Rule dialogue in problem solving environment T-Algebra

Marina Issakova, Dmitri Lepp

University of Tartu

marinai@ut.ee, dmitri@ut.ee

Abstract

This article presents the T-Algebra environment that allows step-by-step solving of algebraic problems in four fields of mathematics. An essential moment in the solving process is the solution step dialogue between the environment and the user. The so-called Action-Object-Input scheme is used for that purpose. Each solution step in T-Algebra consists of three stages: selection of the transformation rule, marking the parts of expression, entering the result of the application of the rule. The last stage can proceed in three different modes: direct entering of the result, structured entering of the result and construction of the result using different components. The presented scheme improves the ability of the program to check the student's solution and respond to the errors made by the student.

Keywords

Algebraic manipulation, Computer-Aided Assessment, Problem solving environments, Step by step solution, Solution step dialogue

Introduction

The calculation of the value of expressions, operations with fractions, transformation of expressions and solving of equations are such issues in the mathematics curriculum that pose difficulties to many students and are relatively labour-intensive for the teachers. There are several reasons, why the use of computers could be of assistance in this matter. The difficulties experienced by the students while solving the problems can be quite variable and require a thorough thought effort from the teacher in order to understand all details. When using traditional instruction technology, the teacher is not able to give prompt advice or draw attention to errors. The problems often include a great number of details and if the student receives the corrected solution from the teacher only a week after the assignment, he may not remember his thoughts at the moment of making the error or the causes of error. It may often be the case that even a principal mistake is disregarded as an error caused by oversight, thus preventing the student from engaging in the analysis of its real causes. For the teacher, checking of assignments in this field is very labour-intensive and he may not be able to discover all errors made in written assignments.

In many countries, the schools use computer algebra systems (DERIVE, Maple, etc.) or software developed on the basis of these systems (StudyWorks, MathCAD) to simplify working with algebraic problems. These programs have not been specifically developed for educational purposes. For many types of school algebra problems, they do indeed enable a simple acquisition of an answer (and in some cases also the solution compiled by the computer), but they are not designed for the situation in which the student solves problems, makes mistakes, requires feedback and advice, etc. Currently, there is a great lack of such step-by-step problem solving environments designed for the students.

One that is worth mentioning is the APLUSIX [1] package for transformations and equations, which provides the students with feedback on the correctness of their step (the students can see on the screen, whether the expression/equation obtained in this step is equivalent to that in the previous step). But this package is unable to handle the solution algorithms of different types of problems and does not provide a precise diagnosis of the errors made.

The University of Stanford has developed extensive and interesting web-based courses for gifted students – EPGY (Education Program for Gifted Youth) – that also include modules for mathematics [2]. The latter incorporate programs TPE (Theorem proof environment) and Felissa [3] enabling the construction of proofs and transformations by selecting necessary steps from the menu. Yet, these

programs have been designed for advanced students, whose solutions need not be checked step-by-step. These programs do not provide a precise diagnosis of the errors.

The issues related to the aforementioned problems are discussed in the following article in three sections. The first section presents a general description of T-Algebra and introduces the features of the system as well as the problems that can be resolved by the program. In addition, it depicts the program's problem resolution window and provides an example of problem resolution by the program. The second section defines the resolution process as a sequence of steps and provides one simple example (in conjunction with a comparison to the school algorithm). This part gives a general overview of the step dialogue used in the program and describes in more detail all the stages of a step. The third part presents a detailed description of the set of rules used in the resolution process (which items need to be marked, what does the program do, which data need to be entered in the result, etc.) complete with examples.

1. Description of T-Algebra

This article presents an environment for solving algebraic problems, T-Algebra, which enables step-by-step problem solving in four fields of mathematics: calculation of the values of numerical expressions; operations with fractions; solution of linear equations, inequalities and linear equation systems; simplification and factorisation of polynomials.

Work in the T-Algebra environment follows the same steps described in the algorithms taught at school. The program monitors, whether the student works according to the algorithm, and supports it with the respective dialogue, diagnoses transformation errors, offers advice on the selection of the next transformation and, if necessary, performs the next step by itself. In addition, the program includes a separate module for exercise compilation and review of the student results.

The contents of the program comply with the curriculum and the problems are chosen based on the types of problems in Estonian textbooks. The selection of problems covers almost all routine exercises that are solved under respective topics.

The environment is being developed by the Master's and Doctoral students of the Institute of Computer Science at the University of Tartu and under the supervision of their instructors. The authors of the content of the program are mathematics teachers and the authors of textbooks for schools. This version is developed as a project financed by the 'Tiger Leap' computerization programme for Estonian schools [4]. During the preparations for the project, the authors were supported from the grant no. 5272 of Estonian Science Foundation [5].

1.1. Expressions in T-Algebra

The main object for the program to work with is the algebraic expression. In this section, we will describe, which expressions are allowed in the program, i.e., which expressions are treated as correct.

The main part of an expression consists of numbers ($0 \ldots 9$) and letters ($a \ldots z$). The expressions in the program enable using powers ($\square^{\square}$), common fractions ($\frac{\square}{\square}$ and $\frac{\square}{\square}$), decimal fractions, and grouping symbols (parentheses $()$ and brackets $[]$). Permitted arithmetic operations are: addition, subtraction, multiplication and division (signs $+ \; - \; \cdot \; :$). In addition, more complicated expressions are realized in the program as well: linear equations (sign $=$), linear inequalities (signs $< \; \leq \; \geq \; >$) and systems of linear equations (sign $\{$).

