A task-oriented agent-based mechanism for mathematical assistant systems[*]

Quoc Bao Vo
FR Informatik, Universität des Saarlandes, 66041 Saarbrücken, Germany
bao@ags.uni-sb.de

Abstract

We present an agent-based mechanism that acts as a mediator module between theorem proving systems and mathematical knowledge bases containing information that is necessary for the constructions of proofs. Unlike the more popular user-oriented mediators who work as information agents to provide the so-called value-added services to the collected data before presenting it to users or user applications, our (multi-)agents are more task-oriented. That is, our agents work in tandem with the user or user application on the tasks the user is trying to solve. This approach is particularly suitable to mathematical knowledge retrieval in theorem proving as (i) checking for applicable axioms/definitions/theorems from the knowledge base can be done independently from the proof search process concurrently carried out by the prover, and $(\ i)$ the prover and the mediator operate on two different search spaces and the search outcome brought about by the mediator can be of great benefit to the prover, e.g. to avoid the prover from exploring many unnecessary or irrelevant proof steps, to keep the prover's search space more manageable and the constructed proof more comprehensible.

1 Introduction

Information is crucial in the process of planning and decision making. However, information and information systems are both increasing in size; there is a tendency towards information overload. This is particularly the case when one wishes to obtain the relevant information from the Internet having only a few descriptors or keywords at hand. Thus, research in software agents, in particular information mediators, has placed much emphasis on the development of mechanisms that allow more useful information to be produced from the raw input data. These include information selection, filtration, and integration. The more prominent research groups working in this area include the collective work on Large-scale Interoperation, Mediation, and Composition (LIC) led by Gio Wiederhold at Stanford University [24], the University of Maryland Information Mediation Project [38], the Information Agents Group at the University of Southern California, Information Sciences Institute [19], the Software Agents Group of the MIT Media Laboratory [35], etc. to mention a few. In general, the mediators provide intermediate services, linking data resources and users' application programs. Their function is to provide integrated information obtained from diverse and heterogeneous data sources. More specifically, the mediators perform (see [43]):

(i) Accessing and retrieving relevant data from multiple heterogeneous resources;

(ii) Abstracting and transforming retrieved data into a common representation and semantics;

[*]A preliminary version of this paper appears in [39].

(iii) Integrating the homogenized data according to matching descriptors and keys;

(iv) Reducing the integrated data to increase the relevance and information density in the result to be transmitted.

Wiederhold coins the term *value-added services* to refer to the above functionalities undertaken by mediators that basically “convert data to information”.

However, as most of these projects aim at information agents whose goals are more focused on assisting users or clients in handling information, they are more *user-oriented.* That is, they are customized to the user by learning the user’s interests/preferences/habits/etc., or getting *personalized.* That way the agent becomes more useful to a particular user. In this paper we argue that there is yet another way in which information agents can act as mediators between information resources and user application, namely *task-oriented.* Task-oriented information agents necessarily have access to the task(s) the user is trying to accomplish and know what information is required to solve such tasks and, thereby, extract the relevant information from the data sources and possibly format/integrate the obtained data to make them readily usable to user or user application. We observe that this aspect of information agents has been largely neglected thus far. We motivate our approach by an application in mathematical assistant systems.

Mathematical assistant systems (MathAS, see e.g. [34, 33] and the references therein) provide users with integrated environments in which various mathematics-related tasks including learning, teaching, referencing, proving theorems, carrying out complex computations, etc. can be accomplished. The long term visions of MathAS is to provide an environment with two ends:

> **High end:** At this end, the system interacts with the human users such as mathematicians or students (who learn mathematics) and also the Digital Libraries that provide the data sources for activities undertaken by the system.
>
> **Low end:** At this end, the system is connected to other systems that provide the services to be used to accomplish its tasks. Examples of such service-providers include: computer algebra systems, traditional theorem provers built by the automated deduction community, model checking systems, databases containing mathematical materials used by other systems, etc.

