Theorema-based TI-92 Simulator for exploratory learning

Youngcook Jun
RISC (Research Institute for Symbolic Computation)
A-4232 Castle of Hagenberg, Austria
Currently at
Department of Computer Education, Sunchon National University, Korea
ycjun@sunchon.sunchon.ac.kr

Abstract

One of the Theorema system's capabilities provides a computational session that enables a developer to simulate an existing graphing calculator such as TI-92. Moreover, the deductive reasoning facility of Theorema allows the simulator to deal with propositional and predicate logic for pedagogical purposes. We present how to apply the use of such a simulator to help students explore mathematical ideas in terms of the blackbox/whitebox principle. This experimental approach is demonstrated with integrated modes of computing, solving and proving. This paper is motivated by how to encourage students to explore his/her own mathematical thinking based on algorithmic and logical operations built into the Theorema system.

1 Introduction

For mathematics learning and teaching, many computer programs are available in schools. Among them, Computer Algebra Systems (CAS) are adopted to help students manipulate algebraic expressions and graphical representations for mathematical objects [5, 6]. Even though most CASs generate immediate answers for given input expressions as a black box, students are encouraged to focus on manipulating various expressions with different problem contexts so that they can engage in high-level thinking by reducing low-level computational burden [8]. In this regard, Buchberger suggested how to use CAS for educational purposes in terms of a black box/white box principle [1]. The transition from the use of black box to that of white box can be followed by a creativity cycle that consists of the several steps: 1) work on various examples 2) try to find any patterns out of them to make any conjecture 3) try to verify the conjecture 4) come up with knowledge bases (i.e., theorems) in terms of building algorithms 5) try one of those algorithms as a black box to new examples, and so on. In this paper, it is assumed that exploratory learning takes place in the steps of 1) and 2). During this exploratory learning, students pursue different layers of mathematical ideas with many examples, already earned concepts and algorithms by experimenting and operating concrete math objects in a constructive way.

To investigate this exploratory learning phase more deeply, we try to build a simulator for algebra (Derive) part in TI-92 with Theorema. Theorema is an integrated suite for teaching logic and mathematics with collection of general and special provers [3]. It can generate automated theorem proofs in a natural language style presented in Mathematica notebooks. Besides the capability of proving, solving and simplifying, Theorema's language constructs are dealt with mathematical expressions that are similar to the conventional mathematics. With Theorema's data structure such as tuple notation , it is relatively easy to manipulate algebraic expressions, logical deduction and theorem proving. For the current work, we only try to simulate Derive part of TI-92 with keyboard layout so that the future version will generate sequences of keystroke maps for various simulated commands. The internal representation of Theorema will deal with keystrokes of a user's action in particular learning mode such as exploratory learning to capture and verify any emerging patterns in terms of equivalent relations and equality. This direction suggests how to demonstrate integrating CAS and automated deduction for educational purposes as an experimental apparatus.

2 Design Principles

The design of this system consists of three parts considering Theorema's distinct features such as computing, solving and proving. The computing part is carried out in the computational session while proving part is done in the proving session. The current Theorema does not provide a rich set of solving mechanisms yet. Derivations in each problem solving are separately designed in this simulator. Derivations are just a collection of intermediate steps without explicit verbalization in a whitebox sense. The reason behind this is that output screen of TI calculator is limited for detailed verbal descriptions for those intermediate steps. How to design such a simulator to mimic the behavior of TI-92 calculator is closely related to Theorema's capability mainly by integrating its computational and proving session.

Given a fixed set of commands and menu clicks, how to compose different commands to come up with pedagogically meaningful output? This viewpoint concerns the technical aspects of manipulating TI-92 keystrokes to help students attain the level of understanding. As students gradually experiment with algebraic expressions and numbers with possible derivations in certain cases, they probably get into the stage of guessing some properties of the target mathematical objects. Such conjecture making seems to motivate the students to attempt to verify certain properties with proving capability of Theorema as a blackbox. The whitebox part of Theorema can be available as a separate Mathematica notebook. With the help of Theorema's built-in provers, the students can confirm or disconfirm their conjectures in the simulated screen. This process guides them to make a concrete algorithm with which it can be served as a blackbox to do experiment with other sets of data. Such a process is described in terms of a spiral, called creativity cycle. This view guides us the main aspects of designing the TI-92 simulator.

