# Distributed Mathematical Problem Solving

JACQUES CALMET KARSTEN HOMANN

*University of Karlsruhe*
*Institute for Algorithms & Cognitive Systems*
*Am Fasanengarten 5 · 76131 Karlsruhe · Germany*
{calmet,homann}@ira.uka.de

**Abstract**

Coupling computer algebra systems and theorem provers enables to extend the capabilities they have when standing alone. We report on an ongoing research project whose long term goal is to provide an open environment for doing mathematics including reasoners and symbolic calculators. It is extensible by users which can construct complex systems by combination and insertion of existing packages. These systems may be based on different logics, formalisms, data structures, interfaces. A result of this work is illustrated by a prototype implementation of an interface between Isabelle and Maple.

**Keywords:** symbolic mathematical computing, theorem proving

## 1 Introduction

It has been said that Artificial Intelligence (AI) is exporting its successes. Two such examples are computer algebra (CA) and theorem proving (TP). They are today independent research fields. We are investigating how it is possible to integrate them anew. It is not surprising that such a goal leads to rely on their original "father": AI. Coupling CA and TP enables to extend the capabilities they have when standing alone. In particular, methods from learning become almost mandatory. This is why the environment we are designing is called $\lambda\epsilon\mu\mu\alpha$[1]. However, we report here only on some basic design features of our approach and not on $\lambda\epsilon\mu\mu\alpha$ as a whole.

[1]Learning Environment for Mathematics and Mathematical Applications

One characteristic is that we are dealing with distributed problem solving in mathematics. Nowadays two types of systems are available for solving mathematical problems: computer algebra systems (CAS) and theorem proving systems (TPS). A straightforward integration is difficult for the following reasons: (1) many systems, especially CAS, are copyrighted and allow neither extension nor connection to other software modules, (2) they were designed, implemented and validated as stand-alone systems, (3) they are based upon particular internal representation formalisms and do not allow external access to the embedded knowledge, (4) usually, no standards or languages for interfacing exist.

Typically, a given problem is transformed into the language of one system and solved using algorithmic (CAS) or deductive (ATP) knowledge. When relying on only one formalism possibly only subproblems can be solved. However, different subproblems are described more naturally in different theories and logics, such as higher-order logics, dynamic and temporal logics. Ad-hoc logics often enable an efficient computation of solutions, what would not be possible with general provers. The availability of several theories, logics, and their interconnections enable a natural and direct formulation of problems. Just as important is a common environment which allows to interchange between theories. [6] describes an architecture for an open mechanized reasoning system (OMRS) with different logics, models, vocabularies, data structures, strategies, and interaction capabilities. Its goal is to provide a framework and a methodology wich allow users to compose existing reasoning modules or add new ones in a "plug and play" manner. However, they do not discuss reasoning by symbolic computation in a model.

In symbolic mathematical computing there are several approaches to develop common, portable, and extensible interfaces as well as restricted communication mechanisms ([7] and figure 2). CAS/$\pi$ [10] describes a graphical user interface which allows communication with already existing CAS as well as with user defined software packages. New tools can be added, connections between modules can be specified, and the user interface (formulae editors, menus, visualization of graphics) can be extended even during runtime. This is achieved by the translation of the underlying system, programming and representation languages respectively. Varying views of the mathematical objects, e.g. graphics, text, formulae in different syntax, can be passed to the corresponding module. To provide the universal means of communicating mathematical information between applications is aim of OpenMath [11], a project to standardize the exchange of mathematical information. A second goal of OpenMath is the development of an open distributed processing of mathematical information. However, the systems are not intended to communicate directly with each other to solve specific subproblems, they are still black boxes. Additionally, no theorem provers can be included yet. To include TPS is an extension of the OpenMath objectives.

The advantages of a possible combination of both approaches are twofold: (1) improved expressive power, (2) more powerful inference capabilities. There are several possible approaches to integrate theorem provers and computer algebra systems. On one hand classical CAS usually offer a straightforward programming language with ad-hoc implementations of rewriting. One approach towards introducing theorem proving in CAS is an extension of Analytica [5], a Mathematica package to prove theorems in elementary analysis which solves an extensive collection of nontrivial mathematical problems. On the other hand some classical TP were extended by techniques of symbolic computing, e.g. Otter allows to call external algorithms out of proofs. Specialized prover packages have been developed which are capable of performing symbolic mathematical computations. However, there are no environments integrating theorem provers and computer algebra systems which consistently provide the inference capabilities of the former and the powerful arithmetic of the latter systems.

