Stimulating Students' Creativity Through Computer-Supported Experiments and Automated Theorem Proving

Wolfgang Windsteiger
RISC Institute, JKU Linz, Austria

Introduction

We present an environment for learning and teaching mathematics that aims at inspiring the creative potential of students by enabling the learners to perform various kinds of interactive computer experiments during their learning process. Computer interactions are both of visual and purely formal mathematical nature, where the computer-algebra system *Mathematica* powers the visualization of mathematical concepts and the tools provided by the mathematical assistant system *Theorema*, which is also based on *Mathematica*, are used for the formal counterparts. We present case studies on equivalence relations and polynomial interpolation, in which we demonstrate the entire bandwidth of computer-support that we envision for modern learning environments for mathematics.

Motivation

Computers are used in mathematics education typically in order

- to perform computations that would be too time-consuming for a student to be done by pencil and paper, e.g. solve a 10x10-system of linear equations,
- to perform operations that a student is not able to at a certain level of education, e.g. solve a system of non-linear equations, or
- to visualize/animate mathematical objects, e.g. an animation of the sequence of Taylor polynomials depending on their degree.

However, such crucial mathematical skills like “the precise use of formal language”, “conjecturing mathematical properties”, “rigorously proving or disproving conjectures”, or “inventing mathematical algorithms” are often attributed to the human ingenuity or creativity and therefore their development is considered not supportable by computers.

We develop course material including *interactive experiments*, where in a first phase students are led to the discovery of mathematical properties or algorithms. In a second phase, they must precisely formulate their discoveries, and in a third phase they use the *Theorema* system to attempt an automated proof of the statement. The *Theorema* system gives a nicely structured human-readable proof in case of success, but even in case of failure it displays the incomplete proof. This allows students to reformulate their conjecture or to modify the knowledge base leading to an iterative process, in which they finally create a correct statement together with its correctness proof. For details on the *Theorema* system, we refer to [TMA].

We want to demonstrate these ideas in two case studies on equivalence relations and polynomial interpolation, respectively. The material is distributed in form of *Mathematica* notebooks, and *Mathematica* is the natural integrated user interface both for studying the material and for working with *Theorema*[1].

1 Of course, we assume that students have a modest understanding of the language of

Case Study: Equivalence Relations and Set Partitions

The material on equivalence relations and set partitions introduces binary relations on a universe and their properties such as reflexivity, symmetry, and transitivity. Next we introduce the concepts “class”, i.e. the set of all elements related to a given one, and “factor set”, i.e. the set of all classes. As soon as we present a new notion, we provide interactive visualization tools (so-called “widgets”), in which a student can experiment with the new entities in order to explore their properties.

Widgets are Java applications, which are linked to *Mathematica* through *JLink*, a communication protocol that allows to connect Java applications to a *Mathematica* kernel. Using JLink it is quite easy to setup graphical user interfaces (GUI) with typical interactive elements such as buttons, pull-down- or pop-up-menues, scrollbars, or text fields like in common modern user interfaces. In the background, however, these GUIs employ *Mathematica* for computations and graphics. In the case of relations and classes, several widgets are provided that use colors to visualize classes. The elements, whose class is to be displayed, the relation, and the universe can be chosen by the student and the widgets update with each user action. In some widgets, students may choose among predefined reflexive/symmetric/transitive relations, an example of a widget is shown in Figure 1.