3 How to build Derive Simulator with *Theorema*

In Theorema, the user can define both logical and mathematical definitions with various quantifiers in computational and proving session. Based on Mathematica [9], Theorema's logical inferencing mechanism allows users to work on conventional (abstract) and computational (concrete) mathematics. The main part of our simulator is coded by a recursive procedure that feeds an input expression one by one. Each expression triggers one of the "ti92-state" rewriting rules to yield an intermediate state to mimic Derive's algebraic commands. With this new intermediate state, another input is fed for the next rule match. This evaluation part is carried out in the Theorema's computational session. The output is written in Mathematica codes for making different box styles with colors to mimic TI-92 screen layout. The current version of the simulator is only close to algebraic manipulation part without a graphics mode. The next sample code illustrates how Theorema's codes look like for the main part that is based on recursion.

```
Definition["TI simulator",
  any[ key, keys, state ],
  TI92↵[⟨key, keys⟩] := TI92↵[⟨key, keys⟩, ⟨⟨ ⟩, ⟨ ⟩, ⟨ ⟩, ⟨ ⟩, ⟨ ⟩, 0, _⟩]
  TI92↵[⟨⟩, state] := state
  TI92↵[⟨key, keys⟩, state] := TI92↵[⟨keys⟩, ti92–state[key, state]]
]
```

```
TI92↵Result[keys_] := TMAout[
    Compute[TI92↵[keys], using → ⟨Definition["TI simulator"], Definition["ti92–state"]⟩]]
```

3.1 Algebraic Manipulation

Most of the content materials tested in this simulator are from written manuals developed by TI and Kutzler[7]. It is noted that some expressions (i.e., equality sign) are not identical with TI-92 expressions. The frame layout is quite elastic since there is no screen layout controller at this point. The whole part of current version covers three directions: numerical, algebraic and logical expression. In Fig 1, various examples are depicted in PC environment. The following code shows how 3 different TI commands (solve, csolve, factor) and 1 built-in Mathematica command (Together) are executed with a simulated screen.

```
TI92↵Result[⟨EX["solve[x^3−1=0]"], "ENTER", EX["factor[x^3−1]"], "ENTER",
    EX["propFrac[(x^3−2x−5)/(x−1)]"], "ENTER", EX["5x−6=2x+15|x=7"], "ENTER"⟩]
```

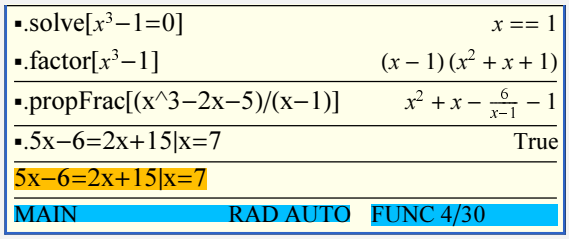


Fig. 1. Theorema's call for simulated TI commands and output layout

From this prototyping, it turns out that most of the Mathematica's commands can emulate Derive's algebraic commands with slight modification. However, some Derive's commands are not available in Mathematica. In such a case, it is programmed with Mathematica language as shown, for example, in the case of proper fraction, "propFrac". It is noted that for the educational purposes, simulated TI commands (eg., solve) only generate real values whereas "csolve" generates complex valued-solutions. Other commands deal with conditional checking with a certain value for a given variable with "|" substitution symbol. This equational checking is also available for experimenting various numerical substitutions for an expression at hand.

3.2 Derivations for equation solving

Equation solving in TI-92 is usually carried out by rewriting terms without showing intermediate steps as a black box. This feature can be modified into a white box considering cognitive fidelities of the students according to different learner levels. In fact, low-level and intermediate level students seemed to prefer following intermediate solving steps [4]. With TI-92, however, students can manipulate different ways of transforming algebraic rules explicitly by extra work on their own. For example, Kutzler introduced several ways to isolating a target variable to find a solution given a system of linear equations. Our simulator rather directly generates a sequence of algebraic transformation including an answer all at once (see stepSolve). Currently, the number of transformation rules is less than 10 since we try to capture general patterns associated with rewriting rules as minimally as possible [5].

