A Microworld for Helping Students to Learn Algebra

Denis Bouhineau, Jean-François Nicaud, X. Pavard, Emmanuel Sander

A Microworld For Helping Students To Learn Algebra

Denis Bouhineau*, Jean-François Nicaud*, Xavier Pavard*, Emmanuel Sander**

*IRIN, 2 rue de la Houssinière, Université de Nantes, UFR Sciences
BP 92208, 44322 Nantes cedex 3, France
{bouhineau, nicaud, pavard}@irin.univ-nantes.fr
**ESA-CNRS "Cognition et Activités Finalisées", Université de Paris 8,
2, rue de la Liberté, 93526 Saint-Denis Cedex, France
sander@univ-paris8.fr

Summary

This paper describes the design principles of a microworld devoted to the manipulation of algebraic expressions. This microworld contains an advanced editor with classical actions and direct manipulation. Most of the actions are available in two or three modes; the three action modes are: a text mode that manipulates characters, a structure mode that takes care of the algebraic structure of the expressions, and an equivalence mode that takes into account the equivalence between the expressions. The microworld also allows to represent reasoning trees. The equivalence of the expressions built by the student is evaluated and the student is informed of the result. The paper also describes the current state of the implementation of the microworld. A first prototype has been realised at the beginning of February 2001.

Keywords: Algebra, microworld, direct manipulation.

1. Introduction

Mathematical microworlds have been developed since the late sixties. The first one was Logo [Papert 80] devoted to recursive programming and geometry. The concepts of mathematical microworld have been established in the mid and late eighties [Thompson 87, Laborde 89, Balacheff & Sutherland 94], and the main features have been exhibited: objects, relationships, operators and direct manipulation. Several microworlds have been realised in geometry and algebra. Some of them are distributed has commercial products and have (or have had) a real success, like Logo and Cabri Geometry [Laborde 89]. The last one has been implemented on pocket calculators and distributed as computer software by Texas Instruments. Microworlds for geometry provide a real contribution to the learning of the domain and have introduced a new concept of geometry called dynamic geometry.

In this paper, we consider microworlds for formal algebra. Algebraland [Foss 87] and the McArthur's system [McArthur et al. 87] can be seen as the first ones. Others software for algebra, like those devoted to the modelling of word problems [Koedinger et al. 97], are not concerned here. In our previous works, we realised what we consider *as second order* microworlds for algebra. Ours systems were advanced prototypes [Nicaud et al. 90, 94]. We conducted a lot of experiments and analysed students' behaviour and learning using these systems [Nguyen-Xuan et al. 93, 99]. We are now designing and realising what we consider as a *first order* microworld for algebra. This system aims at becoming a commercial product. In the future, we will combine the features of first and second order microworlds.

The fundamental question is: What can be a microworld for algebra? In this paper, we define first and second order microworlds. Then we detail first order microworlds and describe the mains specifications of the system we develop and the current state of its realisation.

2. Summary of formal algebra

Before trying to answer the question: *What can be a microworld for algebra,* we need to summarise the domain and to specify the objects and the relationships.

The domain of algebra we consider is composed of algebraic expressions and rules for transforming these expressions. There are problem types like factorisation, equation solving, and calculation of derivatives. Solving a problem consists of transforming the expression of the given problem using correct transformation

rules until a solved form is reached. Correct transformation rules are rules that maintain the equivalence of expressions. We call *formal algebra* this domain.

The concrete objects of algebra are expressions. An expression is a succession of digits (like 12), a symbol (like an unknown x, a parameter m, the constant π) or an operator applied to expressions. An international two-dimensions representation, we call *usual representation*, has been adopted for expressions. For example, $\frac{\sqrt{x}}{12+x^2}$ is a quotient. A quotient is the application of the *division* operator to two arguments. Here, the arguments are $\sqrt{x}$ and $12+x^2$. The first argument is the square root of x (the *square root* operator applied to the variable x); the second argument is the sum of 12 and x^2, etc. Several rules specify what are *well-formed expressions*, in particular the arity rule and the type rules. For example, "=" has two arguments, "=" applies to algebraic arguments and provides a logical result. As a consequence a=b cannot be an argument of a sum because sum applies to algebraic arguments. Let us note $\mathcal{E}$ the set of well-formed expressions considered for a given problem domain.