At the high end, MathAS typically have to deal with various formats of data and representations which are in general informal, and the approaches to problem solving are varied at different levels of abstraction. At the other end, MathAS connect to systems such as theorem provers, computer algebra systems, databases, etc. whose data formats are formal and well-specified. Moreover, at this end the connected systems generally follow pre-defined approaches to problem solving at fixed levels of abstraction.

In order to bridge the gap between the two ends, it’s important that MathAS be able to carry out the reasoning or computation at different levels of abstraction and take advantage of the facilities provided by existing systems to provide the value-added services to the users or user applications. This paper is concerned mainly with the problem of proving mathematical theorems at levels of abstraction suitable to be communicated directly to human users. As such, tools that enable proof search and stepwise construction of proofs must be supported by these systems. Whereas traditional theorem provers work at the logical level, human mathematicians prove theorems at the more abstract levels. To overcome this problem, new approaches have been introduced in which most low level logical operations are abstracted away and mainly domain specific mathematical knowledge is used to guide the construction of the proofs. The area is known as proof planning and such domain specific mathematical knowledge comes in as the strategies or proof methods made available to the proof planner (cf. [11] and [18, 31, 30] and the references therein).

Due to Huang [17] the notion of *assertion* comprises declarative mathematical knowledge such as definitions, theorems, and axioms.[1] This knowledge can be either stored in a well-defined format in mathematical databases KBs or distributed over a network of Digital Libraries. In this paper, we propose a distributed mediator module between KBs and a theorem proving system TP which is independent of the particular proof representation format of TP. We will also outline a sketchy architecture for information mediators that provide access to the less restrictive data sources, e.g. Digital Libraries, where data can be stored in arbitrary formats.

The development of our ideas revolves around the mathematical assistant system ΩMEGA [33] and the current initiative in this project to rebuild the system on top of the proof representation framework in [4]. We furthermore employ ΩMEGA's agent-based search mechanism Ω-ANTS [9] for a distributed modeling of our framework.We will also motivate an application of the approach in a project aiming at a tutorial dialogue system for mathematics.

2 MathAS and theorem proving

In contrast to proofs found in mathematical textbooks, proofs constructed by computer systems, i.e. theorem proving or proof planning systems, are composed of derivations from elementary logic, where the focus of attention is on syntactic manipulations rather than on the underlying semantical ideas. The problem seems to come from the lack of intermediate structures in machine-oriented proofs, e.g. resolution or sequent calculus or natural deduction calculus proofs, that allow atomic justifications at a higher level of abstraction. For instance, Huang [17] introduces the following three levels of justifications which can be found in mathematical proofs:

(i) *Logic level* justifications are simply verbalizations of the logical inference rules, such as the rule of Modus Ponens.

(ii) *Assertion level* justifications account for derivations in terms of the application of an axiom, a definition or a theorem. For instance, an extract from a textbook proof may read:

> "since e is a member of the set A, and A is a subset of B, *according to the definition of subset*, e is a member of B".

(iii) *Proof level* justifications are at a yet higher level. For instance, a proof can be suppressed by resorting to its similarity to a previous proof.

It is the assertion level that is of central interest in this paper because of its direct connection to mathematical databases. Now let's consider an assertion $\mathcal{A}$. There may be several ways in which this assertion can be used depending on the proof situation to which this assertion is introduced. For instance, let $\mathcal{A}$ be the assertion from the above example, i.e. *the definition of subset*:

$$et \quad \subseteq \quad x \;\; Eleme \;\; t\,(x \in \quad x \in \;\;))$$

This assertion allows us to derive: (1) $a \in V$ from $a \in U$ and $U \subseteq V$, (2) $U \not\subseteq V$ from $a \in U$ and $a \notin V$, (3) $\forall x : Eleme\;\; t \;\; x \in U \quad x \in V)$ from $U \subseteq V$, (4) etc.

[1]This notion of assertion which we will at times referred to as *knowledge-based assertions* is not to be confused with the same term that is used by mathematicians to refer to statements which are asserted to a proof. The latter which we call *proof-based assertions* in the rest of this paper consist of derivatives or constructs produced during the proof or some relevant axioms or assumptions introduced to the proof.