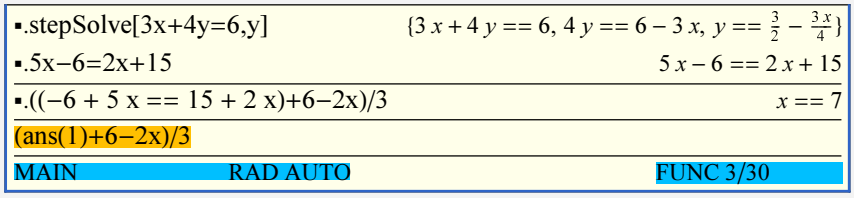
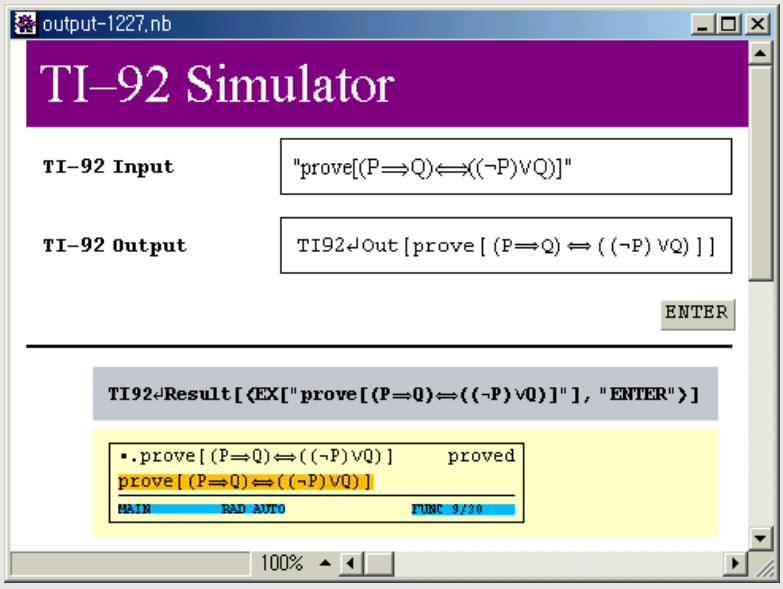


Fig. 2. Commands for step-by-step algebra rewriting

3.3 Propositional Calculus with *Theorema*

The most distinct feature of our simulator comes from the addition of PredicateProver that is a special prover embedded in Theorema. In this experimental version, we try to connect algebraic manipulation with propositional calculus so that students can go through different representations in school mathematics. Due to the fact that propositional calculus is derived by Boolen logic, students can practice logical inference when a certain property occurs during their work along with a creative cycle. This helps students to make a conjecture considering the fact that such a conjecture might be better represented by predicate calculus. As Theorema generates step-by-step natural language illustration in each step of proving, students can grasp how logical deduction goes without much difficulty. A frontend interface to the simulator send a request for proving to one of the Theorema's provers to generate verbalized logical explanations in terms of a proof tree (Figure 3). In this way, students can invoke graphing calculator part as well as logical deduction part to explore deep sides of mathematics. The following figures presents how propositional statement is proven in user-friendly interface. The target task is: prove[(P ⟹ Q) ⟺ ((¬ P) ∨ Q)].



