Inductive Theorem Proving and Computer Algebra in the MathWeb Software Bus

Jürgen Zimmer[1]* and Louise A. Dennis[2]**

[1]Division of Informatics, University of Edinburgh,
jzimmer@mathweb.org
[2]School of Computer Science and Information Technology, University of Nottingham,
lad@cs.nott.ac.uk

Abstract. Reasoning systems have reached a high degree of maturity in the last decade. However, even the most successful systems are usually not general purpose problem solvers but are typically specialised on problems in a certain domain. The MathWeb Software Bus (MathWeb-SB) is a system for combining *reasoning specialists* via a common software bus. We describe the integration of the λ-Clam system, a reasoning specialist for proofs by induction, into the MathWeb-SB. Due to this integration, λ-Clam now offers its theorem proving expertise to other systems in the MathWeb-SB. On the other hand, λ-Clam can use the services of any reasoning specialist already integrated. We focus on the latter and describe first experiments on proving theorems by induction using the computational power of the MAPLE system within λ-Clam.

1 Introduction

Reasoning systems have reached a high degree of maturity in the last decade.[1] However, these systems are usually not general purpose problem solvers but should be seen as *reasoning specialists* with individual expertise in a certain domain. Automated Theorem Provers (ATPs), for instance, are specialised in equality proofs or proof by induction. Some Computer Algebra Systems (CASs) are, for example, specialised on group theory, others in computations in polynomial rings.

The MathWeb Software Bus (MathWeb-SB) [FK99] provides the means for inter-operability of reasoning specialists. It connects a wide-range of reasoning systems (*MathWeb services*), such as ATPs, CASs, model generators (MGs), (semi-)automated proof assistants, constraint solvers (CSs), human interaction

* The author is supported by the European Union IHP grant CALCULEMUS HPRN-CT-2000-00102.

** The author was funded by EPSRC grant Gr/M45030.

[1] Throughout this paper, by *reasoning system* we mean both logic-based Deduction Systems and computation-based systems such as Computer Algebra Systems or Constraint Solvers.

units, and automated concept formation systems, by a common *mathematical software bus*. Reasoning systems integrated in the MathWeb-SB can therefore offer new functionality to the pool of services, and can in turn use all services offered by other systems. The communication between systems in the MathWeb-SB is based on the standards OpenMath [CC98] and OMDoc [Koh00] for encoding mathematical content.

A crucial idea behind frameworks like the MathWeb-SB is that opening existing reasoning specialists and combining them via a common software bus can lead to synergetic effects. By using the expertise of other reasoning specialists, a reasoning system can solve problems that are way beyond its previous problem solving horizon. A typical example for this kind of synergy is the use of external reasoning systems, like CASs, MGs, CSs, and ATPs, in the mathematical assistant system Ωmega [BCF$^+$97]. The use of external reasoners allows Ωmega to solve problems in many different domains, such as group theory, analysis, and economics.

In the last three years we have steadily increased the list of reasoning specialists integrated in the MathWeb-SB. In this paper, we describe our work in progress on the integration of the λ-Clam [RSG98] system, a proof planning system specialised on proof by induction. The advantages of integrating λ-Clam into the MathWeb-SB are twofold: i) λ-Clam can offer its expertise in inductive theorem proving to other reasoning systems, e.g., to the Ωmega proof planner which does not have the expertise to perform inductive proofs, and ii) λ-Clam can use all reasoning specialists already available in the MathWeb-SB.

For our work we have investigated both directions, λ-Clam offering services and λ-Clam using external reasoning systems. Due to space limitations we focus on the latter in this paper, especially on the use of CAS computation in the λ-Clam proof planner. For our first experiments with λ-Clam using CAS we studied theorems proved by the systems *Analytica* [CZ92,BCZ98] and CLAM-Lite [Wal00]. *Analytica* is a theorem prover implemented in the programming language of the CAS *Mathematica*. CLAM-Lite is a simple re-implementation of the proof planner CLAM [BvHHS90] in Maple's programming language. Thus, both systems are based on the (re-)implementation of a theorem prover in the programming language of a CAS. Preliminary results of our experiments show that, in principle, we can prove the same theorems as *Analytica* and CLAM-Lite using a far more general approach which is even easier to implement, namely the combination of existing reasoning systems via a software bus. In our case, the Maple system was already available in the MathWeb-SB which reduced the task to the integration of λ-Clam and the implementation of services (*phrase-books*) for the translation of OpenMath and OMDoc from and to λ-Clam's logic.