Expressions in the program must be mathematically correct and involve various combinations of abovementioned symbols. Here are some examples of correct expressions:

- $\frac{1}{2}-2\frac{2}{3}$,
- $x^2(1+x)^2$,
- $2x-2\le 3-x$.

The following expressions are treated by the program as incorrect:

- $(1+x)\frac{1}{2}-3\frac{0,2}{3}$, because an operator is required between the parentheses and the fraction, and a decimal fraction is not allowed in mixed numbers;
- $a2b+2bc3av-(x-1)^{-4}$, because a multiplication sign is required in monomial multiplication, and constants are not permitted between variables in monomial multiplication.

1.2. Problems resolvable by T-Algebra system

T-Algebra enables the student to solve under the control of the program almost any analogous problems in the given fields (calculation of the values of numerical expressions; operations with fractions; solution of linear equations, inequalities and linear equation systems; simplification and factorisation of polynomials), provided that the original expression/equality/system of equalities has been given in the text of the problem.

This version of the program supports 51 types of problems. We will list some types of supported problems from every field with some specific examples.

1. Calculation of the values of numerical expressions covers the following sub-themes found in textbooks:
    a. Defining the order of operations in an expression. Example: define the order of operations: $11461:(979-822)-(3843+759):177$
    b. Calculation of the value of an integer expression. Example: simplify as much as possible $23\cdot 43+144:6-5\cdot(21-14)$
    c. Calculation of the values of expressions with decimal fractions. Example: perform the indicated operations: $15{,}7-1{,}44:(2{,}79+3{,}21)+(50{,}04-12{,}51):(1{,}2+1{,}8)$
    d. Calculation of the values of literal expressions, if all variable values are given. Example: evaluate: $7894-1984:a+1200, \quad if\ a=31$
2. The program section of operations with fractions enables solving, for example, the following types of problems:
    a. Reduction of a common fraction to its lowest terms. Example: reduce to the lowest terms: $\frac{35}{95}$
    b. Addition and subtraction of similar fractions. Example: combine into a single fraction and simplify: $\frac{17}{26}-\frac{7}{26}+\frac{3}{26}$
    c. Addition and subtraction of mixed numbers. Example: subtract: $3\frac{1}{3}-1\frac{3}{5}$
    d. Multiplication and division of common fractions. Example: perform the indicated operations and reduce to the lowest terms: $\frac{9}{8}:\frac{3}{4}\cdot\frac{4}{9}$

3. The following types of problems are applicable for solving linear equations, inequalities and linear equation systems:

   a. Problems concerning the individual steps of equation solving algorithms. Example: move numbers to the right $7x + 8 = -6$

   b. Solving linear equations in one variable. Example: solve the equation $11(x-2) - 9x = 2$

   c. Solving linear inequalities. Example: solve the inequality $m - 1 \le 2m + 1$

   d. Solving systems of linear equations in two variables. Example: solve the system of linear equations by elimination using addition $\begin{cases} 2x + 3y = 12 \\ x - 3y = -3 \end{cases}$

4. The following types of problem are supported for operating with powers, monomials and polynomials:

   a. Combining like terms. Example: combine like terms:
   $3 - x^2 + 5x - 3x + 4x^2 - 8 - 5x + 2$

   b. Multiplication and division of monomials. Example: multiply the monomials:
   $2a \cdot (-3ab) \cdot (-2a^2c)$

   c. Multiplication of polynomials. Example: multiply the polynomials and simplify:
   $(2a + b + 3c)(2a - b + 3c) - (2a - b + 3c)(2a - b - 3c)$

   d. Factoring. Example: factor out the common factor: $33x^2y + 44xy^2 - 55xy$

1.3. Description of the problem-resolution window

Figure 1 shows the problem-resolution window of the T-Algebra program.

