Experiments with an Agent-oriented Reasoning System

Christoph Benzmüller[1], Mateja Jamnik[2], Manfred Kerber[2] and Volker Sorge[1]

[1]Fachbereich Informatik, Universität des Saarlandes
66041 Saarbrücken, Germany
chris|sorge@ags.uni-sb.de
http://www.ags.uni-sb.de/
[2]School of Computer Science, The University of Birmingham
Birmingham B15 2TT, England, UK
M.Jamnik|M.Kerber@cs.bham.ac.uk
http://www.cs.bham.ac.uk/

Abstract. This paper discusses experiments with an agent oriented approach to automated and interactive reasoning. The approach combines ideas from two subfields of AI (theorem proving/proof planning and multi-agent systems) and makes use of state of the art distribution techniques to decentralise and spread its reasoning agents over the internet. It particularly supports cooperative proofs between reasoning systems which are strong in different application areas, e.g., higher-order and first-order theorem provers and computer algebra systems.

1 Introduction

The last decade has seen a development of various reasoning systems which are specialised in specific problem domains. Theorem proving contests, such as the annual CASC[1] competition, have shown that these systems typically perform well in particular niches but often do poorly in others. First-order provers, for instance, are not even applicable to higher-order problem formulations. Computer algebra systems and deduction systems typically have orthogonal strengths. Whereas many hard-wired integrations of reasoning systems have been shown to be fruitful, rather few architectures have been discussed so far that try to extend the application range of reasoning systems by a flexible integration of a variety of specialist systems.

This paper discusses the implementation of experiments with an agent oriented reasoning approach, which has been presented as a first idea in [BJKS99]. The system combines different reasoning components such as specialised higher-order and first-order theorem provers, model generators, and computer algebra systems. It employs a classical natural deduction calculus in the background to bridge gaps between sub-proofs of the single components as well as to guarantee correctness of constructed proofs. The long term goal is to widen the range of mechanisable mathematics by allowing a flexible cooperation between specialist

[1] CADE ATP System Competitions, see also http://www.cs.jcu.edu.au/~tptp/.

systems. This seems to be best achieved by an agent-based approach for a number of reasons. Firstly, from a software engineering point of view it offers a flexible way to integrate systems. Secondly, and more importantly, the agent-oriented approach enables a flexible proof search. This means that each single system – in form of a pro-active (software) agent – can focus on parts of the problem it is good at, without the need to specify a priori a hierarchy of calls. Currently we still work with a centralised approach and focus on the construction of a single proof object. This means all agents pick up and investigate the central proof object, given in higher-order natural deduction style with additional facilities to abstract from pure calculus layer [CS00]. In case they find that they are applicable in the current proof context they fulfill their task by invoking a tactic by, for instance, calling the external system they encapsulate. After consuming the available resources they come back and make bids in terms of (probably) modified proof objects. Based on heuristic criteria[2] one bid is accepted and executed by the central system while the remaining ones are stored for backtracking purposes. In this sense global cooperation and communication is established in our approach via a central proof object. The benefit is that we have to care only about translations into one single proof representation language, which reduces the proof theoretical and logical issues to be addressed. Furthermore, our central proof object makes use of a human oriented natural deduction format which eases user interaction. For human oriented proof presentation we employ the graphical user interface LOUI [SHB+99] and the proof verbalisation system P.rex [Fie01].

However, extensive communication amongst the agents is currently also a weakness of our system, since too much of the resources are spent on communication. Hence, a future goal is to subsequently reduce this overhead by extending the agents' reasoning capabilities and also by decentralising the approach. A discussion of particular agenthood aspects of our agents will be given in Section 4.

Using the agent paradigm enables us to overcome many limitations of static and hard-wired integrations. Furthermore, the agent based framework helps us to desequentialise and distribute conceptually independent reasoning processes as much as possible. An advantage over hard-wired integrations or even re-implementations of specialised reasoners is that it makes the reuse of existing systems possible (even without the need for a local installation of these systems). Accessing external systems is orchestrated by packages like MATHWEB [FHJ+99] or the logic broker architecture [AZ01]. From the perspective of these infrastructure packages our work can be seen as an attempt to make strong use of their system distribution features.

Our system currently uses about one hundred agents. They are split in several agent societies where each society is associated with one natural deduction rule/tactic of the base calculus. This agent set is extended by further agents encapsulating external reasoners. The encapsulation may be a direct one in case of locally installed external systems, or an indirect one via the MATHWEB framework, which facilitates their distribution over the internet. Employing numerous

[2] For instance, bids with closed (sub)goals are preferred over partial results, and big steps in the search space are preferred over calculus level steps.

