of Proceedings of the 7th International Conference on Systems Research, Informatics & Cybernetics, pp. 130–135, Baden-Baden, 1994.

Integrating Explanation-Based Learning in Symbolic Computing

K. Homann

Universität Karlsruhe
Institut für Algorithmen und Kognitive Systeme
Am Fasanengarten 5 · D-76131 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 of schemata of an application leads to increased problem solving capabilities.

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

Introduction

Problem solving in mathematics and mathematical applications requires sophisticated knowledge acquisition and reasoning techniques on the underlying mathematical laws. These capabilities are not provided by classical computer algebra systems, which offer a powerful collection of algebraic algorithms and usually a straightforward 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.

These features could not 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 (Bauer and Woessner, 1984), (Calmet et al., 1992) and schemata (Chafe, 1975), (Shavlik, 1990).

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

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 (Shavlik, 1990)) 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. A more general task is to generalize 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 offers 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 and Bendix, 1967).

EBL in Mathematically-Based Domains

EBL in mathematically-based domains is a rewriting method for expressions. Unknown properties of variables are computed by applying rules and generalizing its conditions and consequences. There is a substantial difference to classical EBL in the structure of the explanation graph but standard generalization techniques can be applied and the same problems arise (e.g. special versus general schemata). Because of length requisites, this paper only briefly describes the methods and formalisms.

Mathematical schemata, user definitions, domain knowledge, and algebraic algorithms build the background knowledge of a system which is capable of solving problems in mathematically-based domains. *Equation schemata* allow the representation of this knowledge, but with the exception of algebraic algorithms. Therefore, the lack of mathematical algorithms in classical systems led to inefficient and inappropriate representation of the underlying mathematics. Figure 1 gives a brief insight into the hierarchy of some of these mathematical schemata. The next section gives an architecture which allows the integration of algorithms in a schema-based problem solver.

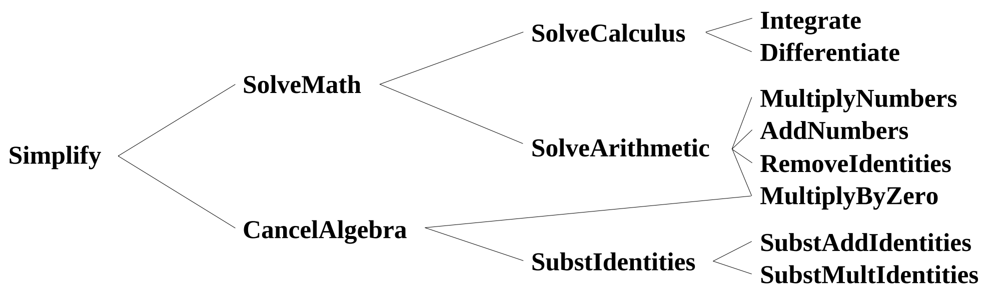

Figure 1: Hierarchy of Mathematical Schemata

A primitive equation schema is defined as an inference rule consisting of a list of variables, preconditions, and a rewrite equation, e.g.

Name	ConstantsOutOfCalculus
Variables	$x, indep, expr$
Preconditions	IndependentOf($?indep$, $?x$)
Equation	$\int(?indep * ?expr)\ d?x = ?indep\ *\ \int ?expr\ d?x$

Name	Differentiate
Variables	$x, n, expr$
Preconditions	Number(x)
Equation	$\frac{d}{dx} expr^n = n\ expr\ \frac{d}{dx} expr^{n-1}$

Equation schemata can be applied by unification of the preconditions and the left hand sides of an equation with a given expression and by substituting the right hand sides. Given problems are solved by applying schemata to eliminate obstacles (Shavlik, 1990) in the calculation of unknown properties of a variable. In the simplest case an obstacle is an unaccaptable variable which, if its value is known, would permit to solve the problem.

When a problem cannot be solved, the teacher is asked for a detailed solution. The allowed steps are: to give an instance of a known expression, to define a new variable, to transform a previous expression (e.g. evaluation at a given mapping), and to introduce new dependencies between variables. 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.

Learning new equations is the result of generalizing explanations when verifying the introduction of new dependencies. This generalization is done by introducing and isolating the primary obstacles, by introducing their cancelers and by cancellation. The adopted algorithm of Shavlik introduces the generalization of numbers which allows to learn dependencies of an arbitrary number of instances of variables and the generalization of the explanation structure which allows to learn equations with much less preconditions (and is thus more general).

Integration into CAS

One of the major drawbacks of this method was the missing algorithmic mathematical knowledge. However, large collections of powerful algebraic algorithms are provided by CAS. They are difficult to use, because their mathematical knowledge, e.g. definitions of mathematical structures, properties of operators of a domain, domains of computation, range of algorithms and their mathematical specification, is hidden in the algebraic algorithms. They are very efficient in computing symbolic solutions by given algorithms but cannot derive new theorems.