We prove through an example, the integration of Maple and Isabelle, that we can improve the capabilities of theorem proving systems. The disadvantage of this kind of interaction lies in the black box behaviour of most CAS and TPS. No system can be plugged into such an environment directly if it cannot explain its solution or provide results and accept information incrementally. The ability to reflect its behaviour has been addressed in the TP community but not in CA, although becoming increasingly important.

The explicit representation of the embedded mathematical knowledge opens the black boxes and allows reflection by explanations. The formalization of schemata allows the environment to understand, control, and modify its behaviour.

To summarize, what is needed is a variety of particular logics, theories, packages, CAS, TPS. Thus we work towards an environment and language for semantically describing both CA and reasoning systems and efficient cooperation formalisms to manage the interaction of several components for doing mathematics.

This paper is organized as follows. First we introduce the notion of an open mechanized mathematics environment and illustrate some kinds of interaction between reasoning systems and symbolic calculators. Section 3 describes an interface between the tactical theorem prover Isabelle and the computer algebra system Maple. Although the examples are elementary, they offer a good insight into problems which can be solved by theorems *and* algorithms. Section 4 gives a contribution to the discussion on how to implement reflection in algebraic engines. Those behave like black boxes because the embedded mathematical knowledge is not represented explicitly.

# 2 Specification and Combination of Components

Providing a high-level mechanism for component combination adequate for prototyping enables to adapt new modules by defining a formal external connection. Reasoning structures [6, 14] represent provisional proof fragments in the construction of derivations or proofs by different reasoning systems. Each proof can be described by such structures. External access to reasoning structures is given by handles [15] which represent local contexts, e.g. collections of known facts, constraints, and assumptions. Open mechanized reasoning systems (OMRSs) can cooperate by exchanging knowledge through handles. The unified interface allows to extend reasoning environments easily by "plug and play".

We extend the notion of reasoning structures and handles for additional interaction and cooperation with computer algebra systems. Such an *open mechanized mathematics environment* (OMME) is defined to contain OMRSs and open mechanized computation systems (OMCSs) leading to

| | | |
|---|---|---|
| Reasoning Theory | = | Sequents + Rules |
| Reasoning System | = | Reasoning Theory + Control |
| OMRS | = | Reasoning System + Interaction |
| OMCS | = | Symbolic Calculator + Interaction |
| OMME | = | OMRS* + OMCS* |

An OMME is a collection of reasoning and computation systems which provide interaction for cooperative problem solving. At this stage, several possible kinds of interaction between a reasoning system (RS) of a OMRS and a symbolic calculator (SC) of OMCS are available [9]. Three examples are given in figure 1.

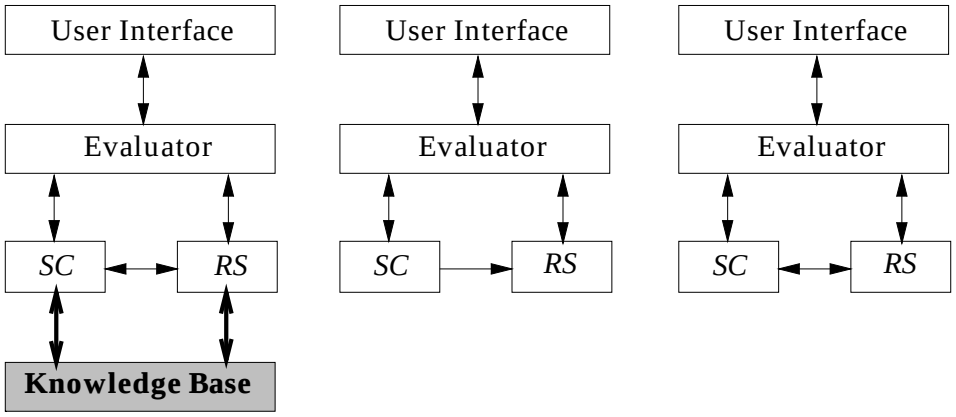


**Figure 1: Interaction between Reasoning System and Symbolic Calculator**

The first kind of interaction allows a direct cooperation of several systems with a common knowledge representation. Due to the stand-alone character and copyright of many systems, such an interaction is impracticable in most cases. However, the

environment should support the direct interoperationability of systems providing such an interaction, for instance Mathematica and Analytica [5].

Secondly, a unidirectional link from one system to another one allows the immediate call of algorithms out of proofs. It extends the capabilities of both providing incremental problem solving. As an example, we have implemented an interface between Isabelle and Maple illustrated in the next section.

