The Eurasia Proceedings of
Educational & Social Sciences (EPESS)
ISSN: 2587-1730

The Eurasia Proceedings of Educational & Social Sciences (EPESS), 2014

Volume 1, Pages 361-366

ICEMST 2014: International Conference on Education in Mathematics, Science & Technology

A PROBLEM GENERATOR SYSTEM TO LEARN FIRST-DEGREE EQUATIONS

Mir Mohammad Reza ALAVİ MİLANİ
Karadeniz Technical University, Computer Engineering Department

Hüseyin PEHLİVAN
Karadeniz Technical University, Computer Engineering Department

Sahereh HOSSEİNPOUR
Karadeniz Technical University, Computer Engineering Department

ABSTRACT: Problem-based learning enhances academic productivity, and improves long-term memory. Thus it is better than traditional instruction. Meanwhile, the teaching of mathematics education is more important than others. Much software is developed for visual learning mathematics, but there is a great need for problem-based systems. This is more visible, due to the increasing proliferation of e-learning education. On the other hand, first degree equations are a uniquely important topic in high school and collage algebra classes, for the simple reason that mastery of a preponderance of later topics requires a student's ability to solve these equations. Such topics include absolute value equation, equation containing fractions, radicals, and an abundance of applications. Proficiency at solving first degree equations in one variable is literally essential to success in an algebra class. Students often do not clearly understand the concepts of these topics and make mistakes when they write homework's or use these concepts in the other topics. In order to help the students to learn these concepts by solving problems, we have proposed a system that generate problems and evaluate learner's answers. In this paper we propose a methodological approach for automatic solving of mathematical equations, especially in terms of the first degree equations in one variable, with the aim of the practicing of mathematics subjects, by using of Computer Algebra System (CAS) tools. The paper also addresses some specific fields such as the simplification and automatic production of mathematical equations.

Key words: First-degree equation, problem-based learning, computer algebra system, problem generator, problem solver

INTRODUCTION

According to the research conducted, it can be argued that the problem-based learning is more effective than traditional learning, and increase students' ability to solve the problems (Farnsworth 1994) On the other hands textbooks cannot be suitable as a source of questions, because they are limited, and not Interactive. Also, textbooks do not usually solve the problem step by step and don't have appropriate visual features for motivation. In these resources, often there aren't facilities for on line helps to students. Therefore, Use of information and computer-based technologies in education, specially physics (e.g.,CAPA) (Kashy et *al*. 1993), mathematics (e.g., Mathway) (Dragon, T et *al*. 2013; Gutierrez-Santos S. et al. Oct.-Dec. 2012) or electronics (e.g., CHARLIE) (Barker, 1997) is widespread. Also can be seen that emerged systems for generating and solving the questions (Wolfram 2011, Bridgeman et *al*. 2000;May 2000; Baldwin, 1996)

Also, due to the importance of mathematical education and necessity to produce numerous questions on different topics in this suject, the importance of using Computer Algebra System (CAS) in this section can be shown. Creation infrastructure for the production of educational software in mathematics, studies on this type of software production methods and production and implementation of software for mathematics education is the future cases that should be considered by such topics scholars.

In this article, we will describe a system that is designed to teach the first degree equations in mathematics. This system has two parts: Tutorial and Test, and with taking deferent levels generate any questions automatically and

- Selection and peer-review under responsibility of the Organizing Committee of the conference

*Corresponding author: Mir Mohammad Reza ALAVİ MİLANİ- e-mail: milani@ktu.edu.tr

gives to users. Step-by-step solving of generated problems, and guidance to the user, in order to correctly solve the problem are the other facilities of this system.
We used java programming language in design of our system. Because of this language is independent from operating systems and can be used in all devices and even the Internet. Also we used JavaCC for parsing of mathematical expression. The proposed system can be used as part of the personalized educational system or used in the e-learning systems as a tool for practice and self-examination.
In the next sections we will discuss more about methodology and generate of first-degree equation parse tree using JavaCC. Also we will dedicate to our automatic procedure for generating first-degree equations. Then, we demonstrate the user interface and educational plan for our proposed method.

METHODS

Proposed Method

For implementation of proposed system, the following three sections will be designed

- Production of first-degree equations
- Solving of first-degree equations
- Designing of User Interface and Appropriate Plan

Equations should be randomly generated with considering level of difficulty in its. Also, for obtaining the result of equation and to get a user guide, we need to create parse tree and automatically solving by the computer.
Below, each of the sections will be explained in detail.

Equation Generation

Discussed system generates an equation as a binary tree. The terminal nodes of the tree contain *Num* or *Var* operands and non-terminal nodes contains *Plus* or *Times* operator. Algorithm of tree generation is recursive and based on grammars that shown in list1.

S → E = E
E → E + E | Num | Var
E → E * E´ | E´ * E
E → (E)
E´ → E´ + E´ | E´ * E´ | Num
E´ → (E´)
Num → [0-9]*
Var → ´x´

List1. CFG Grammar for First degree Equation Generation

Using the rules in list1, we can obtain infinite equation. Figure 1 show one example of generated equation in tree structure.