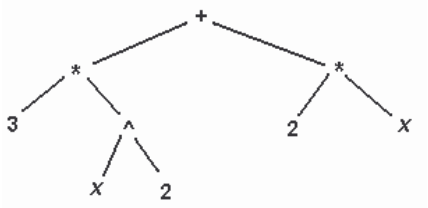

Figure 1: Tree representation of $3x^2+2x$

The usual representation is not the simplest one. The simplest is a tree representation (see figure 1) in which operators and there arguments are very visible and in which it is obvious to find to what arguments an operator applies. But the tree representation is heavy. This explains why it is not used with students and why the usual representation was built. The main difficulties in the usual representation come from hidden operators (like *multiplication* in 2x) and from the priority of operators (x+2=0 is the operator = applied to x+2 and 0 because the priority of + is higher than the priority of =). Parentheses are sometimes necessary, like in x(x+1); they are not in the tree representation.

The basic abstract objects of algebra are numbers. Let us note $\mathcal{K}$ the set of numbers considered for a given problem domain. In secondary education, it is generally the largest set that has been introduced and, during a long period, it is the real numbers. There is a fundamental morphism $\mathcal{M}_1$ from $\mathcal{E}$ to $\mathcal{K}$ that maps operators to functions (e.g., + is mapped to the addition of numbers) and that maps any pair of an expression and a substitution to a number (e.g., 2x+6 with [x: 5] is mapped to 16). This morphism allows using operators in the representation of numbers. Natural numbers have canonical representations in base 10 made of successions of digits. Other numbers have representations that use operators and digits (like –12). Some numbers cannot be represented with operators and digits, and are represented with symbols (like π).

As said above, expressions are transformed to solve problems. For example, <factor $(x+3)^2-1$> can be solved by applying the rule $A^2-B^2 \rightarrow (A-B)(A+B)$ to $(x+3)^2-1$, providing $((x+3)-1)((x+3))+1$, then suppressing the parentheses around x+3, providing (x+3-1)(x+3+1), finally reducing the numbers, providing (x+2)(x+4). Transformation rules are objects of the domain. Applying correct rules is not the only way to solve problems. For example, one can factor x^2-5x+6 telling the solution is (x-3)(x-2) because one has guessed the roots of the polynomial. Actually there is a fundamental domain that gives signification to expressions. The latest example can be justified by "the expressions x^2-5x+6 and (x-3)(x-2) are equivalent because they are associated to the same polynomial". For expressions having an algebraic type, the fundamental domain most often used is the set of functions from $\mathcal{K}$ to $\mathcal{K}$ but it can also be the polynomials or the rational fractions over $\mathcal{K}$. For expressions having a logical type (like equations), the fundamental domain is the set of functions from $\mathcal{K}$ to {false, true}. Let us note $\mathcal{F}$ the fundamental domain and call functions the elements of $\mathcal{F}$, even when they are polynomials or rational fractions. There is a fundamental morphism $\mathcal{M}_2$ from $\mathcal{E}$ to $\mathcal{F}$. Functions are the fundamental abstract objects because they provide the equivalence of expressions: two expressions are equivalent if and only if they are associated to the same function of $\mathcal{F}$ by the morphism $\mathcal{M}_2$..

3. What can be a microworld for algebra

According to Laborde [89]. A microworld is a world of objects and relationships. There is a set of operators able to operate on the objects to create new ones with new relationships. Direct manipulation has an important place in microworlds. Balacheff & Sutherland [94] emphasise on *an epistemological domain of validity* providing the relationship between the representations and what they intend to represent. Thompson [87] tells that mathematical microworlds must facilitate the process of constructing objects and relationships and focus on the construction of meaning. They must use a model not embedded in the curriculum, they must not provide instruction.