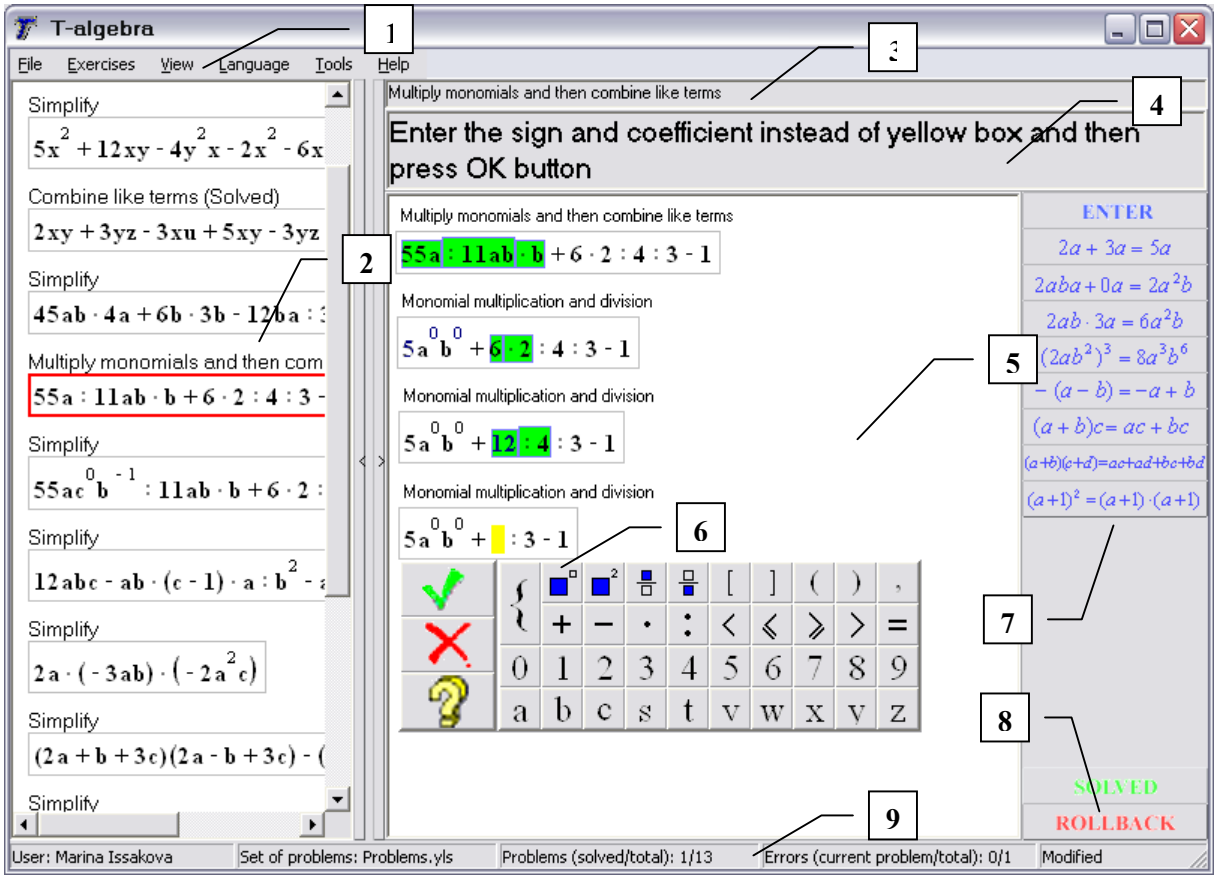


Figure 1. The problem-resolution window of the T-Algebra program

The problem-resolution window has been divided into two logical parts. The left-hand part contains a field displaying a list of problems. The list contains expressions and formulations of problems. In addition, information on the problem resolution is displayed – if a problem has been solved, this is indicated in the list after the problem formulation as shown in the figure. A problem, which is currently being resolved, is displayed in a red box.

The main parts of the window are:

1. The program menu bar, which enables to manipulate with files (Open, Save) and problems (Clear the list of problems, Generate a random problem, Switch to the previous or the next problem), open additional windows (for example, view the error counters – categorized by the types of errors), choose the language of the program, modify program settings, or view help.
2. The list of problems to be resolved, which also shows the results of problem solving. It is also possible to hide the list of problems and to use the whole window for viewing the solution.
3. The text of the problem.
4. Instructions to aid the problem resolution process (indicating what the student should do next: choose the rule to apply next, mark some parts of expression, enter something, etc.).
5. The resolution process for the selected problem – the sequence of steps.
6. The expression box, the buttons for symbols that are not on the keyboard, and the palette for structures (system of equations, fraction, power, etc.).
7. The menu of possible operations.
8. The buttons for confirming the solution or rollbacking the solution steps.
9. The status bar, which shows information about the user and the open set of problems.

Part (5) of the sample window in Figure 1 shows the resolution process for the simplification problem $55a:11ab\cdot b+6\cdot 2:4:3-1$. The solution is not yet complete, but some steps have already been taken. At the first step, ‘monomial multiplication/division’ was selected as the operation, and all the terms contained in the first addend were marked. The powers of the coefficient and the variables were entered in the result. At the second step, 6 and 2 were multiplied in the same way. For the last completed step, the operation ‘monomial multiplication/division’ was picked and the terms 12 and 4 were marked. The selection has been confirmed and, as the next step, the user would have to enter the result of the division in the yellow box on the next line.

2. Solution process scheme (step dialogue)

The solution of problems in T-Algebra consists of several steps as described in the previous section. There are two different possibilities for taking a step in interactive programs – direct entering of the step result or conversion by some rules (commands). T-Algebra uses the second option – conversion by rules. In this section, we will describe the dialogue used for the individual steps of the solution.

Each solution step in T-Algebra consists of three stages:

1) selecting a transformation rule (action),
2) marking the parts of expression (object),
3) entering the result of the application of the selected rule (input).

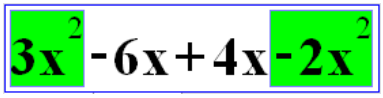
Hereinafter we will refer to this scheme as the Action-Object-Input scheme, based on its three stages. This type of scheme was first used in the program Polynom, developed by one of the authors of this article as his Master’s thesis, which was presented [6] at the ICTMT6 conference (October 2003, Volos, Greece).

In this article, we present the upgraded version of the scheme in which each stage of the step contains multiple choices. In simple cases, the second and the third stage may be skipped. It is also possible to let the program complete these stages automatically. In more complicated cases, entering of some additional information might be required by the program after the necessary parts of the expression have been marked in order to generate the next step (for example, when substituting the variable with its value or other expression, the latter should be entered manually).

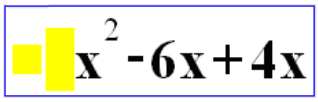
In the next sections, we will provide the reasons for choosing such a scheme. We will also discuss some potential advantages of this scheme. Each stage of the solution step will be described. In the current section, we will present one solution step as a simple example of how the step corresponds to the algorithm taught at school.