The structure of this paper is as follows. We first give brief descriptions of the MathWeb-SB and the λ-Clam system in sections 2 and 3 respectively. In section 4, we describe the integration of λ-Clam into the MathWeb-SB. We also describe the reasoning services offered by λ-Clam and discuss the use of external reasoning systems in inductive theorem proving in λ-Clam. In section 5 we present first results from experiments with λ-Clam using the CAS Maple.

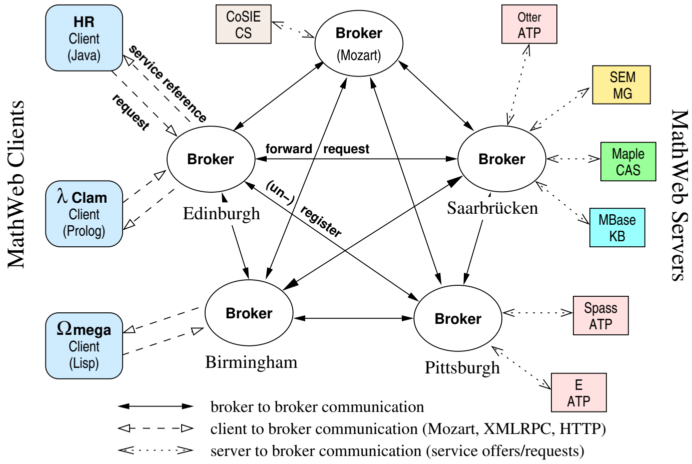

Fig. 1. The MathWeb Software Bus

We draw a conclusion in section 6 and list various directions for future work in section 8.

2 The MathWeb Software Bus

The MathWeb Software Bus [FK99] is a platform for distributed automated reasoning that supports the connection of a wide range of mathematical services by a common software bus.[2] The MathWeb-SB provides the functionality for turning existing reasoning systems into mathematical services that are homogeneously integrated into a networked proof development environment. Each system benefits from being able to use the features of the other components in the MathWeb-SB. The MathWeb-SB is implemented in the multi-paradigm programming language Mozart which enables easy distribution of applications over a LAN or the Internet [Smo95,Con]. This allows the MathWeb-SB to offer a stable network of distributed mathematical services that is in every day use. Client applications can access 23 different reasoning systems, for instance, the CASs Maple, MagMa, and CoCoA, the CS $\mathcal{C}o\mathcal{SIE}$, mediators, and MGs such as Mace and Satchmo. Moreover, the MathWeb-SB integrates nine first order ATPs, such as Otter, Spass, Bliksem, and E.

The current architecture of the MathWeb system is depicted in Fig. 1. Local *brokers* provide routing and authentication information to the mathematical services. MathWeb-SB servers (wrappers) encapsulate the actual reasoning systems and offer the mathematical services to their local broker. Brokers register to other brokers running remotely and thus build a dynamic web of brokers. Client

[2] Further information about the MathWeb-SB is available at `http://www.mathweb.org/mathweb/`.

applications such as the ΩMEGA system, HR, or λ-Clam, connect to one of the MathWeb brokers and request services. If the requested service is not offered by a local reasoning system, the broker forwards the request to all known remote brokers until the service is found. If the requested service is found, the client application receives a reference to a newly created *service object*. The client can then send messages directly to this service object.

The MathWeb-SB originated in the use of external reasoning specialists in the ΩMEGA system. It turned out that the use of external reasoners in proof planning can give valuable support to a human user constructing a proof plan and can significantly extend the problem solving horizon of an automated proof planner. Currently, ΩMEGA's use of external reasoning systems via the MathWeb-SB can be classified into the following categories:

- the use of CAS computations during proof planning
- controlling the proof planning process with the help of CAS computation
- sending subgoals to first order (higher order) ATPs
- finding witness terms for existentially quantified variables using model generators and constraint solvers

The successful use of these reasoning systems in ΩMEGA was one of the main motivations for the work described in this paper, namely the integration λ-Clam into the MathWeb-SB. On the one hand, λ-Clam can now use all reasoning specialists available in the MathWeb-SB. On the other hand, λ-Clam can offer its expertise in inductive theorem proving to other reasoning systems, e.g., the ΩMEGA proof planner which does not provide the proof planning methods and the control knowledge needed for proofs by induction.