A theorem prover/proof planner, called a *prover* from now on,[2] that operates on the calculus level can only achieve such conclusions after a number of proof steps to eliminate the quantifiers and other connectives such as implication and conjunction. On the other hand, most human mathematicians would be satisfied having those conclusions derived in one step from the assertion. Furthermore, such a prover would have to search in a huge search space for every formula is necessarily decomposed according to the logical quantifiers/connectives it contains and it must account for all the resulting proof situations.

As has been discussed in the introductory section, knowledge-based proof planning constructs (or, plans for) proofs that are cognitively adequate for human users. Since (knowledge-based) proof planners operate on conceptual levels that are generally more abstract than the logic level, they require more advanced infrastructure and employ special data structures designated for such purposes. The next subsection briefly discuss these issues.

2.1 Formalization of the problem

We take as the starting point for our approach the proof development environment ΩMEGA [33] whose core consists of a proof planner together with a hierarchical plan data structure ($\mathcal{PDS}$). The proof format in ΩMEGA is based on the Natural Deduction (ND) calculus introduced by Gentzen [14]. A linearized version of ND proofs as introduced by Andrews [3] is employed. In this formalism which is implemented in the proof planner in ΩMEGA [18], an ND proof is a sequence of proof lines , each of them is of the form:

$$\texttt{Label}\ \ \Delta \vdash \textit{Derived-formula}\quad \texttt{Rule}\ \textit{premise-lines})$$

where `Rule` is a rule of inference in ND or a method, which justifies the derivation of the *Derived-formula* using the formulae in the *premise-lines*. `Rule` and *premise-lines* together are called the justification of a line. Δ is a finite set of formulae which are the hypotheses the derived formula depends on.

A problem of proof planning consists of a theorem (to be proved) and the assumptions to be used to prove the theorem. A proof planner operates on a set of methods to be used to construct a proof plan following a set of proving strategies.[3] The theorem and assumptions are expressed as deduction lines in a $\mathcal{PDS}$ where all the assumptions are marked as *closed* and the theorem is marked as *open*. The proof planner then uses the methods to come up with actions to update the $\mathcal{PDS}$. The aim of the proof planning process is to reach a closed $\mathcal{PDS}$, that is one without open lines.

Methods (which include the more restrictive notion of *tactics*) and strategies facilitate the most distinguished features of knowledge-based proof planning in comparison to traditional theorem provers. In a $\mathcal{PDS}$, several different levels of abstraction are maintained and kept consistent. As the proof planner evolves the proof objects at one level, corresponding proof objects at other levels are possibly created to keep them accordingly updated. (Note that it's not always possible to obtain corresponding proof objects at other abstraction levels, except those at the calculus level.) The calculus level, which is also the lowest level of abstraction maintained in the $\mathcal{PDS}$, makes sure that the produced proof is a correct one irrespective of the level of abstraction it is planned. Figure 1 illustrates the architecture of a hierarchical proof planner with a 3-dimensional $\mathcal{PDS}$ being embedded in a 2-dimensional proof planning representation. The third dimension of representation in a proof planner (which is neglected from figure 1 for clarity) contains the strategies and further meta-knowledge such as control rules (see [27]) to allow more comprehensive and sophisticated problem solving approaches.

[2]Remark that we do not rule out the possibility of having the user as one of the (high level) strategies used by proof planning. Thus, the user can (partially or completely) intervene in the process of planning a proof with his own strategies of how the proof should proceed.

[3]Details about methods and strategies are beyond the scope of the present paper. The interested reader is referred to [31, 30].