Let the problem be the following: simplify the expression $3x^2 - 6x + 4x - 2x^2$. When solving the problem on paper, the student would at first examine the expression and then decide to combine like terms. Then he would underline the like terms he wants to combine with a line, and enter the resulting expression after the equality sign. The program follows principally the same scheme of actions. The corresponding solution step consists of the following three stages:

1) selecting a transformation rule – the student selects from the rule list the rule of combining like terms;
2) marking the parts of expression – the student marks all the monomials similar to x^2, using the mouse and selection buttons;



3) entering the result of the application of the selected rule – the program copies unchanged parts of the expression onto the next line and asks the student to enter the sign and the coefficient before the new monomial.



The following example should give an idea of the structure of the solution step dialogue in more sophisticated cases, as well as of the level of detail of the step and of the level of detail in error checking and diagnosis.

For example, when transforming polynomials, the solution step could be comprised of the following stages (if an error message is displayed at any checking stage, the student must first correct the error himself or let the program correct the error in order to proceed to the next stage):

1) the student picks from the menu the rule of combining like terms;
2) the program checks, whether it is possible to apply such transformation at this stage;
3) the student selects in the expression a group of like terms;
4) the program checks, whether the selected parts of the expression are actually like terms, and it also checks, whether these terms can be combined (i.e., whether they belong to the same polynomial); then the program displays the next line of the expression;
5) the student enters in the resulting expression the sign and the coefficient of the new monomial;
6) the program checks, whether the entered parts are correct and the whole expression is still equivalent to the expression displayed in the previous line, and then displays the resulting expression in the next line of the solution.

For each action of the student, the program gives specific instructions ('Choose the rule to apply next', 'Select terms to combine', 'Enter the sign and the coefficient in the resulting expression', etc.). The student can cancel the step at any moment. It is also possible at any stage of the step to ask the program for help and let the program complete certain stages automatically.

Thus, the whole process of solution follows this algorithm:

1) the student completes a solution step,
2) if the answer is not obtained, repeat step 1),
3) if the student obtains an answer, he has to confirm it and in some cases make additional choices (for example, when solving linear equations, the student has to choose, whether the equation has one solution or has no solution or has an infinite number of solutions).

2.1. Action-Object-Input scheme

Interactive programs, in which the user processes some kind of objects (text, image, table, etc.), usually allow the user to apply different menu-selectable operations. The user can apply the operations in different order. If the operations are applied to the objects with the same structure, it is normal to use the so-called **Object-Action** scheme in which the user first selects objects and then chooses an operation to apply to these objects. Such scheme is used, for example, in changing the font of a paragraph, copying text, etc. Most computer algebra systems also use this scheme.

However, when the arguments of different operations have very diverse structures (monomials, polynomial, parentheses, etc.), it is difficult to apply the **Object-Action** scheme. It is not clear before the operation is selected, what information needs to be entered to specify the object (whether the object is a monomial or a polynomial, an expression in brackets, and exponential expression, etc.). In this case, an **Action-Object** scheme is preferable, i.e. the user first selects an operation and then marks objects to which the operation will be applied.

This second scheme was also chosen for T-Algebra. As each rule can only work with certain types of objects (monomials, polynomials, expressions in brackets, variables, fractions, etc.), it should be clear beforehand, which objects ought to be marked. The **Action-Object** approach gives the possibility to mark parts of expression far removed from one another (for instance, similar terms that are separated by several other terms).

Finally, the chosen scheme is more suitable for working with the resolution algorithms used at school. The algorithm tells the student, what step should be taken next; this enables him to mark the required parts of the expression, and finally enter the result of the operation.

In T-Algebra, the **Action-Object** scheme was upgraded with a third component – entering the parts of the resulting expression (**Input**). This gives the student the possibility to participate in the solution process. And it enables the program to check the knowledge and skills of the student.

2.2. Selection of a rule of transformation

At the first stage of each solution step, the student has to choose the rule that he is going to apply. He would make the same decision also when using the school algorithm, but in this case he would usually not write this decision in the solution. When the teacher checks the solution, he has to understand, which rule the student wanted to apply.

It would be almost impossible to write a program that would understand what the student wanted. Consequently, a program can check only a limited number of attributes without the information on the intentions of the student. Practically, the only thing that can be checked is whether the expression in

the next line is equivalent to that in the previous line. This is the checking mechanism used by the Aplusix [1] program.

However, when the program has the information on which rule is applied, it is able to check a number of different attributes. Firstly, such information gives the program the ability to check, whether the selected rule is applicable, i.e., to determine the student's skill of choosing the correct rule. This enables the program to estimate, whether the student knows the algorithm used for solving this type of problems.

The second advantage given to the program by this information is that it can check more efficiently, whether the student's actions on the next stages of the step are correct: e.g., did the student mark the parts of the expression that are suitable for the selected rule, did he enter correct parts in the resulting expression, etc.

When the rule is selected and the objects are marked, the program has sufficient information to generate an expression for the next line, where the student has to enter manually some parts of the result. It is also possible for the program to calculate the expression in the next line automatically, because it has all required information. Thus, the program is able to check, whether the student entered correct parts in the next line. The user can also ask for help and let the program complete automatically the current stage or the whole step. Depending on the work mode, help features can be disabled. Information on errors can also be switched off – in this case, the program lets the student make errors in the solution. Such mode can be used for testing the knowledge of the student.

In the case, when the student makes a mistake and picks a wrong rule, the program does not immediately inform the student about the error. This gives the student a chance to correct the error without assistance. If the user cancels the step before starting to mark the parts of the expression, the error counter does not increase. The program does not proceed to the next stage, when the student has selected unsuitable rule or objects, because it would be impossible to generate the expression in the next line.

