To appear in K. Homann, S. Jacob, M. Kerber, H. Stoyan (Eds.), Proceedings of Workshop on Representation of Mathematical Knowledge, 12th European Conference on Artificial Intelligence, Budapest, 1996.

Hybrid Representation for Specification and Communication of Mathematical Knowledge

Jacques Calmet[1] and **Karsten Homann**[1] and **Indra A. Tjandra**[2]

Abstract. It is well known that mathematicians switch to different views of a problem when needed. They can represent a problem at a formal, conceptual, heuristic, algorithmic or constraint level whenever necessary. To represent Mathematics within several formalisms has been the subject of many research projects. However, only few knowledge-based systems manage translations of representations between theories.

The goal of this paper is twofold. One the one hand, we report on a hybrid knowledge representation and reasoning system called MANTRA. The system provides four different knowledge representation methods – first-order logic, terminological language, semantic networks, and production rules – distributed into a three levels architecture. Specifications of mathematical domains of computation and their inherently related type inference mechanisms can be transformed into knowledge bases.

On the other hand, we argue that a main requirement when designing future environments is the capability to cooperate and to integrate by communicating mathematical knowledge among/through *mathematical services* based upon restart-able computation and reasoning. Therefore, structures representing intermediate results in any kind of mathematical computation must also be considered.

1 INTRODUCTION

The design of systems for mathematical problem solving recently emerged form the development of specialized packages in environments called mathematical assistants. Such environments rely on sophisticated AI techniques: reasoning, representation and cooperation.

It is well known that mathematicians switch to different views of a problem when needed. They can represent a problem at a formal, conceptual, heuristic, algorithmic or constraint level whenever necessary. To represent Mathematics within the theories of several formalisms has been subject of many research projects. However, only few knowledge-based systems manage translations of representations between theories. The aim of the QED project [14] is to build a single, distributed, computerized repository that rigorously represents all important, established mathematical knowledge. Such a repository includes heterogeneous mathematical objects such as theorems, equations, definitions, algorithms, domains of computations ...

Because of adequacy and efficiency of different formalisms, we argue that systems for representing mathematical knowledge should be hybrid by means of combining several formalisms and paradigms. Additional requirements for such systems are a clear semantics explaining the meaning of the expressions and decidability of all involved inference problems. This papers describes such a system which provides four different knowledge representation methods – first-order logic, terminological language, semantic networks, and production rules – distributed into a three levels architecture.

Although the system is designed for the representation of mathematical knowledge, to explicitly specify mathematics in terms of hybrid knowledge representation formalisms is awkward and inappropriate. We introduce a framework for specifying mathematical domains of computation and their inherently related type inference mechanisms as well as for transforming those specifications into knowledge bases of the knowledge representation system.

There has been a number of works to integrate systems performing any kind of mathematical computation, i.e. combining computer algebra systems (CAS) in CAS/π and OPENMATH [1], combining theorem provers (TPS) in OMRS [11] and many others. The integration of CAS and TPS in a common environment has not yet led to powerful systems. However, some prototypes were developed which prove the advantages of such a combination, e.g. ANALYTICA [9], interfaces between HOL and MAPLE [12], and ISABELLE and MAPLE [2]. We classify communication and cooperation methods for such environments in [5]. As a result, a system for representing mathematical knowledge must also provide capabilities to represent the contexts and intermediate results of reasoning and computation. We introduce this knowledge in terms of reasoning and computation structures.

The paper is organized as follows. The hybrid knowledge representation system MANTRA is introduced in section 2. Mathematical knowledge is represented by several epistemological formalisms which can be accessed and combined by hybrid inference algorithms. A framework for the specification of mathematical domains of computation is given in section 3. The specifications can be transformed and are represented by MANTRA. Section 4 sketches structures for cooperated mathematical problem solving. Again, the structures can be represented by labeled graphs of the hybrid knowledge representation system.

[1] Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe, Am Fasanengarten 5, D-76131 Karlsruhe, Germany, {calmet,homann}@ira.uka.de

[2] School of Computer Science, University of Windsor, Windsor, Ontario N9B 3P4, Canada, ono@cs.uwindsor.ca

ECAI 96. *12th European Conference on Artificial Intelligence*
Workshop on Representation of Mathematical Knowledge
Edited by H. Stoyan, K. Homann, S. Jacob, M. Kerber

2 HYBRID KNOWLEDGE REPRESENTATION

The representation of mathematical objects requires efficient and adequate encodings together with powerful inference capabilities. We designed and implemented MANTRA[3][3, 7, 4], a system that supports the representation of mathematical knowledge.

