Explanation-Based Learning in Computer Algebra Systems

KARSTEN HOMANN

Universität Karlsruhe
Institut für Algorithmen und Kognitive Systeme
Postfach 6980 · D-76128 Karlsruhe · Germany
homann@ira.uka.de

Abstract

An approach to integrate explanation-based learning into computer algebra systems is given. Schemata are learned by generalizing explanations of a teacher and by generalizing numbers. We outline the architecture of an intelligent environment for learning mathematics and its advantages. A unified treatment of mathematical rules and schemata of an application leads to more powerful problem solving capabilities.

Keywords: Explanation-Based Learning, Completion, Symbolic Computations.

1 Introduction

Problem solving in mathematics and mathematical applications requires sophisticated knowledge acquisition and reasoning techniques on the underlying mathematical laws. These capabilities were not provided by classical computer algebra systems, which offer a powerful collection of algebraic algorithms and usually a primitive programming language. Although AXIOM allows the definition of new abstract data types including properties of operators and started a new generation of systems, no AI methods (e.g. automated theorem proving, learning) are provided.

We report on the integration of machine learning into computer algebra systems (CAS), which is one task in the development of an intelligent environment for symbolic mathematical computations called $\lambda\epsilon\mu\mu\alpha$[1]. The paradigm of explanation-based learning (EBL) was chosen because it offers:

- construction of composed objects by analyzing how components can be combined,
- aid to formulate the solution to a given problem,
- improvement of performance by experience,
- consideration of user definitions,
- explanations of solution steps.

[1] $\mathcal{L}$earning $\mathcal{E}$nvironment for $\mathcal{M}$athematic and $\mathcal{M}$athematical $\mathcal{A}$pplications

These features couldn't be integrated into classical CAS because their mathematical knowledge was given implicitly within complex implemented algorithms. The first step in the proposed approach is to build a comprehensible representation of the mathematical knowledge in terms of abstract computational structures [BaWo84, CaTj90] and schemata [Chaf75, Shav90].

Several approaches to learning in mathematically based domains have been developed. They cover learning by heuristics, empirical learning, learning by testing, inductive learning, learning by analogies, and explanation-based learning. However, none of them (except [Shav90]) generalized the structure of explanations or generalized numbers. An inference rule resulting from generalizing numbers subsumes an infinite class of rules learned by standard EBL and describes the situation after an indefinite number of inferences. The more general task is generalizing the structure of explanations where the order of the applied schemata or the schemata themselves are generalized.

On the other hand, the automated theorem proving community offered many reasoning systems for mathematics (e.g. Ontic, Nuprl,...) and completion algorithms. None of them tried to learn rules incrementally by EBL, where one can avoid the undecidability problems of the underlying algorithm (e.g. Knuth-Bendix [KnBe67]).

2 EBL in Mathematically-Based Domains

Figure 1 gives a brief insight into some of the schemata used by a simplifier. Mathematical schemata, user definitions, domain knowledge, and algebraic algorithms form the background knowledge of a computer algebra system.

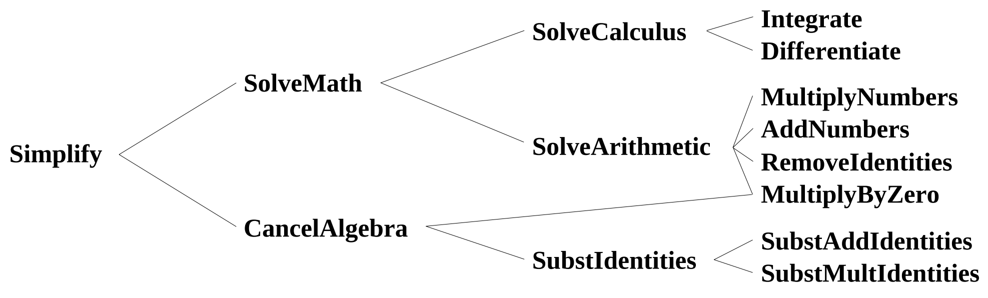

Figure 1: Mathematical Schemata

A simple equation schema is defined as an inference rule consisting of a list of variables, a precondition, and a rewrite equation. Equation schemata can be applied by unifying the preconditions and the left hand sides with a given expression and substituting the right hand sides. Given problems are solved by applying schemata to eliminate obstacles [Shav90] in the calculation of unknown properties of a variable. An explanation why this is an appropriate solution to this problem is generated, the achieved schema is generalized to solve other problems, and finally, the knowledge base of all schemata is updated by the new generalized schema.