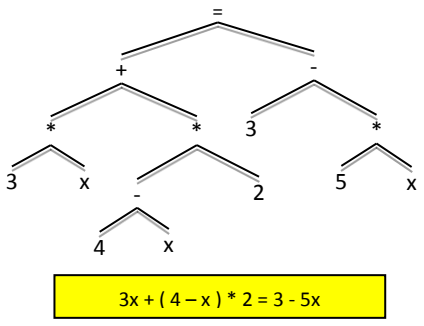

Figure 1 Example of an equation that generated by grammar in list1

According to rules of grammar in list1, the size of generated equations is purely randomly and may be very long. To solve this problem, we can be assigned a certain value as probability for each rules of grammar. The value of

some rules will be increase or decrease associated with the level of nodes in tree. Qua the probabilities of terminal nodes will be increased and the probabilities of non-terminal nodes will be decreased. Thus with increasing the size of tree, probabilistic of generating terminal node will be increase too.
Notice that, to generate equations with specefic complexity, we can use parameters for increase/decrease of probabilities amount. Thus in system we can get degree of difficulty from users and generate desired equations.
Generated equation can be stored in a database or displayed to the user. For obtain the string of equation from tree, we must be traversed tree in in-order like fashion. Consider that the equation is evaluated by traversing of original tree thus we should have already saved a copy of the tree. Also, the intermediate results are stored in the nodes of the tree and used to check the user's answer. The x-coordinate of the operands are also stored in the nodes, and are used to verify the under braces in user's answer.

Equation Solving

In this paper, a CAS-like system is presented for the step-by-step solution of problems. The core calculation mechanism of the system, which is based on symbolic calculation techniques, can be coded by using a Java-like programming language. The input data of the system are textual expressions used for the representations of mathematical problems, which can be parsed via a compiler-compiler tool such as JavaCC.
In this paper we present a method to calculate the first-degree equation symbolically by using the common tools of compiler designs as well as utilizing the CAS methods. Before addressing how it is done, however, first the equations received from the user should ideally be examined in terms of the verified entry (the rules related to a first-degree equation) and the resulting process should be transformed to a tree structure known as parse tree. Therefore, we examine the suggested system in separate phases with the following:

Parse Tree Construction

The general form of an expression is grammatically defined using a context-free grammar (CFG). Since JavaCC generates LL (k) parsers, this grammar must be LL (k) one. The design of a scanner and parser is based on the lexical and phrase structure of mathematical expressions. Thus, a CFG grammar is developed to recognize and parse such equations. In addition to the form of the input data the grammar must also represent the appropriate priority level and associativity of operators. This task is followed by the conversion of the grammar to the LL(k) one. The resulting LL(1) grammar is shown in Table 1.
A scanner splits the input data into a sequence of tokens which are the atomic parse elements of that input data.

Table 1: An LL (1) grammar for first-degree equations.

Rule	Method
S → E "=" E $	parse()
E → ("+" | "-")? T E´	expr()
E´ → ("+" | "-") T E´ | λ	
T → Num ("x")? | "x" (Num)?	term()
Num → (D)+ ("." (D)+)?	
D → ["0"-"9"]	

Simplification

The next task is the simplification of the resulting tree, which is one of the most important and underlying operations in symbolic computation systems. This has many difficulties in the implementation because some concepts of simplification are naturally challenging. For example, responding the question "which of the states shown in the following figure is simplified?" justifies this claim.
In this step, reusing the tree obtained from the previous step, all nodes associated with the tree are visited again; this time the simplification operations are performed for every node. For example, if visited node has an operator +, which is connected to two child nodes with numbers, the sum of these two numbers are returned as the simplified form of the node. The Figure 2 represents this example.

```
if OpNode instanceof Pluse then
        If RChild and LChild instanceof Num then return Num(RChild.val()+LChild.val())
```

```
End if
```

Fig. 2 Pseudo-code for a node with two *Num* child

In the above state, if both child nodes are of the same variable, the addition operation can be done as Figure 3. Note that, in the Figure 3 pseudo-code, the class *Var* is used as a variable, and the coefficient of variable x is placed inside the *Var* objects. For example, *Var*(3) is used to represent the expression 3**x*. The simplification stage can encounter some particular expressions. To illustrate this, let us consider more complex expressions represented by a tree such as

Plus(*Var*(5), *Plus*(*Num*(3),*Var*(2)))

The represented expression is 5**x*+3+2**x*. Since the child nodes are not of the same type, the parent one is not able to simplify, while the expression is equivalent to the simplified one 7**x*+3, represented by *Plus*(*Var*(7),*Num*(3)). This problem can be resolved by the transformation of the binary tree into multiple one and then applying the simplification. Figure 1 shows the case graphically. In this way, the simplifying function can generate the simplest form of the expression.

```
if OpNode instanceof Plus then
    If RChild and LChild instanceof Num then
 return new Num(RChild.val()+LChild.val())
    If RChild and LChild instanceof Var then
       return new Var(RChild.val()+LChild.val())
End if
```

Fig. 3: Pseudo-code for a node with two *Num* or *Var* Child

Presentation function

At the each stage, the equation tree should comprehensively and suitably be transformed to a mathematical expression to be represented. For this, a function should be designed that takes a tree as an input and then returns a mathematical expression related to that tree as a string. Performed by using the concepts of classes and recursive calling, by meeting every node and depending on the operator type of the node, this function generates and returns appropriate strings. Of course, it should be noticed that for representing mathematical expressions, the appropriate use of parentheses is of specific importance, which should specifically be considered in the design of the function.

