modes of functioning.

Allowing students to freely build and transform algebraic expressions, even simple expressions, means building an advanced editor of algebraic expressions. We have developed such an editor from the questions "What do users expect when they do action A?" and not from the question "What is the simplest implementation of action A with the data structure we have?". Of course, this lead immediately to display and edit the expression in their usual two-dimensions representation, but this is just the first step. The other steps consist of having often a structured behaviour (also called algebraic behaviour) and sometimes an unstructured behaviour for flexibility.

In this paper, we only consider input with mouse and keyboard. Input with a pen and hand writing introduces facilities which have been investigated by some projects [34]. We also limit our examples to *simple* algebraic expressions, in the sense that integrals, matrix, etc., are not considered. Actually, we think that these simple algebraic expressions contain almost all the editing difficulties.

This paper shortly describes, in section 2, the Aplusix system (in its current distributed version), some of its experiments, and a recent module devoted to tree representations of algebraic expressions.

Section 3 develops the notion of algebraic expression and of representation of algebraic expressions. Characteristics of software systems which display and edit algebraic expressions are explored in section 4. Section 5 is devoted to natural editing of algebraic expressions, and the conclusion introduces the idea of *smart editor* of algebraic expressions.

2. Aplusix

2.1. Short description of Aplusix

Aplusix is organized on the base of two editors, an advanced editor for algebraic expressions, and an editor for algebraic reasoning; these two editors have been constructed as microworlds. They permit numerical calculations and formal algebra activities like expansion, factorisation, and resolution of equations, inequations, and systems of equations, with verification of the equivalence of expressions during the reasoning. There is an immediate feedback showing whether two consecutive expressions are equivalent or not. Aplusix also provides solution and score on a sub-domain.

In Aplusix, numbers may be written as integers, decimals, fractions and square roots. Exponents must be integers (positive, negative or null). The domains of the 2007 version are described in table 1.

Type of exercise	*Domain of Verification of the calculations.*	*Domain of Solutions and Scores.*
Numerical calculation	The expressions must not include variables.	The expressions must not include variables.
Expansion	Polynomial expressions of one or several variables.	Polynomial expressions of one or several variables.
Factorisation	Polynomial expressions of one variable and maximum degree 4 or of two variables and maximum degree 2.	Polynomial expressions of 1 or 2 variables and maximum degree 2.
Solving equations	Polynomial equations of one unknown and maximum degree 4 and rational equations giving such polynomial equations.	Polynomial equations of one unknown with maximum degree 2.
Solving inequations	Polynomial inequations of one unknown and maximum degree 4 and rational inequations giving such polynomial equations.	Polynomial inequations of one unknown and maximum degree 1.
Solving systems of equations	Systems of equations with maximum 10 equations and 10 unknowns.	Systems of equations with 2 equations and 2 unknowns.

Table 1. *Domains of the 2007 version of Aplusix.*

Commands for executing certain algebraic actions are available. These commands can be enabled or not, powerful or not, according to parameters set by the teacher, (they have to be adapted to the current level of understanding of the students in order to only present calculations they can do without difficulty). With this feature, such a computer system can provide an introduction to the proper use of a Computer Algebra System. These commands are 'Calculate', 'Expand and reduce', 'Factor' and 'Solve'. See example in figure 1.

Besides the usual way of using Aplusix, called "training mode", with immediate feedback, we implemented a "test mode" without immediate feedback. At the end of a test (often 10 exercises), or later, students can enter in a "self-correction mode" where they see their final answer with a score and with feedback, and can correct their errors with the help of immediate feedback. See figure 2.

Last, the microworlds are embedded in a piece of software whose concern is the practical organisation of the learning process (login of the students, organisation of their work according to some prepared scenario, recording of the activities, scoring and statistical analysis of these activities made available for the teacher, tuning of the software with parameters, automated generation of exercises...). A longer description is given in [26].

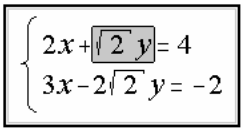
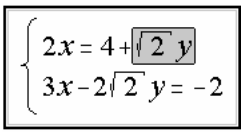
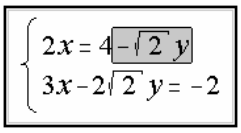
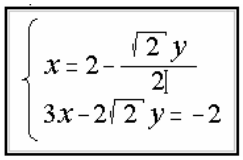
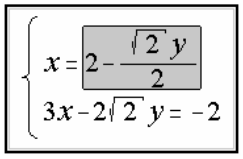
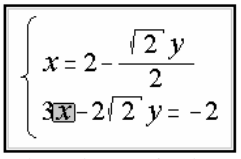
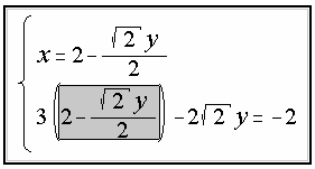
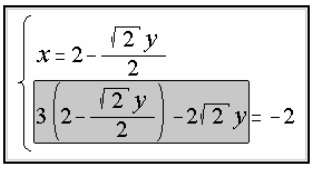
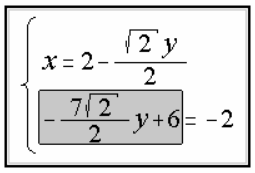
<table>
<tr>
<td> 
1) Anna duplicates the given exercise and selects $\sqrt{2}y$</td>
<td> 
2) She drags $\sqrt{2}y$ and drops it on the right hand side.</td>
<td> 
3) She hits the “minus” key.</td>
</tr>
<tr>
<td> 
4) Anna deletes 2 on the left, changes 4 to 2 on the right and inserts 2 as denominator of $\sqrt{2}y$</td>
<td> 
5) She selects the value of x and makes a copy in the clipboard.</td>
<td> 
6) Then she selects x in the second equation.</td>
</tr>
<tr>
<td> 
7) And makes a paste that produces a substitution (the parentheses are added by the system).</td>
<td> 
8) Then she selects the left hand side of the second equation.</td>
<td> 
9) And she applies the “expand and reduce” command using the pop-up menu.</td>
</tr>
</table>

Figure 1. *How to solve a system of equations using* drag&drop *and the command “Expand and reduce”.*

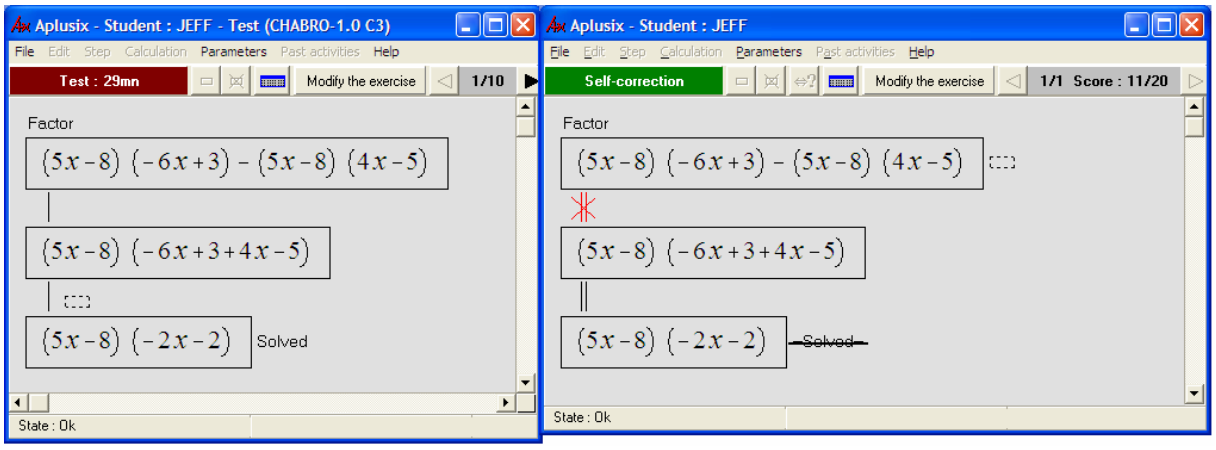


Figure 2. *On the left, Jeff has solved an exercise in the test mode, without feedback. On the right, he is looking at the result in the Self-correction mode. He can see his score (11/20), an incorrect calculation and a stricken “Solved”. He can click on “Modify the exercise” to correct his error.*