In the following, we briefly describe the λ-Clam system and its main reasoning features.

3 The λ-Clam Proof Planner

λ-Clam [RSG98] is a higher-order proof planning system. It is a descendant of the CLAM [BvHHS90] series and is specialised for proof by induction, but it is also intended to allow the rapid prototyping of automated theorem proving strategies. It is implemented in Teyjus λ-PROLOG which is a higher order extension of standard PROLOG.

Proof Methods. λ-Clam works by using depth-first planning with *proof methods*. *Atomic methods* encode more or less complex reasoning steps ranging from quantifier elimination to the step case of an induction proof or difference reduction between a goal and its hypotheses. Atomic methods can be composed to *compound methods* using *methodicals* [RS01]. Compound methods offer the means to define search strategies at various levels of abstraction. A proof plan produced by λ-Clam is a tree of the subgoals generated during proof labelled with the atomic method applications applied to that subgoal. The leaves of the tree are the trivial subgoals: *true*. These proofs are not fully formal but are closer to the formality of proofs found in mathematical textbooks.

Rippling is one of the most widely used and most powerful proof methods. It is used for difference reduction, particularly in inductive proof planning. It was first introduced in [BSvH+93]. The implementation of rippling in λ-Clam is based on the theory presented by Smaill & Green [SG96] who proposed a version that naturally coped with higher-order features. The main advantages of rippling are that it allows an equation to be treated as a rewrite in both directions without loss of termination and provides useful information for automatically patching failed proof attempts.

Proof Critics. The methodical expressions which make up compound methods provide a guide to which proof methods should be chosen at any given stage of the proof. Knowing which method is expected to apply gives additional information in case the system generating the plan fails to apply it. Since heuristics are employed in the generation of proof plans it is possible for a proof planning system to fail to find a plan even when one exists. To this end *proof critics* [IB96] can be employed to analyse the reasons for failure and propose alternative choices. The proof critics currently implemented in λ-Clam allow for generalising the original goal, modifying the chosen induction scheme or speculating a missing lemma (rewrite rule).

4 λ-Clam in the MathWeb-SB

We integrated λ-Clam into the MathWeb-SB using the methodology that has been successfully used for the integration of other systems like the ΩMEGA proof planner or the higher order theorem prover TPS: A MathWeb-SB *wrapper* implements the interface to the services offered by λ-Clam and has full control over λ-Clam by simulating user input using socket communication. λ-Clam itself can use the wrapper to access other services as external reasoning systems.

4.1 Services offered by λ-Clam

All services offered and used by λ-Clam are based on the OpenMath [CC98] and the OMDoc [Koh00] standard. We implemented a translation service (a phrase-book) to translate incoming formulae, definitions, and conjectures into λ-Clam's higher order abstract syntax. For the sake of efficiency, this translation is performed by the MathWeb-SB wrapper. λ-Clam currently offers two services to the MathWeb-SB:

`planProblem` This service takes an OMDoc document, containing a single conjecture, as an argument. The service starts the λ-Clam proof planning mechanism on the conjecture. In our current implementation, the service expects the conjecture to be about natural number arithmetic. We plan an extension of the service such that clients can also provide the theory the conjecture is defined in.

Client applications using the `planProblem` service can use optional arguments to determine which proof strategy (compound method) λ-Clam should use for the planning attempt, and to give a time limit in seconds. In the current implementation, the service simply returns the OpenMath symbol *true* if λ-Clam could find a proof plan within the given time limit, and $false$ if no proof plan could be found.

`ripple` We argued in section 3 that rippling is one of the most successful heuristics for guiding (inductive) proof planning. Therefore, λ-Clam offers its rippling mechanism as a separate service to the MathWeb-SB. The service is given a single input formatted using the OMDoc standard. The OMDoc must contain a non-empty set of rewrite rules, formalised as lemmas, and a goal sequent $H \vdash \varphi$ as a conjecture. The `ripple` service tries to reduce the difference between φ and the best suitable hypothesis in H using the rewrite rules. The `ripple` service also tries to apply fertilisation to reduce the goal φ to the trivial goal *true*. As a result, the service `ripple` returns an OMDoc which contains the resulting proof planning goal as a sequent $H \vdash \varphi'$.