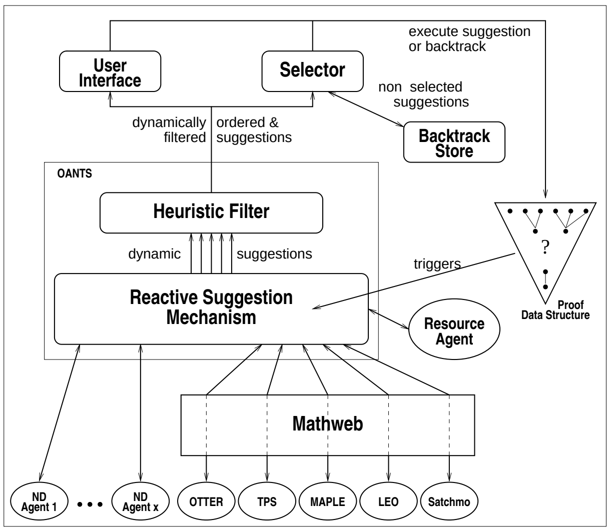

Fig. 1. System architecture.

agents, amongst them powerful theorem provers which are computationally expensive, requires sufficient computation resources. Hence, it is crucial to build the whole system in a customisable and resource adaptive way. The former is achieved by providing a declarative agent specification language and mechanisms supporting the definition, addition, or deletion of reasoning agents (as well as some other proof search critical components and heuristics) even at run-time. For the latter, the agents in our framework can monitor their own performance, can adapt their capabilities, and can communicate to the rest of the system their corresponding resource information. This enables explicit (albeit currently still rudimentary) resource reasoning, facilitated by a specialised resource agent, and provides the basic structures for resource adaptive theorem proving. Further details on the resource and adaptation aspects are addressed in [BS99].

The rest of the paper is structured as follows: Section 2 presents the main components of the system architecture. Experiments with the architecture are sketched in Section 3. In Section 4 we provide an overview of the features of our approach and discuss related work. A conclusion/outlook is given in Section 5.

2 System Architecture

The architecture of our system is depicted in Fig. 1. The core of the system is written in Allegro Common Lisp and employs its multi-processing facilities. The choice of Common Lisp is due the fact that Omega, our base system, is implemented in this programming language; conceptually it can be implemented in any multi-processing framework.

Initial problems, partial proofs as well as completed proofs are represented in the **Proof Data Structure** [CS00] and the natural deduction infrastructure provided by the core system, Omega [BCF$^+$97].

Our approach builds on the **Reactive Suggestion Mechanism** Oants [BS01] as a reactive, resource adaptive basis layer of our framework. Triggered by changes in the proof data structure this mechanism dynamically computes applicable commands with their particular parameter instantiations and calls external reasoners into the current proof state. An important aspect is that all agent computations in this mechanism are de-sequentialised and distributed. The idea of this reactive layer is to receive results of inexpensive computations (e.g., the applicability of natural deduction rules) quickly while external reasoners search for their respective proof steps within the limits of their available resources, until a suggested command is selected and executed. A special resource agent receives performance data from the agents, which monitor their own performance, in order to adjust the system at run time. Heuristic criteria are used to dynamically filter and sort the list of generated suggestions. They are then passed to the selector and/or the user. We give here some sensible heuristic criteria. Does a suggestion close a subgoal? Is a subgoal reduced to an essentially simpler context (e.g., reduction of higher-order problems to first-order or propositional logic)? Does a suggestion represent a big step in the search tree (proof tactics/methods) or a small step (base calculus rules)? Is the suggestion goal directed? How many new subgoals are introduced?

Agents as well as heuristic criteria can be added/deleted/modified at run time. Due to lack of space OANTS cannot be described here in detail; for this we refer the reader to [BS01].

OANTS provides agents that do computations on the basic natural deduction calculus. It also provides agents that invoke additional proof tactics/methods and external reasoning systems. The external reasoning systems are called by the agent-shells indirectly via the MATHWEB system. That is, the agents themselves are realised as concurrent Lisp processes in the core system. These processes activate themselves and make calls to MATHWEB services when their applicability criteria are fulfilled (this contrasts calls by human users to external systems in interactive proof environments).

We extended the approach from [BS01] in the context of our work to integrate partial proofs as results from the external reasoning systems into the overall proof as well as to store different alternative subproofs simultaneously. Moreover, we extended OMEGA's graphical user interface LOUI to be able to display different subproofs of external reasoners as choices for the user.

The **Mathweb system** realises calls to external reasoners which may be distributed over the internet. In our most recent experiments we extensively tested the new ONE-MATHWEB system which is based on a multi-broker architecture. Each broker has knowledge about its directly accessible reasoning systems, and also about urls to other ONE-MATHWEB brokers on the internet. For example, in our experiments the reasoning agents gained access to the computer algebra system MAPLE running in Saarbrücken. For this we simply had to inform the Birmingham MATHWEB broker (which for license reasons cannot offer a MAPLE service locally) about the existence and url of the Saarbrücken broker. The Saarbrücken broker then connects the Birmingham broker (which receives and answers to the requests of the reasoning agents) with the MAPLE service. Currently our system links up with the computer algebra systems MAPLE and GAP running in Saarbrücken, and locally with the higher-order theorem provers LEO and TPS, the first-order theorem prover OTTER (employed also as our propositional logic specialist), and SATCHMO (employed as a model generator). MATHWEB is described in detail in [FHJ+99].