2.2. Experimentation of Aplusix

Many experiments have been conducted since 2002 in France, prepared by the researchers in mathematics education of our team. Most of them were carried out in the usual functioning of the class, using the computer lab and supervised by the teacher. Other experiments occurred in Brazil, Canada, Italy, India, and Vietnam. Different goals were pursued, generally two or three simultaneously.

Perception of Aplusix by students and teachers
The general opinion of teacher who used Aplusix is the following: The students work more, gain confident, work with autonomy and acquire knowledge. Teachers save time because of students' autonomy and of prepared exercises. They are better able to help students having difficulties with mathematics. See detailed opinions in [3].
Students like Aplusix. Even some students who dislike mathematics used Aplusix with pleasure. See comments of students in [2].

Contribution of Aplusix to the students' learning
Experiments with pre-test and post-test have been conducted, either in learning or remedial situations. Generally there were one or two students per computer. In one case in India, the students were in groups of 4. All the results are very positive; see [25]

Use of Aplusix during all the school year
Some teachers in France used Aplusix each time the subject studied can be exercised with Aplusix. At the end of the school year, they observed better result than previous classes. See [25].

Gathering data for student modelling
As our team also works in automatic student modelling, some experiments had the goal to collect data for later analysis in the lab. See results in [27].

2.3. Distribution

Aplusix has currently publishers in France (since July 2005), UK (since January 2007), and Italy (since January 2007). The distribution will start in Benelux in January 2008. New publishers and distributors will be searched.

2.4. Extension to tree representations of algebraic expressions

In the framework of the European project ReMath [29], a new module devoted to tree representations of algebraic expressions has been developed [5]. The goal is to help students understanding algebraic expressions with a new register which is closer to the definition of algebraic expressions. This extension has been fully integrated to Aplusix (to a prototype): the students can choose between 4 representations and modes at any step, except when the teacher has added constraints to the exercise.

The 4 representations and modes are:

1) The ***usual representation*** (or natural representation) which is the previous representation of Aplusix with its editor.
2) The ***free tree representation*** which displays and edits expressions as trees without scaffolding: students can place in nodes whatever they want.
3) The ***controlled tree representation*** which displays and edits expressions as trees with scaffolding: internal nodes of the trees must be known operators and must have a correct number of children (correct arity); leaves must be integers, decimals, or variables.
4) The ***mixed representation*** which is the *controlled tree representation* with the possibility to have complex expressions in the leaves in their usual representations. See figure 3.

Two particular types of exercises have been added:

1) Transform a usual representation into a tree representation.
2) Transform a tree representation into a usual representation.

The first experiments of the tree representation will occurred in France and Italy at the end of 2007. They are currently analysed.

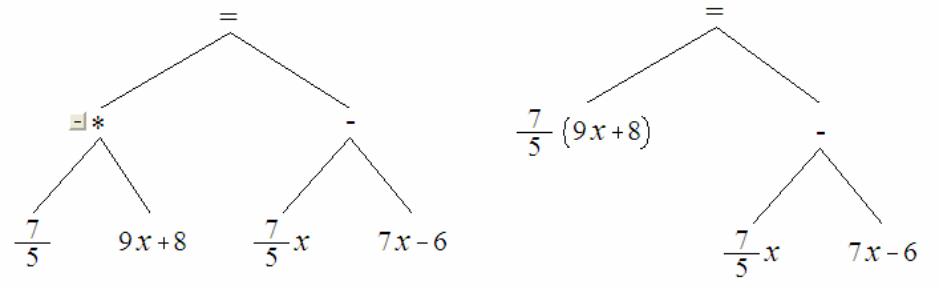


Figure 3. *In the mixed mode, the tree is partially expanded and can be expanded or collapsed using "+" and "-" button which appear when the mouse cursor is near a node. When one clicks on the "-" button near "*" on the tree on the left, one gets the tree on the right.*

3. Algebraic expressions and their representations

3.1. Introduction

Algebraic expressions aim at representing mathematical objects, but they must be associated to a representation system to produce a concrete representation on paper or screen. For example, "the sum of x power 2 and 3", "x^2+3", "x^2+3", and "xx+2+1" are 4 representations of the same polynomial. The 3 first are representations of a same algebraic expression (ways of combining x, 2 and 3), while the fourth is a representation of another algebraic expression which corresponds to the same polynomial. We can see here 3 levels: the mathematical objects (e.g., polynomials) which are abstract, the representations which are concrete (ink on paper or pixels on screen), and algebraic expressions which are at an intermediate level. This intermediate level is abstract as one needs a representation system to draw an algebraic expression.

3.2. Definitions

Given a set of symbols of operators (e.g., {+, –, *, /, ^, sqrt, =, ≠, <, ≤, >, ≥, and, or, not}), a set of symbols of terminal objects (e.g., the integers, the logical values *true*, *false*) and a set of symbols of variables (e.g., {x, y, z}), we define an algebraic expression as a finite construction obtained from the recursive definition given below.

An algebraic expression is:

- a symbol of terminal object,
- or a symbol of variable,
- or a symbol of operator applied to arguments which:
    - are algebraic expressions,
    - are in the right number (correct arity),
    - and have correct types.

This definition is borrowed from the rewrite rule theory [6]

As indicated in the introduction, algebraic expressions are abstract and need a representation system to be represented with concrete elements. We call *adequate representation system* a representation system having the three following properties.

Argument property: Each argument of a symbol of operator is represented.

For example, we do not consider a representation system in which the symbol of operator "+" applied to "2" and "3" produces "5". Such representation system is a calculation system, not a representation system for algebraic expressions.

Operator property: Each symbol of operator is represented. Exceptions are possible when there are rules allowing knowing what are the implicit (not represented) operators.

The usual representation has implicit operators. Some are indicated by the relative position of the arguments (power, subscript). When this is not the case, the implicit operator is times.

Structure property: The representation indicates which the arguments of each symbol of operator are. Exceptions are possible when all the ways of understanding a representation correspond to the same mathematical object.

The natural language representation (one way of using it) allows to write or say "x plus 2 times y" in which one does not know whether the second argument of plus is "2" or "2 times y". This is an important problem, because the two ways of understanding the representation do not correspond to the same polynomial. Hence, the structure property is not verified by the natural language representation. It means that it must be limited to unambiguous expressions (a strong limitation).
The usual representation allows to write "–2*x*" in which one does not know whether the argument of "–"is "2" or "2*x*". Here, there is no problem as both interpretations correspond to the same polynomial.