2.2.1. Set of rules

The algorithms defined in the school textbooks were followed as closely as possible in the design of the rules. Much support in constructing the rules was provided by school math teachers and authors of textbooks. Some rules can be applied in varying ways. For example, the program could complete the step automatically (study mode). In addition, there can be variations at the stage of entering the results – in some cases the student has to enter the result on the keyboard, in other cases the result should be constructed from the parts of the expression, etc.

We have attempted to make the rules *polymorphic* so that one and the same rule could be applied to several kinds of objects. For example, the rule of combining like terms can be applied to separately marked monomials, numbers (monomials in which all variables are in the power of 0), fractions, and also to larger selections of whole polynomials (consisting of all like terms). This gives the student the opportunity to apply a once learnt rule in different expressions and even in different kinds of exercises.

The designed set of rules must also be *complete*, i.e., all exercises in these particular fields of school mathematics should be solvable with these rules. The different rules in the program cover almost all definitions, operation descriptions and rules found in the textbooks. Several chapters in the textbooks contain multiple topics with each topic introducing its own particular rules and even specific types of exercises. For example, in the field of monomials and polynomials, every sub-topic introduces some new information and defines new rules.

Some rules are limited to a particular field, i.e., they are used only in the exercises of that field. This includes, for example, the rules for processing the whole linear equations, inequalities and systems of linear equations or parts of them (reversing the sides, moving terms to other side etc.), which are specific to this field.

Other rules have a more general character and can be used almost in any field. Such general rules are, for example, all arithmetical operations with numbers, clearing of parentheses, etc.

For user convenience, all rules are grouped according to their fields of application. Some rules can be disabled in some exercises. Thus, when solving problems related to the topics at the beginning of the curriculum, the rules required for solving more advanced problems can be disabled.

2.3. Marking the parts of expression

At the second stage of each step, the student has to mark the parts of the expression to which the selected rule should be applied. In the case of some rules, there is nothing to mark – in these cases the rule will be applied to the expression as a whole or applied automatically. An example could be the rule of ordering the monomials, which automatically orders all monomials in the expression. When solving problems on paper, the students usually do not mark the parts of expressions, except in specific cases (e.g., when combining like terms, they underline the like terms). When the teacher checks the solution, he must be able to understand the thoughts of the student.

In a program, such checking is practically impossible – if the program does not have information on which objects the selected rule was applied to, its checking ability will remain very limited. This kind of automatic checking is further complicated by the fact that in an exercise book, the student may include a number of transformations in one step and he may arbitrarily alter the sequence of parts (the laws of commutativity).

The marking of objects has been used in other teaching environments as well, e.g., Aplusix [1] and Felissa [3]. In Aplusix, a part of the expression (always only one) is marked before automatic simplification or factoring. In Felissa, two parts of an expression need to be marked sometimes.

However, the realization of marking in T-Algebra differs from marking in other environments. The main difference is that in T-Algebra it is possible to mark an unlimited number of terms of an expression and these terms can be located far from each other (and be separated by other terms).

In order to mark a part of an expression, with the Expression Editor in the marking mode (see Figure 2), this part should be selected (either with mouse or keyboard) just like in a regular text editor and then the user should press the  button. To remove marking, the student would have to select the same part and press the  button.  buttons are for moving between the marked parts. When the user has finished marking, he should confirm the selection by pressing the  button.

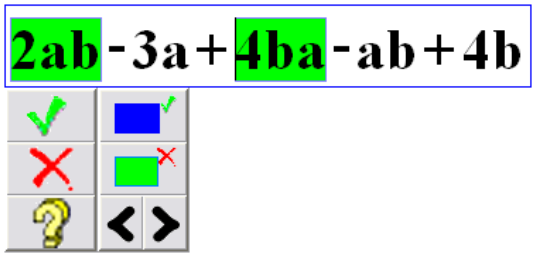


Figure 2. Expression Editor of T-Algebra program in the marking mode

The information on the objects of the rule applied at this step of the solution enables the program to diagnose a number of attributes. First, the program checks, whether a correct part of the expression has been marked. Clearly, there is no point in letting the user mark only a part of the number (e.g., mark only *2* from the number 123) or mark the operator after the number (e.g., mark *12+* in the expression *12+13*). The program allows marking various types of sub-expressions: whole numbers, single variables, monomials, expressions in parentheses, fractions or their combinations with operators. This preliminary check does not depend on the selected rule and is applicable with all rules.

Second, the program checks, whether the marked parts are appropriate for the implementation of the selected operation. This checking already depends on the selected rule. For example, when 'combining like terms' has been selected, the program begins by checking, whether the selected terms are monomials, whether these monomials are like (same variables in same powers), and finally, whether they can be combined (are in the same sum).

If the user has finished marking, he clicks on the confirmation button. The described checks are performed after pressing this button. If the results of a check indicate that the rule is not applicable to the marked parts, the user is informed about the error. In this case, the program does not enable moving forward with the solution, because it cannot generate the expression for the next step based on incorrect selection.

The program does not inform the student immediately after an unsuitable part of the expression was marked, because the student should have a chance to correct the error without assistance. He can always remove the incorrect marking before confirming the selection.

2.3.1. Possible problems with marking the parts of the expression

While designing the functionality of marking the parts of expressions, we had to solve three didactic problems. This subsection introduces all three problems and their solutions.