Once the reactive suggestion mechanism dynamically updates and heuristically sorts the list of suggestions, which are commands together with their particular parameter instantiations, it passes the list on to the **selector**. Its main task is to automatically execute the heuristically preferred command, and hence, initiate an update of the proof data structure. Furthermore, the selector stores the non-optimal, alternative command suggestions in a special store. The information in this store is used when backtracking to a previous state in the proof data structure becomes necessary. Instead of a complete initialisation the reactive suggestion mechanism is then simply initialised with the already computed backtracking information for the current proof context. Backtracking is caused when the reactive layer produces no suggestions or when a user defined maximal depth[3] in the proof data structure is reached.

The **backtrack store** maintains backtracking information for the proof data structure. This information includes representations of the suggestion computations that have been previously computed but not executed. Additionally the store maintains the results of external system calls modulo their translation in the core natural deduction calculus. That is, the immediate translation of external system results is also done by the reactive suggestion layer, and the results of these computations are memorised for backtracking purposes as well. If the system or the user selects to apply the result of an external system, the proof data structure is updated with the translated proof object. Future work will include investigating whether the backtrack store should be merged with the proof data structure. The idea is that each single node in a proof directly maintains its backtracking alternatives instead of using an indirect maintenance via the backtracking store.

The tasks of the **user interface** in our framework are:

1. To visualise the current proof data structure and to ease interactive proof construction. For this purpose we employ OMEGA's graphical user interface LOUI [SHB+99].
2. To dynamically present to the user the set of suggestions, which pop up from the reactive layer to the user, and to provide support for analysing or executing them. This is realised by structured and dynamically updated pop-up windows in LOUI.
3. To provide graphical support for analysing the results of external systems, that is, to display their results after translation/representation in the proof data structure. We achieve this by extending LOUI so that it can switch between the global proof data structure and locally offered results by external systems.
4. To support the user in interacting with the automated mechanism and in customising agent societies at run-time.

From an abstract perspective, our system realises proof construction by going through a cycle which consists of assessing the state of the proof search process,

[3] Iterative deepening proof search wrt. to the maximal depth is conceptually feasible but not realised yet.

evaluating the progress, choosing a promising direction for further search and redistributing the available resources accordingly. If the current search direction becomes increasingly less promising then backtracking to previous points in the search space is possible. Only successful or promising proof attempts are allowed to continue searching for a proof. This process is repeated until a proof is found, or some other terminating condition is reached.

3 Experiments

In this section we report on experiments we conducted with our system to demonstrate the usefulness of a flexible combination of different specialised reasoning systems. Among others we examined different problem classes:

1. Set examples which demonstrate a cooperation between higher-order and first-order theorem provers. For instance, prove:
   $\forall x, y, z_\bullet (x = y \cup z) \Leftrightarrow (y \subseteq x \land z \subseteq x \land \forall v_\bullet (y \subseteq v \land z \subseteq v) \Rightarrow (x \subseteq v)$
2. Set equations whose validity/invalidity is decided in an interplay of a natural deduction calculus with a propositional logic theorem prover and model generator. For instance, prove or refute:
   (a) $\forall x, y, z_\bullet (x \cup y) \cap z = (x \cap z) \cup (y \cap z)$
   (b) $\forall x, y, z_\bullet (x \cup y) \cap z = (x \cup z) \cap (y \cup z)$
3. Concrete examples about sets over naturals where a cooperation with a computer algebra system is required. For instance (*gcd* and *lcm* stand for the 'greatest common divisor' and the 'least common multiple'):
   $\{x | x > gcd(10,8) \land x < lcm(10,8)\} = \{x | x < 40\} \cap \{x | x > 2\}$
   This set is represented by the lambda expression
   $(\lambda x_\bullet x > gcd(10,8) \land x < lcm(10,8)) = (\lambda x_\bullet x < 40) \cap (\lambda x_\bullet x > 2)$
4. Examples from group theory and algebra for which a goal directed natural deduction proof search is employed in cooperation with higher-order and first-order specialists to prove equivalence and uniqueness statements. These are for instance of the form
   $[\exists \circ_\bullet Group(G, \circ)] \Leftrightarrow [\exists \star_\bullet Monoid(M, \star) \land Inverses(M, \star, Unit(M, \star))]$
   Here *Group* and *Monoid* refers to a definition of a group and a monoid, respectively. $Inverses(M, \star, Unit(M, \star))$ is a predicate stating that every element of M has an inverse element with respect to the operation $\star$ and the identity $Unit(M, \star)$. $Unit(M, \star)$ itself is a way to refer to that unique element of M that has the identity property.

We will sketch in the following how the problem classes are tackled in our system in general and how the proofs of the concrete examples work in particular.

3.1 Set examples

The first type of examples is motivated by the shortcomings of existing higher-order theorem provers in first-order reasoning. For our experiments we used the LEO system [BK98], a higher-order resolution prover, which specialises in extensionality reasoning and is particularly successful in reasoning about sets.