MANTRA was implemented according to the following design principles: (i) *several cooperating formalisms* are better than a unique representation formalism, (ii) a *clear semantics* explaining the meaning of the knowledge representation language is fundamental and (iii) all algorithms involved must be *decidable* and reasonably fast. The decidability of all algorithms involved is achieved by adopting a four-valued semantics. The architecture of the system is shown in figure 1.

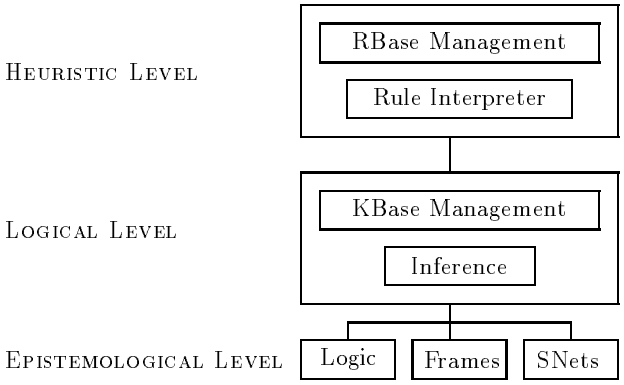

Figure 1. The Architecture of the System

The system provides four different representation formalisms – first order logic, frames, semantic nets and production systems – which can interact through hybrid inference algorithms – assertional reasoning, terminological reasoning, inheritance with exceptions and heuristic programming and any combination. The motivation is that several cooperating formalisms ought to enhance the expressiveness and inference power of the system. We adopt a knowledge representation approach consisting of a representational theory, explaining which knowledge is to be represented by which formalisms, and of a common semantics to define the relationship between expressions of different formalisms in a semantically sound manner.

Each formalism standing alone is inadequate for representing mathematical objects. However, hybrid representations and inferences provide adequacy and efficiency. The logical level supports inference mechanisms and knowledge base management – creation, modification and query. Finally, the heuristic level consists of production rules and Horn clauses providing capabilities for representing procedural knowledge and ad hoc rules for the hybrid inferences.

Due to the interconnections among the different epistemological methods a single data abstraction consisting of directed graphs has been selected. The system has been implemented in Common Lisp and uses the object-oriented extension CLOS to define and manipulate the knowledge bases. This increases the modularity of the system and makes it easy to modify. A graphical user's interface allowing the visualization and definition of knowledge is provided. It has been implemented in C with XToolkit.

The soundness of the semantics is mandatory for specifying mathematical domains of computation. The next section introduces a specification language and its transformation into MANTRA.

3 SPECIFYING MATHEMATICAL DOMAINS OF COMPUTATION

Although MANTRA is designed for the representation of mathematical knowledge, to explicitly specify mathematics in terms of hybrid knowledge representation formalisms is awkward and inappropriate. Formal specification and query languages denote and access mathematical domains and objects.

We introduced a framework, FORMAL, for specifying mathematical domains of computation and their inherently related type inference mechanisms as well as for transforming those specifications into knowledge bases of MANTRA in [17, 6]. FORMAL involves an algebraic specification language, a method to transform specifications into knowledge bases and MANTRA as illustrated in figure 2.

The specification language FORMAL-Σ provides modular and well-structured specifications. It is well-suited to specify parametric and inclusion polymorphisms in a unified way.[4] The underlying formalism is based upon the so-called homogeneous *unified algebras* [15] allowing the treatment of sorts as values.

A specification is represented by a module, e.g. the module `SemiGroup` is specified as a *basic module* possessing the constant `SemiGroup`. It also possesses the function symbol `o` that is represented in the `Operation` part together with its functionality. The property of the function symbol is expressed by means of Horn clauses with equality.

```
(Module SemiGroup
 (Define (Constants SemiGroup)
         (Operations (o (SemiGroup SemiGroup)
                        -> SemiGroup))
         (Clauses (Imply (: (a b c) SemiGroup)
                         (= (o (o a b) c)
                            (o a (o b c)))))))
```

To achieve the executability of specifications they are compiled into an executable representation by FORMAL-Θ. A specification is transformed into a knowledge base according to its syntax tree in a bottom up manner. Each transformation step makes use of a particular transformation rule represented in the following form:

$$\frac{I}{\mathbf{R}\quad 0} \quad \left\langle \begin{array}{l} C_1 \\ \cdots \\ C_n \end{array} \right.$$

The above form is also equivalent to $\dfrac{\{C_1, \cdots, C_n\}}{\vdash R[I, O]}$

[3] Modular Assertional, semantic Network and Terminological Representation Approach

[4] We use the notions of parametric and inclusion polymorphism according to [8].