It is thus natural to integrate the methods of explanation-based learning in mathematically based domains in symbolic computing. This allows to use very efficient algebraic algorithms for mathematical problem solving and schema-based representation and acquisition of learned solutions. We give a schematic overview of the resulting architecture in figure 2.

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,
- introduction of algorithm schemata which provides the meta knowledge for generalizing algebraic algorithms in the cancellation graph,
- representation of generalized schemata as well as special cases (generality versus operationality).

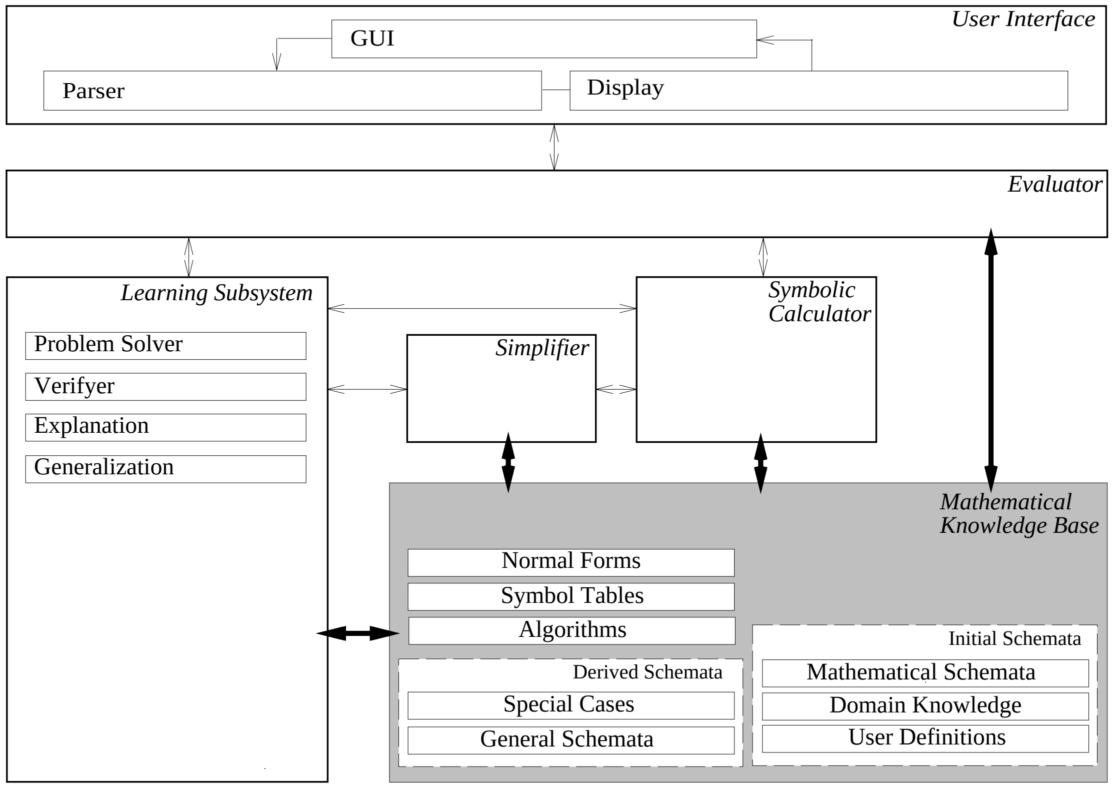

Figure 2: Schematic Integration of a Learning Subsystem into the Architecture of CAS

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 of the initial and derived schemata. The learning component, consisting of a schema-based problem solver, a verifier, and functions to generate explanations and their generalization, derives schemata in both general and special forms.

A proposed interaction between a symbolic calculator (SC) and a learning component (LC) is illustrated in figure 3.

The foreseen benefits of this approach are:

- schema-based problem solving using the symbolic calculator (e.g. symbolic integration, differential equations).
  The main advantage of using the algorithms of the SC to simplify and compute predicates is a huge extension of the build-in mathematics of the system. Several of the problems of schema-based computations can be avoided, e.g. use of associativity and distributivity. Additionaly, the computations by the symbolic calculator are much more efficient and powerful, e.g. equation schemata ConstantsOutOfCalculus and Differentiate of the last section compared to algorithms for integration and differentiation of the SC. To preserve the generalization capability, this kind of interaction requires the explicit representation of conditions and results of algorithms. We introduced the notion of *algorithm schemata* which enables this