The first problem involved the dilemma, whether the program should require marking parts with or without their preceding operators. In Figure 3, the part on the left has been marked with the operator and the part on the right without the operator.

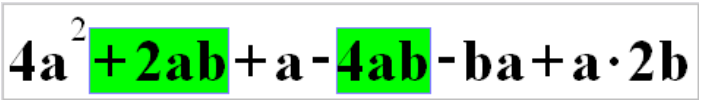


Figure 3. Marking the parts with and without operators

The sign before a term of an expression is very important and it can cause many errors in student exercises. When the students are required to mark the parts without operators, this will be problematic for the students, who want to mark these operators. A separate problem is a minus sign before the first part (or before a part in parentheses), which does not indicate an operation but a negative number. When the students are required to mark the parts with the operators, this will be problematic for other students, who have been used to marking only parts without the signs that stand before them. When the student underlines the parts in an exercise book, he does not draw the line exactly from one margin to another; sometimes the sign is included in the operation – if this has become a habit, this could cause problems.

This problem was solved by allowing the marking of parts both with and without the sign, while the program will always consider that the sign was marked. For example, when the parts are marked as shown in Figure 3, the program will generate in response the expression shown in Figure 4. It must be noted that the minus sign before the second part disappears and the student should calculate the answer before entering the result (which is natural), even though the minus sign was not marked.

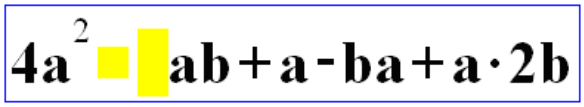


Figure 4. The field for entering the result after combining the parts marked in Figure 3

The second problem appears, when there is a need to mark larger blocks. It was debated, whether the user should mark each part separately or should he be allowed to mark them as a single item if they stand next to each other. An example of such expression is shown in the figure 5. This problem was solved by allowing the marking of parts as one large item, while the program itself will divide it into parts for further processing. However, when a wrong part is marked so that one half is from one part and the other from another part, the program reports a marking error.

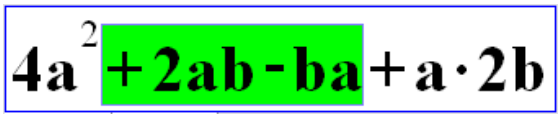


Figure 5. Marking of several parts as one item

The third problem was related to the sequence of operations. Namely, it is taught at the school that additions and subtractions and multiplications and divisions should be done from left to right. Yet, when the user understands operations correctly, he may apply them in any suitable sequence. This may be wrong didactically, but when solving the problems in an exercise book, we often make such simplifications. Consequently, it was decided to retain such possibility in the program. Figure 6 provides an example to illustrate this problem in the case of multiplication and division – it is easier to divide six by three and then multiply by two. In the result, the user should enter the multiplication sign and the result of dividing six by three.

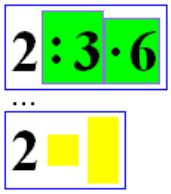


Figure 6. The problem of the sequence of operations

2.4. Entering of the result of the application of the rule

At the third stage of each step, the student should enter some parts of the expression that result from the previously selected operation. The program generates the expression in the next line based on the selected rule and marked parts, and leaves blank certain important parts of the new expression. In the case of some rules, there is nothing to enter – the program will automatically generate the expression in the next line. An example could be the rule of ordering the monomials, which automatically orders all monomials in the expression. When solving problems at school, the students always have to write an expression of the same length after the equality sign. Consequently, they try to reduce their workload by making several transformations at once. The program makes the work easier for the students by copying the parts of the expression that remain unchanged so that the students would have to enter only the parts that were modified. As a result, only one transformation can be made in each step. This makes it easier for both the teacher and the program to check the solutions.

The results can be entered on the keyboard or on the button panel. After the results have been entered, the entry must be confirmed by clicking on the  button. The parts of the expression that the student has to enter are highlighted with yellow boxes. A screenshot of the Expression Editor in the results entry mode is shown in Figure 7.

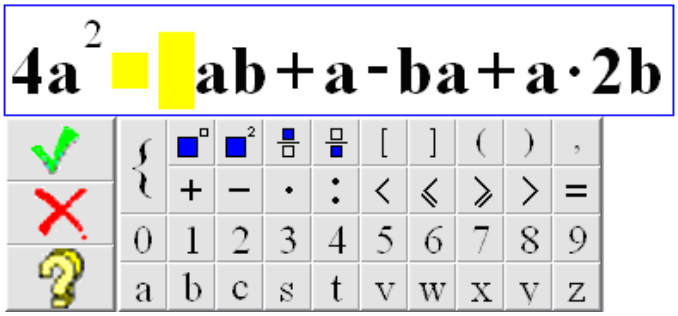


Figure 7. The results entry mode

The shape and the number of user-definable parts depend on the selected rule and marked parts. In the case of some rules, the operators (e.g., + and – signs for clearing the parentheses), the monomial coefficient and its sign (for combining like terms), the powers of variables, etc., need to be entered.

While entering the results, the program protects other parts of the expression from modification – the expression can be modified only in highlighted locations. This makes it easier for the program to check the solution and, in addition to checking the equivalence between the new expression and the previous step, it also enables checking the correctness of separately entered parts, thus improving the overall responsiveness of the program to errors.