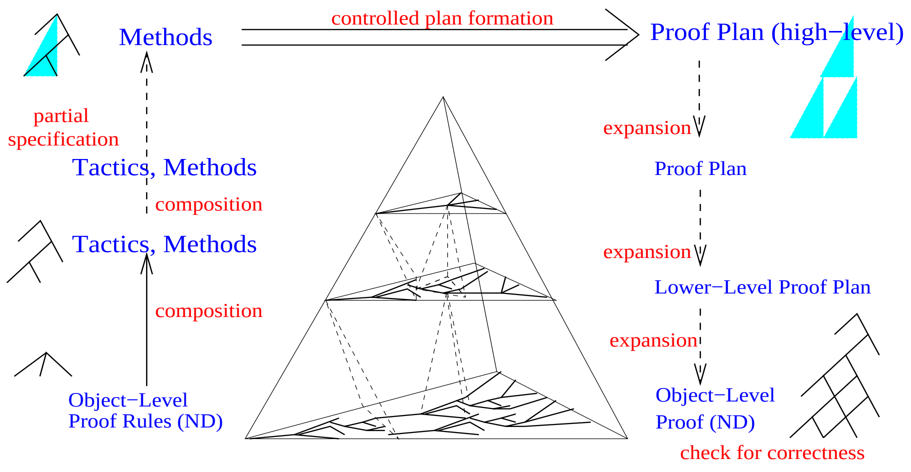


Figure 1: Hierarchical knowledge-based proof planning.

As application of lemmata lies at heart of most (non-trivial) mathematical proofs, it is important that this issue be addressed in a realization of any proof planner. In the approach proposed in this paper, the assertion level is embedded among the abstraction levels used during the process of plan formation. The assertions (which includes lemmata and definitions) thus play the role similar to that of methods. That is, the assertion agents, which are described below, operate in parallel with the proof planner to offer applicable assertions in the current proof situation.

In ΩMEGA, a proof planning process starts with a task, a data structure designed to encapsulate a complete (sub-)problem. Formally, a *task* is a pair $\langle$ $P_{L_{open}}$ consisting of an open line of the $\mathcal{PDS}$, L_{open}, and a set of lines from the $\mathcal{PDS}$, $P_{L_{open}}$. L_{open} is called the *task line* whose formula is called *task formula*. Members of $P_{L_{open}}$ are called the *support lines* or *supports* for the task line L_{open}.

Now consider the situation in which the prover is confronting a list of tasks, called an *agenda* in the ΩMEGA system, that it needs to solve in order to prove the intended theorem. Among the available strategies/methods, the prover could possibly ponder whether there is a definition/axiom or some previously proved result that it can use in the current proof situation to (i) either obtain further closed lines serving as intermediate steps for solving one of the tasks; (ii) or refine a goal task (on some open proof line) to some subtasks which can be resolved by further proof steps. It now boils down to the question of how assertions are to be handled by the proof planner. Since the prover certainly does not want to disrupt the proving strategy it is currently pursuing, it would be more helpful if the applicable assertions could be found and suggested to the prover by some independent assistants. Assertion agents are introduced to play the role of these independent assistants.

3 Modeling assertion agents

In this section we propose a module, which we call M below, that models assertion application as distributed search processes in the Ω-ANTS approach [9]. This agent-based formalism is the driving force behind a distributed proof search approach in ΩMEGA. It enables the distribution of proof search among groups of reasoning agents.

3.1 Formalization

First we briefly sketch the general application scenario that motivates our approach. We assume a scenario where a theorem prover TP is connected to a mathematical knowledge base KB. TP is currently focusing on a proof task $\mathcal{T} = \langle\ P_{L_{open}}$ and candidate assertions $\{\mathcal{B}_i\}$ are determined in KB and handed $L_{open}\rangle$ over to our assertion module M. The task of M is to compute with respect to proof task $\mathcal{T}$ all possible logical consequences of the available assertions $\mathcal{B}_i$.

We propose to create for each assertion $\mathcal{B}_i$ one associated instance $AG_{\mathcal{B}_i}$ of a generic *assertion agent* AG. The generic assertion agent AG is based on the algorithm Assertion-Application provided in [40, 41]. We emphasize that this algorithm only depends on the (logical) formula of the focused assertion and a further set of formulae for the proof context (encoded in the current proof task), and both are specified as parameters of Assertion-Application. Each assertion agent instance $AG_{\mathcal{B}_i}$ computes and suggests the logical consequences of $\mathcal{B}_i$ in the proof context $\mathcal{T}$ to our module M which passes them further to TP.