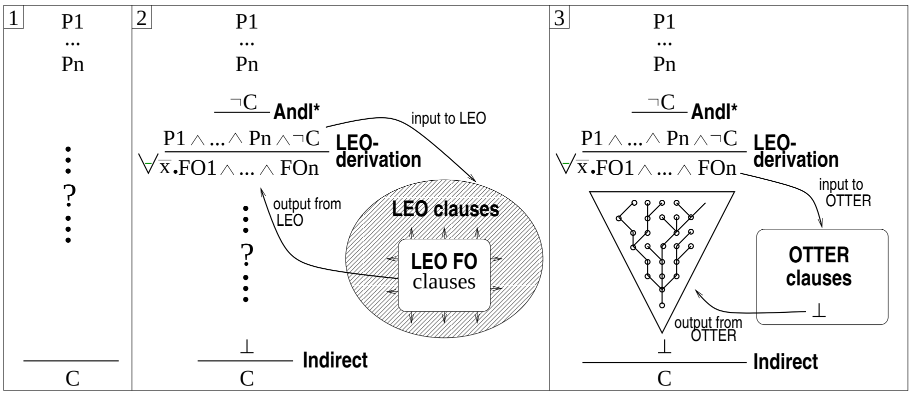

Fig. 2. Agent based cooperation between LEO and OTTER.

Initialised with a set problem LEO tries to apply extensionality reasoning in a goal directed way. On an initial set of higher-order clauses, it often quickly derives a corresponding set of essentially first-order clauses.[4] Depending on the number of generated first-order and other higher-order clauses LEO may get stuck in its reasoning process, although the subset of first-order clauses could be easily refuted by a first-order specialist.

For our examples the cooperation between LEO and the first-order specialist OTTER works as depicted in Fig. 2. The initial problem representation in the proof data structure is described in Part 1 of Fig. 2. The initialisation triggers the agents of the reactive suggestion layer which start their computations in order to produce suggestions for the next proof step.

The agent working for LEO first checks if there is any information from the resource agent that indicates that LEO should stay passive. If not, it checks whether the goal C is suitable for LEO by testing if it is a higher-order problem. In case the problem is higher-order the agent passes the initial problem consisting of the goal C and the assumptions $P_1, \ldots, P_n$ to LEO. While working on the input problem (as indicated by the shaded oval in Part 2 of Fig. 2) LEO derives (among others) various essentially first-order clauses (e.g., $\mathrm{FO}_1 \ldots \mathrm{FO}_n$). For the particular type of cooperation described here, it is important that after a while this subset becomes large enough to be independently refutable. If after consuming all the resources made available by the reactive suggestion layer LEO still fails to deliver a completed proof, it then offers a partial proof consisting of a subset of first-order and essentially first-order clauses (after translation into prenex normal form, e.g., $\forall \overline{x}.\, \mathrm{FO}'_1 \wedge \ldots \wedge \mathrm{FO}'_n$, where the FO'_i are disjunctions of the literals of FO_i and $\overline{x}$ stands for the sequence of all free variables in the scope). In case LEO's suggestion wins over the suggestions computed by other agents, its partial result is represented in the proof data structure and the reactive suggestion mechanism is immediately triggered again to compute a suggestion for the next possible proof step. Since LEO's partial result is now the new subgoal of the partial proof, first-order agents, like the one working for OTTER, can pick it up and ask OTTER to prove it (see Part 3 of Fig. 2). If OTTER signals a

[4] By essentially first-order we mean a clause set that can be tackled by first-order methods. It may still contain higher-order variables, though.

successful proof attempt before consuming all its given resources, its resolution proof is passed to the natural deduction translation module TRAMP [Mei00], which transforms it into a proper natural deduction proof on an assertion level.[5]

We experimented with 121 simple examples, that is, examples that can be automatically proved by LEO alone. The results showed that the command execution interval chosen by the selector is crucial, since it determines the computation time *ct* made available to the external systems.

- If *ct* is sufficiently high, then the problem is automatically proved by LEO (in case of simple examples that can be solved by LEO alone).
- If *ct* is not sufficient for LEO to come up with a proof, but still enough to produce a refutable subset of essentially first-order clauses, then a cooperative proof is constructed as described above.
- If *ct* is not sufficient to even guarantee a subset of refutable essentially first-order clauses, then the problem is tackled purely on natural deduction level, however not necessarily successfully.

We also solved several examples which cannot be solved with LEO alone. One of them is the concrete example given above, which, to our knowledge, cannot be easily solved by a single automated theorem prover. In our experiments, LEO alone ran out of memory for the above problem formulation, and OTTER alone could not find a proof after running 24 hours in auto mode on a first-order formulation of the problem. Of course, an appropriate reformulation of the problem can make it simple for systems like OTTERto prove this new formulation.

3.2 Set equations

The second type of set examples illustrates a cooperation between automated natural deduction agents, a propositional prover and a model generator. The proofs follow a well-known set theoretic proof principle: they are constructed first by application of simple natural deduction agents that reduce the set equations by applying set extensionality and definition expansion to a propositional logic statement. This statement is then picked up by an agent working for a propositional logic prover (here we again use OTTER encapsulated in another agent shell with a slightly modified applicability check and a different representation translation approach) and a counter-example agent which employs SATCHMO. The logic statement is then either proved or refuted. Thus, valid and invalid statements are tackled analogously in all but the last step.

