

Error Diagnosis and Categorization in Problem Solving Environment Using Action-Object-Input Scheme

Dmitri Lepp

dmitri@ut.ee

University of Tartu, Institute of Computer Science
Estonia

Abstract: This paper describes the student error diagnosis component of the T-algebra problem solving environment. In T-algebra students solve different expression manipulation problems step-by-step using different transformation rules and following the three stage input dialogue. When designing the T-algebra we followed the principle that all the necessary decisions and calculations at each solution step should be made by the student. For making a single solution step the student has to choose a transformation rule from the list of available rules, select the operands (certain parts of expression) for this rule and input the result of application of the rule.

Introduced scheme gives the student possibility to make different kinds of errors. The errors can be made either when selecting the rule or unsuitable objects of the rule or while applying the rule – inputting the resulting expression. The program diagnoses different errors and attempts to identify the misconception underlying the mistake. The program checks whether suitable rule is selected, whether the form and number of selected objects and also form of entered result is suitable for the selected rule. T-algebra also checks for the correctness of every single part of the result entered into separate boxes. In general the introduced solution step scheme gives the program more information on the student's intentions than simple entering of the transformed expression. The program can explicitly and without guessing evaluate the decisions about operation and operands. The program can compute the result corresponding to the intentions of the student, and compare it with actual input. This makes the diagnosis of student errors more precise and easier to implement.

The paper also presents the categorization of errors types that are diagnosed. As an example two transformation rules are discussed (combining like terms, multiplication of polynomials) and lists of all diagnosable error types and common student misconceptions are presented. This categorization is what distinguishes T-algebra from other problem solving environments. We have designed 20 different categories and put all diagnosed error types into them. Categories include for example selection of objects of wrong form, selection of objects that are not compatible, errors in form of entered subexpression, calculation errors, errors in calculating the sign of entered subexpression and other categories.

1. Introduction

Expression manipulation problems are very important in school mathematics. Simplifying and factoring the algebraic expressions are such issues in the mathematics curriculum that pose difficulties to many students. 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. Using computer power may help teachers and decrease time spent on checking the assignments.

This paper presents the T-algebra environment, 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. When designing the T-algebra we followed the principle that all the necessary decisions and calculations at each solution step should be made by the student. For that purpose a three-stage solution step dialogue [7] was designed. According to this dialogue, to make a solution step in T-algebra the student has to complete the following three stages:

- choose a transformation rule from the list of available rules,

- select the operands (certain parts of expression) for this rule,
- input the result of application of the rule.

The student is able to make mistakes at each of the mentioned stages of the single step: choose the rule that is not applicable, select improper operands for the rule or make a mistake in calculating the result, etc. The introduced solution step dialogue offers good possibilities for T-algebra to diagnose all these errors. The T-algebra environment uses a categorisation of common student mistakes, so that it may identify the cause of a particular error. The categorisation has resulted from an empirical study that involved both students and teachers. Some results of this study in respect of the topic of solving linear equations are published by Issakova [5]. In addition, this empirical study provided information about the types of problems to be included in the environment, as well as the rules and solution algorithms needed for solving such problems.

The architecture of T-algebra environment follows the main line of Intelligent Tutoring Systems (ITS) architectures. T-algebra includes four major functional components of ITS which, according to Wenger [12], are Domain knowledge, Student model, Pedagogical knowledge and Interface.

The domain representation of the T-algebra contains knowledge of solution algorithms that are taught at school for the problem types implemented. It also contains knowledge of transformation rules that can be used for problem solving – T-algebra is able to apply all the rules automatically (it has the knowledge required for selecting the right operands and calculating the right result of application of the rule). This gives the program the ability to check the students' solutions and perform error diagnoses. The program interface and the dialogue between the user and the environment were described by Issakova et al. [6].

In this article we mainly describe the error diagnosis and the student's model implemented in T-algebra. The second part of the article gives an overview of related work on student modelling in other similar problem solving environments. The third part describes the T-algebra environment, solution step dialogue and gives some examples of applications of the rules. The fourth part describes the error diagnosis in T-algebra, presents the categorisation of possible errors and describes principles of student modelling in T-algebra.

2. Related Work

In many countries, the schools use computer algebra systems (Derive, Maple, MathCAD, StudyWorks etc.) to simplify the work 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, but they are not designed for the situation in which the student solves problems, makes mistakes, requires feedback and advice, etc. Therefore, we are developing the T-algebra environment, where the student can do all the above. There are some similar systems that are worth mentioning such as Aplusix [8, 9], EPGY Felissa [11] and MathXpert [2]. In this section we highlight the difference of our approach in comparison with these systems.