The previous definition only concerns well-formed expressions. Note that as far as editing of algebraic expressions is concerned, a notion of ill-formed expressions is also necessary, first, because during the construction and the modification of a well-formed expression, one passes by stages where the expression is ill-formed (e.g., when operators are defined but not the arguments, to get 3+x, we have the intermediates stages 3 and 3+, the last one being ill-formed), second because there are situations were we want to allow ill-formed expressions, like in education at some stages.

3.3. Representations

We list here the main representations of algebraic expressions.

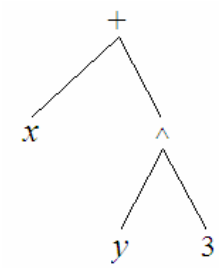
Representation in natural language
Example: *The sum of x and the power of y and 3.*
In this example, the symbols of operators have been changed (sum instead of +, power instead of ^).
It is the historical representation system. In Europe, scientists and mathematicians were still using it in the late 15^{h} century (1478, the Trevise's arithmetica "*ARTE DELL'ABACO*" shows first effort to introduce abstract notation: In 1484, Nicolas Chuquet introduces exponent notation, and Johann Widmann d'Eger replaces p (piu) and m (minus) by + and –; In 1491, François Viète introduces letters for unknown and parameters [11]).
This natural language representation does not verify the *structure property.* As a consequence, it is possible only for small expressions.

Functional representations
Examples: `+(x, ^(y, 3))` `sum(x, power(y, 3))` `(+ x (^ y 3))`
These representations have followed the invention of programmable calculators and some modern programming languages. They are very close to the definition and have no ambiguity. Large

expressions are difficult to understand by human, but not by computers. The first one is the representation used in high level languages, like ProLog, working with terms which are trees. The last one is the representation used in functional languages, like Lisp.

Linear representation
Example: x+y^3
This representation needs priorities between operators, rules for operators of same priority and parentheses. At the beginning, it is the representation used for source code of computer programs. Because of the use of CAS (Computer Algebra Systems), or because it is easier to implement, the linear representation is largely used on the web and among computer based learning environment in mathematics. Large expressions are difficult to understand by human, but not by computers. In [30], Sangwin and Ramsden report surprising variety for the representation of expressions, even at the elementary levels between different systems, and regret the difficulties faced by students due to the necessary mediation needed between traditional mathematical notations and the requirements of such strict or loose syntax.

Tree representation
Example:



This representation shows very clearly the structure of the expression. It needs neither priority between operators, nor parentheses. It is easy to read by humans for short expressions but it is difficult for large expressions. It can be interesting at some stage of the learning of algebra to help understanding the structure of algebraic expressions.

Natural representation
Example: $x + y^3$
This representation is the modern representation introduced during the cultural movement of the Renaissance in the 15th century. More than two centuries have been necessary to achieved it. In particular the representation of product has a very long history which begins with abbreviation 'm' (1545, Stifel), replaced by 'in' in Vietes (1591), then modern '×' (1637 Oughtred) and '.' by Leibniz (1698), simple juxtaposition was also used like an old habit [9]. This is the representation usually used on paper and blackboard. This is why we call it *natural representation*, by analogy with *natural language*. It can also be called *usual representation*.
The natural representation is ergonomic but complex: it has implicit operators (like × in *xy* and ^ in y^3), it needs priorities between operators and parentheses.

TeX / LaTeX
Example: $x+y^3$
TeX / LaTeX are representation systems used by many scientific communities. *TeX* was defined in 1977 by Knuth, *LaTeX* was defined in 1980 by Lamport. It is a standard de facto, used by Wikipedia, and numerous pieces of software and web environments.

Recent computer representations (MathML, OpenMath, OMDoc)
These representations are pretty recent (MathMl 1.01 was released in 1999, OpenMath 1.0 Standard was released in 2000, OMDoc 1.0 was released in 2000). These representations are computer oriented and based on extensions of XML. They are tree-based representations. Table 2 shows an example of representation in Content MathML, Presentation MathML, and OpenMath. They are not supposed to

be read by humans, only by computers. Actually, humans need, and have difficulties, to read them. Computer systems exchange data with those formats (especially with MathML). They also have difficulties, a major one is, for MathML and OMDoc which reuses MathML format, that multiple ways to represent any particular mathematical expression are offered. MathML is a permissive representation system. This is a good point for the usability of the representation, internally to computer systems, but it raises problems for communication between systems. Numerous examples can be found of MathML representations produced by a system which could not be read by another one. Sometimes, even the producer system of a MathML representation is not able to read the file it produced itself.

Content MathML	***Present. MathML***	***OpenMath***

`<apply>`
`  <plus/>`
`  <ci>x</ci>`
`  <apply>`
`  <power/>`
`  <ci>y</ci>`
`  <cn>3</cn>`
`  </apply>`
`</apply>`
	`<math xmlns=…>`
	`  <mrow>`
	`  <mi>x</mi>`
	`  <mo>+</mo>`
	`  <msup>`
	`  <mi>y</mi>`
	`  <mn>3</mn>`
	`  </msup>`
	`  </mrow>`
	`</math>`
		`<OMOBJ>`
		`  <OMA>`
		`  <OMS cd="arith1" name="plus"/>`
		`  <OMV name="x"/>`
		`  <OMA>`
		`  <OMS cd="arith1" name="power"/>`
		`  <OMV name="y"/>`
		`  <OMI>3</OMI>`
		`  </OMA>`
		`  </OMA>`
		`</OMOBJ>`

Table 2. *Representation of* $x + y^3$ *in Content MathML, Presentation MathML, OpenMath.*

Readability

The readability of representations is very different depending the reader is a human or a computer. Table 3 shows readability rates according to the authors' opinion.

Representation name	***Example of small representation***	***Human readability***	***Computer readability***
Natural language	The sum of x and the power of y and 3	10%	1%
Functional representation	`sum(x, power(y, 3))`	10%	100%
Linear representation	x+y^3	50%	100%
Tree representation	*See above*	30%	1%
Natural representation	$x + y^3$	100%	1%
Latex	$x+y^3$	10%	100%
MathML and OpenMath	*See above*	1%	100%

Table 3. *Seven sorts of representation and their readability. Readability means capacity to read medium and large representations as there are (e.g., characters and lines on a two dimensions area for the tree representation). The percentages come from the authors' opinion.*

Note that the very low computer readability of the tree representation and the natural representation correspond to a situation which is not the most frequent: the case where symbols are written on a two dimensions area like a paper sheet that would be scanned. A more usual situation is a computer screen where the representation is drawn by the computer from an internal representation of the expression. In that case, the computer has not to read the representation, it has just to act on it and use its internal representation to do that.

Algebraic expressions are abstractions

While in classical contexts, like education, algebraic expressions appear as representations of semantic objects (e.g., $3x(x^2 - 4)$ is a representation of a polynomial and $3x^3 - 12x$ is another representation

of the same polynomial), the current section shows that algebraic expressions are abstractions having several sorts of representations.

3.4. Syntax and semantics

The definition of algebraic expressions given in 3.2 contains concrete elements (the symbols), the global notions (how symbols of operators apply to arguments, arity, types) being abstract. However, it does not tell us how to represent practically algebraic expressions. The reason is because it intends to give a definition of objects without using a particular representation system. Hence, it is an abstract definition in the sense that representational aspects are abstract.