In case (2a) of our concrete examples several $\forall_I$ (universal quantification introduction in backward reasoning) applications introduce $(a \cup b) \cap c = (a \cap c) \cup (b \cap c)$ as new open subgoal. Set extensionality gives us $\forall u. u \in (a \cup b) \cap c \Leftrightarrow u \in ((a \cap c) \cup (b \cap c))$. A further $\forall_I$ application and subsequent definition expansions (where $a \cup b := \lambda z. (z \in a) \vee (z \in b)$, $a \cap b := \lambda z. (z \in a) \wedge (z \in b)$, and $u \in a := a(u)$) reduce this goal finally to $(a(d) \vee b(d)) \wedge c(d) = (a(d) \wedge c(d)) \vee (b(d) \wedge c(d))$

[5] While TRAMP already supports the transformation of various machine oriented first-order proof formats, further work will include its extension to higher-order logic, such that also the proof step justified in Fig. 2 with 'LEO-derivation' can be properly expanded into a verifiable natural deduction proof.

which contains no variables and which is a trivial task for any propositional logic prover. In case (2b) we analogously derive $(a(d) \vee b(d)) \wedge c(d) = (a(d) \vee c(d)) \wedge (b(d) \vee c(d))$, but now a model generator agents presents the counter-model $a(d), b(d), \neg c(d)$. That is, it points to the set of all d such that $d \in a$, $d \in b$, but $d \notin c$. Hence, the model generator comes up with a counter-example to the expression in (2b).

We have experimented with an automatically and systematically generated testbed consisting of possible set equations involving $\cap, \cup$, *set-minus* operations up to nesting depth of 5 in maximally 5 variables. We classified 10000 examples with our system discovering 988 correct and 9012 false statements. Naturally, the correct statements are probably also solvable with the cooperation of LEO and OTTER.

3.3 Examples with computer algebra

The next type of examples has cross-domain character and requires a combination of domain specific systems. In order to tackle them we added a simplification agent which links the computer algebra system MAPLE to our core system. As an application condition this agent checks whether the current subgoal contains certain simplifiable expressions. If so, then it simplifies the subgoal by sending the simplifiable subterms (e.g., $x > gcd(10, 8)$) via MATHWEB to MAPLE and replaces them with the corresponding simplified terms (e.g., $x > 40$). Hence, the new subgoal suggested by the simplification agent is: $(\lambda x. x > 2 \wedge x < 40) = (\lambda x. x < 40) \cap (\lambda x. x > 2)$. Since no other agent comes up with a better alternative, this suggestion is immediately selected and executed. Subsequently, the LEO agent successfully attacks the new goal after expanding the definition of $\cap$. We have successfully solved 50 problems of the given type and intend to generate a large testbed next.

3.4 Group theory and algebra examples

The group theory and algebra examples we examined are rather easy from a mathematical viewpoint, however, can become non-trivial when painstakingly formalised. An example are proofs in which particular elements of one mathematical structure have to be identified by their properties and transferred to their appropriate counterparts in an enriched structure. The equivalence statement given above in (4) where the unit element of the monoid has to be identified with the appropriate element of the group are in this category. In higher-order this can be done most elegantly using the description operator ι (cf. [And72] for description in higher-order logics) by assigning to the element in the group the unique element in the monoid that has exactly the same properties. In the context of our examples we employed description to encode concepts like the (unique) unit element of a group by a single term that locally embodies the particular properties of the encoded concept itself. If properties of the unit element are required in a proof then the description operator has to be unfolded (by applying a tactic in the system) and a uniqueness subproof has to carried out.

However, an open problem is to avoid unnecessary unfoldings of the description operator as this may overwhelm the proof context with unneeded information.

The idea of the proofs is to divide the problems into smaller chunks that can be solved by automated theorems provers and if necessary to deal with formulae involving description. The ND search procedure implemented in OANTS has the task to successively simplify the given formulae by expanding definitions and applying ND inferences. After each proof step the provers try to solve the introduced subproblems. If they all fail within the given time bound the system proceeds with the alternative ND inferences. The quantifier rules introduce Skolem variables and functions when eliminating quantifications. These are constrained either by the application of a generalised *Weaken* rule, using higher-order unification, or by the successful solution of subproblems by one of the provers, which gives us the necessary instantiation. Problems involving higher-order variables (for which real higher-order instantiations are required) can generally not be solved (in this representation) by first-order provers. However, once an appropriate instantiation for the variables has been computed a first-order prover can be applied to solve the remaining subproblems. Substitutions for introduced Skolem variables are added only as constraints to the proof, which can be backtracked if necessary.

When a point is reached during the proof where neither applicable rules nor solutions from the provers are available, but the description operator still occurs in the considered problem, two theorems are applied to eliminate description. This results in generally very large formulae, which can then again be tackled with the ND rules and the theorem provers.

In our experiments with algebra problems we have successfully solved 20 examples of the described type.