Unlike T-algebra the Aplusix system does not collect any additional information on the student's intentions in making the solution steps. The student simply enters the next line expression, so the program is able to diagnose only the equivalence of the expressions. In the Aplusix program the student decides himself how many changes to make to the expression in one line before moving to the next line. Recently the authors of the Aplusix program were engaged in research devoted to student's modelling with a lattice of conceptions [9] which should help to diagnose student errors.

Sets of correct and incorrect rules were designed and the heuristic search diagnosis algorithm implemented for the linear equation solving problems (which is also one of the topics in T-algebra).

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. The latter incorporate programs TPE (Theorem proof environment) and Felissa [11] 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 MathXpert system uses transformation by rules, but unlike in T-algebra the student does not have the possibility to make mistakes. When making a solution step the student selects the part of expression, the program offers a list of rules applicable to that selection and after the student has selected a rule the program automatically applies it. According to Beeson [2] “...the final design of MathXpert's user model emerged: it does not model any specific student, but rather an ideal student who is prepared to learn the selected topic”. MathXpert still contains a user model – it tailors the output depending on the student's level (entered by the teacher).

3. Problem Solving in T-algebra

In T-algebra students solve different algebraic problems step-by-step using transformation rules. Work in the T-algebra environment follows the same steps (rules) as the algorithms taught at school. However, unlike other rule-based environments T-algebra does not apply the selected rule automatically – the user has to enter the result of application of each rule. As mentioned in the introduction, each solution step in T-algebra consists of three stages: selection of the transformation rule, selection of operands and entering of the result of application of the rule. The introduced scheme gives the program more information on the student's intentions than simple entering of the transformed expression (like in Aplusix). This allows better diagnosis and program help.

Problem solving with the program is very similar to working with pencil and paper. Consider the following example: simplify the expression $2xz - 3x + z - 2x - xz$. 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 and write the resulting expression after the equality sign. The program follows principally the same scheme of action – the corresponding solution step consists of the following three stages:

- The student selects the rule of combining like terms from the list of available rules.
- The student marks (Fig. 3.1) all the monomials similar to x using the mouse and selection buttons – the program checks whether the selected parts of the expression are actually like terms and whether these terms can be combined.
- The program copies unchanged parts of the expression onto the next line and asks the student to enter the resulting monomial. After entering is confirmed the program checks whether the entered parts are correct and the whole expression is equivalent to the expression in previous line.

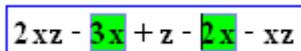

$$2xz - 3x + z - 2x - xz$$

Figure 3.1 Selection of objects for applying the rule

The input stage can proceed in three different modes (Fig. 3.2, left to right): free input, structured input and partial input. Possible input in different boxes is constrained by the type of the box. In figure (Fig. 3.2) input boxes are filled with the correct results. When applying the rules the student has to enter the result of application of the selected rule only, without any other simplifications.

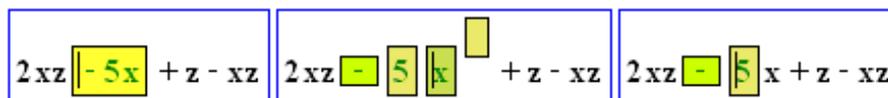


Figure 3.2 Input stage in three different modes when applying the rule Combine like terms

The following example (Fig. 3.3) shows different stages of application of the rule Open parentheses to expression $-3(2z-5)=4z-15$ in structured input mode.

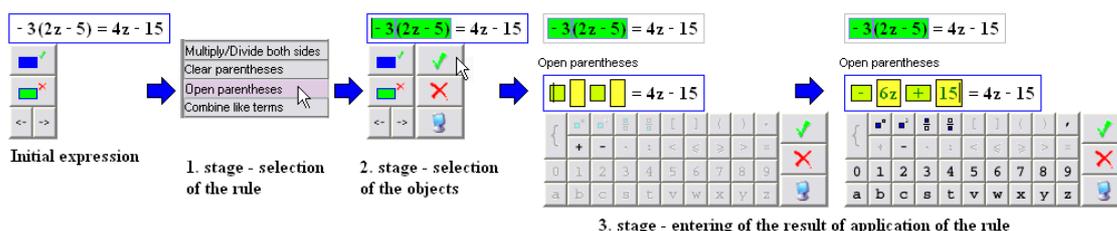


Figure 3.3 Three stages of the step

4. Error diagnosis and categorization in T-algebra

Student modelling as the model of a learner represents the computer system's belief about the learner's knowledge. The student model is the essential component in individualized learning via an intelligent instructional system that is able to adapt to the learner. In T-algebra we use the student model to help the teacher in the assessment of student's knowledge, so according to the survey [3] the student model in T-algebra is evaluative.