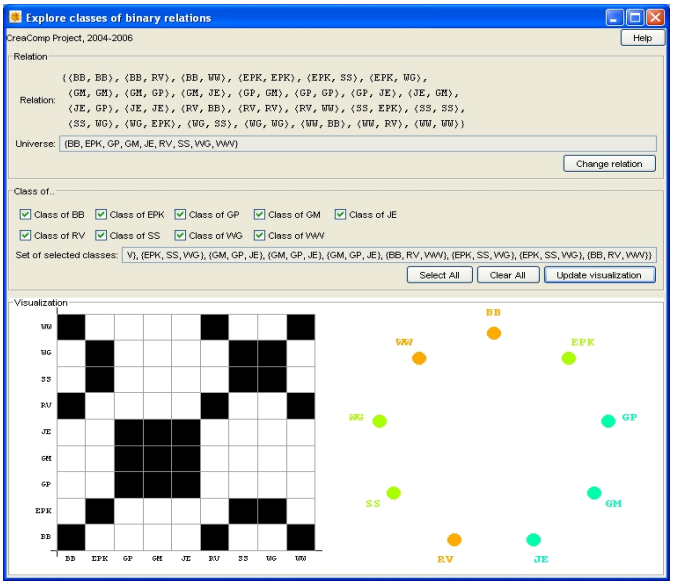

Figure 1: Example of a GUI for equivalence relations

The experiments with these tools establish an imagination about classes and their dependence on the relation's crucial properties. The classes' properties that are highlighted are non-emptyness and the “covering property”. After playing with these widgets a user could conjecture that for *equivalence relations* all classes are non-empty and each element is contained in exactly one of the classes, which is one of the key properties about equivalence classes. For a proof of this theorem, we proceed step by step using and proving several intermediate results on the way. In a first step, we concentrate on relexive relations only and the respective widget supports the idea that every element should be contained in it's own class. Whatever conjecture a student will make, we require them to state it *formally* using the *Theorema* language. In order to facilitate the input, *Mathematica* provides input palettes for special mathematical symbols, the *Theorema* system offers additional specialized palettes for mathematical formulae, and the learning material

mathematical formulae and they underwent training in basic handling of *Mathematica* and the *Theorema* system.

already contains templates that the students are only required to complete. For instance, after having interactively explored classes of reflexive relations, the students would be confronted with a Lemma-template of the form

Lemma["in own class", any[A,R], ...]

where "..." is a placeholder for any formula to be entered by the student. The *Theorema* system can now be employed to prove the lemma named "in own class". In *Theorema*, the call of an automated prover needs the statement to be proven, a knowledge base, the prove-method, and sometimes additional optional parameters to be specified by the user. From the point of view of didactics, the most interesting aspect here is the specification of the knowledge base, which we emphasize by giving students the opportunity to experiment with different knowledge bases, while keeping all other parameters of the prove-call fixed. Therefore, after completing the the lemma-template above, the student can press a prove-button, which will first ask the user to compose a knowledge base before trying a completely automated proof. Again using GUI technology, arbitrary knowledge available in the current *Theorema* session can be adjoined to the knowledge base, and of course, the resulting proof heavily depends on the chosen knowledge. In case of an unsuccessful proof, the user can inspect the failing proof and then adjust the knowledge base, ask the system for a hint, or revisit the theorem. In the concrete example above, a student might complete the template to

Lemma["in own class", any[*A*, *R*], $\forall_x x \in class_{R,A}[x]$]

thereby forgetting the side-condition to force R to be reflexive on A. The proof of this statement cannot succeed, and the incomplete *Theorema* proof will display the reason for failure: for an arbitrary set A_0, arbitrary $x_0 \in A_0$, and arbitrary relation R_0 on A_0 we need to show $\langle x_0, x_0 \rangle \in R_0$. The knowledge base, however, only contains the definitions of "class" and "reflexivity", whereas nothing is known about R_0. From this proof and an augmented Lemma-template, the student might get the insight that

Lemma["in own class", any[*A*, *R*], $reflexive_A[R] \Rightarrow \forall_x x \in class_{R,A}[x]$]

is a better way to formulate what has been observed in the interactive experiments earlier. The automated proof of this version of the Lemma goes through without problems. For details we refer to our article in [SCE].

Case Study: Polynomial Interpolation Algorithms

The case study on polynomial interpolation concentrates on strategies for algorithm development. We represent the interpolation polynomial *P* as a linear combination

$$P = \sum_{i=1}^{n} a_i B_i$$

of appropriate base polynomials B_i and show in a first attempt for an interpolation algorithm, that the basis {1, x, x^2, ...} always leads to a system of linear equations for the coefficients in the linear combination, which in this case coincide with the polynomial coefficients. We then concentrate on grasping the idea of *Lagrangian interpolation*. We fix $a_i = y_i$, where y_i are the given values to be interpolated, and ask ourselves, whether the basis polynomials B_i can be adjusted such that *P* interpolates y_i on all given support points x_i. We provide a GUI again to adjust the basis polynomials, where a student can

- enter support points x_i and the respective values y_i and
- adjust the values of $B_j(x_i)$ for each combination of *i,j*.

The main window of the GUI then displays all interpolation points (x_i,y_i), all basis polynomials B_i, and the interpolation polynomial *P* as shown in Figure 2.