Finally, in the case of bidirectional interaction arbitrary combinations of algorithms and theorems can be applied. This interaction can be generalized to a *mathematics-software bus* [7] as shown in figure 2. The generalization also includes both previous cases: a common knowledge representation module can be connected to the bus and some components may only provide unidirectional links. The bus must provide a standardized protocol to exchange information between mathematical applications.

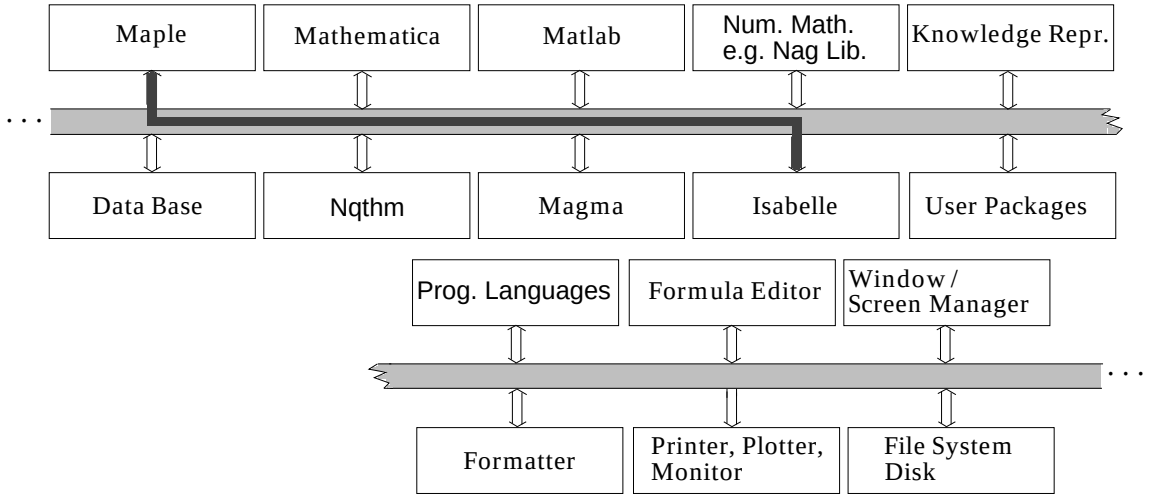


**Figure 2: Mathematics-Software Bus**

# 3 Interfacing Isabelle and Maple

As an example of combinig a reasoner and a computer algebra system we have implemented an interface between Isabelle [13] and Maple [4]. This interface is achieved by extending Isabelle's simplifier, but without any modification of Maple. Maple behaves as a black box in this cooperation, a disadvantage we discuss in the following section.

The tactical theorem prover Isabelle implements a meta-logic which is used to specify logics to be used for reasoning (object-logics). The computation of the CAS

is embedded into the prover as a schematic formula

$$[\![ \mathrm{P}_1, \ldots, \mathrm{P}_n ]\!] \Rightarrow \mathrm{t} \equiv \mathrm{t}'$$

with $P_1, \ldots, P_n$ premises, $t'$ a term computed by Maple. The CAS may simplify only distinct terms and cannot reduce terms containing logical connectives. Therefore, we define a semantic by providing consistency with respect to a signature [1]. The extension of Isabelle's simplifier is done by defining simplification sets which control the simplification tactics. We introduce a new class of rules we call evaluation rules [1] which give access to the CAS. To integrate Maple within an Isabelle theory two steps were required: specification of Maple's concrete syntax and definition of these evaluation rules. Figure 3 illustrates the achieved interaction.