This is precisely, how the program performs the checking: at first it checks, whether the new expression is equivalent to the previous one and whether the entered parts are equivalent to the parts calculated by the computer. If they are equivalent, no further checking is required. If the expressions are not equivalent, it is possible to check the correctness of each entered part to produce a more specific diagnosis. The program tries to determine, whether the student has made a standard error, which often occurs in the solutions of other students. The program has sufficient information (the rule and marked parts) to calculate the right answer on its own and compare it to the value of the part entered by the user.

If the user has finished entering, he clicks on the confirmation button. The described checks are performed after pressing this button. If it turns out that the entered expression is not equivalent to the previous one, the program may still let the student continue the solution (e.g., in the case of a test). Normally, however, the program informs the user about the errors and tries to identify the precise cause of the error.

2.4.1. Construction of the result using different components

In the case of some rules, it is very difficult to propose an expression in which certain parts would have to be entered. An example of this is the rule of polynomial multiplication. The teaching at school emphasizes primarily the way of compiling the result. In the program, a special mode has been designed for this rule (and some other rules) in which the students have to compile the result.

A separate window will be opened in which single parts of two polynomials are presented to the student. The student should form pairs of the products using the drag-and-drop technique. After the pairs have been formed, the student confirms his choice and the program returns to the solution window in which the student has to fill the gaps for plus and minus signs in between the created pairs as sown in Figure 8.

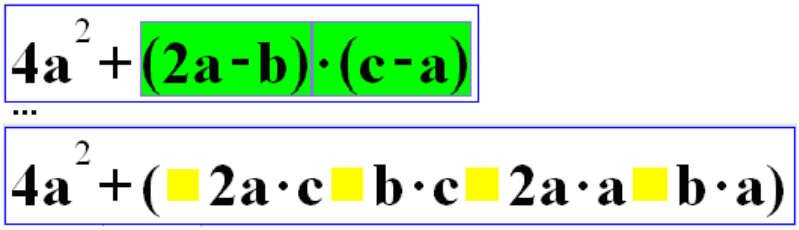


Figure 8. Entering the results after construction

3. Descriptions and examples of rules

Problem solving in the developed program takes place step-by-step and at each step, a particular operation – rule – is applied to the expression. A specific set of rules has been designed for each field. For example, let us consider the rules for equations, equalities and systems of equations. These operations are used at school to solve the aforementioned problems – we have designed the rules with the same names in our system.

1. *Evaluate*
2. *Order*
3. *Combine*
4. *Clear parentheses*
5. *Factor out common factors*
6. *Reverse the sides*
7. *Move terms to other side*
8. *Add to the sides*
9. *Subtract from the sides*
10. *Multiply the sides*
11. *Divide the sides*
12. *Substitute variable*
13. *Add equations*

Some of these rules are applicable in multiple fields (for example, the rules *Combine* or *Clear parentheses*), others are specific to this particular field (for example, the rules *Reverse the sides* or *Add equations*).

Let us take a closer look at some of the rules, paying particular attention to the ways of performing each operation. The description of operations uses a common scheme:

- how are the parts of the expression marked,
- what is controlled by the program,
- what and how should be entered after the marking has been confirmed,
- what does the program do automatically,
- an example of applying the rule.

3.1. Rule *Combine*

The program allows combining in a single step only like terms of the same type. Yet, all parts of the expression can be marked (including the ones that are inappropriate for combining). If some parts that are inappropriate for implementing the operation have been marked, this will be reported at the confirmation of the marking. The program does not require that all like terms of the same type should be marked.

After the marking has been completed, the expression is copied onto the next line and all like terms are replaced with one, which is highlighted with the yellow box instead of the coefficient. The student has to enter the coefficient manually. The program also asks the user to enter a sign before the new part in a smaller yellow box. The program uses these two parameters to evaluate the correctness of the step.

The program considers as like terms the parts in which the variables are in different order, even if the variables therein have not been grouped. It means, for example, that the parts $2ab^2$ and $3b^2a$ or $2a^3b^2$ and $4a^2bab$ are considered as like terms. If the marked parts had varying forms, then the program proposes the form of the variable of the leftmost like term.

Figure 9 shows an example of the application of this operation in which three parts have been marked in the initial expression, and in the result the student has to enter in the yellow boxes the sign and the coefficient, respectively.

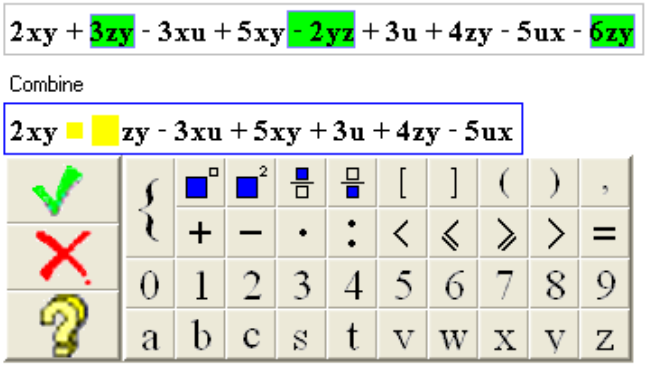


Figure 9. Applying the rule of combining similar terms

3.2. Rule *Move terms to other side*

The rule is applicable to linear equations and linear inequalities. According to the school algorithms, the parts containing a variable should be moved to one side of the equation (usually the left) and free parts should be moved to the other side by reversing the signs of all moved parts. The program allows moving the marked parts to the other side and does not require that the parts containing a variable should be moved to the left and free parts to the right. Moving can take place from both sides simultaneously.

In order to apply the rule, the student has to mark the addends that he wants to move to the other side. The program checks at the confirmation of marking, whether the selected parts are appropriate. When inappropriate parts have been marked, the program displays a respective error message.