3.1.1 Defining assertion agents

Generic assertion agents are implemented in Ω-ANTS agent specification language whose details can be found in [9]. The philosophy behind the Ω-ANTS mechanism in ΩMEGA is to support an agent society that works simultaneously with a proof planner or a human mathematician who works interactively with ΩMEGA to prove a certain theorem. The behavior of these agents is therefore more or less reactive since, otherwise the suggestions made by the agents may be well behind the mathematician's reasoning. Assertion agents work amongst other agents who provide various services for the prover including: examining applications of (logical) inference rules on the involved connectives/quantifiers, searching for assumptions relevant to a particular goal task, searching for intermediate proof steps that are achievable from the premises and possibly relevant to a solution of the current goal task, etc. The following LISP-like agent specification, which is based on the declarative agent specification language described in [9], demonstrates the implementation of a generic assertion agent:

```
(agent~defagent assertionApplication c-predicate
  (for Conc)
  (uses assertion pre-requisite)
  (exclude Prem)
  (definition (assertion-appl
                 (:param assertion)
                 (:param pre-requisite)
                 Conc)))
```

The above assertion agent is defined as a `c-predicate` agent indicating that its search is restricted to open proof lines, i.e. possible conclusions. As assertion can be applied in any direction, e.g. backward, forward or even sideward, it is necessary that generic `s-predicate` and `p-predicate` assertion agents be defined. `s-predicate` agents search the support lines for possible premises whilst `p-predicate` agents search for multiple premises satisfying certain application conditions from the set of closed lines. The proof lines this generic agent looks for are instantiations of the argument `Conc`, given in the `for`-slot. The `use`-slot contains two arguments `assertion` and `pre-requisite` indicating that the agent requires these two parameters to be instantiated before it can complete the computation for the rest of the instantiation. The `exclude`-slot on the other hand determines that this agent does not complete any partial instantiation that has already contained an instantiation for the argument `Prem` which stands for the supported proof lines required by the corresponding proof rule (see below). The idea for the exclusion constraints is to suppress redundant or even false computations. The `definition`-slot of course contains the computation steps to be carried out when the agent is invoked.

Rewriting agents: A useful subclass of assertion agents. Mathematical theories often come with equality and a set of equations that relate the syntactical objects, or mathematical expressions. Rewriting agents check for applicability of equations and, depending on the current proof situation, make suggestion about rewriting steps to either simplify a mathematical expression or transform it to a known form, i.e. one to which a known solution is related. Rewriting agents were proved to be extremely useful in proofs that involve a lot of algebraic computations upon the mathematical expressions occurring in the proof objects. Moreover, rewriting agents were recently extended to deal with logical equivalence as follows: A proof situation can be equivalently transformed to another proof situation by replacing (under an appropriate instantiation) a subformula with another logically equivalent subformula if the gap between the premises and the goal formula in the resulting proof situation is judged to be smaller than that in the original proof situation.

3.1.2 Agent-based architecture

Our approach separates the search for applicable assertions in the following sense: While the prover TP may consider different options about what rules, methods, or strategies to use, a set of assertion agents is also working spontaneously and concurrently to find out and suggest the applicable assertions. In Ω-ANTS, agents compute and propose ΩMEGA-commands that invoke proof rules applicable in the current proof situation. Such proof rules, if applied, transform the current proof state to a new proof state by modifying the $\mathcal{PDS}$. Rating criteria may be specified and employed in Ω-ANTS to heuristically sort the computed consequences before they are (dynamically) passed to the TP as alternative options for the continuation of the theorem proving process. Figure 2 depicts the main features of the Ω-ANTS system with the assertion agents embedded. For further details about the system architecture of the Ω-ANTS mechanism, the reader is referred to [8].