The services offered by λ-Clam can be used by other reasoning systems via the MathWeb-SB. First experiments with the use of the `planProblem` service and the `ripple` service within the ΩMEGA proof planner were successful. We formalised problems in the natural number theory of ΩMEGA and implemented proof planning methods that call the `planProblem` and the `ripple` service of λ-Clam to close open subgoals or to reduce a subgoal to a simpler one, respectively.

One advantage of passing lemmas from ΩMEGA's theories as rewrites to λ-Clam's `ripple` service is that the rewriting process is completely independent of λ-Clam's theories. Thus, λ-Clam can be used as an abstract rewriting engine whose termination is guaranteed. However, the current version of λ-Clam does not maintain a trace of the positions of the subterms to which a rewrite rules was applied. The latter would allow the `ripple` service to tell a client application, like ΩMEGA, exactly which rewrite has been applied to which subterm of the planning goal not just the rewrite rule that had been applied to that goal. ΩMEGA could then use this information to construct a natural deduction proof for the reasoning steps performed during rippling.

4.2 λ-Clam using MathWeb Services

For the use of external reasoning systems, we extended λ-Clam by a module which abstracts from socket communication details and offers a convenient interface to MathWeb-SB wrapper. Furthermore, we implemented a module to translate λ-Clam's formulae into the OpenMath standard using the core OpenMath Content Dictionaries (CDs). Thus, the module forms the second half of an OpenMath phrase-book for λ-Clam. Using the `mathweb` module, λ-Clam can now access every reasoning system available in the MathWeb-SB. We are currently thinking of the following applications of external reasoners in λ-Clam:

Using ATPs to prove simple subgoals: This approach has already been used in the ΩMEGA proof planner to restrict the search space. In some cases, even higher order problems can be transformed to first order problems and then sent to one or more first order ATPs in the MathWeb-SB.

Using ATPs on the control level: It is a known shortcoming of λ-Clam that it does not check the consistency of hypotheses after performing a case split. This leads to λ-Clam missing easy proofs by inconsistency. Modern ATPs are very efficient and could detect trivial inconsistencies in a few milliseconds. We therefore try to prune inconsistent search paths in proof planning with the help of ATPs like Otter.

CAS computation in proof planning: Due to our positive experience with CAS computations in many proof planning domains, we think that the use of CASs in inductive proof planning can enable λ-Clam to solve problems it couldn't solve before. The rewriting capabilities of a CAS can complement the rewriting of λ-Clam and can thus enhance the reasoning capabilities of λ-Clam.

In our work, we followed all three applications of external reasoning system but focused on the last application, namely the use of Maple's simplification in inductive proof planning. On the Maple side, we therefore extended the `simplifyTerm` service. This service now accepts a term (formula) in OpenMath representation. It returns the OpenMath representation of the term (formula) resulting from the call of the `simplify` function of Maple. Before this extension, `simplifyTerm` accepted only Maple syntax. To allow the access of external reasoning systems in the proof planner, we implemented two new proof methods for λ-Clam. The method `otter` can be used to send an open subgoal to the ATP Otter. The `maple_simplify` method calls the `simplifyTerm` service of Maple on the current subgoal and introduces the resulting formula as a new subgoal. λ-Clam treats both these systems as oracles and simply labels the appropriate node in the proof plan with the name of the Oracle.

In the following section we present first experience with the use of Maple in λ-Clam.

5 Using Maple in λ-Clam

Our experiments should rather be seen as a *proof of concept* than as an extensive case study. So far, we have focused on theorems taken from two works: The *Analytica* theorem prover and the CLAM-Lite proof planner. All theorems have been formalised in λ-Clam's theories and will be available in version 3.2 of λ-Clam. In the following, we describe the theorems we have been looking at so far and give examples for the rewriting (simplification) performed by Maple in the different cases. As we mentioned above, the communication between λ-Clam and Maple is fully based on the OpenMath standard, but for the sake of readability we use a convenient mathematical notation here.