In formal algebra, as said above, the concrete objects are expressions. The basic relationship between the objects is the structural relationship over the expressions: 12 is the first argument of the second argument of $\frac{\sqrt{x}}{12+x^2}$. The second relationship is the equivalence relationship. There are two main sorts of operators.

The first sort operators allow building any well-formed expression: they are the operators of *algebraic expression editors*. We term *first order* the algebraic microworlds that focus on these operators. First order algebraic microworlds are included in many word processing systems. They are called *equation editor*. We will show below that most of them have a poor relationship with the semantic objects of algebra. The McArthur's system [McArthur et al. 87] has feature of a *first order* algebraic microworlds.

The second sort operators are the transformation rules that allow building new steps in a reasoning process. We term *second order* the algebraic microworlds that focus on transformation rules. Several second order algebraic microworlds have been realised in the past [Bundy & Welham 81, Foss 87, Thompson 89, Oliver & Zukerman 90, Beeson 96]. The previous systems we developed in our team were of that class. Students solved problems by applying transformation rules to selected sub-expressions. They were not allowed to build a step another way. The community did not really consider these systems as microworlds, may be because they teach (something refused by Thompson), may be because they do not have the first sort operators, may be because people who realised these systems emphasised more on the tutoring aspects.

In the rest of the paper, we focus on first order algebraic microworlds we just call algebraic microworlds. In section 4., we consider the classical features of an editor for an algebraic microworld. In section 5., we tackle the issue of direct manipulation.

4. Main actions of the editor of an algebraic microworld

Objects and the structural relationship

A microworld must allow the user to create objects. In the case of algebra, this means to have an editor for algebraic expressions. The visual fidelity imposes the usual 2D representation for the interface of the editor. This representation contains text parts that can be seen as successions of characters (like 2x+1) and bloc parts well delimited (like $\frac{x}{1+x}$ and $\sqrt{x+1}$).

The conceptual fidelity imposes to take the structural relationship into account. So, we propose to give to students well-formed expressions, as much as possible. For that purpose, expressions with missing arguments are completed with something, for example "?" (e.g., typing + at the end of 2x provide 2x+?) and expressions that cannot be completed are highlighted (as spelling errors in MS-Word).

The main actions of an editor are: input, delete, select something, apply an operator to something, copy something, delete something, cut something, paste something. Let us consider these actions through the structural relationship. This implies that the above something is a sub-expression. We call *the structure mode*, a mode where well-formed sub-expressions are preserved during the actions and *text mode*, a mode where well-formed sub-expressions are not preserved. Note that the two modes are useful, the former to avoid destroying what is already built, the later to let the user be able to do everything.

Proposals for the implementation of the actions

Input from the keyboard (or input by clicking a button), when the expression has an insertion point: in order to keep a well formed expression, the input of a text operator adds "?" when necessary; in the structure

mode, the input of a bloc operator adds "?" for each argument; in the text mode, the input of a bloc operator uses the environment for the arguments (e.g., typing / after x in xy provides $\frac{x}{y}$).

Delete the right or the left of the input cursor: in a text part, only one character is deleted except when the character is "?" where the operator requiring it is also deleted (e.g., from 2x+?+1, the cursor being between + and ?, <del> provides 2+1); when a bloc is concerned (including parentheses), the entire bloc is deleted in the structure mode and only the operator is deleted in the text mode (e.g., from $2\sqrt{x}$, the cursor being after 2, <del> provides 2 in the usual mode and 2x in the text mode).

Select a sub-expression: the structural relationship suggests selecting only well formed sub-expressions. Note that 2x+4 in 2x+y+4 and –mp in 1–2mxp are well-formed sub-expressions. The natural idea for the selection is to consider the smallest sub-expression that includes the dragged area. Such actions were implemented in our previous prototypes and were appreciated by the students and the teachers.

Apply an operator: when a sub-expression is selected, we consider that the input of an operator means "apply this operator to the sub-expression". For an operator having several arguments, the sub-expression becomes the main argument, the other arguments are "?".

Copy: copies the selected sub-expression in the clipboard.