Our experiments show that the cooperation between different kinds of reasoning systems can fruitfully combine their different strengths and even out their respective weaknesses. In particular, we were able to successfully employ LEO's extensionality reasoning with OTTER's strength in refuting large sets of first-order clauses. Likewise, our distributed architecture enables us to exploit the computational strength of MAPLE in our examples remotely over the internet. As particularly demonstrated by the last example class the strengths of external systems can be sensibly combined with domain specific tactics and methods, and natural deduction proof search.

Note that our approach does not only allow the combination of heterogeneous systems to prove a problem, but it also enables the use of systems with opposing goals in the same framework. In our examples the theorem prover and the model generator work in parallel to decide the validity of the current (propositional) goal.

Although many of our examples deal with problems in set theory they already show that the cooperation of differently specialised reasoning systems enhances the strengths of automated reasoning. The results also encourage the application of our system to other areas in mathematics in the future. However, there is a bottleneck for obtaining large proofs, namely the translation between the different systems involved, in particular, in the presence of large clause sets.

4 Discussion

Our work is related to blackboard and multi-agent systems in general, and to approaches to distributed proof search and agent-oriented theorem proving in particular. Consequently, the list of related work is rather long and we can mention only some of it. We first summarise different facets of our approach which we then use to clarify the differences to other approaches and to motivate our system design objectives. Our system:

(1) aims to provide a cognitively adequate assistant tool to interactively and/or automatically develop mathematical proofs;
(2) supports interaction and automation simultaneously and integrates reactive and deliberative proof search;
(3) maintains a global proof object in an expressive higher-order language in which results of external systems can be represented;
(4) employs tools as Loui [SHB+99] or P.rex [Fie01] to visualise and verbalise proofs, i.e., communicate them on a human oriented representation layer;
(5) couples heterogeneous external systems with domain specific tactics and methods and natural deduction proof search; i.e., our notion of heterogeneity comprises machine oriented theorem proving as well as tactical theorem proving/proof planning, model generation, and symbolic computation;
(6) reuses existing reasoning systems and distributes them via Mathweb (In order to add a new system provided by Mathweb the user has to: a) provide an abstract inference step/command modelling a call to the external reasoner, b) define the parameter agents working for it, and c) (optional) adapt the heuristic criteria employed by the system to rank suggestions. Due to the declarative agent and heuristics specification framework these steps can be performed at run time.);
(7) supports competition (e.g., proof versus countermodel search) as well as cooperation (e.g., exchange of partial results);
(8) follows a skeptical approach and generally assumes that results of external reasoning system are translated in the central proof object (by employing transformation tools such as Tramp [Mei00]) where they can be proof-checked;
(9) employs resource management techniques for guidance;
(10) supports user adaptation by enabling users to specify/modify their own configurations of reasoning agents at run-time, and to add new domain specific tactics and methods when examining new mathematical problem domains;
(11) stores interesting suboptimal suggestions in a backtracking stack and supports backtracking to previously dismissed search directions;
(12) supports parallelisation of reasoning processes on different layers: term-level parallelisation is achieved by various parameter agents of the commands/ abstract inferences, inference-level parallelisation is supported by the ability to define new powerful abstract inferences which replace several low level inferences by a single step (a feature inherited from the integrated tactical theorem proving paradigm), and proof-search-level parallelisation is supported by the competing reasoning systems.

Taken individually none of the above ideas is completely new and for each of these aspects there exists related work in the literature. However, it is the combination of the above ideas that makes our project unique and ambitious.

A taxonomy of parallel and distributed (first-order) theorem proving systems is given in [Bon01]. As stated in (12), our approach addresses all three classification criteria introduced there: parallelisation on term, inference, and search level. However, full or-parallelisation is not addressed in our approach yet. This will be future work.

A very related system is the TECHS approach [DF99] which realises a cooperation between a set of heterogeneous first-order theorem provers. Partial results in this approach are exchanged between the different theorem provers in form of clauses, and different referees filter the communication at the sender and receiver side. This system clearly demonstrates that the capabilities of the joint system are bigger than those of the individual systems. TECHS' notion of heterogeneous systems, cf. (5) above, however, is restricted to a first-order context only. Also symbolic computation is not addressed. TECHS [DF99] and its even less heterogeneous predecessors TEAMWORK [DK96] and DISCOUNT [ADF95] are much more machine oriented and less ambitious in the sense of aspects (1)–(4). However, the degree of exchanged information (single clauses) in all these approaches is higher than in our centralised approach. Unlike in the above mentioned systems, our interest in cooperation, however, is in the first place not at clause level, but on subproblem level, where the subproblem structure is maintained by the central natural deduction proof object. Future work includes investigating to what extend our approach can be decentralised, for instance, in the sense of TECHS, while preserving a central global proof object.

In contrast to many other approaches we are interested in a fully skeptical approach, cf. (8) and the results of some external reasoners (e.g., for OTTER TPS, and partially for computer algebra systems) can already be expanded and proof checked by translation in the core natural deduction calculus. However, for some external systems (e.g., LEO) the respective transformation tools still have to be provided. While they are missing, the results of these system, modelled as abstract inferences in natural deduction style, cannot be expanded.