5.1 Analytica Theorems

The *Analytica* system [CZ92,BCZ98] is a theorem prover implemented in the programming language of the CAS *Mathematica*. It is based on rewriting and uses term simplification and other computations available in *Mathematica*. E. Clarke and X. Zhao used their system to prove rather complex theorems in real-valued standard analysis, such as theorems about inequalities, limit theorems, finite sums, and about the convergence of series. *Analytica* can also prove that a function found by Weierstrass is continuous over the real numbers but nowhere differentiable.

Since λ-Clam is specialised to inductive proofs we first tried to prove *Analytica* problems that require induction. We formalised the two induction problems presented in [CZ92] and [BCZ98]. The first theorem gives a closed representation for the Fibonacci numbers:

Theorem 1. *Let $Fib(n)$ be the nth Fibonacci number, i.e. $Fib(0) = 0$, $Fib(1) = 1$, and $\forall n \in \mathbb{N}. n \geq 2 \Rightarrow Fib(n) = Fib(n-1) + Fib(n-2)$. Then the following holds:*

$$\forall n. Fib(n) = \frac{(1+\sqrt{5})^n - (1-\sqrt{5})^n}{2^n \sqrt{5}}.$$

Proving Theorem 1 requires complex reasoning about the square root function and exponentiation which is very hard if not infeasible to encode as rewriting rules in λ-Clam. λ-Clam therefore uses MAPLE's simplification whenever the rewrite rules initially given are not sufficient to perform the required reasoning steps. λ-Clam uses the following induction scheme for the proof planning attempt:

$$\frac{P(0) \wedge P(s(0)) \wedge \forall n. (P(n) \wedge P(s(n)) \Rightarrow P(s(s(n))))}{\forall n. P(n)},$$

where s is the successor function on the natural numbers. In the step case $(P(n) \wedge P(s(n)) \Rightarrow P(s(s(n)))$ of the inductive proof, λ-Clam uses the definition of the Fibonacci numbers to rewrite the goal until no rewriting is applicable anymore. At this point of the proof λ-Clam tries to apply the `maple_simplify` method which performs the following rewriting:

$$\frac{(1+\sqrt{5})^n - (1-\sqrt{5})^n}{2^n \cdot \sqrt{5}} + \frac{(1+\sqrt{5})^{n+1} - (1-\sqrt{5})^{n+1}}{2^{n+1} \cdot \sqrt{5}} = \frac{(1+\sqrt{5})^{n+2} - (1-\sqrt{5})^{n+2}}{2^{n+2} \cdot \sqrt{5}} \rightsquigarrow$$
$$\frac{1}{10} \cdot 5^{\frac{1}{2}} \cdot 2^{-n} \cdot (3 \cdot (1 + 5^{\frac{1}{2}})^n - 3 \cdot (1 - 5^{\frac{1}{2}})^n + (\ 1 + 5^{\frac{1}{2}})^n \cdot 5^{\frac{1}{2}} + (1 - 5^{\frac{1}{2}})^n \cdot 5^{\frac{1}{2}} =$$
$$\frac{1}{10} \cdot 5^{\frac{1}{2}} \cdot 2^{-n} \cdot (3 \cdot (1 + 5^{\frac{1}{2}})^n - 3 \cdot (1 - 5^{\frac{1}{2}})^n + (1 + 5^{\frac{1}{2}})^n \cdot 5^{\frac{1}{2}} + (1 - 5^{\frac{1}{2}})^n \cdot 5^{\frac{1}{2}}.$$

λ-Clam can easily reduce the resulting goal to the trivial goal by applying equality reflexivity.

The second theorem proved by λ-Clam is taken from [BCZ98] and gives a closed form of a modified geometric sum:

Theorem 2. *For all natural numbers $m \neq 1$ the following holds:*

$$\forall n. \sum_{k=0}^{n} \frac{2^k}{1+m^{2^k}} = \frac{1}{m-1} + \frac{2^{n+1}}{1-m^{2^{n+1}}}$$

During the planning for this theorem, one crucial simplification step that cannot be performed by λ-Clam, but by MAPLE, occurs in the step case: $\frac{1}{-1+m} + \frac{2}{1-m^2} \rightsquigarrow \frac{1}{1+m}$.