Paste a sub-expression: the sub-expression is inserted at the place of the cursor or the selection. In the structure mode, parentheses are added when necessary (pasting x+1 after 2 in 2y provides 2(x+1)y). In the text mode, they are not (pasting x+1 after 2 in 2y provides 2x+1y).

Delete a sub-expression: deletes the selected sub-expression and an operator when necessary.

Reasoning and the equivalence relationship

An important activity in formal algebra consists of solving problems by successive transformations of expressions, with conservation of the equivalence. After the edition of expressions we have to consider the representation of the reasoning processes for an algebraic microworld. A reasoning process contains not only successive transformations of expressions, with conservation of the equivalence, but also backtracks and sub-problems. A backtrack is sometimes realised when solving a problem. It consists of going back to a former step to try another way to solve the problem (so the steps have the form of a tree). A sub-problem is sometimes extracted and solved to help to solve a problem. Given a problem <T e>, T being the type and e the expression, the most frequent situation for a sub-problem consists of having a problem <T1 e1>, where e1 is a sub-expression of e, such that the solved form of the sub-problem may benefit to the original problem. For example, <solve A=0>, when A is a polynomial having a degree higher than 1, benefits from the resolution of the sub-problem <factor A>.

We consider that an algebraic microworld must represent reasoning processes, the equivalence between expressions, backtracks and sub-problems.

Propositions for the implementation of reasoning processes

In our previous prototypes, we implemented all the above features. The link between equivalent steps was an arrow and corresponds to a tutored action. We think now that it would be more accurate for an algebraic microworld to use a double arrow meaning "these expressions are equivalent". Backtrack was implemented by the possibility to have several successors to any step. We think good to keep that. Sub-problems were linked to problems with a special arrow. We think now that it would be better to solve sub-problems on other places of the sheet or on other sheets. As for the expressions, the system must highlight the expressions that are not equivalent.

Existing systems

Microsoft Equation, FrameMaker, StarOffice [Microsoft Equation, Adobe FrameMaker, StarOffice] offers the user a specific interface to define and represent algebraic expressions. The arrangement of the elements of an expression according to the international two dimensions usual representation, the specific drawing of the numerous algebraic operators, the adoption of certain typographical conventions for the writing of numbers, variables and function names, justify an independent interface. Formulae or expressions

are obtained using preset juxtapositions of boxes to be filled with text or with other formulae. The result is generally satisfactory on the graphic level but certain expressions are very badly represented, and the algebraic aspect of the editors is often less developed than their graphic features. For example, it is common not to be able to differentiate expressions of the type of 2^{2^2} -that is (2^2)^2- from the expression 2^{22} -that is 2 with the power 22- (Adobe FrameMaker, StarOffice). The graphic aspect is often very developed without any algebraic control, so, the user can move the elements of his choice inside an expression to adjust their positions, one pixel or more away; several modes for parentheses are proposed to obtain $(((x+1)x+2)x+3)x+4$ or (((x+1) x+2) x+3) x+4; the sizes, the styles and the fonts are adjustable by the user for a better visual depiction but all that is performed without any algebraic control. Thus certain objects obtained are algebraically ill-formed, e.g., incomplete expressions, unbalanced parentheses (Equation Microsoft). Sometimes, in addition, the constructions with juxtaposition of boxes respect the subjacent algebraic structures but the result has some trouble. For example, in $2x^3$, it is impossible to select x^3 which concerns two different boxes, on the contrary x+2 can be selected in 5x+23, although it is not an algebraic sub-expression, because these elements are not in different boxes. From a general point of view, all the manipulations permitted on the expression have to be performed with respect to the structure of boxes, and that structure is more graphical than algebraic, as a consequence, to move and interpret the place of the cursor is a difficult operation because there are many graphic positions which do not correspond to algebraic positions.

5. Direct manipulation

In a microworld, direct manipulation allows to displace objects taking into account the relationships. Expressions, sub-expressions and operators can be manipulated through the structural and equivalent relationships or without taking care of these relationships.