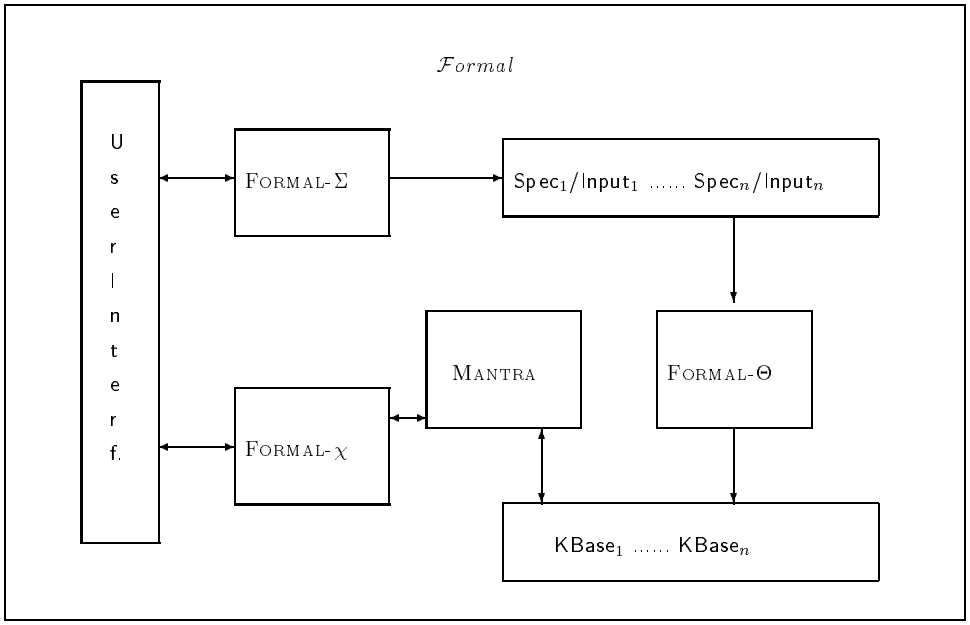

Figure 2. Overview of FORMAL

The intended meaning of such a rule is the following. I and O are called input and output scheme respectively. I is part of a specification in FORMAL-Σ and O is the corresponding representation of I in the language of MANTRA. A program scheme is a term from $W(PL \cup X)$, the term algebra over $PL \cup X$, i.e. a term over PL (programming language) containing free variables from a countable set X of typed scheme parameters. $C_1 \cdots C_n$ are applicability conditions which are Horn clauses over an enrichment of $PL \cup X$, i.e. they may contain additional syntactic and semantic predicates over program schemes.

A transformation rule is correct if it constitutes a valid inference, i.e. if the program schemes I and O are in the semantic relation indicated by R whenever the applicability conditions are valid. The correctness of the transformation of a specification, i.e. the specification and its corresponding representation in MANTRA are semantically equivalent, is verified by giving a proof for each transformation rule that there is a morphism from I to O according to the semantic predicate R.

The transformation of a specification can be outlined as follows: The signature is represented by means of frames provided at the epistemological level of MANTRA. The carrier is a distributive lattice in a unified algebra and is also modeled by frames. The Horn clauses imposed on the specification can be represented at the heuristic level of MANTRA.

Finally, the role of FORMAL-χ consists in processing queries given by the user, e.g. to simplify a term or to define the type of an expression.

4 STRUCTURES FOR MATHEMATICAL SERVICES

There have been recent efforts to combine systems for mathematical computing and reasoning [1, 2, 9, 11, 12, 13]. The design of future environments for mathematical problem solving will include algorithmic and logical services. A formal specification of mathematical services is given in [13] consisting of systems together with interfaces for restart-able computation and reasoning. Consequently, structures representing intermediate results in any kind of mathematical computation must also be considered. This section is restricted to the aspects of a suitable representation of structures. [13, 5] describe coordination and cooperation of communicating structures among mathematical services.