5.2 CLAM-Lite Theorems

In [WNB92], T. Walsh, A. Nunes and A. Bundy presented automatically synthesised closed forms of sums. Recently, T. Walsh implemented the simple "proof planning" shell CLAM-Lite in the CAS Maple and used his system to prove the correctness of some of these closed forms using Maple computations [Wal00].

We formalised all theorems of [WNB92] in λ-Clam and tried to prove them using Maple via the MathWeb-SB. The theorems and results are shown in Fig. 2. A plus sign in the column for CLAM-Lite or λ-Clam means that the respective

No	Closed Form	CLAM-Lite	λ-Clam using Maple
3	$\sum i = \frac{n\cdot(n+1)}{2}$	+	+
4	$\sum i^2 = \frac{2n^3+3n^2+n}{6}$	+	+
5	$\sum i + i^2 = \frac{2n^3+6n^2+4n}{6}$	+	+
6	$\sum a^i = \frac{a^{n+1}-1}{a-1}$	+	+
7	$\sum ia^i = \frac{(n+1)a^{n+1}-a\cdot\frac{a^{n+1}-1}{a-1}}{a-1}$	+	-
8	$\sum(i+1)\cdot a^i = \frac{(n+1)a^{n+1}}{a-1} - \frac{a^{n+1}-1}{(a-1)^2}$	+	+
9	$\sum \frac{1}{i(i+1)} = \frac{n}{n+1}$	+	-
10	$\sum Fib(i) = Fib(n+2)-1$	-	+

Fig. 2. Closed forms proved by CLAM-Lite and by λ-Clam. All sums are over i from 0 to n, $a \neq 1$

system could find a proof plan for the problem at hand. A minus indicates failure.

The proof of Theorem 3 in λ-Clam is very similar to the proof presented in [Wal00] and Maple performs the same rewriting in the step case of the proof. The reason for λ-Clam's failure on Theorems 7 and 9 is more a technical than a conceptual problem: The normal form of terms returned by Maple is sometimes much bigger than the original term and due to a bug in Teyjus λ-PROLOG (the implementation language of λ-Clam), λ-Clam can currently not read in terms that exceed a certain size.

Interestingly, CLAM-Lite could not find a proof plan for Theorem 10 although it had access to the same version of Maple that λ-Clam is using. On the other hand, λ-Clam could find a proof plan, using Maple to perform the following rewrite at the end of the step case:

$$(Fib(n) + Fib(n+1) - 1) + Fib(n+1) = (Fib(n+1) + (Fib(n) + Fib(n+1))) - 1$$
$$\rightsquigarrow 2 \cdot Fib(n+1) + Fib(n) - 1 = 2 \cdot Fib(n+1) + Fib(n) - 1,$$

which again is difficult to perform solely with rewriting in λ-Clam because we would have to define a special term ordering to guaranty the termination of the rewriting system. Walsh names the simplicity of CLAM-Lite's planning methods as a reason for CLAM-Lite's failure on Theorem 10. The fact that λ-Clam could prove the theorem underlines our thesis that it is a better choice to combine existing deduction systems via protocols instead of re-implementing them on top of a Computer Algebra System.

6 Conclusion

We describe the integration of the λ-Clam proof planner that is specialised to inductive theorem proving into the MathWeb-SB. We described the advantages of this integration: λ-Clam can now offer its expertise in proof by induction to other reasoning systems in the MathWeb-SB and can use the reasoning services already offered in the MathWeb-SB. We focused on the latter and described first experiments with the use of the CAS MAPLE in λ-Clam to prove theorems about Fibonacci numbers and closed forms of sums. λ-Clam could *not* find proof plans for these theorems without MAPLE's computational power. This paper describes work in progress and we hope to present more results in the final version.

7 Related Work

Several frameworks for the combination of reasoning systems have been developed. The Logic Broker Architecture (LBA) [AZ00] is most similar to the MathWeb-SB. It is a platform for the distributed automated reasoning based on the distribution facilities of the CORBA middle-ware and the OPENMATH standard.