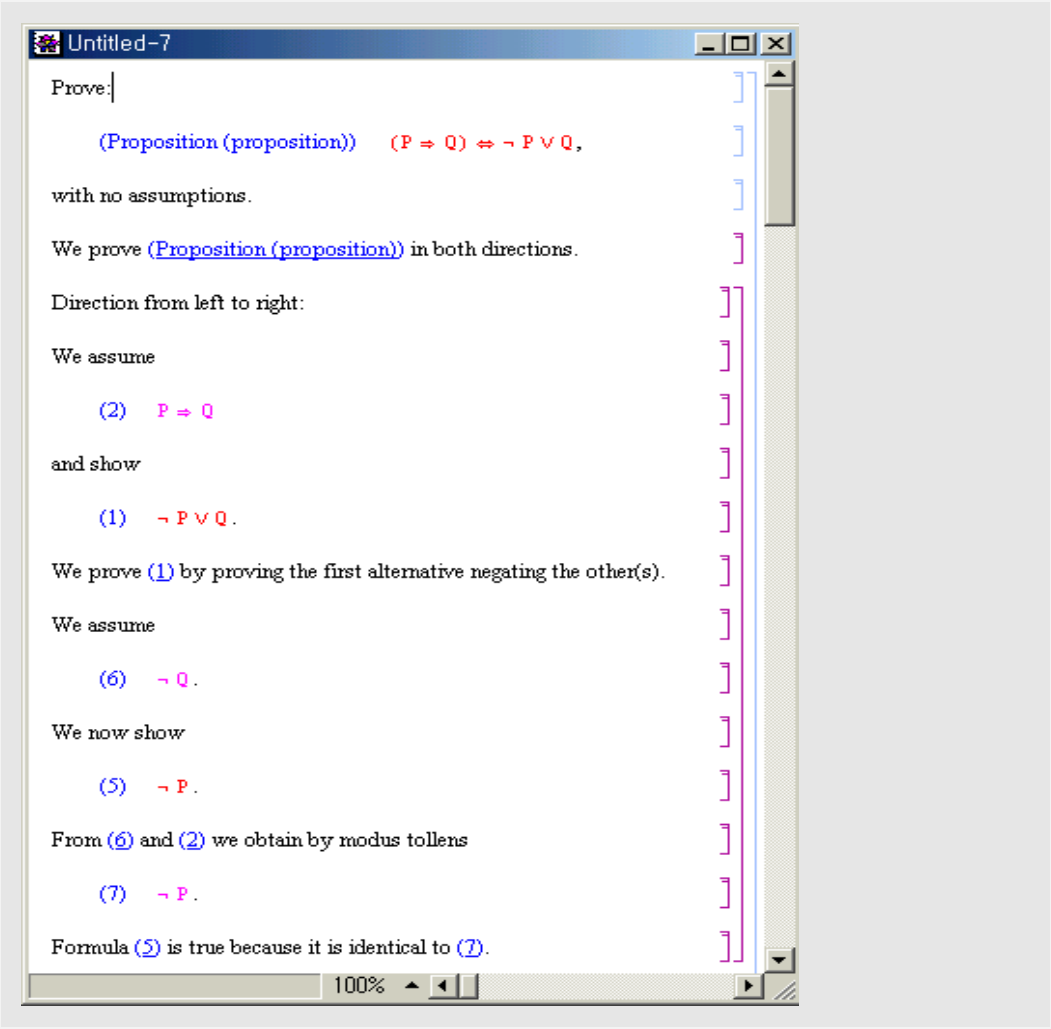


Fig. 3. The simulator's screen and PredicateProver's step-by-step proof

4 Exploratory learning with syllogism tasks

According to Buchberger [2], the use of Theorema can be applied to theory exploration rather than proving isolated theorems. Considering the fact that secondary mathematics textbooks contain only several dozens of theorems in geometry, proving is not regarded as main mental activities in secondary schools. However, proof training is indispensable for high-level mathematics. This aspect implies the importance of making students be engaged with reasoning tasks. With this motivational thing in mind, we consider traditional logical tasks, called syllogisms. There are three methods to deal with them: diagram-based, equational logic and predicate calculus. All of these methods might be difficult for high school students. However, students can visualize their thinking with diagrams and translate mathematical thinking into algebraic formula. At the end, they can verify their conclusions by implicitly calling PredicateProver available in Theorema system. By reading step-by-step annotations in each proved text, students can understand how each proof is derived.

Compared to expert mathematicians who explore mathematics ideas from scratch to grasp coherent ideas in an integrated way by working with each part (layer) of the whole procedure, students usually tend to follow a trial-and-error approach. In doing so, they can use formulae, diagrams, equations to find meaningful derivations. In this paper, it is assumed that algebraic transformation rules with predicate calculus can lead students to explore various properties as they accumulate algorithmic thinking. For example, a student might want to draw a valid conclusion for the following task: “All philosophers are professor. All professors are logicians.” To tackle with such a task, one may use a Venn Diagram by abstact set notation. Given a statement, “All As are Bs. All Bs are Cs.”, the valid conclusion, “All As are Cs,” might be easily drawn either propositional calculus or a Venn Diagram.

Another interesting method is to transform such syllogism premises into equations. The following equations are those in this case: A*B=A, B*C=B. Then A=A*B=A*B*C=A*C, so we have A*C=A which means All As are Cs. To verify this conclusion, we can call Theorema’s predicate prover by converting two premises into predicate calculus form. As for another task, “All A are B, Some B are not C”, the internal code for checking the conclusion “Some A are not C” looks like in the following way with the result “proved”:

```
Proposition["AabObc→Oac", any[A, B, C], (A ⊆ B ∧ B ⊈ C) ⇒ A ⊈ C]
```

```
Prove[Proposition["AabObc→Oac"],
    using → ⟨Definition["All A are B: Aab"], Definition["Some A are not B: Oab"]⟩,
    by → PredicateProver] // Last
```

```
proved
```

The main prover, called PredicateProver, usually starts with negating the conclusion to find any occurred contradiction. In some cases, the prover produces “failed” result, which means the conclusion can not be proved. Also notice that there are several syllogisms that do not have valid conclusions. In those syllogisms, it is rather difficult for students to encode two premises in a correct way to draw any conclusion. In such cases multiple representations are very helpful to assist their thinking. In syllogism tasks, there are two ways of represent syllogisms: equational logic and Venn diagram. For the first representation, syllogisms are represented in algebraic forms just like high school algebra solving. The simulator part can also produce step-by-step solutions using equational rewriting method. Another representation,.Venn diagram, requires rather sophiscated implementation details that execute conversion process from a Venn diagram to algebraic equations and vice versa. The implementation of this Venn diagram conversion process is left as part of logico-graphic feature available in the future version of Mathematica.