First-Degree Equation Calculator

In this step, having the underlying function, a tree related to the first-degree equation solution can conveniently be created. Figure 4 show a flowchart for first-degree equation solving.

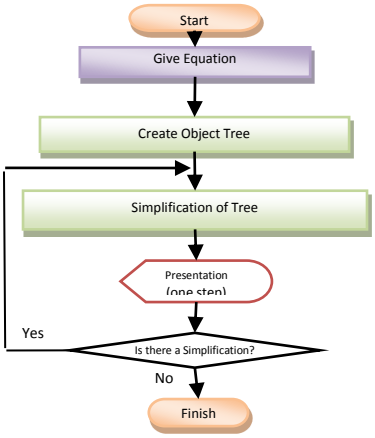

Fig. 4: Flowchart for First-Degree Equation Solving

According to the flowchart above, having received an equation from the user and having generated an initial parse tree, the simplification and presentation function must be called in each round of loop. Thus we can show step-by step solution of equation (one step in each cycle).

User Interface and Educational Plan

The user interface of proposed method can be allows the user to choose from different levels of difficulty and system will be generate equation based on specific level. The default is intermediate level of difficulty. Our system presented an equation that generated based on user's preference of difficulty. The user is expected to solve the equation by specification and writing intermediate results. Entered intermediate answers will be evaluated by system and feedback will be display properly. To obtain appropriate results, the user may enter either the tutorial or test mode to solve the presented equation.

Tutorial Mode

In this mode, the user may operate by repeatedly clicking on the "Next Step" button. On each click, system displays a step of solution followed by the result of one simplification. Therefore, the user can observe the sequence in which equation is solved, and learn this subject.

Test Mode

In this mod, the user in order to solve the problem must enter intermediate results as describe before. The user may have her/his answer checked at any stage of solving the problem. Each stage of answer will be evaluate by system and provide a feedback to user. And if the answer is correct, user can be continuing to her/his task otherwise this stage should be repeated by user.

CONCLUSION

In this work, a CAS-based system was developed for teaching and automatically solving problems related to first-degree equation . By the step-by-step representation of a problem, students can be helped to learn this mathematical subject. Also, the possibility of generating different solutions to a problem and automatically generating a similar new problem to the problem solved can improve students, learning capabilities. So, students will be able to automatically access the infinite number of questions, without spending cost and time, and observe their different solutions step by step. This feature helps them increase their ability to solve a problem, and prepares them for tests. On the other hand, utilizing the system teachers will be able to produce and use diverse questions based on templates designed for each grade, saving time.

REFERENCES

Baldwin, D. (1996). Three years' experience with gateway labs. ACM SIGCSE Bulletin, 28(SI), 6-7.

Barker, D. S. (1997, November). CHARLIE: A computer-managed homework, assignment and response, learning and instruction environment. In Frontiers in Education Conference, 1997. 27th Annual Conference. Teaching and Learning in an Era of Change. Proceedings. (Vol. 3, pp. 1503-1509). IEEE.

Bridgeman, S., Goodrich, M. T., Kobourov, S. G., & Tamassia, R. (2000, May). PILOT: An interactive tool for learning and grading. In ACM SIGCSE Bulletin(Vol. 32, No. 1, pp. 139-143). ACM.

Bridgeman, S., Goodrich, M. T., Kobourov, S. G., & Tamassia, R. (2000). SAIL: a system for generating, archiving, and retrieving specialized assignments using LATEX. ACM SIGCSE Bulletin, 32(1), 300-304.

Dragon, T., Mavrikis, M., McLaren, B. M., Harrer, A., Kynigos, C., Wegerif, R., & Yang, Y. (2013). Metafora: A web-based platform for learning to learn together in science and mathematics. Learning Technologies, IEEE Transactions on,6(3), 197-207.

Farnsworth, C. C. (1994) Using computer simulations in problem-based learning. In proceedings of Thirty Fifth ADCIS conference, Omni Press, Nashville, TN, 137-140

Gutierrez-Santos, S., Geraniou, E., Pearce-Lazard, D., & Poulovassilis, A. (2012). Design of Teacher Assistance Tools in an Exploratory Learning Environment for Algebraic Generalization. Learning Technologies, IEEE Transactions on, 5(4), 366-376.

Kashy, E., Sherrill, B. M., Tsai, Y., Thaler, D., Weinshank, D., Engelmann, M., & Morrissey, D. J. (1993). CAPA-An integrated computer-assisted personalized assignment system. American Journal of Physics, 61(12), 1124-1130.

Wolfram, S. (2011). Wolfram Alpha. 2011. URL: http://www. wolframalpha. com. Stand, 25