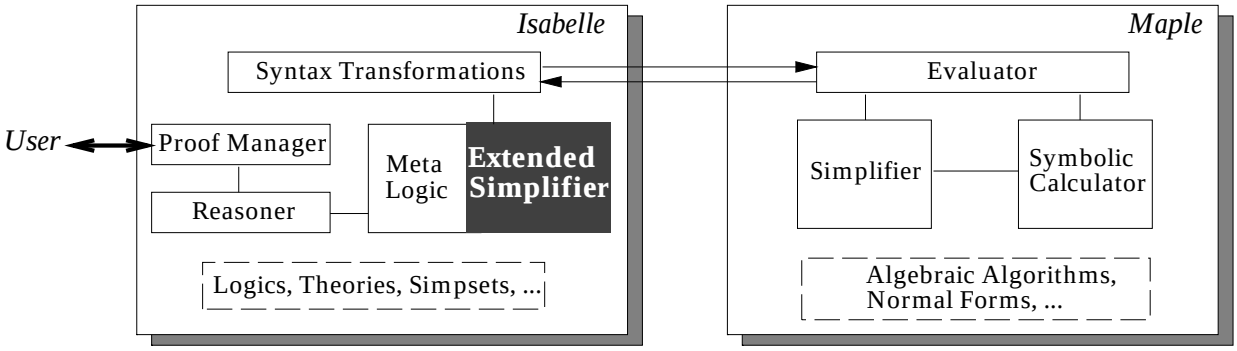


**Figure 3: Schematic Link between Isabelle and Maple**

The combination of both systems enables to prove new theorems by symbolic computations and reasoning, e.g. arithmetics with numbers, finite and infinite summations, and proofs by induction on polynomial expressions [1]. Examples where efficiency is improved by external calls to Maple are $1 + 9 * 11 = 100$ (trivial to a human but not to a theorem prover) or $\sum_{n=0}^{k} (2\,n + 1) = (k + 1)^2$. The application of Maple avoids using the inefficiency and complexity of the Peano arithmetic.

An example of a theorem hard to prove by Isabelle alone is $\forall n \geq 5 : n^5 \leq 5^n$. Maple can be used to expand the products in Isabelle's proof by induction, e.g. $(n + 1)^5 = n^5 + 5\,n^4 + 10\,n^3 + 10\,n^2 + 5\,n + 1$.

# 4 Reflection, Black Boxes and Schemata

For the purpose of useful integration, mathematical components must be able to interact with the other modules in sophisticated ways. Deciders and black boxes which only provide their solution (yes, no, or result of algebraic algorithms) are not adequate in general, because they neither do work incrementally nor accept questions about their internal knowledge. In the best case they query other components

for information about entities not in their domain of expertise. There has been some work on an architecture for structuring and specifying integrable reasoning modules [15, 6] which opens the black boxes.

Components for symbolic mathematical computing can be introduced by extending the notion of handles [15] only available in reasoning systems. They represent the information contained in the local context of reasoning specialists, e.g., collections of facts or assumptions.

The main problem when generalizing to CAS is that such handles are difficult to introduce since a CAS is not reflective in the sense of [12]:

> A reflective system is a system that has an internal representation of itself, and the ability to use such a representation to understand, control, and modify its own behavior.

One of the ideas of our approach is to introduce a schema-based representation of mathematical structures and algorithms. Examples of such schemata are given (see [9] for details) in figure 4. They can be represented in an extension of our hybrid knowledge representation system Mantra [2].

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

| *Name* | **Group** |
|---|---|
| *Based-On* | Monoid |
| *Sorts* | Gr |
| *Operators* | inv _ :: Elt $\rightarrow$ Elt |
| *InitialProps* | $\forall x \in$ Elt: $inv(x)\ f\ x = ne$ |

| *Name* | **gcd-primitive**$(?a, ?b) = ?g$ |
|---|---|
| *Signature* | $?A \times ?A \rightarrow ?A$ |
| *Constraints* | isa ($?A$, UniPoly$(x, UFD)$) |
| *Definition* | |
| *Subalgs* | primitive-part<br>pseudo-remainder<br>content<br>gcd<br>multiply |
| *Theorems* | |
| *Function* | GcdPrimitive |

**Figure 4: Examples of Equation, Type, and Algorithm Schemata**

Moreover, the resulting components are able to extend their meta-knowledge incrementally by explanation-based learning [8].

This explicit representation allows reflection by several kinds of explanation: (1) Which algorithms were used? (2) Which parameters and types are required / given? (3) What is the mathematical meaning of the solution? (4) Why is the output the solution?

The explicitly represented knowledge of a symbolic calculator (schemata) or reasoning system is called a *context*. These contexts are a generalization of handles.

Let $c_1, c_2, ...$ range over such contexts. There is a pre-order $c_1 \trianglelefteq c_2$ on contexts expressing that $c_2$ is reachable by $c_1$ by adding new information. The relation is transitive and reflexive and behaves consistent with $\preceq$ defined on handles [15].

Symbolic calculators must accept information incrementally and provide their mathematical knowledge. The two important families of operations on contexts are telling facts and asking questions. These primitives, called $tell$ and $ask$, are also the primitives of the logical level of our hybrid knowledge representation system. Some example interrogations $ask(c, q)$ of Mantra's knowledge bases representing contexts are: (1) is formula $f$ of $q$ entailed by $c$, (2) is attribute $a$ of $q$ computable with context $c$, (3) solve $q$ by computation with context $c$. The answer might be four-valued $\{yes,\ no,\ unknown,\ external\}$ where $external$ means $c$ can try to solve $q$ by external call to another context $c'$ perhaps requesting another OMRS or OMCS. $tell(c, k)$ tells a context new facts to give a new context with additional information $k$ ($c \trianglelefteq tell(c, k)$). We also adopt formalisms for extraction and composition of contexts.