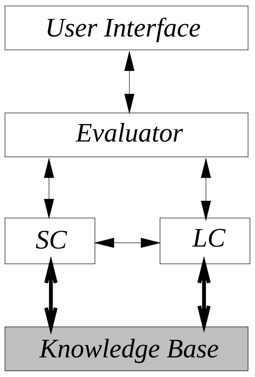

Figure 3: Interaction between Symbolic Calculator and Learning Subsystem

representation, e.g. schema for the algorithm `gcd`

Name	$\mathsf{gcd}(?a, ?b) = ?g$
Signature	$?A \times ?A \rightarrow ?A$
Constraints	isa ($?A$, EuclideanRing)
Definition	$(?g \mid ?a) \wedge (?g \mid ?b) \wedge (\forall c \in ?A : (c \mid ?a) \wedge (c \mid ?b) \Rightarrow (c \mid ?g))$
Subalgs	
Theorems	$\mathrm{gcd}(u, v) = \mathrm{gcd}(v, u)$ $\mathrm{gcd}(u, v) = \mathrm{gcd}(v, u \ mod \ v)$ $\mathrm{gcd}(u, 0) = u$
Function	

This reduces the proof of a teacher's solution because mainly substitutions of variables must be computed by equations, e.g. steps (1)–(5), (8)–(10) in the verification of the solution to the energy problem are efficiently performed by the SC

$$
\begin{array}{lll}
 & & \frac{d}{dt}(\frac{1}{2}\, M_1\, V_{1,y}^2(t) + M_1\, g\, X_{1,y}(t)) \\
(1) & \textbf{SeperateDerivatives} & = \frac{d}{dt}(\frac{1}{2}\, M_1\, V_{1,y}^2(t)) + \frac{d}{dt}(M_1\, g\, X_{1,y}(t)) \\
(2) & \textbf{ConstsOutOfDerivatives} & = \frac{1}{2}\, M_1\, \frac{d}{dt}(V_{1,y}^2(t)) + M_1\, g\, \frac{d}{dt}(X_{1,y}(t)) \\
(3) & \textbf{Differentiate} & = \frac{2}{2}\, M_1\, V_{1,y}(t) \frac{d}{dt}(V_{1,y}(t)) + M_1\, g\, \frac{d}{dt}(X_{1,y}(t)) \\
(4) & \textbf{MultiplyNumbers} & = 1\, M_1\, V_{1,y}(t) \frac{d}{dt}(V_{1,y}(t)) + M_1\, g\, \frac{d}{dt}(X_{1,y}(t)) \\
(5) & \textbf{RemoveIdentities} & = M_1\, V_{1,y}(t) \frac{d}{dt}(V_{1,y}(t)) + M_1\, g\, \frac{d}{dt}(X_{1,y}(t)) \\
(6) & \textbf{SubstCalculus} & = M_1\, V_{1,y}(t)\, A_{1,y}(t) + M_1\, g\, V_{1,y}(t) \\
(7) & \textbf{SubstToCancel} & = M_1\, V_{1,y}(t)\, \frac{F_{net,1,y}(t)}{M_1} + M_1\, \frac{F_{net,1,y}}{M_1} V_{1,y}(t) \\
(8) & \textbf{SubstMultIdentities} & = 1\, V_{1,y}(t)\, F_{net,1,y}(t) + 1\, F_{net,1,y}\, V_{1,y}(t) \\
(9) & \textbf{RemoveIdentities} & = V_{1,y}(t)\, F_{net,1,y}(t) + F_{net,1,y}\, V_{1,y}(t) \\
(10) & \textbf{SubstAddIdentities} & = 0\, \frac{kg\, m^2}{s^3}
\end{array}
$$

Algorithm schemata must support references to input variables in the output which allows to generalize the application of the algorithms.

- learning mathematical schemata of CAS by EBL.
  EBL has shown remarkable success in learning heuristics to guide the application of algorithms and equations, e.g. LEX2 (Mitchell, 1983) for symbolic integration.

- modifying EBL to incrementally complete the properties of operators in abstract computational structures (Calmet and Tjandra, 1990).
- extraction of mathematical schemata from algebraic algorithms.

Conclusion

An insight into EBL in mathematically-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 conducted 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.

References

Bauer, F.L. and H. Woessner (1984); Algorithmische Sprache und Programmentwicklung (2nd Edition); Springer

Calmet, J., Homann, K. and I.A. Tjandra (1992); Unified Domains and Abstract Computational Structures; Proc. International Conference on Artificial Intelligence and Symbolic Mathematical Computing (ed. Calmet, J. and J.A. Campbell); Springer, LNCS 737 (pp. 166–177)

Calmet, J. and I.A. Tjandra (1990); Learning Complete Computational Structures; Proc. 5th International Symposium on Methodologies for Intelligent Systems (Selected Papers); Knoxville, TN (pp. 63–72)

Chafe, W. (1975); Some Thoughts on Schemata; Theoretical Issues in Natural Language Processing I; Cambridge, MA (pp. 89–91)

Knuth, D.E. and P.B. Bendix (1967); Simple Word Problems in Universal Algebras; Oxford (pp. 263–298)

Mitchell, T.M. (1983); Learning and Problem Solving; Proc. 8th International Joint Conference on Artificial Intelligence; Karlsruhe, Germany (pp. 1139–1151)

Shavlik, J.W. (1990); Extending Explanation-Based Learning by Generalizing the Structure of Explanations; Pitman, London