The PROSPER project [DCN+00] aims at developing the technology needed to deliver the benefits of formal specification and verification to system designers in industry. The central idea of the PROSPER project is that of a *proof engine* (a custom built verification engine) which can be accessed by applications via an application programming interface (API). This API is supposed to support the construction of CASE-tools.

Our experiments were strongly inspired by our positive experience with the use of CAS computations in the ΩMEGA proof planner [KKS96]. Similar work has been done for an integration of CASs in the higher order theorem prover HOL [HT93] and for the MAPLE-PVS interface described in [DGKM01].

8 Future Work

The preliminary results we got from our experiments with the use of CAS in inductive proof planning were promising and in the near future we will extend our experiments to a fully-fledged case study. We are going to formalise more theorems about closed forms of summations and try to find proof plans for these theorems. We plan a more detailed comparison of our work with [Wal00] also on the basis of quantitative results (runtime comparisons).

λ-Clam offers the means to formalise and prove theorems in non-standard analysis (NSA) [MFS02]. Using NSA, λ-Clam could already find proof plans for the limit theorems LIM-PLUS and LIM-TIMES and some other analysis theorems, e.g., the mean-value theorem. In contrast to the classical ϵ-δ proofs, NSA proofs tend to be much shorter and more intuitive. Hence, it would be interesting to use λ-Clam's NSA theory to construct alternative proofs for some of the complex theorems proved by the *Analytica* system using the computational power of CASs available in the MathWeb-SB.

We hope in future to experiment, where appropriate, with the communication of plans between systems. This would allow λ-Clam's proof plans to be replayed in an LCF-style theorem prover to produce fully formal proofs. It would also allow the output of oracles such as MAPLE to be expanded within λ-Clam to alleviate concerns about the combinations of different logics etc.

The proper use of context in the communication between reasoning systems is still an open research question. Context can not only reduce the amount of information that has to be transfered between systems, it is also crucial to establish more complex forms of collaboration and coordination between reasoning systems. The λ-Clam proof planner, for instance, offers the powerful *critics* mechanism which analyses the failure of a proof attempt and gives feedback to the user about possible ways of correcting the proof. This feedback can include generalising the original goal, modifying the chosen induction scheme, or speculating a new rewrite rule. Potentially this feedback could also be given to another reasoning system using λ-Clam and this would involve far more complex interactions in terms of context. For instance, if λ-Clam were to suggest modifying the induction scheme this new proposed scheme might have to be transmitted back to the client system for verification in terms of its own logic. To enable this form of fine-grained interaction between reasoning systems we plan to develop a general notion of context in inter-system communication and to use λ-Clam as a prototypical reasoning system that builds up a context.

References

[AZ00] A. Armando and D. Zini. Towards Interoperable Mechanized Reasoning Systems: the Logic Broker Architecture. In A. Poggi, editor, *Proceedings of the AI*IA-TABOO Joint Workshop 'From Objects to Agents: Evolutionary Trends of Software Systems'*, Parma, Italy, May 2000.

[BCF+97] C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, K. Kohlhase, A. Meirer, E. Melis, W. Schaarschmidt, J. Siekmann, and V. Sorge. ΩMEGA: Towards a mathematical assistant. In W. McCune, editor, *Proc. of the 14th Conference on Automated Deduction*, volume 1249 of *LNAI*, pages 252–255. Springer Verlag, 1997.

[BCZ98] A. Bauer, E. Clarke, and X. Zhao. Analytica — an Experiment in Combining Theorem Proving and Symbolic Computation. *Journal of Automated Reasoning (JAR)*, 21(3):295–325, 1998.

[BSvH+93] A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, and A. Smaill. Rippling: A heuristic for guiding inductive proofs. *AI*, 62:185–253, 1993. Also available from Edinburgh as DAI Research Paper No. 567.

[BvHHS90] A. Bundy, F. van Harmelen, C. Horn, and A. Smaill. The Oyster-Clam system. In M. E. Stickel, editor, *Proc. of the 10th International Conference* *on Automated Deduction*, pages 647–648. Springer-Verlag, 1990. LNAI No. 449. Also available from Edinburgh as DAI Research Paper 507.

[CC98] O. Caprotti and A. M. Cohen. Draft of the Open Math standard. The Open Math Society, `http://www.nag.co.uk/projects/OpenMath/omstd/`, 1998.