In mathematics teaching and computer science, it is a good practice to distinguish clearly what is relevant to syntax and what is relevant to semantics.

Even if this definition contains abstract elements, it lays on the syntactical side. According to this definition, "+" is a constructor of expressions, an operator which can be applied to 3 and 5, giving 3+5, not a function which would have made a calculation and which would have produced 8.
It is not a strictly syntactical definition of algebraic expressions, because some concrete representational aspects are abstracted, it is an abstract syntactical definition.
A good example of the gap between a strictly syntactical definition and our abstract syntactical definition is given by the absence of parentheses. Parentheses are syntactical elements needed in some representation systems, they are not mathematical operators and some representational systems do not use them. That is why we do not include them in the definition of algebraic expressions.

Semantic objects are usually associated to algebraic expressions, e.g., numbers, polynomials, rational fractions, functions, sets (like sets of solutions of equations). Such associations are made by the association of a function to each symbol of operator, for example, for the "+"symbol, it can be the addition of numbers, or the addition of polynomials, or the addition of functions.
The choice of semantic objects introduces a general notion of equivalence: given a set of semantic objects, two algebraic expressions are equivalent if and only if they are associated to the same semantic object, their *denotation*.
The definition is not on the semantic side, "+" applied to symbol 1 and symbol 2 is not considered to be the same as "+" applied to symbol 2 and symbol 1 (even if theses two objects share the same denotation)

See more on syntax and semantics in [24].

4. Representations used and editing performed in few software

We have achieved a small survey to evaluate computer systems representing and editing mathematical expressions. The purpose was not to be exhaustive, neither as regards computer systems, nor as regards representations and editing. The goal was to answer only two simple questions:

- Is there any consensus in favour of a representational system?
- Is there any consensus in favour of simple editing actions?

The study was not performed to find the best practices. Systems were not evaluated in that sense. Next section will propose ideas about good practices, from a mathematical point of view.

In this survey, a panel of about twenty pieces of software has been evaluated, see table 4. They can be regrouped in three classes: CAS (computer algebra systems), Editors (text processors), ILE (interactive learning environments). For each, a small session has been realized in order to represent a

simple expression (a simple quadratic equation with one parameter: $x^2 + mx + \frac{4}{13} = 0$) and to try to edit it (just a simple copy and paste of a subpart).

Symbol	Software	Version	Class	Other	Reference
Ed-1	Amaya	9.54	Editor		[1]
ILE-2	Aplusix II	1.02	ILE		[26]
ILE-3	DragMath	0.5	ILE	Web version	[7]
Ed-4	Formulator	3.7	Editor		[8]
Ed-5	Integre MathML Equation Editor	1.2	Editor		[12]
ILE-6	Leactivmath	2	ILE	Web version	[14]
Ed-7	MathCast	0.88	Editor		[16]
ILE-8	Mathdrag'n	0.7.1	ILE		[17]
CAS-9	Mathematica	5.1	CAS		[18]
Ed-10	MATHIWYG	2.0	Editor	Web version	[19]
Ed-11	MathType	6.0	Editor		[20]
CAS-12	Maxima (wxMaxima)	5.13	CAS		[21]
Ed-13	Microsoft equation editor	3.0	Editor		[22]
CAS-14	Mupad	4.0	CAS		[23]
Ed-15	Openoffice	2.2	Editor		[28]
Ed-16	Scientific Notebook	5.5	Editor		[31]
Ed-17	SciWriter	3.0	Editor		[32]
Ed-18	Texmacs	1.0.5	Editor		[33]
Ed-19	Webeq	2.0	Editor	Web version	[35]
ILE-20	Wiris		ILE	Web version	[36]
CAS-21	Xcas	0.7.2	CAS		[37]

Table 4. Software representing and editing mathematical expressions.

4.1. Representation used and input mode

All the systems have adopted the natural representation for the display of mathematical expressions, i.e., $x^2 + mx + \frac{4}{13} = 0$.

However, 48% have a dual representation system (linear+natural or MathML+natural or Latex+natural or custom_form+natural), see table 5. For most of them, a representation system is associate to the keyboard and is used for the input of the expression (it is the linear representation or a tex/latex/custom representation), and the other representation system is devoted to the display (it is the natural representation). Note that the display can be immediate or delayed until the expression is finished and the user ask for it. Sometimes, other output representations are also available (MathML LaTeX or image) or other displays (with box structure, or tree). A minority of them, essentially CAS systems (CAS-12, 14, 21), do not allow editing in the natural representation.

45% software which only use one system of representation (i.e. the natural representation) allow or use in fact a linear representation of the expression for input (i.e. x^2+mx+4/13=0), see table 5. The others use click and/or drag panel of operators or block/box constructors or a virtual keyboard to input the expression, see figure 4 for panel of operator, block/box constructor and virtual keyboard.

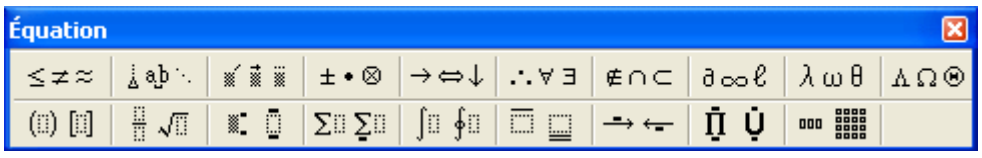


Figure 4. Panel of operators, block/box constructors and virtual keyboard from [22,26].

When using a linear representation, except a minority (ILE-2, 6), systems have the following behaviour: after having typed "x^2" or "1/2" the insertion point moved vertically (which is what is expected as the "^" sign and the "/" sign introduce vertical representation) and when a "+" is typed, the insertion point and the "+"sign come back to the base line.
Software ILE-2 and 6, propose to leave the insertion point at the vertical position reached. In order to come back to the base line, the user has to click at the desired position, or to use a movement key.
But nothing is simple, there is not only two general behaviours, there is lack of coherence: software Ed-5 behaves as ILE-2 and 6 for "/" and as the others for "^".

Except for ILE-8, incomplete mathematical expressions are well accepted, but the way they are represented differs: most of the time, a blank is drawn, ILE-20 leaves a box, ILE-2 leaves a symbolic argument. 29% software (CAS-9, 12, 14, ILE-2, 6, Ed-7) send an error message.

Except the natural representation, we do not have observed any consensus on the use of any other representation system (not even linear, latex, or MathML which could have been natural candidates). Even for the natural representation, some minor differences can be observed on the way expressions are displayed, most of them are almost invisible (just a question of one pixel or two). For the reader, it is a good point, because it means that the differences, when they exist, are not annoying and do not introduce any inconvenient.

For example, about the alignment of the horizontal bar of "+", "–" and "$\frac{1}{2}$",

- half of them respect the alignment but
- Ed-16 has "–" below the two others,
- ILE-8 has the fraction bar below the two others,
- Cas-12 has "+" above the two others,
- Cas-14 has "–" above the two others,
- Cas-21 has "+" above "–" and "/" below "–".

(tests have been performed on WindowsXP, with default font and size).
Hence, we can give the following conclusion: *there is no consensus for a common representational system, i.e., no representation system asserting property of general compatibility between software* (natural representation could not be a candidate, except in the eyes of the reader, because it means image and images are not suitable data format for that kind of exchange). Diagram 1 shows clearly the lack of consensus.