Example

We illustrate this by giving an example of an application to physics (see [Shav90] for further details). The initial physical knowledge consists of the definition of velocity, acceleration and Newton's laws.

When a problem can't be solved, the teacher is asked for a detailed solution. The allowed steps are: giving an instance of a known expression, defining a new variable, transforming a previous expression (e.g. evaluation at a given mapping), and introducing new dependencies between variables. Learning new equations is the result of generalizing explanations when verifying the last step.

Given three balls with their velocity and mass at time A, the problem is to determine V_1 at time B, when V_2 and V_3 are given (no external forces).

The problem solver stops after one blind substitution and the teacher has to give a solution. Suppose that the first step of this solution is

$$M_{i_1} V_{i_1,x}(A) + M_{i_2} V_{i_2,x}(A) + M_{i_3} V_{i_3,x}(A) = M_{i_1} V_{i_1,x}(B) + M_{i_2} V_{i_2,x}(B) + M_{i_3} V_{i_3,x}(B) \; .$$

After verifying the equation, an explanation why it holds is evaluated and generalized to get the final result shown in figure 3. Comparing to the result of standard EBL (figure 2) one can note that the structure as well as the numbers could be generalized.

Variables: c, t, i_1, i_2, i_3
Precondition: $\text{IsComponent}(c) \wedge \text{IsTime}(t) \wedge$
$\text{Not}(\text{ZeroValued}(M_{i_1})) \wedge \text{Not}(\text{ZeroValued}(M_{i_2})) \wedge \text{Not}(\text{ZeroValued}(M_{i_3})) \wedge$
$\text{IndependentOf}(M_{i1}, t) \wedge \text{IndependentOf}(M_{i2}, t) \wedge \text{IndependentOf}(M_{i3}, t) \wedge$
$i_1 \neq i_2 \wedge i_1 \neq i_3 \wedge i_2 \neq i_3 \wedge \text{Permutation}(\{i_1, i_2, i_3\}, ObjectsInWorld) \wedge$
$\text{ZeroExpression}(\text{ValueOf}(F_{ext,i_1,c}) + \text{ValueOf}(F_{ext,i_2,c}) + \text{ValueOf}(F_{ext,i_3,c}))$

$$M_{i_1} V_{i_1,c}(t) + M_{i_2} V_{i_2,c}(t) + M_{i_3} V_{i_3,c}(t) = constant$$

Figure 2: Learned Schema with Standard EBL

Variables: c
Precondition: $\text{IsComponent}(c) \wedge \forall i \in ObjectsInWorld : \text{Not}(\text{ZeroValued}(M_i)) \wedge$
$\forall i \in ObjectsInWorld : \text{IndependentOf}(M_i, t)$

$$\frac{\partial}{\partial t} \sum_{i \in ObjectsInWorld} M_i V_{i,c}(t) = \sum_{i \in ObjectsInWorld} F_{ext,i,c}(t)$$

Figure 3: Learned Generalized Schema

3 Integration into CAS

To extend the mathematical capabilities of EBL and to benefit from the knowledge and algorithms of CAS, we give a schematic overview of the resulting architecture in figure 4.

The integration leads to some theoretical and technical problems:

- common representation of variables, equations, expressions in a mathematical knowledge base (e.g. additional indexing of variables in the explanations),
- explicit separation of calculation schemata and algebraic algorithms,