[Con] The Mozart Consortium. The mozart programming system. http://www.mozart-oz.org/.

[CZ92] E. Clarke and X. Zhao. Analytica – A Theorem Prover for Mathematica. Technical Report CMU//CS-92-117, Carnegie Mellon University, School of Computer Science, October 1992.

[DCN+00] L. A. Dennis, G. Collins, M. Norrish, R. Boulton, K. Slind, G. Robinson, M. Gordon, and T. Melham. The prosper toolkit. In *Proc. of the 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS-2000*, LNCS, Berlin, Germany, 2000. Springer Verlag.

[DGKM01] M. Dunstan, H. Gottliebsen, T. Kelsey, and U. Martin. A maple-pvs interface. *Proc. of the Calculemus Symposium 2001*, 2001.

[FK99] A. Franke and M. Kohlhase. System description: MathWeb, an agent-based communication layer for distributed automated theorem proving. In Harald Ganzinger, editor, *Proc. of the 16th Conference on Automated Deduction*, volume 1632 of *LNAI*, pages 217–221. Springer Verlag, 1999.

[HT93] J. Harrison and L. Théry. Extending the HOL Theorem Prover with a Computer Algebra System to Reason About the Reals. In C.-J. H. Seger J. J. Joyce, editor, *Higher Order Logic Theorem Proving and its Applications (HUG '93)*, volume 780 of *LNCS*, pages 174–184. Springer Verlag, 1993.

[IB96] A. Ireland and A. Bundy. Productive use of failure in inductive proof. *JAR*, 16(1–2):79–111, 1996. Also available as DAI Research Paper No 716, Dept. of Artificial Intelligence, Edinburgh.

[KKS96] M. Kerber, M. Kohlhase, and V. Sorge. Integrating Computer Algebra with Proof Planning. In Jaques Calmet and Carla Limongelli, editors, *Design and Implementation of Symbolic Computation Systems; International Symposium, DISCO '96, Karlsruhe, Germany, September 18-20, 1996; Proc.*, volume 1128 of *LNCS*. Springer Verlag, 1996.

[Koh00] M. Kohlhase. OMDoc: An open markup format for mathematical documents. Seki Report SR-00-02, Fachbereich Informatik, Universität des Saarlandes, 2000. `http://www.mathweb.org/omdoc`.

[MFS02] E. Maclean, J. Fleuriot, and A. Smaill. Proof-planning non-standard analysis. In *Proc. of the 7th International Symposium on Artificial Intelligence and Mathematics*, 2002.

[RS01] J. Richardson and A. Smaill. Continuations of proof strategies. In Maria Paola Bonacina and Bernhard Gramlich, editors, *Proc. of the 4th International Workshop on Strategies in Automated Deduction*, Siena, Italy, June 2001.

[RSG98] J.D.C. Richardson, A. Smaill, and I. Green. System description: Proof planning in higher-order logic with lambda-clam. In C. Kirchner and H. Kirchner, editors, *Proc. of the 15th International Conference on Automated Deduction*, volume 1421 of *LNCS*, pages 129–133. Springer-Verlag, 1998.

[SG96] A. Smaill and I. Green. Higher-order annotated terms for proof search. In Joakim von Wright, Jim Grundy, and John Harrison, editors, *Theorem*

Proving in Higher Order Logics: 9th International Conference, TPHOLs'96, volume 1275 of *LNCS*, pages 399–414, Turku, Finland, 1996. Springer-Verlag. Also available as DAI Research Paper 799.

[Smo95] G. Smolka. The Oz programming model. In Jan van Leeuwen, editor, *Computer Science Today*, volume 1000 of *LNCS*, pages 324–343. Springer-Verlag, Berlin, 1995.

[Wal00] T. Walsh. Proof Planning in Maple. In *Proc. of the CADE-17 workshop on Automated Deduction in the Context of Mathematics*, 2000.

[WNB92] T. Walsh, A. Nunes, and A. Bundy. The use of proof plans to sum series. In D. Kapur, editor, *Proc. of the 11th Conference on Automated Deduction*, volume 607 of *LNCS*, pages 325–339, Saratoga Spings, NY, USA, 1992. Springer Verlag.