As regards input, we have observed a large variety of modes (linear input and linear+natural output (delayed), or linear input and immediate natural output, virtual keyboard or box constructors). It involves for the user a necessary learning period and possible errors, abandons or lost of time. As regards students, it could be an important point to take into account. That point become more and more significant as representation and input of mathematical objects progress on the web and elsewhere, become more and more common, without any consensus.

	Linear input accepted	Dual system of representation (display)	Representation available as output (file)	+–/ aligned and same thickness	a+b can be written vertically a i.e. + b
Software	ILE-2, Ed-5, ILE-6, Ed-7, ILE-8, CAS-9, CAS-12, CAS-14, Ed-15, CAS-21 **(10/21)**	Ed-1, ILE-3, Ed-7, CAS-9, Ed-10, CAS-12, CAS-14, Ed-15, Ed-19, CAS-21 **(10/21)**	Latex: ILE-3, CAS-9, CAS-12, Ed-11, Ed-16, Ed-17, Ed-18 **(7/21)** MathML: Ed-1, ILE-3, Ed-4, Ed-5, Ed-7, CAS-9, Ed-11, CAS-12, Ed-16, Ed-17, Ed-18 **(11/21)**	ILE-2, ILE-3, Ed-4, Ed-5, ILE-6, CAS-9, Ed-10, Ed-11, Ed-17, Ed-18 **(10/21)**	Ed-1, Ed-4, CAS-9, Ed-10, Ed-11, Ed-13, Ed-16, Ed-17, Ed-19 **(8/20)**

Table 5. Differences in representation and input between software systems

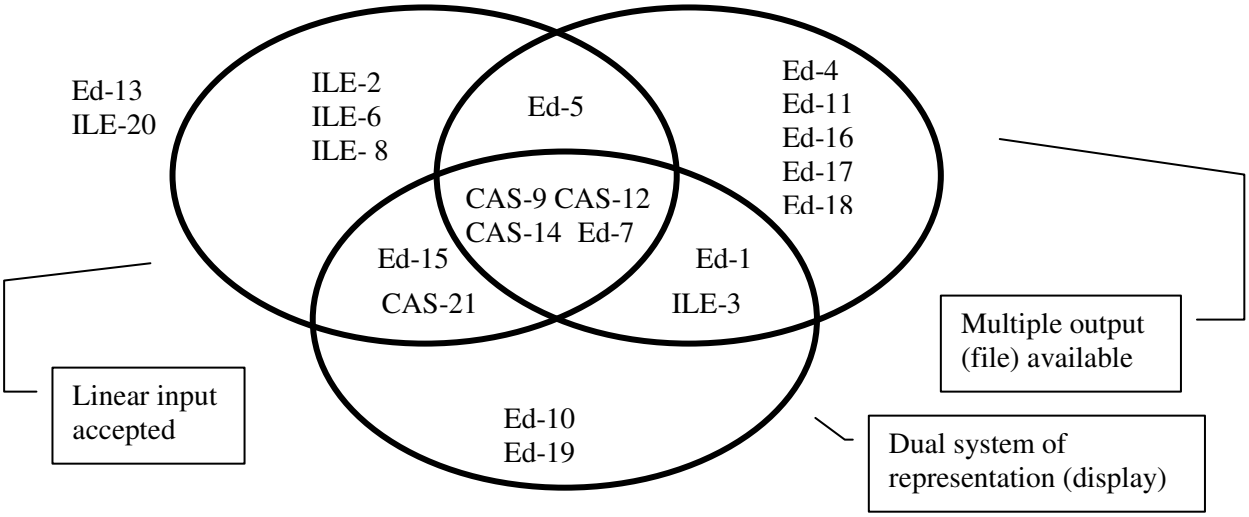


Diagram 1. Differences in input/output representation between software systems

4.2. Insertion point position, movement and simple editing

For editing acts, we have focus on the editing of natural representations. Four basic tests have been performed to evaluate software editors. They all have been executed on the same expression: $x^2 + mx + \frac{4}{13} = 0$.

- The first test consists of finding all the positions the insertion point can reach. In the linear representation (which is x^2+mx+4/13=0), there are 14 positions for the insertion point.
- The second test consists of placing the insertion point at the leftmost position, and navigating inside the expression with the right arrow key to reach the rightmost position. For the linear representation, this test gives 13 movements.
- The third test was to try to select 2 and m. For the linear representation this leads to have selected 2+m.

- The last test was to copy this selection, and to paste it in 13, between 1 and 3. For the linear representation, it gives x^2+mx+4/12+m3=0.

First test, number of positions for insertion point. 48% software systems give 17 positions, this does not correspond to the number of positions in the linear representation, see table 6. They propose, for example, supplementary positions around the fraction.
28% gives more than 17 positions (insertion point with greater vertical extent) or less (some places are unreachable, for example inside 13, between 1 and 3).
24% do not give insertion point.
Close to the linear representation, most of software systems propose two positions between x and 2: one after x, at the same level, another one before 2 at the level of 2.

Second test, number of movements to reach the end. Most software systems need as number of movements, the number of positions minus 1, see table 6. But some introduce new positions (Ed-5, 15) which necessitate more movements and some do no go through all the positions which lead to fewer movements. For example ILE-2, Cas-14, Ed-16, 19 do not go through 13 (the denominator), and Ed-18 does go through 13 but not through the numerator 4 (maybe because 13 is longer than 4).
All systems cross x² from left to right passing through the two positions shown in figure 11 between x and ² having the same horizontal position. For all, between x and ², one right arrow stroke does not go to the left, but stay at the same horizontal position and goes upward.
For all the 33% which go through the numerator and the denominator (always in that order), there is one movement which leads to a leftward movement (from leftmost part of the numerator, to the rightmost part of the denominator) even if it is the right key which is pressed.

Third test, selection of 2 and m. The selection of "2+m" corresponds to the linear representation selection; "x²+mx" corresponds to the selection of the smallest mathematical sub-expression containing "²" and "m". Selections "²+mx" and "x²+m" are incoherent because they preserve a mathematical construction but not the other, see figure 5. But all modes of selection are present in some software systems. By order of importance, "2+m": 33%, "²+mx": 24%, "x²+mx": 9% and "x²+m":5%. 29% software systems do not permit selection of ² and x, see table 6.

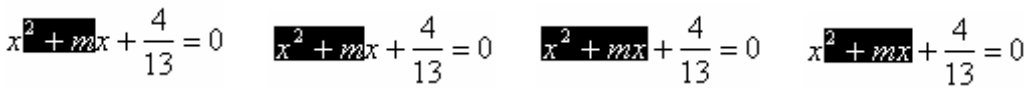


Figure 5. *Minimal selection of ² and x (depending on the system).*

Last test, copy/paste the selection in 13. The test gives almost always the same result, when it can be performed, which is $x^2 + mx + \frac{4}{1^2 + m3} = 0$ (when the selected element is ²+m), or $x^2 + mx + \frac{4}{1x^2 + mx3} = 0$ (when the selected element is x²+mx), and so on (depending on the selected element). This is a textual copy/paste.
The exception is given by ILE-2, which introduces parentheses, and leads to an algebraic insertion (the insertion is an algebraic sub-expression of the global expression) $x^2 + mx + \frac{4}{1(x^2 + m)3} = 0$.

