Designing Next-Generation Training and Testing Environment for Expression Manipulation

Rein Prank, Marina Issakova, Dmitri Lepp, and Vahur Vaiksaar

University of Tartu, Institute of Computer Science, Liivi Str 2, 50409 Tartu, Estonia
{prank, marinai, dmitri, vax}@ut.ee

Abstract. T-algebra is a project for creating an interactive learning environment for basic school algebra. Our main didactical principle has been that all the necessary decisions and calculations at each solution step should be made by the student, and the program should be able to understand the mistakes. This paper describes the design of our Action-Object-Input dialogue and different input modes as an instrument to communicate three natural attributes of the step: choice of conversion rule, operands and result.

1 Introduction

When the student solves an expression manipulation task, he should at each step

1. Choose a transformation rule corresponding to a certain operation.
2. Select the operands (certain parts of expressions or equations) for this rule.
3. Replace them with the result of the operation.

For proper learning of this difficult area an environment should be available where all the necessary decisions and calculations at each solution step would be made by the student, and the environment would be able to understand the mistakes.

Traditional classroom cannot be such environment because the teachers are not able to discover and correct the mistakes in time. Existing software does not address the spectrum of arising problems in whole complex. On the one hand, some programs require from the student only execution of the first and (partially) the second substep. Using computer algebra systems, the student selects at best only the transformation rule and a part of the expression; the transformation itself is made by the computer. In addition, computer algebra systems have only a small number of very powerful operations (Simplify, Solve, ...) that enable to get the answer of a task in one step but not to build traditional step-by-step solution. The latter deficiency has been overcome in rule-based learning environments [1, 6] that have much more detailed rules. But they also perform the selected transformations mainly automatically.

Some other programs (for instance APLUSIX [4]) use paper-and-pencil like dialog design where a transformation step consists purely of entering the next line. However, such input does not provide the program with information about the decisions made by the student at stages 1 and 2. As a consequence, practically the only error that can be diagnosed is the non-equivalence with the previous line.

In the University of Tartu our first environment for step-wise expression manipulation (in Propositional Logic) was implemented about fifteen years ago [5]. The step

dialogue was built using Object-Action scheme. The student had to mark some subformula and enter the result of conversion (there was also a working mode with conversion rules in the menu). The program was able to verify separately selection of operand and performed conversion.

Using our long positive experience, we try now to integrate the above-described rule-oriented and input-oriented approaches. In 2004 we received funding from Estonian School Computerisation Foundation ‘Tiger Leap’ and launched a project, called T-algebra, for expression manipulation tasks in four areas: calculation of the values of numerical expressions; operations with fractions; solving of linear equations, inequalities and linear equation systems; simplification and factoring of polynomials.

2 General Design of Solution Steps – Action-Object-Input Scheme

Working with T-algebra, each step consists of three substeps:

1. Selection of the operation from the menu,
2. Marking the operand(s) in expression,
3. Entering the result of the operation.

The following figure demonstrates third substep after selection of rule *Collect like terms* and marking of appropriate operands.

Fig. 1. Entering the result in free input mode

How much does the amount of input grow in comparison with simple entering of the result? Our step dialogue requires from the student two preceding actions. First of them is only one click on the mouse. At the second substep the student has to mark one or more parts of the expression. As a compensation, the program is able to copy the rest of the expression automatically to the next line. If the dialogue would consist only of the third substep of our step, the student should enter the passive parts of the expression from the keyboard or mark and copy them one by one. This means that addition of the second substep does not change the amount of the work.

For third substep the program has three input modes: *free*, *structured* and *partial* mode (Figures 1 and 2 demonstrate corresponding input areas).

In free input mode the program generates one input box inside of the expression in the next line. The student should enter in the box one expression replacing all marked parts from the previous line.

In structured input mode the information about the actual rule and operands is used to create the pattern of the result automatically, using different input boxes for signs, coefficients, variables and exponents.