Drag & drop a sub-expression

A selected sub-expression s can be dragged & dropped inside an expression e as a word in a word processing system. Such action changes the expression. Let us call u the expression obtained. In the structure mode, the constraints we consider comes from the structural relationship, they are: (1) u is a well formed expression, (2) s is a sub-expression of u, (3) sub-expressions of e that does not include s are left unchanged. Note that these constraints introduce sometimes parentheses around s. In the text mode, the only constraint is the first one and parentheses are never added. When s is a sub-expression of u, s is still selected and a new drag & drop may be performed. In the text mode, when s is not a sub-expression of u, there is no selection in u at the end of the action, see an example in section 6, figure 4. The aim of drag & drop in these two modes is to allow the user to build the expression (s)he wants to.

The aim of drag & drop in the equivalent mode is very different; it is to maintain the equivalence. So we add constraint (4) u must be equivalent to e. With constraints (1) to (4), the only general move that can be done consists of displacing an argument of a commutative operator. To get a more powerful equivalent drag & drop, we withdraw constraint (3) so that sub-expressions of e that does not include s may be changed.

Let us take as an example the expression $(y-1)(x+x^2)$, select the first occurrence of x and drag it between the two parentheses. We may expect to get (y-1)x(1+x) which is a factorisation. An explanation for factoring in such context may be that the moved expression is an argument of a sum, which is an argument of a product, the action being performed as “get x as a common factor of the sum and insert it as a factor of the product”. Note that it is necessary to withdraw constraint (3) for that, because x^2 has been changed in x.

The examples, table 1, show that a major place is given to factor. Even in situations where no product appears, factors are used (e.g., #4). These examples suggest to consider the equivalent drag & drop as “extract or insert the sub-expression as a factor when possible” (which strongly differs from the explanation suggested in the previous paragraph). We must remark that the results are built by the application of algebraic rules (which explain how the equivalence is maintained) and that reduction rules are sometimes applied (when 2 is inserted in 3x-6, we expect to get 6x-12) that may make disappear the sub-expression (in that case, the inversion of the process is not possible).

#	Expression	Selected sub-expression	Place of the drop	Expected result
1	$3x-1+x^2$	x^2	Before 3	x^2+3x-1
2	$(y-1)(x+x^2)$	first occurrence of x	Between)(	$(y-1)x(1+x)$
3	$(xy)^2$	x	Before (	x^2y^2
4	$\sqrt{4+x}$	4	Before sqrt	$2\sqrt{1+\frac{x}{4}}$
5	$\sqrt{2+x}$	2	Before sqrt	$\sqrt{2}\sqrt{1+\frac{x}{2}}$
6	$\frac{x}{x+y}$	first occurrence of x	Denominator	$\frac{1}{1+\frac{y}{x}}$

Table 1, examples of equivalent drag & drop with basic operators.

Considering the structure, there are several general situations for a drag & drop: (1) an argument of an operator is drag and drop over another argument (e.g., #1, #8, #10, inverse of #2, #4, #5); (2) a sub-expression going out (e.g., #2, #3, #4, #5); (3) a sub-expression going out then going into another (e.g., inverse of #2, #3, #4, #5). This issue concerns either structure or equivalent drag & drop.

It is natural to consider the mouse point as the destination for the drop. But this point does not determine a unique position in the structure, e.g., the point after = in $3x=x^2(x-1)+2$ can be considered as a point in the sum, in the product or in the power. This issue concerns either structure or equivalent drag & drop.

For the non-basic operators, equivalent drag & drop may be defined for some situations as in table 2. Note that the actions are complex and that the link with factors disappears.

Expression	Selected sub-expression	Place of the drop	Expected result
$\ln(xy)$	x	before ln	$\ln x+\ln y$
$\ln x^3$	3	before ln	$3\ln x$
$\cos(x+y)$	x	before cos	$\cos x\cos y-\sin x\sin y$

Table 2, examples of equivalent drag & drop with non-basic operators.