With versatile features available from TI92 simulator, it is then up to students who can touch upon several layers of math contents to investigate any meaningful properties with several provers embedded in the Theorema. In this way, computer algebra in TI-92 can be combined with Theorema’s solving and proving functionality to promote students’ explarotary learning. The overall demonstration of this simulator indicates future work to enhance several things including interface, algebraic expressions, interactive solving with feedback and meta-rules for invoking Theorema’s several provers.

5 Conclusion

This paper illustrates how the combiniation of computer algebra system and automated deduction system can be applied for pedagogical purposes. With the assumption that secondary mathematics learning can be directed to student-oriented exploratory learning instead of passive learning, it is the aim of this study to design and develop TI-92 simulator that can mimic TI graphics calculator based on Mathematica and Theorema. Even though computing and solving is dominant in most parts of secondary school mathematics, the current trend of using CAS suggests various ways of utilizing it to promote students to engage herself/himself in more conceptual levels of mathematical thinking. Theorema's computational and proving capability has shown that how TI calculators can be implemented for pedagogical purpose with research-oriented CAS such as Mathematica and Theorema. The future study will deal with verification method that ensures pedagogically and logically sound ground in each problem solving step. The emphasis of exporatory learning cast in this paper only suggests how CAS-oriented simulator can be pedagogical tool that can help students to pursue his/her own ideas in an intergrated environment of CAS and automated deduction system.

There are two things that need reflective thoughts for the use of CAS and automated deduction for pedagogical purposes layed out in this paper. First, characteristics of students' exploratory learning might be different compared of mathematician's exploratory style. More detailed analysis of student-centered exploratory activities are required. This aspect might shed light on how to design CAS with possible use of automated deduction for school mathematics. The second point reflects which equivalent commands or methods are directly appealing to students' exploratory learning. For example as shown $\text{prove}[(P \Longrightarrow Q) \Longleftrightarrow ((\neg P) \vee Q)]$, the propositional statement can be checked with truth tables available in TI-92 calculator for school mathematics even though TI-92 simulator can provide predicate calculus styles. From students' perspective, it seems very important for them to select the most appealing commands out of equivalent classes given a comand or method.

Acknowledgements

This work has been carried out in the frame of the Theorema project at the RISC institute, supported by the "Spezialforschungsbereich for Numerical and Symbolic Scientific Computing" (SFB F013) at the University of Linz. I thank Dr. Bruno Buchberger who helped me to prepare this paper.

References

[1] Buchberger, B. (1990). *Should students learn integration rules?* ACM SIGSAM Bull.(24), 10-17.

[2] Buchberger, B. (1999). *Theory exploration versus theorem proving*. Mathematica Notebook for the invited talk at the Calculemus Meeting, Trento, July 11, 1999.

[3] Buchberger, B., Jebelean, T., Kriftner, F., Marin, M., Tomuta, E. ,& Vasaru, D.(1997). A Survey of the Theorema Project. *Proceedings of ISSAC'97*, *W. Kuechlin (ed)*, ACM Press, 384-391.

[4] Jun, Y. (1995). *Learning how to solve linear equations by teaching the computer: Development and formative evaluation.* Unpublished doctoral dissertation, University of Illinois at Urbana-Champaign.

[5] Kajler, N. (Ed.) (1998). *Computer-Human Interaction in Symbolic Computation.* Springer-Verlag/Wien.

[6] Karian, Z. (Ed.) (1992). *Symbolic computation in undergraduate mathematics education.* MAA Notes 24, Mathematical Association of America.

[7] Kutzler, B. (1997). *Introduction to the TI-92*. bk teacheware.

[8] Tall, D. (Ed.) (1991). *Advanced mathematical thinking.* Kluwer Academic Pub.

[9] Wolfram, S. (1996). *The Mathematica Book*. Wolfram Media and Cambridge University Press.