Interaction and automation are addressed by the combination of ILF & TECHS [DD98]. With respect to aspects (6)–(12), especially (10), there are various essential differences in our approach. The design objectives of our system are strongly influenced by the idea to maintain a central proof object which is manipulated by the cooperating and competing reasoning agents, and mirrors the proof progress. This central natural deduction proof object especially eases user interaction on a human oriented layer, cf. (3) and (4), and supports skepticism as described above. In some sense, external systems are modelled as new proof tactics. Extending the background calculus and communication between them is currently only supported via the system of blackboards associated with the current focus of the central proof object. This relieves us from addressing logical issues in the combination of reasoning systems at the proof search layer. They are subordinated and only come into play when establishing the soundness of contributions of external reasoners by expanding their results on natural deduction layer. A centralised approach has advantages in the sense that it keeps the integration of n heterogeneous systems, with probably different logical contexts, simple and it only requires n different proof (or result) transformation tools to natural deduction arguments. In particular the overall proof construction is controlled purely at the natural deduction layer.

However, experiments indicated that aside from these advantages, the bottleneck of the system currently is the inefficiency in the cooperation of some external systems, especially of homogeneous systems specialised in resolution style proving which cannot directly communicate with each other. Future work therefore includes investigating whether the approach can be further decentralised without giving up much of the simplicity and transparency of the current centralised approach.

With the centralisation idea, we adopted a blackboard architecture and our reasoning agents are knowledge sources of it. In the terminology of [Wei99] our reasoning agents can be classified as reactive, autonomous, pro-active, cooperative and competitive, resource adapted, and distributed entities. They, for instance, still lack fully deliberative planning layers and social abilities such as means of explicit negotiation (e.g., agent societies are defined by the user in OANTS and, as yet, not formed dynamically at run-time [BS01]). In this sense, they are more closely related to the HASP [NFAR82] or POLIGON [Ric89] knowledge sources than to advanced layered agent architectures like INTERRAP [Mül97]. However, in future developments a more decentralised proof search will make it necessary to extend the agenthood aspects in order to enable agents to dynamically form clusters for cooperation and to negotiate about efficient communication languages.

5 Conclusion

In this paper we presented our agent-based reasoning system. Our framework is based on concurrent suggestion agents working for natural deduction rules, tactics, methods, and specialised external reasoning systems. The suggestions by the agents are evaluated after they are translated into a uniform data representation, and the most promising direction is chosen for execution. The alternatives are stored for backtracking. The system supports customisation and resource adapted and adaptive proof search behaviour.

The main motivation is to develop a powerful and extendible system for tackling, for instance, cross domain examples, which require a combination of reasoning techniques with strengths in individual domains. However, our motivation is not to outperform specialised systems in their particular niches. The agent paradigm was chosen to enable a more flexible integration approach, and to overcome some of the limitations of hardwired integrations (for instance, the brittleness of traditional proof planning where external systems are typically called within the body of proof methods and typically do not cooperate very flexibly).

A cognitive motivation for a flexible integration framework presented in this paper is given from the perspective of mathematics and engineering. Depending on the specific nature of a challenging problem, different specialists may have to cooperate and bring in their expertise to fruitfully tackle a problem. Even a single mathematician possesses a large repertoire of often very specialised reasoning and problem solving techniques. But instead of applying them in a fixed structure, a mathematician uses own experience and intuition to flexibly combine them in an appropriate way.

The experience of the project points to different lines of future research. Firstly, the agent approach offers an interesting framework for combining automated and interactive theorem proving on a user-oriented representation level (and in this sense it differs a lot from the mainly machine-oriented related work). This approach can be further improved by developing a more distributed view of proof construction and a dynamic configuration of cooperating agents. Secondly, in order to concurrently follow different lines of search (or-parallelism), a more sophisticated resource handling should be added to the system. Thirdly, the communication overhead for obtaining large proofs is the main performance bottleneck. More efficient communication facilities between the different systems involved have to be developed. Contrasting the idea of having filters as suggested in [DF99] we also want to investigate whether in our context (expressive higher-order language) abstraction techniques can be employed to compress the exchanged information (humans do not exchange clauses) during the construction of proofs.

Further future work includes improving several technical aspects of the current Omega environment and the prototype implementation of our system that have been uncovered during our experiments. We would also like to test the system in a real multi-processor environment, where even the agent-shells for external reasoners can be physically distributed – currently, the agent-shells, which are local, make indirect calls (via Mathweb) to the external systems. Furthermore, we will integrate additional systems and provide further representation translation packages. The agents' self-monitoring and self-evaluation criteria, and the system's resource adjustment capabilities will be improved in the future. We would also like to employ counter-example agents as indicators for early backtracking. Finally, we need to examine whether our system could benefit from a dynamic agent grouping approach as described in [FW95], or from an integration of proof critics as discussed in [IB95].

Acknowledgements

For advice, support, and encouragement we thank Alan Bundy, Michael Fisher, Malte Hübner, Andreas Franke, and Jürgen Zimmer. This work was supported by EPSRC grants GR/M99644 and GR/M22031 (Birmingham), and the SFB 378 (Saarbrücken).