Manipulation of operators

Parentheses and bloc operators define areas the user may want to change. For that purpose, we consider direct manipulations of these operators: "(" and ")" may be moved independently by drag & drop; the field of fraction bars and radicals may be changed with handles that appears when the operator is selected. For example, $2x+\sqrt{3y+2}+5$ may be transformed in $2x+\sqrt{3y+2+5}$ or $2x+\sqrt{3y}+2+5$ using the right handle of the radical sign.

Existing systems

The Graphing Calculator [Graphing Calculator] that equips Apple's computers since 1994 presents a very convincing algebraic editor of expressions although it is limited to the usual functions. The expressions are always well-formed and not ambiguous. In addition, the software allows manipulating the expressions algebraically with the mouse, according to the author's concept "the calculator preserves equality" during these manipulations. But this last point is not always perfectly respected. For example, using drag & drop, $\sqrt{\frac{a}{b}}$ can be transformed into $\frac{\sqrt{a}}{\sqrt{b}}$, and $x+1$ can be transformed into $x(1+\frac{1}{x})$ without indication of a domain of validity. Only the operands can be manipulated with this software. We would like being sometimes able to move the brackets or the signs equal, extend fraction bars or radicals. Finally, one can regret that these manipulations are generally carried out and gradually made the expressions more and more complicated and that these manipulations are not reversible: replaced in its initial position, an expression can not find its initial form.

6. The Aplusix microworld for algebra

Our team planed to realised a first order algebraic microworld in September 2001. The Aplusix microworld will include two main activities: first to allow the student to build expressions, second to allow the student to solve a problem on an algebraic expression. A first prototype has been realised at the beginning of February 2001 and a first experiment is planed for mid 2001. The recording of the interactions is already implemented. Every elementary action is recorded in order to analyse the students' behaviour in the laboratory by hand or by programs. The first experiments will be conducted with 15 years old students in the mid 2001 in order to evaluate the two activities.

The development is realised with Borland Delphi for windows. It current state is described here.

The 2D display of expressions is implemented with the following operators: + - × / ^ √ = ≠ < > ≤ ≥ and or not. The selection, including the selection of several arguments of a commutative operator) is implemented (see figure 2). Insert, delete, copy, cut, past are implemented in the text and structure modes.

The display of the reasoning tree is implemented (figure 4). The evaluation of the equivalence of two expressions A and B is realised on $\mathcal{F}$ with morphism $\mathcal{M}_2$. when A and B are polynomials of one variable and when A and B equations with one unknown and a degree less than 4.

The direct manipulation is implemented in the text and structure modes (figure 3). The equivalent drag & drop is implemented only for moving an argument in a commutative operator. A complex equivalent drag & drop is not necessary for our first experiments with 15 years old students and needs a deeper reflection; our goal is not to implement a pretty mechanism but a mechanism having an epistemological value.

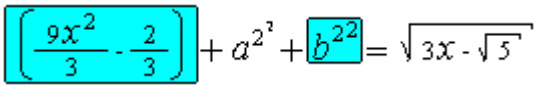

Figure 2: Display of an expression in the usual 2D representation with selection

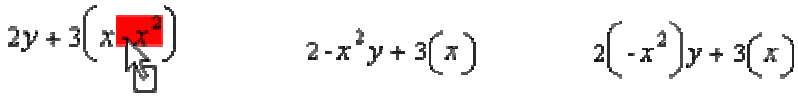

Figure 3: A drag & drop. The selected expression (on the left) is dropped between 2 and y with the text mode in the middle and with the structure mode on the right.