Conclusions from that subsection correlate the previous one, *there is no consensus in the editing actions*, event the simplest one (place an insertion point).

Number of positions for insertion point in $x^2+mx+4/13=0$	10: Ed-15	13: Ed-5	14: ILE-3	17: ILE-2, Ed-4, ILE-6, CAS-9, Ed-10, Ed-11, Ed-13, Ed-16, Ed-18, ILE-20	18: Ed-19	20: Ed-1, Ed-17	No insertion point: Ed-7, ILE-8, CAS-12, CAS-14, CAS-21
Number of movements to reach the end	13: ILE-2, Ed-11, Ed-13, Ed-16	14: ILE-6, Ed-18	16: Ed-4, CAS-9, Ed-10, Ed-19, ILE-20	17: Ed-15	19: Ed-1, Ed-17	25: Ed-5	No movement: ILE-3, Ed-7, ILE-8, CAS-12, CAS-14, CAS-21
Smallest selection with ^{2} and m	^{2}+m: Ed-1, ILE-6, Ed-7, Ed-11, Ed-13, Ed-16, ILE-20	x^2+m: Ed-4, CAS-9, Ed-17, Ed-18, Ed-19	x^2+mx: ILE-2, Ed-10	^{2}+mx: ILE-8	No selection: ILE-3, Ed-5, CAS-12, CAS-14, Ed-15, CAS-21		

Table 6. Differences in insertion point position, movement and simple editing between software editors

From both previous subsections, we can conclude that existing software have not reached a consensus for presentation, exchange and editing of mathematical expressions. The result is a confusing landscape for users and students trying to learn mathematics with computers. The need for a work on homogenization of representation and editing of mathematical representation is clear.

5. Editing natural expressions

From now, we only consider *natural representations of algebraic expressions* which are called *natural expressions* for simplification purpose. We consider that humans prefer to use this representation system and discuss the way they are graphically represented and how they can be edited.

5.1. The text&box view

Natural expressions are drawn in a 2D (two-dimensions) space and can be viewed as composed of texts (sequences of characters), of boxes with invisible borders and of drawings of particular operators. Boxes are necessary when the natural representation of an expression cannot be made in one-dimension. Boxes can contain one-dimension expressions and other boxes according to different 2D modes of association. For example, a fraction is composed of a line (drawing of the *divide* operator) with a box above and a box under for the arguments (which are any expressions). We can also consider a surrounding box for a fraction. See figure 6.

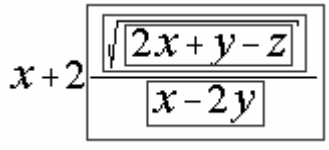


Figure 6. *Text&box view. The borders of the boxes have been drawn.*

There are box operators like *divide*, *power sqrt*, *and* (when it is represented by a left brace "{") , and text operators like +, –, sin, =, <, or.

5.2. Text&box editors

We call text&box editors, editors which follow the text&box view without additional features. This is the case of many 2D editors like Microsoft Equation Editor [22], MathType 5.2 [20] WIRIS [15, 36] and the editor used in the LeActiveMath ILE [14]. Figure 7 shows two examples of expressions displayed by a text&box editor.

These editors allow many sorts of ill-formed expressions: any character can be input at any place of a text part. The only elements which have a clear meaning are the box operators. They are generally obtained through buttons. Often, characters that would benefit to be associated to box operators (like "/" and the *divide* operator) are not.

$$(x/2+5\left(\frac{x}{3}+y\right)+4) \qquad (hello <<< \left(\frac{+-/*\%}{\sqrt{world}}=\right)++$$

Figure 7. *Two Text&box expressions that can be built with the Microsoft Equation Editor, MathType and WIRIS. On the left, a strange quasi-well-formed expression: the "/" key has been used for x over 2 and the fraction button has been used for x over 3; the parentheses keys have been used at the beginning and at the end, and a parenthesis button has been used inside.*

5.3. Syntactical basic enhancement

Here are four features in order to improve the basic editing of natural expressions:

1) Operators recognition: "/" is recognized as *divide* and produces a fraction, "sqrt" is recognized as *square root* and produces a square root, "and" is recognized as *and* and produces a left brace "{", etc.
2) Parenthesis recognition: "(" and ")" are recognized as parentheses and associated to existing parentheses when possible, see figure 8.
3) Arity representation: when an operator does not have enough arguments, places for the missing arguments are drawn, see figure 9,
4) Incorrect type indication, see figure 10,

$$\left(2+\frac{3}{4}\right] \qquad \left(2+\frac{3}{4}\right)$$

Figure 8. *From Aplusix: On the left, there is an unbalanced "(". When a ")" is typed, it is associated to this"(" and the parenthesis size is adapted to the content.*

$$2+\frac{3}{?}+?$$

Figure 9. *From Aplusix: "?" represents a missing argument, both in box operators like* divide *and in text operators like* plus*.*

$$2+\frac{3}{x=5}$$

Figure 10. *From Aplusix: "x=5" is drawn in blue to indicate an incorrect type.*

5.4. Insertion point and insertion point's movements

From a general editor viewpoint, one must be able to place the insertion point at any place where a character can be inserted. However, a question arises here: What are the atoms? In the 3.2 definitions, the atoms are the symbols of terminal objects (e.g., integers), of variables and of operators. Actually, one can go at a sub-atomic level as these atoms are sometimes made of several characters. Is it adequate to have cursor positions inside integers? Our answer is yes, because if we want to change

140527 into 140627, we would like to be able to place the insertion point after 5, delete 5 and type 6, otherwise, we would have to delete 140527 and type the 6 digits of 140627. Is it adequate to have cursor positions inside text operators (like 'sin')? Our answer is yes and no, with a preference for no because, as text operators have short names, it is not a problem to delete one in one keystroke and to type another. Figure 11 provides the behaviour of *Microsoft Equation Editor*. It has an adequate behaviour with the insertion point going into text operators.
We consider that there are two inadequate behaviours: (1) The impossibility to place the insertion point inside integers (e.g., the Integre math editor and the OpenOffice equation editor for the natural representation part) and (2) When several insertion point positions lead to the same result when one types a letter or a digit, see figure 12.

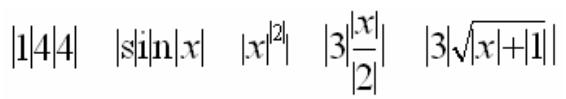


Figure 11. *From Microsoft Equation Editor: insertion point positions. Note the possibility to place the insertion point inside numbers, inside text operators (e.g.,* sin*), before, inside and after exponents, fractions and square roots.*

Figure 12. *From Integre MathML Equation Editor: two insertion point positions when one would like to have only one. In both positions, if one types "y", one gets the same result (on the right).*

As regards the horizontal movements, we consider that the adequate behaviour consists of always moving to the right (or the left). Crossing from left to right the numerator of a fraction then going to the denominator is inadequate as it is a right arrow which goes to the left or the reverse. Passing in the 2 positions between x and ² in x² in figure 11 is the general behaviour. It can be improved by avoiding going in the exponent area, considering that this is a secondary argument which can be reached with an up arrow keystroke or a click. No studied system has this behaviour. We plan to implement it in the next version of Aplusix.

Vertical movements are less frequent and more complex. The problem consists of choosing the closer box with regards to two criteria: the vertical and horizontal positions of the boxes. Figure 13 shows the behaviour of the Microsoft Equation Editor in 3 particular situations. This is a good behaviour. In Aplusix, we do not have a good treatment of the vertical movements at the present time.