When checking paper assignments the teacher looks through the student solutions to find all the mistakes made and the number of correctly solved problems. Then the teacher evaluates the mistakes and correct solutions and calculates the final grade. T-algebra uses the same approach. In the case an error is diagnosed T-algebra searches for a possible misconception that can lead to such an error.

T-algebra uses an expert-based student model – the student model is assumed to be a subset of the expert model. First, the expert domain is modelled as a set of correct rules – rules implemented in T-algebra. The learner is modelled as a subset of these correct rules, plus a set of incorrect rules (or actually a set of typical errors made when applying these rules).

We had to collect some background knowledge for building a student model system:

- The correct procedures, algorithms, transformation rules, concepts, principles and/or strategies of a domain (called the theory of that domain).
- The misconceptions held and other errors made by a population of students in the same domain (called the bug library [1]).

For that purpose we searched through the results of the related works and conducted our own surveys [4, 5, 10]. School textbooks were reviewed in order to find all the rules used for making solution steps and algorithms for solving the problems. The students' paper works were studied –

which steps the students make while solving problems on paper, what are the common mistakes for most students when applying certain rules? When designing the input stage for the rules we tried to leave a possibility for making most of the common mistakes, which we identified. According to the survey [3] in actual practice it is generally impossible to have a complete bug library. Furthermore, the results of [10] suggest that different groups or populations of students (students from different schools) may need different bug libraries. Therefore, we focused only on the most common mistakes leaving the possibility to add more diagnostic procedures in future.

4.1. Error Diagnosis in T-algebra

A set of correct and incorrect rules is often used for building a student model (for example in [9]). In the case that free expression transformation is allowed in the program, it has to guess which correct rules were used to reach the resulting expression from the initial one. If the expressions are not equivalent then the heuristic search should be applied in order to find which incorrect rules were possibly used.

In T-algebra expressions are manipulated using the set of implemented rules. The system does not offer incorrect rules but the students can make mistakes when applying available rules: select a rule that is not applicable, select unsuitable operands for the rule or enter an incorrect result of the rule. The design of problem solving in T-algebra is such that as soon as an error is found the student cannot proceed to further stages of the step or to the next steps before the error is corrected. T-algebra uses the action-object-input step dialogue and it gives some advantages over environments with pure input:

- the program can explicitly and without guessing evaluate the decisions about operation and operands,
- the program can compute the result corresponding to the intentions of the student, and compare it with actual input.

T-algebra performs error diagnosis each time when the user confirms his input – after selection of operands for the rule and after entering the result of the rule. We prefer not to check whether the selected rule is applicable immediately after selection – the student is given the possibility to cancel the rule if he does not find suitable operands. Each time when an error is diagnosed the program displays an appropriate error message to the student, tries to diagnose whether the error is one of the typical ones (common misconception) and records it in the model of student.

After the student confirms his selection of operands the program first checks whether the rule is applicable to any operands in the current expression. Then prior to any rule-specific considerations the program checks whether all selected parts are syntactically (not mathematically) correct subexpressions. Syntactical mistakes are usually technical ones caused probably by the lack of attention, not by any misconceptions.

When the operands are syntactically correct and the program diagnoses an error (unsuitable operands) then it tries to identify a misconception that caused the error. Error can be caused either by misunderstanding of the order of operations (add some numbers or terms which are the parts of a product, etc.) or misunderstanding of a particular rule. A number of rule-specific checks are performed:

- the form of the objects is checked (for example, like terms for combining should be monomials),

- the positions of the objects are checked (for example, numbers for adding should belong to the same sum expression),
- some rule specific compatibility requirements are checked (for example, the condition of like terms, the condition of fraction that can be reduced, etc.).

The mistakes listed in examples are quite common for pupils. T-algebra can easily detect them because the student selects the rule and the operands explicitly. It would be very hard to diagnose such errors from the resulting and initial expressions only.