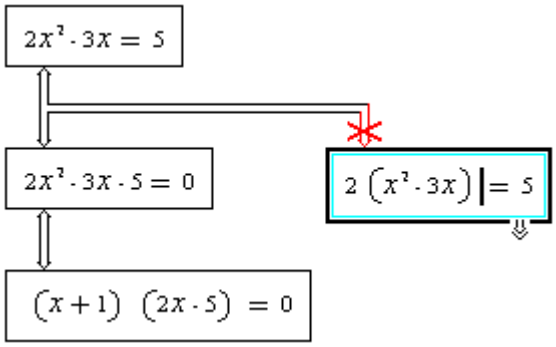

Figure 4: A reasoning tree with non equivalent expressions

Later realisations in the framework of our first order algebraic microworld will include: (1) the representation of new operators, a column operator to represent systems of equations in the usual form, ∑, ∫, etc., (2) a powerful equivalent drag & drop (3) the manipulation of fraction bars and radicals, (4) the representation and display of sub-problems, (5) the extension of the evaluation of the equivalence (we will use results of applied mathematics and constraint programming), and (6) a redisplay function. Of course, we will also add features or realise modifications suggested by the experiments.

We plane to combine next this microworld with our previous works, adding transformations rules, knowledge to solve problems, to evaluate students' resolutions, to explain, to provide hint.

7. Domain of validity and discussion

To conclude, we would like to discuss the *domain of validity* as emphasised by Balacheff & Sutherland [94] who defined fours dimensions in mathematics for the *epistemological domain of validity*:

The first dimension is the set of problems, which the microworld allows to be proposed. For the *building expression* activity of the Aplusix microworld, it is any well-formed expression using the operators and the constants included in the system and respecting the types. For the *reasoning* activity, the set of problems is constituted by any problem concerning an algebraic expression that belongs to the class of expressions the system is able to compare for the equivalence relationship. It is currently the polynomials of one variable with any sort of form and any sort of coefficients and equations with one unknown having a degree less than 4. Other classes of expressions will be added later. Note that this is independent of the problem type (factor, expand, solve, etc.). The problem type must just be an algebraic problem type, i.e., must be such that a problem is invariant when the expression is replaced by an equivalent expression.

The second dimension is the nature of the tools and the objects provided by its formal structure. The objects of the Aplusix microworld are algebraic expressions as they are defined in the rewrite rule theory [Dershowitz & Jouannaud 89], which is the most achieved theory on expressions. The nature of the tool is to allow to build expressions and to allow comparing expressions according to the equivalence relationship.

The third dimension is the nature of the phenomenology over its formal structure. The phenomenology at the interface of the Aplusix microworld has a maximum of fidelity concerning the representation of expressions. It is our permanent first principle for the interface: *implement the usual 2D representation*. Concerning the edition features, as exposed in this paper, we defined them using fundamental concepts of the domain (the structure and equivalence relationships). The display of the equivalence is a natural phenomenon as the equivalence is a fundamental feature of any algebraic reasoning.

The fourth dimension is the sort of control the microworld makes available to users and the feedback provided. For the *building expression* activity, the student may execute any editing action. The major difference with other editors is that ill-formed expressions are highlighted. Most of the time, actions that lead to incorrect expressions are completed in order to get a well-formed expression (instead of refusing the action). This is a form of feedback. When the student disagrees with the interpretation of the system, (s)he can undo the action. For the *reasoning* activity, the student may produce a new step at any time and has the indication of the equivalence as fundamental feedback. This category of feedback, already realised by McArthur [McArthur et al. 87] in an elementary context, has not been deeply experimented. We are impatient to observe how it can benefit to the students' learning.

Balacheff & Sutherland [94] also evoked a *didactic domain of validity* for microworlds. On that issue, we will just indicate two points. First, the Aplusix microworld is fundamentally an algebraic world, not a pre-algebraic one. For example, there is no subtraction in it. The minus sign is a unary symbol (a-b is the sum of a and the opposite of b). So students who are not ready to be plunge into a strict algebraic context will probably not benefit from the use of the system. Second, the system can be parameterised by the user or the teacher. A text file contains the description of the operators. It is possible to activate or deactivate an operator; it is possible to change some feature of an operator. For example, one can allow or not variables in a denominator or in a square root. These parameters allow to custom the environment to a particular level.

We have designed an algebraic microworld an epistemological way, and we are ending the first prototype. We hope that most of our choices will be good, but we are not sure. The answer will come from the users and we will analyse carefully the discrepancies between the behaviour of the system and the user expectations, in order to correct minor problems and to learn from the major ones and correct them too. Major problems, and the lesson they provide, will be published.