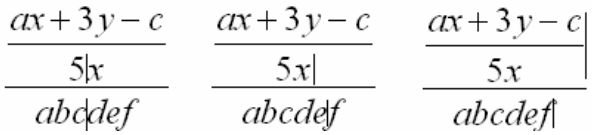


Figure 13. *From Microsoft Equation Editor when a up arrow is typed: on the left, the insertion point after c goes after 5; on the middle, the insertion point after e goes after x; on the right, the insertion point after f goes after the upper fraction (in this case, one may prefer to go after the x of* 5*x).*

5.5. Enhanced Backspace and Delete

In tex&box editors, Backspace with an insertion point inside a box and placed on the left of the content does nothing (Microsoft Equation Editor) or suppresses the box (MathType), see figure 14. This can be improved by deleting the operator. For a unary operator, one can delete the operator and keep the argument, as WIRIS does for the square root in a text way (figure 15) and as Aplusix does in a structured way, see figure 16. For a binary operator, one can keep the main argument or the two arguments. See in figure 17 how to suppress common denominators with Aplusix. Similar behaviour can be implemented for the delete key.

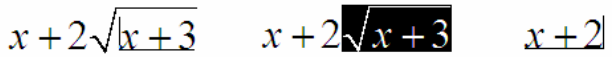


Figure 14. *From MathType. The insertion point is inside the square root, on the left. Hitting backspace selects the square root and its content. Hitting backspace again suppresses the square root.*

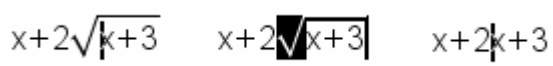


Figure 15. *From WIRIS. The insertion point is inside the square root, on the left. Hitting backspace selects the square root (without the content). Hitting again backspace suppresses the square root and keeps the content in a text way.*

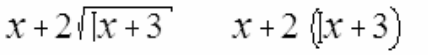


Figure 16. *From Aplusix. The insertion point is inside the square root, on the left. Hitting backspace suppresses the square root in a structured way that makes parentheses appear.*

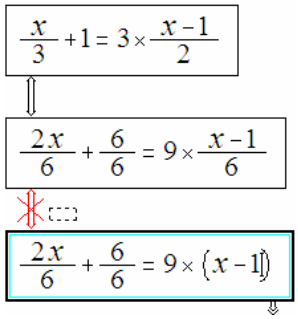


Figure 17. *From Aplusix. The student put all the terms to the common denominator 6. Then, he/she wants to suppress the 3 denominators. He/she has only to place the insertion point on the right of each denominator and hit twice backspace (first, the system replaces 6 by ? and second it deletes the denominator and keeps the numerator in a structured way, that is why parentheses appear).*

5.6. Algebraic selection

Text&box editors allow the selection of parts of the expression regardless the structure, see figure 18. The selection is not linked with the notion of sub-expression.

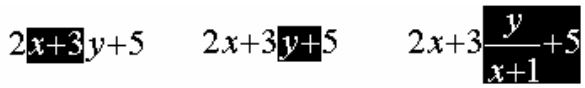


Figure 18. *From Microsoft Equation Editor, MathType and WIRIS: Three examples of selection.*

When places for missing arguments are drawn, it is possible to implement a selection which respects the structure, i.e., the sub-expression. For example, in $(2x+y)(z+1)-2$, let us drag left to right over x, x is selected; when we continue over +, the selection becomes $2x+1$, when we continue over ")", the selection becomes $(2x+y)$, when we continue over "(" the selection becomes $(2x+y)(z+1)$.
Furthermore, when a selection is present, *crtl-click* can be implemented to extend the selection for associative operators as in many systems, see figure 19.
We call *algebraic selection* this selection mechanism.

Figure 19. *From Aplusix. On the left, 4x is selected. A ctrl-click on 3 or x or 2 on the right part produces the selection displayed on the right in order to have a selection of sub-expression.*

5.7. Operations on a selection

Input on a selection

In *Text&box* editors insertion over a selection of a character (e.g., *x*, 4, =) replaces the selection (this is the general mechanism of text editors) while a click on a button corresponding to a box operator applies the operator to the selection, see figure 20.

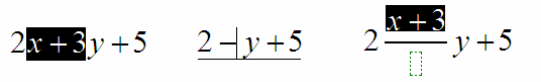


Figure 20. *From MathType: What happens with a selection when one hits "–" (in the middle) or clicks on the fraction button (on the right).*

This can be made more homogenous this way: (1) When the input is an operator, a box operator or not, obtained with a hit of a key or a click on a button, applies the operator to the selection as main argument; (2) In the other cases, replace the selection by the input. In the case of the minus sign, it can be improved by considering a change of sign, i.e., if the selection is –3x, getting 3x instead of –(–3x), see figure 21.

Figure 21. *From Aplusix. A hit of "–" over a selection of the form "–A".*

Paste over a selection

When an editor implements an algebraic selection, paste over a selection can be performed according to an algebraic way which correspond to a substitution, see figure 22.

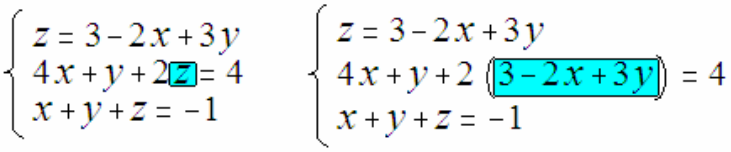


Figure 22. *From Aplusix. Paste when z is selected and 3-2x+3y is in the clipboard. This operation is viewed as a substitution and parentheses are added when necessary.*

5.8. Paste on the insertion point and Drag&drop

In texts, paste and drop on the insertion point consists of placing a copy of the clipboard at the insertion point side to side with the text of this place. This has meaning because concatenation has meaning for texts. For algebraic expressions, the link between expressions is made by operators. If we want to combine 4 and 12, we choose to do that with "+" or "/", etc. Placing side to side *x* and *y*, which gives *xy*, has meaning because there are implicit operators (times in this situation). A general behaviour for an editor would be to let the user choose the operator. But this may be a bit heavy. Another way would be to provide a way to easily change the operator used for paste. In the case of Aplusix, we have implemented the second choice, limiting the operators to the operators of variable arity and of the right type, see figures 23 and 24.

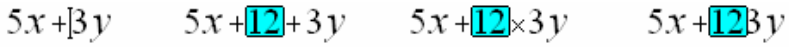


Figure 23. *12 is pasted when the insertion point is before 3. Aplusix pastes 12 with the "+" operator (because it is the upper numerical operator of variadic arity at this position). A hit on the "AltGr" key changes the paste operator to "*". Another hit on the "AltGr" key changes the paste operator to the composition of numbers.*

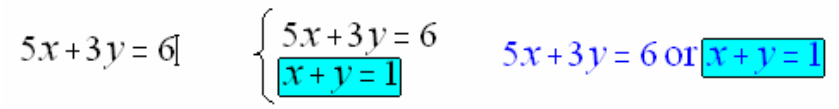


Figure 24. *x+y=1 is pasted when the insertion point is after 6. Aplusix pastes x+y=1 with the "and" operator (possible logical operator of variadic arity at this position). A hit on the "AltGr" key changes the paste operator to "or".*

This is a structural, or algebraic, way of performing paste and drop.