Partial input mode is a modification of structured input where the program fills some boxes automatically and leaves to the student only the boxes for the components that are crucial for particular rule (signs, coefficients, exponents).

Fig. 2. Input areas for $6ab^2$ in structured and partial mode

There are a few transformations that need some addition to our general input scheme. For example, in the case of multiplication of polynomials the program creates only one monomial-structured group of boxes and our virtual keyboard contains a button for adding the next group. In the case of addition of fractions in two structured modes we decided to ask in separate window and check separately the common denominator, etc. Exceptions are described in [3].

3 What Does T-Algebra Check?

T-algebra has no checkpoint after selection of the rule, and first two substeps are checked together. It means that if the student has selected impossible operation, then he has the possibility to cancel the step himself. If he selects irrelevant operand(s) as well then T-algebra diagnoses a marking error. If the selected rule does not correspond to “official” algorithm then the program displays a warning because in many cases the order of application of the rules is insignificant but current version of T-algebra does not implement any deeper analyse.

Errors in marking (when not misclicks) tend to refer to serious gaps in student’s understanding of the whole task, priority order of operations or of selected rule. T-algebra checks the following items:

1. Are the marked parts syntactically correct subexpressions?
2. Do the marked terms have the form required for operands of selected rule? (The like terms for combining should be monomials etc.)
3. Do the marked terms satisfy compatibility requirements for selected rule? (For example, are they really like terms?)
4. Do the marked terms satisfy location requirements (belong to the same sum or product etc)?

The input of third substep is checked in the following order.

1. Is the entered expression syntactically correct?
2. Has the input the structure required by the result of application of the selected rule?
For example, the result of multiplication of monomial with polynomial should be the sum of monomials and the result of combining like monomials should be one monomial.
3. Is the entered expression equivalent to the marked (i.e., replaced) part and is the result of substitution equivalent to the previous line?
4. For some rules the program checks yet whether the operation was really performed. For example, $6/12$ cannot be reduced to $60/120$ or $5/10$ or $6/12$. Some details of error checking in equation solving tasks are described in [2].

4 Conclusion

Our main interest has been such a dialogue from where the program could understand the solution steps of the student and give intelligible feedback for weaker students who tend to make many mistakes. In our first version of the program we are not able to realize all the potential of our interface and to implement very detailed diagnosis. We start from giving some level of feedback and add further details if the experiments demonstrate that the pupils really need more explanations. But we have seen that implementation of the items listed in Section 3 was quite easy and the error messages that refer to the rules and operands were helpful for the pupils.

References

1. Beeson, M.: Design Principles of Mathpert: Software to support education in algebra and calculus. In: Kajler, N. (ed.): Computer-Human Interaction in Symbolic Computation. Springer-Verlag (1998) 89-115
2. Issakova, M.: Possible Mistakes During Linear Equation Solving on Paper and in T-Algebra Environment. In: Proceedings of the 7th International Conference on Technology in Mathematics Teaching, Bristol, UK, Vol. 1 (2005) 250-258
3. Lepp, D.: (2005) Extended Solution Step Dialogue in Problem Solving Environment T-Algebra. In: Proceedings of the 7th International Conference on Technology in Mathematics Teaching, Bristol, UK, Vol. 1 (2005), 267-274
4. Nicaud, J., Bouhineau, D., & Chaachoua, H.: Mixing microworld and cas features in building computer systems that help students learn algebra. *International Journal of Computers for Mathematical Learning*, 5(2), (2004) 169-211
5. Prank, R.: Using Computerised Exercises on Mathematical Logic. *Informatik-Fachberichte*, Vol. 292, Springer-Verlag (1991) 34-38
6. Ravaglia, R., Alper, T., Rozenfeld, M., Suppes, P.: Successful pedagogical applications of symbolic computation. In: Kajler, N. (ed.) *Computer-Human Interaction in Symbolic Computation*, Springer-Verlag (1998) 61-88