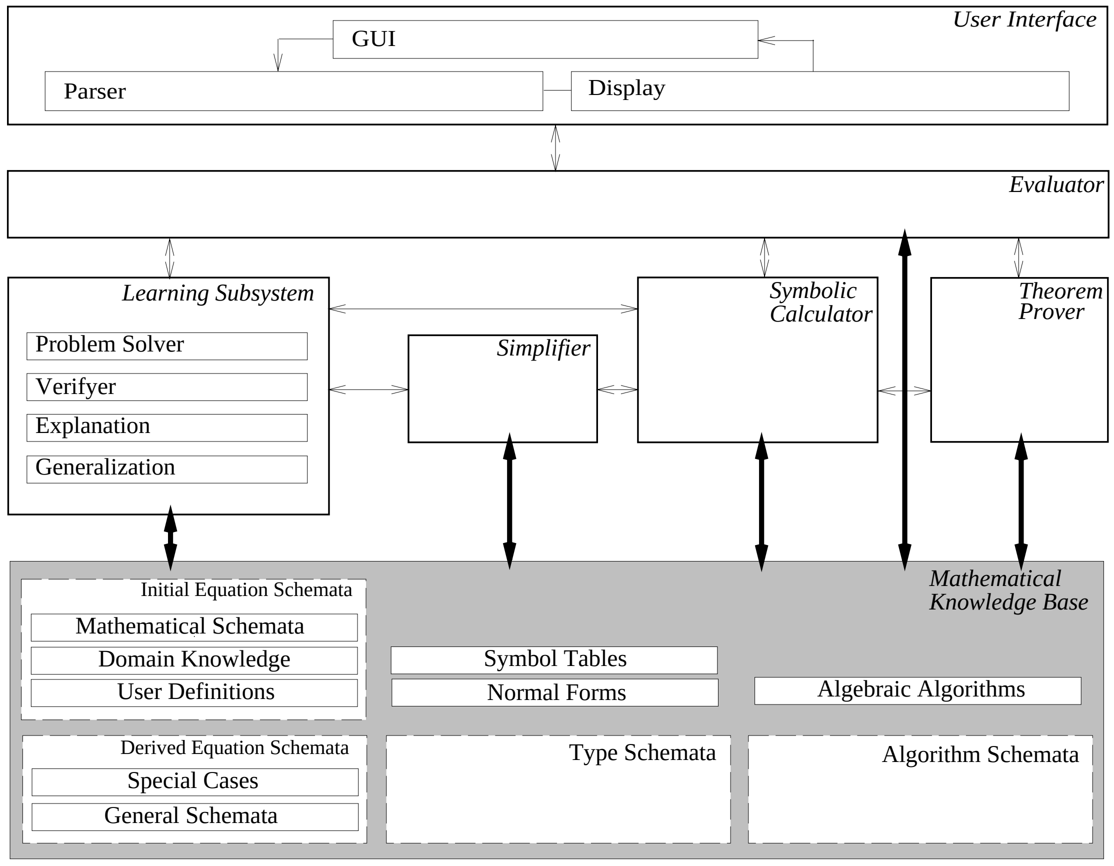

Figure 4: Schematic Integration of a Learning Subsystem into the Architecture of CAS

- representation of generalized schemata as well as special cases (generality versus operationality).

The user interface is extended to provide frames and graphs for handling schemata. It can display the explanations about solutions of specific problems. These problems are solved by the evaluator by applying schemata (learning subsystem) or algebraic algorithms (symbolic calculator) making use of the definitions in the symbol tables of the mathematical knowledge base. This knowledge base also consists of the normal forms of the simplifier, the algebraic algorithms of the symbolic calculator, as well as initial and derived schemata. The learning component, consisting of a schema-based problem solver, a verifyer, and functions to generate explanations and their generalization, derives schemata in both general and special form.

Some promising advantages of this approach are:

- learning mathematical schemata of CAS by EBL,
- schema-based problem solving using the symbolic calculator (e.g. symbolic integration, differential equations),
- modifying EBL to incrementally complete the properties of operators in ACS,
- extraction of mathematical schemata from algebraic algorithms.

4 Conclusion

An insight into EBL in mathematical-based domains is given. The methods presented rely on a schema-based problem solver, and new schemata are learned by EBL. Integrating this approach into CAS leads to new promising capabilities.

Among others, further research must be investigated to reformulate the notion of computational structures including both schemata and algorithms, automated verification of schemata and algorithms, and new applications of the intelligent environment. This environment should also allow the integration of automated theorem provers.

We started an application in coding theory, where a source can learn new codes and schemata, when and why the coding and decoding should automatically change in case too few or too many errors or burst errors occur.

References

[BaWo84] F.L. Bauer, H. Woessner, *Algorithmische Sprache und Programmentwicklung (2nd Edition)*, Springer, 1984.

[CaTj90] J. Calmet, I.A. Tjandra, *Learning Complete Computational Structures*, Fifth International Symposium on Methodologies for Intelligent Systems (Selected Papers), pp 63–72, Knoxville, TN, 1990.

[Chaf75] W. Chafe, *Some Thoughts on Schemata*, Theoretical Issues in Natural Language Processing I, pp 89–91, Cambridge, MA, 1975.

[KnBe67] D.E. Knuth, P.B. Bendix, *Simple Word Problems in Universal Algebras*, pp 263–298, Oxford, 1967.

[Shav90] J.W. Shavlik, *Extending Explanation-Based Learning by Generalizing the Structure of Explanations*, Pitman, London, 1990.