When applying the rules, the program copies the unchanged parts of the expression to the new line and protects them from modification. It only lets the user to enter the result of applying the rule – the user has to fill yellow boxes with the result or with essential parts of the result (Fig. 3.2). After the student confirms his input the program performs different checks on the correctness of the result. Having full information on the rule and objects, the program is able to apply the rule itself and compare the student's result with the correct one.

First, T-algebra checks the correctness of the syntax of the entered expression. Then the system checks the equivalence of the result entered by the student and the result calculated by the expert module of the system. The students solve problems in T-algebra by applying certain rules from the predefined set of transformation rules. When making the solution step the student has to enter the exact result of application of the rule at the input stage (only the modified part of the expression). This entered part of the result in most cases should be in a certain form (for example, when combining like terms it should be a single monomial). Therefore, after checking the equivalence T-algebra checks whether the entered part of the result is in the correct form. Next, a check of required extra symbols and brackets is performed (for example, when multiplying $2*(x-y)$ in expression $2*(x-y)*(a-b)$ the entered part of the result should be put into brackets). After these checks, if the entered expression is not equivalent to the correct one, T-algebra tries to diagnose the place and the cause of an error with more precision. Depending on the form of the result and the rule being applied T-algebra checks for equivalence of separate components – coefficient of a monomial, power of variable, etc. It also tries to identify a common misconception (for example, powers of variables are multiplied instead of adding when multiplying the monomials).

4.2. Categories of Errors

In order to build the student model we had to collect some typical errors (bugs) that student make in their solutions, in addition to rules. In T-algebra students apply transformation rules to make solution steps – so they learn different rules (the same that are presented in school textbooks) and practice applying them. T-algebra identifies erroneous applications of these correct rules and diagnoses misconceptions if possible.

We have structured all possible and diagnosable errors as a tree – each node in the tree corresponds to a certain error type and leaves in the tree correspond to possible misconceptions or to such errors where T-algebra is currently unable to identify a misconception (Fig. 4.1). The root of the tree is the most general error. The first level nodes represent all available rules – so errors are first categorised by the rules. Each transformation rule node has three child nodes (level 2) – errors and warnings in selecting the rule, errors in selecting operands and errors in entering of the result. The child nodes of the first node are errors in selecting a rule that is not applicable and warnings about selecting the rules that do not correspond to the current steps in solution algorithms. The child nodes of each 'errors in selecting operands' node (level 3) represent some rule-specific errors in

respect of the form, positions or other compatibility requirements of the selected operands and these nodes may have child nodes (leaves, level 4) with causes of these errors – misconceptions. The child nodes of each ‘errors in entering the result’ node (level 3) represent errors made in entering the result – the syntactical errors, errors in the form of the result, the non-equivalence, need of required extra symbols and some rule-specific errors. The non-equivalence errors of most rules are divided to errors in calculating some specific parts of the result (coefficient of a monomial, power of a variable etc.) and these in turn are divided to different misconceptions.

Each time an error is found the T-algebra error diagnosis module tries to identify the exact error and a misconception if possible, i.e., it tries to move the error in a tree to as low level (closer to the leaves) as possible. When an appropriate node in the tree is found that corresponds to the diagnosed error the model of a student is altered – the weight of a specific node is increased.

The set of all different typical errors and misconceptions that T-algebra is able to diagnose is too large to be presented here. Therefore, we have chosen two rules which have different forms of the resulting expressions and present the structured sets of diagnosable errors and misconceptions in the applications of these rules. These representative rules are *Combine like terms* and *Multiply the polynomials*.