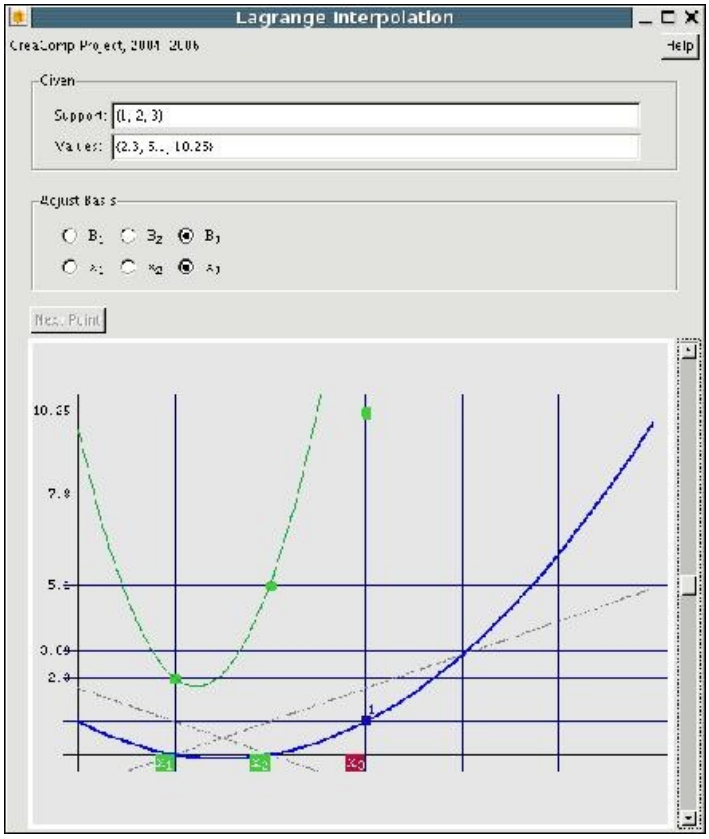

Figure 2: GUI for adjusting Lagrange basis polynomials

Support points x_i are marked green, if $P(x_i)=y_i$, otherwise they are red. The GUI forces the student to iterate *j=1,...,n*, i.e. it starts with *j=1* and there is only one basis polynomial B_1, whose value needs to be chosen at x_1. Once all points x_i are green for *i=1,...,j* the student is allowed to proceed the next interpolation point, i.e. increase *j* by 1. All basis polynomials $B_1,\ldots,B_j$ are left unchanged from the previous step, there is a new basis polynomial B_{j+1}, and the student must (re-)adjust $B_k(x_i)$ for all *i,k=1,...j+1* using the radio buttons on top and the scrollbar available to the right. The degrees of all basis polynomial increases by 1 automatically, and it is not too difficult to conjecture the crucial property of the *Lagrange basis polynomials*, namely $B_i(x_i)=1$ and $B_j(x_i)=0$ for all $i\neq j$. In a similar experiment, students can then explore the well-know product-formula for the basis polynomials.

In a second part, we let students explore the Neville-algorithm for interpolation, which can be formulated as a recursion that computes a degree *n-1* interpolation polynomial at *n* support points from two degree *n-2* polynomials interpolating at the first and the last *n-1* support points, respectively. Similar to the style shown in the equivalence relation case study, students can prove the main theorem on Neville's algorithm, and a GUI allows to

inspect the recursive behaviour of the algorithm. It shows that straight-forward recursive implementation would lead to a tremendously inefficient procedure and leads students to the idea of organizing the algorithm in an iteration according to the “Neville scheme”.

Conclusion

We have presented interactive course material based on *Mathematica* and *Theorema*. Its main feature is the *stimulation of mathematical creativity* through *interactive experiments* and the *integration of automated proving into mathematics teaching*. This approach puts the students' own activity into the center of attention and it encourages them to *explore mathematics* by themselves as opposed to just *present facts* as is often practiced in traditional maths teaching. The material can accompany conventional teaching in the classroom, but it can as well be used for distance teaching, home study or similar learning scenarios.

Acknowledgements

The work presented in this paper is joint work with Susanne Saminger and Günther Mayrhofer in the frame of the project “CREACOMP” directed by Prof. Bruno Buchberger, Prof. Erich Peter Klement, and Prof. Günter Pilz at JKU Linz.

References

[TMA] B. Buchberger, A. Craciun, T. Jebelean, L. Kovacs, T. Kutsia, K. Nakagawa, F. Piroi, N. Popov, J. Robu, M. Rosenkranz, W. Windsteiger. *Theorema: Towards Computer-Aided Mathematical Theory Exploration*. Journal of Applied Logic **4**(4), pp. 470-504. 2006. ISSN 1570-8683.

[SCE] G. Mayrhofer, S. Saminger, W. Windsteiger. *CreaComp: Experimental Formal Mathematics for the Classroom*. In: Symbolic Computation and Education, Shangzhi Li, Dongming Wang, and Jing-Zhong Zhang (ed.), pp. 94-114. 2007. World Scientific Publishing Co., Singapore, New Jersey, ISBN 978-981-277-599-3.

About the Author

Dipl.-Ing. Dr. Wolfgang Windsteiger

RISC Institute

Johannes Kepler University Linz

Schloss Hagenberg,

A-4232 Austria.

email: Wolfgang.Windsteiger@risc.uni-linz.ac.at

Phone: +43 732 2468 9960

Fax: +43 732 2468 29960