5.9. Equivalent Drag&drop

Besides this structural drag&drop, there is also an equivalent drag&drop which consists of moving a sub-expression inside a global expression and preserving the equivalence of the global expression.
Such functionality is implemented in the GraphingCalculator [10]. This software allows to manipulate the expressions algebraically with the mouse, according to the author's concept "the calculator preserves equality" during these manipulations.

Moving arguments of a commutative operator is the first natural form of equivalent *drag&drop* (e.g., moving $3x^4$ to the left in $2x^3+x^2+3x^4$ provides $2x^3+3x^4+x^2$ if the drop is before x^2 and $3x^4+2x^3+x^2$ if the drop is before $2x^3$). In this particular situation, the structural *drag&drop* does the same thing if the drop is made with the operator.
A second natural form of equivalent *drag&drop* consists of moving an additive sub-expression from a side of an equation (or inequation) to the other side, changing its sign (e.g., in $x^2+5x=-6$ moving -6 to the left provides $x^2+5x+6=0$).
A third natural form of equivalent *drag&drop* consists of moving a multiplicative sub-expression from a side of an equation (or inequation) to the other side (e.g., in $5x=-6$ moving 5 to the right provides $x=-\frac{6}{5}$). Other forms are proposed in table 7. They are based on factorisations or reductions.

Expression	Selected sub-expression	Place of the drop	Result	Type of action
$3x^2-1+x^2$	x^2	over $3x^2$	$4x^2-1$	Reduction
$(y-1)(x+x^2)$	first occurrence of x	Between)(	$(y-1)x(1+x)$	Factorisation
$(xy)^2$	x	Before (	x^2y^2	Factorisation
$\sqrt{4+x}$	4	Before sqrt	$2\sqrt{1+\frac{x}{4}}$	Factorisation

Table 7. *Examples of equivalent* drag&drop *with basic operators.*

6. Conclusion: toward *smart editors* of algebraic expressions

There is a representation which has been built by humans during several centuries and which is called here *natural expression*. If, at some learning stage, it is good for students to use the natural language representation and the tree representation, and to perform changes of semiotic systems to better understand the notion of algebraic expressions, the rest of the time there is no reason for humans to use another representation system.

Software developers are going little by little to fully manipulate natural expressions. First, many of them took care of the presentation. Second, some of them took care of the editing in a text&box framework. The next step consists of introducing manipulations which correspond more to the user

needs and to mathematics, leading to what we would like to call *smart editors*. Many answers can be found by asking the *right question* which is: "As a user, what would I like to get when I do this editing action?" instead of considering that "With my internal representation the answer to this action is that". But we will still have a problem: as there are several ways to answer to the right question, smart editors will have different behaviours and the last step will be standardisation.

7. References

[1] Amaya, http://www.w3.org/Amaya/

[2] Aplusix: Comments of students. http://aplusix.imag.fr/en/ *(Information / Comments (students))*

[3] Aplusix: Opinions of teachers. http://aplusix.imag.fr/en/ *(Information / Opinions (teachers))*

[4] Beeson M. (1996). Design Principles of Mathpert: Software to support education in algebra and calculus, in: Kajler, N. (ed.) *Human Interfaces to Symbolic Computation*, Springer-Verlag.

[5] Bouhineau, D., Chaachoua H., Nicaud, J.F., Viudez C. (2007). Adding new Representations of Mathematical Objects to Aplusix. *Proceedings of the ICTMT-8 conference.* Hradec Králové, Czech Republic.

[6] Dershowitz N, Jouannaud J.P. (1989). Rewrite Systems. *Handbook of Theoretical Computer Science, Vol B, Chap 15*. North-Holland.

[7] DragMath, http://www.dragmath.bham.ac.uk/

[8] Formulator, http://mmlsoft.com/projects/formulator/

[9] Freudenthal, Hans. Notations et symboles utilisés en mathématiques., in Encyclopædia Universalis, (1968).

[10] Graphing Calculator, http://www.PacificT.com

[11] *Ifrah, Georges, Histoire Universelle des Chiffres, Éditions Robert Laffont, Paris 1994.*

[12] Integre MathML Equation Editor, http://www.integretechpub.com/zed/

[13] Koedinger, K.R., Anderson, J.R., Hadley, W.H. and Mark, M.A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education 8*, (pp. 30–43).

[14] LeActiveMath project. http://www.leactivemath.org/

[15] Marquès D., Eixarch R., Casanellas G., Martínez B., Smith T. (2006). WIRIS OM Tools a Semantic Formula Editor. Porceedings of MathUI 2006.

[16] MathCast, http://mathcast.sourceforge.net/home.html

[17] MathDrag'n, http://mathdragn.squarespace.com/

[18] Mathematica, http://www.wolfram.com/

[19] Mathiwyg, http://www.terradotta.com/index.cfm?FuseAction=MathIWYG.Demo

[20] MathType 5.2. http://www.dessci.com/en/products/mathtype/

[21] Maxima (wxMaxima), http://wxmaxima.sourceforge.net/wiki/index.php/Main_Page

[22] Microsoft Equation Editor. http://office.microsoft.com/en-gb/word/HP051902471033.aspx

[23] Mupad, http://www.sciface.com/

[24] Nicaud, J.F., Bouhineau, D. and Gélis J.M. (2001). Syntax and semantics in algebra. *Proceedings of the 12th ICMI Study Conference.* The University of Melbourne.

[25] Nicaud J.F., Bittar M., Chaachoua H., Inamdar P., Maffei L. (2006). Experiments With Aplusix In Four Countries. *International Journal for Technology in Mathematics Education, Volume 13, No 1.*

[26] Nicaud, J.F., Bouhineau, D., Chaachoua H. (2004). Mixing Microworld and CAS Features for Building Computer Systems that Help Students to Learn Algebra. *International Journal of Computers for Mathematical Learning 9, p. 169-211.*

[27] Nicaud, J.F., Chaachoua H., Bittar M. (2006).Automatic calculation of students' conceptions in elementary algebra from Aplusix log files. Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS2006), LNCS n° 4053, Springer-Verlag.

[28] OpenOffice, http://fr.openoffice.org/

[29] ReMath project. http://remath.cti.gr/

[30] Chris .J. Sangwin, P. Ramsden, (2007) Linear syntax for communicating elementary Mathematics in Journal of Symbolic Computation 42 920–934.

[31] Scientific Notebook, http://www.mackichan.com/

[32] SciWriter, http://www.soft4science.com/

[33] Texmacs, http://www.texmacs.org/

[34] Will Thimbleby, Harold Thimbleby (2005) A novel gesture-based calculator and its design principles, Proceedings 19th. BCS HCI Conference.

[35] WebEq, http://www.dessci.com/en/products/webeq/

[36] WIRIS editor. http://www.wiris.com

[37] Xcas, http://www-fourier.ujf-grenoble.fr/~parisse/giac_fr.html

Les cahiers Leibniz ont pour vocation la diffusion des rapports de recherche, des séminaires ou des projets de publication sur des problèmes liés au mathématiques discrètes.

Pour soummettre un articles dans les cahiers,
`http://www.g-scop.inpg.fr/CahiersLeibniz/`

Responsables de la publication : Nadia Brauner et András Sebő
ISSN : 1298-020X - © Laboratoire G-SCOP