8. References

[Adobe FrameMaker] V5.0, http://www.adobe.com/products/framemaker/main.html

[Balacheff & Sutherland 94] N. Balacheff, R. Sutherland, Epistemological domain of validity of microworlds, the case of Logo and Cabri-géomètre. In : Lewis R., Mendelshon P. (eds) Proceedings of the IFIP TC3/WG3.3 : Lessons from learning (pp.137-150). North-Holland, 1994.

[Beeson 96] M. Beeson, Design Principles of Mathpert: Software to support education in algebra and calculus, in: Kajler, N. (ed.) *Human Interfaces to Symbolic Computation*, Springer-Verlag, 1996.

[Bundy & Welham 81] A. Bundy and B. Welham. Using Meta-level Inference for Selective Application of Multiple Rewriting Rule Sets in Algebraic Manipulation. Artificial Intelligence, Vol 16, no 2, 1981.

[Dershowitz & Jouannaud 89] N. Dershowitz and J.P. Jouannaud. *Rewrite Systems*. In Handbook of Theoretical Computer Science, Vol B, Chap 15. North-Holland. 1989.

[Foss 87] Foss C.L. *Learning from errors in ALGEBRALAND*. IRL report No IRL87-0003, 1987.

[Graphing Calculator] V1.2, http://www.PacificT.com

[Koedinger et al. 97] Koedinger K. R., Anderson J. R., Intelligent Tutoring Goes To School in the Big City. International Journal of Artificial Intelligence in Education, 8, 30-43. 1997.

[Laborde 89] J.M. Laborde, Designing Intelligent Tutorial Systems: the case of geometry and Cabri-géomètre, *IFIP WG 3.1 Working Conference on Educational Software at the Secondary Education Level,* Reykjavik, 1989.

[McArthur et al. 87] D. McArthur, C. Stasz and J. Y. Hotta. Learning problem-solving skills in algebra. Journal of education technology systems, 15, p 303-324, 1987.

[Microsoft Equation] V3.0, http://www.mathtype.com/msee

[Nguyen-Xuan et al. 93] A. Nguyen-Xuan, J.F. Nicaud, J.M. Gélis, F. Joly : An Experiment in Learning Algebra with an Intelligent Learning Environment. Actes de PEG'93, Edinbourgh, 1993.

[Nguyen-Xuan et al. 99] A. Nguyen-Xuan, A. Bastide, J.F. Nicaud, "Learning to match algebraic rules by solving problems and by studying examples with an intelligent learning environment, proceedings of *Artificial Intelligence in Education*, Le Mans, 1999.

[Nicaud et al. 90] J.F. Nicaud, C. Aubertin, A. Nguyen-Xuan, M. Saïdi, P. Wach : APLUSIX: a learning environment for student acquisition of strategic knowledge in algebra. Actes de PRICAI'90. Nagoya, 1990.

[Nicaud et al. 94] J.F. Nicaud, J.M. Gélis, M. Saïdi, A. Nguyen-Xuan, "The APLUSIX project: a computer-aided teaching of algebra. Methodology, theoretical foundations, realisations and experiments", CALISCE, p 369-376, Paris, septembre 1994.

[Oliver & Zukerman 90] Oliver J., Zukerman I. *DISSOLVE: An Algebra Expert for an Intelligent Tutoring System.* Proceeding of ARCE, Tokyo, 1990.

[Papert 80] S. Papert. Mindstorms. Children Computers, and Powerful Ideas. Sussex: Harvester.

[StarOffice] V5.1 http://www.sun.com/staroffice

[Thompson 87] P. W. Thompson, Mathematical Microworlds and Intelligent Computer Assisted Instruction, in *Artificial Intelligence and Instruction*, Addison Wesley, Reading MA-USA (1987).

[Thompson 89] P. W. Thompson. Artificial Intelligence, Advanced Technology, and Learning and Teaching Algebra. In S. Wagner and C. Kieran: Research issues in the learning and Teaching of Algebra. Lawrence Erlbaum. 1989.