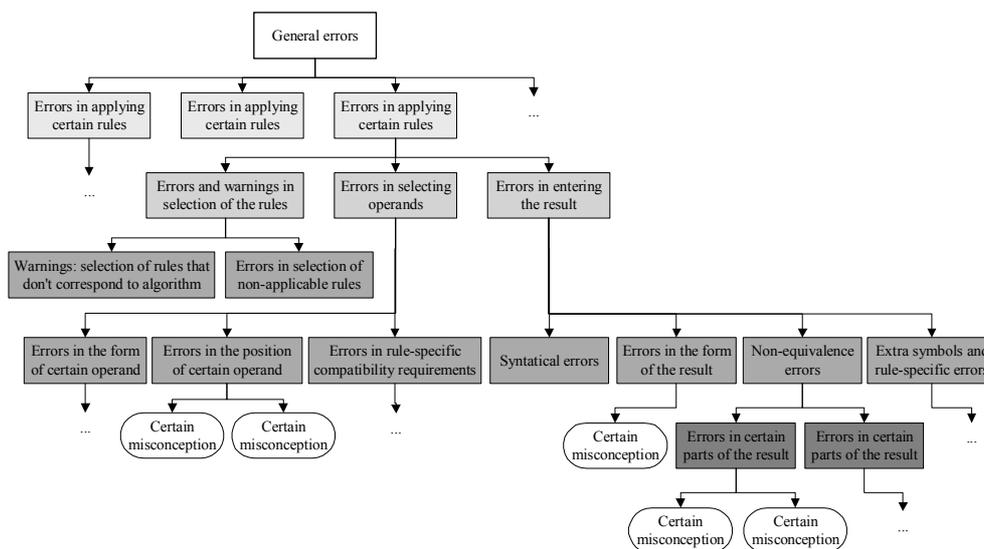


Figure 4.1 Tree structure of errors and misconceptions

First we present the errors that T-algebra is able to diagnose when student selects and applies the Combine like terms rule. In order to apply this rule the student has to select at least two similar monomials from one sum expression. More than two terms may be selected but T-algebra does not require selecting all the possible similar terms as operands of a single step. Only one group of similar terms can be combined at a single step (in $2x+y+3x+2y$ it is possible to combine either $2x$ and $3x$ or y and $2y$ at one step). After confirming selection of the operands T-algebra copies the remaining parts of the expression and asks the student to enter the resulting monomial. Depending on the input mode program asks (Fig. 3.2) the student to enter only a coefficient and operation sign or the whole monomial (either into one box or into a structure – set of boxes for coefficient, operation sign, variables and powers).

When selecting similar terms to combine the following errors and misconceptions are diagnosed:

- no operands selected,

- one of the selected parts is not a monomial,
- only one monomial is selected,
- one of the selected monomials is not similar to others,
- the selected monomials are not members of the same sum,
 - possible misconception of the order of operations $a+a*b$
- the selected monomial is in brackets (should be selected together with brackets).

When entering the resulting monomial, the following errors are diagnosed first:

- the result box is empty (but it should not be),
- the result is syntactically incorrect,
- the result is not a monomial,
- the operation sign is missing (but should be present),
- the monomial is not equivalent to the correct one.

The last error can be diagnosed further:

- the operation sign is incorrect (should be opposite),
- an error in calculating the coefficient,
 - possible cause: did not take “-“ into account,
 - possible cause: did not add all the coefficients,
- the monomial is not similar to operands.

The last error can be diagnosed further:

- a new variable is introduced,
- the power of the variable is incorrect,
 - possible cause: adding the powers of the variables,
 - possible cause: multiplying the powers of the variables.

The second example rule is Multiply the polynomials. In order to apply this rule the student has to select exactly two polynomials (with embracing brackets) in one product. Only one pair of polynomials can be multiplied at a single step. After confirming the selection of the operands the program copies unchanged parts of the expression and asks the student to enter the result (polynomial). Depending on the input mode the program offers different number and kinds of boxes: single box for the whole result (free input), boxes for coefficients, operation signs and powers of variables (partial input) or the structure for the first monomial and possibility to add more of them (structured input). The rule has to be applied exactly – combining of like terms is not allowed in the result.

When selecting polynomials for multiplication the following errors and misconceptions are diagnosed:

- no operands selected,
- one of the selected parts is not a polynomial,
- only one polynomial is selected,
- more than two polynomials are selected,
- the selected polynomials are not in the same product,
 - possible misconception of the order of operations.

When entering the resulting polynomial, the following errors are diagnosed first:

- the result box is empty (but it should not be),
- the result is syntactically incorrect,
- the result is not a polynomial,
- the operation sign is missing (but should be present),
- the resulting polynomial should be put into brackets,
- the student has probably combined similar terms (but he should not),
- the polynomial is not equivalent to the correct one.

The last error can be diagnosed further:

- the number of monomials in the entered result differs from the number of monomials in the correct result,
- the correct result does not contain any monomial similar to the one entered by the student,
 - it is probably similar to a monomial calculated on the basis of the misconception that the powers of the variables should be multiplied,
- the correct result contains a monomial similar to the one entered by the student but the coefficients are different.

We have designed 20 different categories and put all diagnosed error types into them. Categories include for example selection of objects of wrong form, selection of objects that are not compatible, errors in form of entered subexpression, calculation errors, errors in calculating the sign of entered subexpression and other categories. When solving the problems it is possible to review statistics on all errors made (numbers of errors of each category) as well as error situations themselves.