# 5 Conclusion

We report on an ongoing research project whose long term goal is to provide an environment for doing mathematics including reasoners and symbolic calculators. It is extensible by users which can construct complex systems by combination and insertion of existing systems. These systems may be based on different logics, formalisms, data structures, interfaces. A result of this work is to integrate CA and TP. This is illustrated by a prototype implementation of an interface between Isabelle and Maple. The work in progress consists of:

- One part of this project extends the concept of reasoning structures and reasoning theories to provide additional structuring and specification of symbolic mathematical calculators. For this purpose, a complete characterization of both reasoning systems and symbolic calculators must be developed.
- The mathematics-software bus is defined to have a primitive cooperation protocol. It is specified only for transfering messages to the bus. This bus and the protocol have to be extended by a control unit and specification of communication and cooperation mechanisms.

Among the many questions arising from such a project are the following ones: What must be specified to interconnect and integrate reasoning and calculation devices? Is it possible to interact with systems which are not intended to be used interactively? How can systems deal with information about entities not in their domain of expertise? What mathematical concepts and structures are necessary to establish a semantics for an open mechanized mathematics environment?

# References


[1] C. Ballarin, K. Homann, J. Calmet, *Theorems and Algorithms: An Interface between Isabelle and Maple.* In C. Traverso (Ed.), Proceedings of International Symposium on Symbolic and Algebraic Computation (ISSAC'95), Montreal, Canada, ACM Press, 1995, to appear.

[2] G. Bittencourt, J. Calmet, K. Homann, A. Lulay, *MANTRA: A Multi-Level Hybrid Knowldege Representation System.* In T. Pequeno, F. Carvalho (Eds.), Proceedings of the XI Brazilian Symposium on Artificial Intelligence, pp. 493–506, 1994.

[3] J. Calmet, J.A. Campbell, *Artificial Intelligence and Symbolic Mathematical Computing,* Editors, Lecture Notes in Computer Science, Springer Verlag, 1994, to appear.

[4] B.W. Char, K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan, S.M. Watt, *Maple V Language Reference Manual,* Springer Verlag, 1992.

[5] E. Clarke, X. Zhao, *Analytica – An Experiment in Combining Theorem Proving and Symbolic Computation,* Technical Report CMU-CS-92-147, Carnegie Mellon University, 1992.

[6] F. Giunchiglia, P. Pecchiari, C. Talcott, *Reasoning Theories – Towards an Architecture for Open Mechanized Reasoning Systems,* 1994, to appear.

[7] G. Gonnet, *Semantics and Purposes of OpenMath,* slides of talk presented at Second OpenMath Workshop, 1992.

[8] K. Homann, *Integrating Explanation-Based Learning in Symbolic Computing.* In J.W. Brahan, G.E. Lasker (Eds.), Advances in Artificial Intelligence – Theory and Application II, Volume II, pp. 130–135, 1994.

[9] K. Homann, J. Calmet, *Combining Theorem Proving and Symbolic Mathematical Computing.* In [3].

[10] N. Kajler, *CAS/PI: a Portable and Extensible Interface for Computer Algebra Systems.* In Proceedings of International Symposium on Symbolic and Algebraic Computation (ISSAC'92), pp. 376–386, ACM press, 1992.

[11] A. van Leeuwen, M. Roelofs, A. Strotmann, *Objectives of OpenMath,* draft version 0.8.3, 1995.

[12] P. Maes, *Issues in computational reflection.* In P. Maes, D. Nardi (Eds.), Meta-Level Architectures and Reflection, pp. 21–35, North-Holland, 1988.

[13] L.C. Paulson, *Isabelle — A Generic Theorem Prover,* Lecture Notes in Computer Science 828, Springer Verlag, 1994.

[14] C.L. Talcott, *Towards a Framework for Specifying Components of Automated Reasoning Systems: A report on work in progress.* In TTCP XTP-1 Workshop on Effective Use of Automated Reasoning Technology in System Development (EUARTSD), 1992.

[15] C.L. Talcott, *Reasoning Specialists Should be Logical Services, Not Black Boxes,* Proceedings of CADE-12 Workshop on Theory Reasoning in Automated Deduction, pp. 1–6, 1994.