References

[ADF95] J. Avenhaus, J. Denzinger, and M. Fuchs. DISCOUNT: A system for distributed equational deduction. In *Proc. of RTA-95, LNCS* 914. Springer, 1995.

[And72] P. Andrews. General models, descriptions and choice in type theory. *Journal of Symbolic Logic*, 37(2):385–394, 1972.

[AZ01] A. Armando and D. Zini. Interfacing computer algebra and deduction systems via the logic broker architecture. In [CAL01].

[BCF$^+$97] C. Benzmüller et al. ΩMEGA: Towards a mathematical assistant. In *Proc. of CADE–14*, *LNAI* 1249. Springer, 1997.

[BJKS99] C. Benzmüller, M. Jamnik, M. Kerber, and V. Sorge. Agent Based Mathematical Reasoning. In *CALCULEMUS-99, Systems for Integrated Computation and Deduction*, volume 23(3) of *ENTCS*. Elsevier, 1999.

[BK98] C. Benzmüller and M. Kohlhase. LEO – a higher-order theorem prover. In *Proc. of CADE–15*, *LNAI* 1421. Springer, 1998.

[Bon01] M. Bonacina. A taxonomy of parallel strategies for deduction. *Annals of Mathematics and Artificial Intelligence*, in press, 2001.

[BS99] C. Benzmüller and V. Sorge. Critical Agents Supporting Interactive Theorem Proving. *Proc. of EPIA-99*, LNAI 1695, Springer, 1999.

[BS01] C. Benzmüller and V. Sorge. OANTS – An open approach at combining Interactive and Automated Theorem Proving. In [CAL01].

[CAL01] *CALCULEMUS-2000, Systems for Integrated Computation and Deduction.* AK Peters, 2001.

[CS00] L. Cheikhrouhou and V. Sorge. $\mathcal{PDS}$ — A Three-Dimensional Data Structure for Proof Plans. In *Proc. of ACIDCA'2000*, 2000.

[DD98] I. Dahn and J. Denzinger. Cooperating theorem provers. In *Automated Deduction—A Basis for Applications*, volume II. Kluwer, 1998.

[DF99] J. Denzinger and D. Fuchs. Cooperation of Heterogeneous Provers. In *Proc. of IJCAI-99*, 1999.

[DK96] J. Denzinger and M. Kronenburg. Planning for distributed theorem proving: The teamwork approach. In *Proc. of KI-96*, *LNAI* 1137. Springer, 1996.

[FHJ$^+$99] A. Franke, S. Hess, Ch. Jung, M. Kohlhase, and V. Sorge. Agent-Oriented Integration of Distributed Mathematical Services. *Journal of Universal Computer Science*, 5(3):156–187, 1999.

[FI98] M. Fisher and A. Ireland. Multi-agent proof-planning. CADE-15 Workshop "Using AI methods in Deduction", 1998.

[Fie01] A. Fiedler. *P.rex*: An interactive proof explainer. In R. Goré, A. Leitsch, and T. Nipkow (eds), *Automated Reasoning – Proceedings of the First International Joint Conference, IJCAR*, *LNAI* 2083. Springer, 2001.

[Fis97] M. Fisher. An Open Approach to Concurrent Theorem Proving. In *Parallel Processing for Artificial Intelligence*, volume 3. Elsevier, 1997.

[FW95] M. Fisher and M. Wooldridge. A Logical Approach to the Representation of Societies of Agents. In *Artificial Societies.* UCL Press, 1995.

[IB95] A. Ireland and A. Bundy. Productive use of failure in inductive proof. *Special Issue of the Journal of Automated Reasoning*, 16(1–2):79–111, 1995.

[Mei00] A. Meier. TRAMP - transformation of machine-found proofs into ND-proofs at the assertion level. In *Proc. of CADE–17*, *LNAI* 1831. Springer, 2000.

[Mül97] J. Müller. A Cooperation Model for Autonomous Agents. In *Proc. of the ECAI'96 Workshop Intelligent Agents III*, *LNAI* 1193. Springer, 1997.

[NFAR82] H. Nii, E. Feigenbaum, J. Anton, and A. Rockmore. Signal-to-symbol transformation: HASP/SIAP case study. *AI Magazine*, 3(2):23–35, 1982.

[Ric89] J. Rice. The ELINT Application on Poligon: The Architecture and Performance of a Concurrent Blackboard System. In *Proc. of IJCAI-89.* Morgan Kaufmann, 1989.

[SHB$^+$99] J. Siekmann et al. LOUI: Lovely OMEGA user interface. *Formal Aspects of Computing*, 11(3):326–342, 1999.

[Wei99] G. Weiss, editor. *Multiagent systems: a modern approach to distributed artificial intelligence.* MIT Press, 1999.

[Wol98] A. Wolf. P-SETHEO: Strategy Parallelism in Automated Theorem Proving. In *Proc. of TABLEAUX-98*, *LNAI* 1397. Springer, 1998.