The formal specification of structures for the representation of derivations and application of algorithms are based on two kinds of theories. Reasoning theories are defined in [11] to consist of a sequent system $S_{sys} = \langle S, C, \models, I, \_[\_] \rangle$, a set of identifiers Id and a rule set $\tilde{r} \in Rset[S_{sys}, Id]$. Computation theories are introduced in [13] as an object system $O_{Sys} = \langle O, C, \models, I, \_[\_] \rangle$ and a set of named algorithms $\tilde{a} \in Algs[O_{Sys}, Id]$. Sequents and objects allow the use of schematic variables and can be instantiated.

By extending notions and notations given in [11] we define reasoning and computation structures as labeled graphs consisting of edges and two kinds of nodes. The nodes are labeled by their corresponding sequents or objects and justifications or explanations respectively. To enable vertical flexibility the explanations and justifications may contain nested reasoning structures together with an instantiation. This enables to nest structures within others to achieve better presentations and readability.

Let $RT = \langle S_{Sys}, Id, \tilde{r} \rangle$ and $CT = \langle O_{Sys}, Id, \tilde{a} \rangle$ be arbitrary but fixed reasoning and computation theories. A basic structure consists of two kinds of nodes and edges together with the corresponding labeling maps. A basic reasoning structure rs_I is illustrated in figure 3. The labeling of the sequent nodes was omitted because of readability. Figure 4 illus-

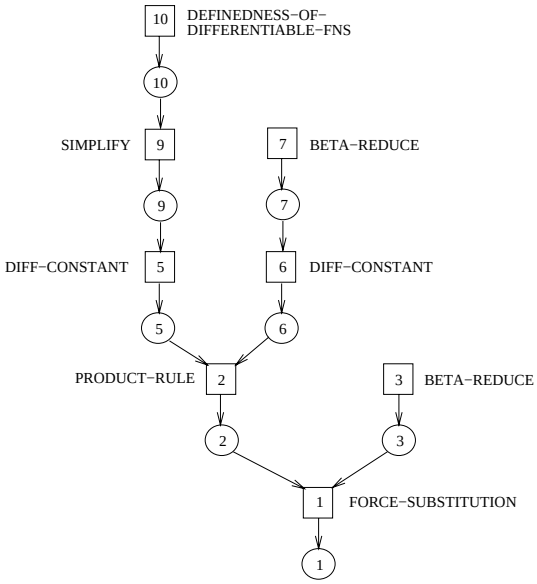

Figure 3. Reasoning structure rs_I

trates the graph of a basic computation structure without constraints. The structure represents the computation of a predicate `SquareFree` by executing three algorithms. Reasoning

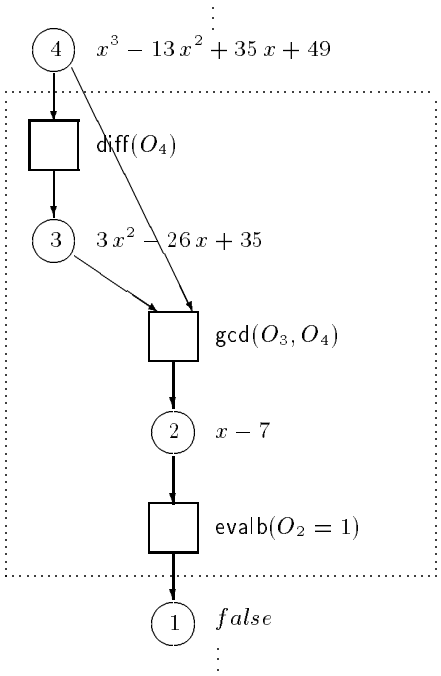

Figure 4. Parts of a computation structure cs_I

and computation structures can be instantiated, generated and manipulated by primitive operations. They are given in [11] and [13] respectively as well as theorems about reachability, instantiation of derivation, elimination of nesting and derivation of trees. The structures can be represented as directed graphs in MANTRA.

5 CONCLUSION AND FURTHER RESEARCH

This paper deals with the representation, specification and communication of mathematical knowledge. We illustrate a hybrid system which is well suited for the representation of mathematical knowledge because of the decidability of all inference problems and its capability to combine several representation paradigms. A framework for specifying mathematical domains of computation and their related type inference mechanisms can be transformed into the hybrid system. Finally, we argue that future systems for mathematical knowledge representation must be capable to manage and represent structures for cooperative mathematical problem solving.

We designed and implemented interfaces along the ideas presented in this paper. The combination of the automated theorem prover DTP and the CAS MAGMA gives solutions to mathematical problems which were unsolvable by a CAS when standing alone. Another interface between the tactical theorem prover ISABELLE and MAPLE was implemented by extending the simplifier of ISABELLE by new kinds of rules to call external functions of the CAS. The interactive proof involves computation structures with only one algorithm application. The semantic of the interaction is given in [2].