4.3. Student Model and Assessment

The student model in T-algebra consists of two components: the model of student knowledge of the rules and the model of errors. The student model records proficiency scores for each issue – rule correctly applied, error of certain type made. The proficiency score shows the level of confidence of the program in anticipating that the student would apply certain rules correctly or make certain kinds of mistakes. T-algebra holds proficiency score for each of the available rules as well as for each node in the tree of error types. T-algebra uses its own heuristics for updating the scores each time when a certain rule is applied correctly or any kind of error is diagnosed. A history-based model is used, so the teacher can track the student's performance over time.

In addition to proficiencies the teacher can add his own scores for each correctly solved problem and penalty scores for different error types. When the students solve problems T-algebra calculates the score and the teacher can grade the student work according to this score. When grading the student work the teacher can review the whole solutions, the errors made (number of errors of different types and the erroneous situations) and the model of the student. When displaying the number of errors made by the student, T-algebra is able to collect similar errors from different rules and display them as a single sum (for example, errors in the order of operators, syntactical errors or arithmetic errors).

5. Conclusions

In this paper we presented the T-algebra problem-solving environment with its integrated error diagnosis and a student model that is intended to help the teacher in assessing the students.

As the T-algebra environment is not finished yet we could not conduct any thorough studies and evaluate the error diagnosis of T-algebra to ensure that the environment is indeed effective in helping the students to solve the problems. We have already made a series of trials with students and collected some positive results of using T-algebra – error diagnosis procedures were found useful and error messages helped the students to correct the mistakes.

Although the development of T-algebra is still in progress, we foresee it as a promising tool for learning, teaching and testing the skills of solving expression manipulation problems.

Acknowledgements The author was supported by Estonian Doctoral School in Information and Communication Technologies.

References

- [1] Baffes, P., Mooney, R. (1996). Refinement-Based Student Modeling and Automated Bug Library Construction. *Journal of Artificial Intelligence in Education* 7(1) 75-116.
- [2] Beeson, M. (2002). MathXpert: un logiciel pour aider les élèves à apprendre les mathématiques par l'action. *Sciences et Techniques Educatives* 9(1-2). English translation 'MathXpert: Learning Mathematics in the 21st Century' is available <http://www.mathcs.sjsu.edu/faculty/beeson/Papers/English-ste/English-ste.html>.
- [3] Diogene (2002). Survey on Methods and Standards for Student Modelling. Public project report of the DIOGENE project. Available at <http://www.crimpa.it/diogene/archive.html> (last viewed 26.09.2006).
- [4] Hall, R. (2002). An Analysis of Errors Made in the Solution of Simple Linear Equations. *Philosophy of Mathematics Education Journal* 15.
- [5] Issakova, M. (2005). Possible Mistakes during Linear Equation Solving on Paper and In T-algebra Environment. *Proceedings of the 7th International Conference on Technology in Mathematics Teaching*, Vol. 1. Bristol, UK 250-258.
- [6] Issakova, M., Lepp, D., Prank, R. (2005). Input Design in Interactive Learning Environment T-algebra. *Proceedings ICALT-2005: The 5th IEEE International Conference on Advanced Learning Technologies*. Kaohsiung, Taiwan 489-491.
- [7] Lepp, D. (2005). Extended Solution Step Dialogue in Problem Solving Environment T-algebra. *Proceedings of the 7th International Conference on Technology in Mathematics Teaching*, Vol. 1. Bristol, UK 267-274.
- [8] Nicaud, J., Bouhineau, D., Chaachoua, H. (2004). Mixing microworld and cas features in building computer systems that help students learn algebra. *International Journal of Computers for Mathematical Learning* 5(2) 169-211.
- [9] Nicaud, J., Chaachoua, H., Bittar, M., Bouhineau, D. (2005). Student's modelling with a lattice of conceptions in the domain of linear equations and inequations. *Proceedings of AIED 05 Workshop 1: Usage Analysis in Learning Systems*. Amsterdam, the Netherlands 81-88.
- [10] Payne, S., Squibb, H. (1990). Algebra malrules and cognitive accounts of errors. *Cognitive Science* 14 445-481.
- [11] Ravaglia, R., Alper, T., Rozenfeld, M., Suppes, P. (1998). Successful pedagogical applications of symbolic computation. *Kajler, N. (ed.): Computer-Human Interaction in Symbolic Computation*. Springer-Verlag, Berlin Heidelberg New York 61-88.
- [12] Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann Publishers, Los Altos, CA.