If marking has been done correctly, the equation or inequality is written in the work environment onto the next line so that the selected parts with small yellow boxes are on the other side of the equality sign. The student should enter + or – signs in the yellow boxes. The correctness of these signs is checked, when the correctness of the step is evaluated.

Figure 10 provides an example of the use of this rule. In the initial equation, two addends on the left are marked for moving to the right and three parts on the right are marked for moving to the left. As a result, the next line displays an equation in which the moves have been accomplished. Before the moved parts are small yellow boxes that require a sign to be entered.

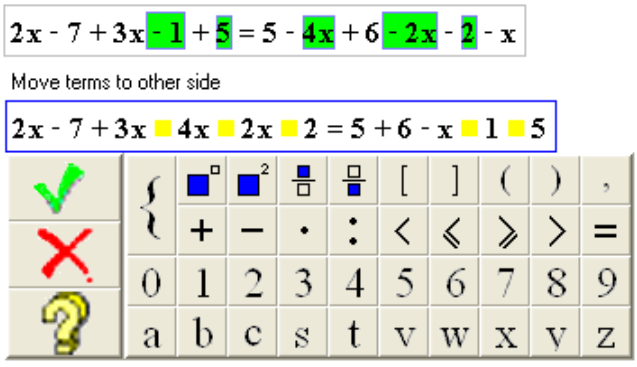


Figure 10.

Applying of the rule of moving terms to other side

3.3. Rule *Substitute variable*

This rule can be applied to the calculation of the value of an expression containing variables, checking the solutions of a linear equation, checking the validity of linear inequality and solving a linear equation system in two variables by substitution. In the case of a linear equation or inequality, the variable is simply replaced by a number; in the case of a linear equation system, the variable may be replaced by an expression.

In order to carry out the operation, the student has to mark one occurrence of the variable that is to be replaced. In order to solve an equation system, the variable should be marked in the equation in which it is to be replaced (thus, the replacement will not take place in the other equation).

At the confirmation of the marking, the program checks, whether the selected part is a variable (a respective message is displayed if it is not). If the selected part is suitable for the application of the rule, a small window will be opened to enter the number or expression that should replace the variable. If an expression is entered, the program checks, whether it is a number given in the text of the problem, or in the case of an equation system, whether the replaced variable is expressed through other variables in any of the equations.

After the entry has been confirmed, the program copies the expression onto the next line of the main window. All occurrences of the variable (in the case of an equation system, all its occurrences in the equation in which it was marked) are replaced with the entered expression, highlighted with yellow boxes, and the user is given the opportunity to correct the expression where necessary, e.g., by adding parentheses or a multiplication sign.

Figure 11 provides an example of the use of this rule. Let the problem be the following: 'Check, whether the number 2 is the solution of the equation' and the expression is $3a - a = 5$. In the equation, the variable a is marked, and in the small window the number 2 is entered. The line of the result provides the opportunity to correct the parts in yellow boxes (in this example, it is necessary to insert a multiplication sign in the first box between the numbers 3 and 2 to produce multiplication and not simply the number 32).

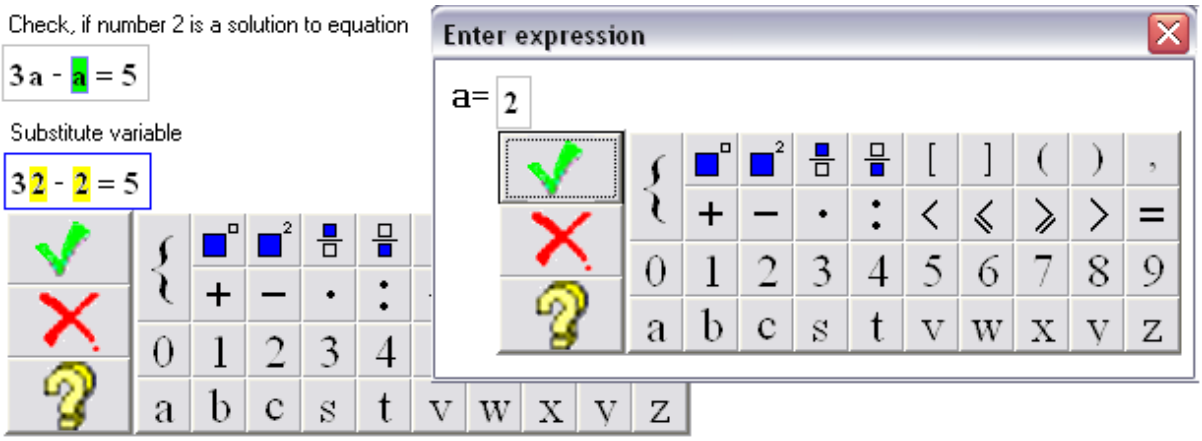


Figure 11. Applying the rule of variable substitution

References

1. http://aplusix.imag.fr/index-en.htm, last viewed 28.06.2004
2. http://epgy.stanford.edu/courses/math/, last viewed 28.06.2004
3. http://epgy.stanford.edu/TPE/index.html, last viewed 28.06.2004
4. http://www.tiigrihype.ee/eng/index.php, last viewed 28.06.2004
5. http://www.etf.ee/index.php?keel=ENG, last viewed 02.07.2004
6. D. Lepp. Program for exercises on operations with polynomials. 6th International Conference on Technology in Mathematics Teaching. Volos-Greece, October 2003. 365-369