Among the work in progress is the development of an interface between the interactive proof system IMPS and the CAS CALVIN[5] which allows restart-able and incremental application of algorithms by managing contexts. They can be used to guarantee correctness of computations, e.g. requesting the context of n when integrating x^n.

REFERENCES

[1] J. ABBOTT, A. VAN LEEUWEN, A. STROTMANN
Objectives of OpenMath. Submitted to Journal of Symbolic Computation, 1995.

[2] C. BALLARIN, K. HOMANN, J. CALMET
Theorems and Algorithms: An Interface between Isabelle and Maple. In A.H.M. LEVELT (Ed.), Proceedings of International Symposium on Symbolic and Algebraic Computation (ISSAC'95), pp. 150–157, ACM press, 1995.

[3] G. BITTENCOURT
An Architecture for Hybrid Knowledge Representation, Dissertation, University of Karlsruhe, 1990.

[4] G. BITTENCOURT, J. CALMET, K. HOMANN, A. LULAY
MANTRA: A Multi-Level Hybrid Knowledge Representation System. In T. PEQUENO, F. CARVALHO (Eds.), Proceedings of the XI Brazilian Symposium on Artificial Intelligence, pp. 493–506, 1994.

[5] J. CALMET, K. HOMANN
Classification of Communication and Cooperation Mechanisms for Logical and Symbolic Computation Systems. To appear in F. BAADER, K.U. SCHULZ (Eds.), Proceedings of First International Workshop on Frontiers of Combining Systems (FroCoS'96), Kluwer Series on Applied Logic, 1996.

[6] J. CALMET, I.A. TJANDRA
A Unified-algebra-based specification language for symbolic

[5] The CAS CALVIN is being developed by our students at University of Karlsruhe.

computing. In A. Miola (Ed.), Proceedings of International Symposium on Design and Implementation of Symbolic Computation Systems (DISCO'93), pp. 122–133, LNCS 722, Springer, 1993.

[7] J. Calmet, I.A. Tjandra, G. Bittencourt
Mantra: Shell for Hybrid Knowledge Representation. In Tools for AI, pp. 164 –171, IEEE Computer Society Press, 1991.

[8] L. Cardelli and P. Wagner
On understanding types, data abstraction and polymorphism. In Computing Survey, 17(4), pp. 471 – 522, 1985.

[9] E. Clarke, X. Zhao
Combining Symbolic Computation and Theorem Proving: Some Problems of Ramanujan. In A. Bundy (Ed.), Automated Deduction (CADE-12), pp. 758–763, LNAI 814, Springer, 1994.

[10] W.M. Farmer, J.D. Guttman, F.J. Thayer
Contexts in Mathematical Reasoning and Computation. In Journal of Symbolic Computation 19:201–216, 1995.

[11] F. Giunchiglia, P. Pecchiari, C. Talcott
Reasoning Theories – Towards an Architecture for Open Mechanized Reasoning Systems. To appear in F. Baader, K.U. Schulz (Eds.), 1996, Proceedings of First International Workshop on Frontiers of Combining Systems (FroCoS'96), Kluwer Series on Applied Logic, 1996.

[12] J. Harrison, L. Théry
Extending the HOL Theorem Prover with a Computer Algebra System to Reason About the Reals. In J.J. Joyce, C.-J.H. Seger (Eds.), Higher Order Logic Theorem Proving and its Applications (HUG'93), pp. 174–184, LNCS 780, Springer, 1993.

[13] K. Homann
Symbolic Mathematical Problem Solving by Cooperation of Algorithmic and Logical Services, (in German), Dissertation, University of Karlsruhe, to appear, 1996.

[14] Roman Matuszewski
The QED Workshop II, (Ed.) Proceedings, Technical Report L/1/95, Warsaw University, published electronically at URL http://www.mcs.anl.gov/qed/index.html, 1995.

[15] P.D. Mosses
Unified algebras and institution. In Logics in Computer Science, pp 304 – 312. IEEE Press, 1989.

[16] P.F. Patel-Schneider
A Decidable First-Order Logic for Knowledge Representation, Proceeding of IJCAI 9, 1985.

[17] I.A. Tjandra
Algebraic Specification of Mathematical Domains of Computation and Polymorphic Types in Computer Algebra, (in German), Dissertation, University of Karlsruhe, 1993.