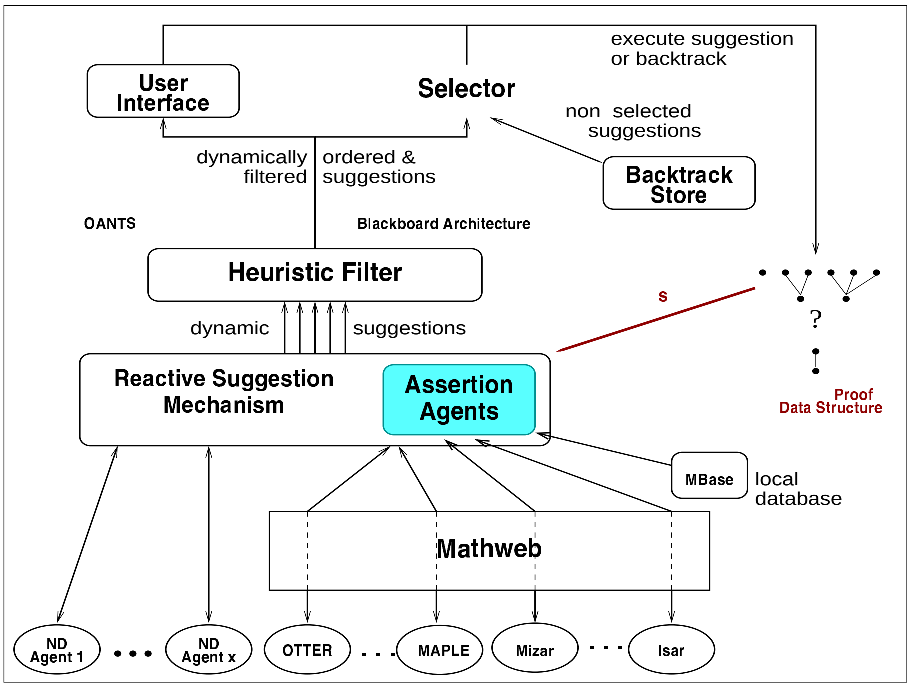


Figure 2: System architecture of an agent-oriented reasoning system.

In figure 2:

- MBASE, which will be discussed in more details below, is the local database of mathematics. At the moment, all assertions come from MBASE or must be translated to the format adopted by MBASE before they can be passed to the assertion module M.

- MATHWEB [13] is an agent-based logical broker that provides a communication channel between different systems to allow them to talk to each other and to exploit the facilities provided by others. In this figure, MATHWEB allows ΩMEGA to exploit the reasoning capabilities provided by external systems such as the first-order theorem prover OTTER or the computer algebra system MAPLE. Moreover, MATHWEB also allows ΩMEGA to connect to external databases of mathematics such as the MIZAR and ISAR libraries. Through MATHWEB, data from these libraries are requested and reformatted to be compliant to the internal representation of the ΩMEGA system. Then they are structured into the hierarchy of theories in MBASE before being passed to the assertion module M.

- The ND Agent$_i$'s at the lower left corner of the figure provide ΩMEGA with the calculus level reasoning capability. Recall that when the proof objects at some level of abstractions are evolved by the prover, we have to try to keep the proof objects at other levels up-to-date, in particular those at the calculus level. In the end, the correctness of the planned proof must be guaranteed by a corresponding proof at the calculus level, or the object-level.

Depending on the size of the knowledge base KB there could be too many applicable assertions passed to M and also too many ways an assertion can be applied to be handled in practice.

For instance, going back to our running example again, let our current task consist of the open proof line $\vdash e \in U$ and contain no other assumption about U being the superset of any other set or e being a member of some other set with which U can have a subset relationship. However, the assertion $\mathcal{A}$ is applicable in this situation and the outcome of applying our algorithm to this open line is the new open proof line (corresponding to a derived subgoal): $\vdash \exists \quad et \quad \subseteq U \wedge e \in \quad$) which is, even though logically correct, not a very useful subgoal to be pursued as there are infinitely many ways in which the variable could be instantiated.

We sum up the above argument by claiming that restricted application of assertion is necessary. One possible and simple restriction is to impose prerequisite(s), such as simple syntactical criteria or domain restrictions, when selecting the candidate assertions that are passed from KB to M. For instance, regarding our running example, a simple but useful heuristic to prove the membership of an object wrt. a set using the subset definition is that the task include a closed line stating that is a superset of some other set. It is this restricted version of assertion application that has been implemented in the agent-based mechanism Ω-ANTS in the ΩMEGA system.

While such heuristic constraints are not too difficult to realize in toy problem domains, it could be a challenging problem in more complex mathematical domains. It's no longer appropriate just to consider the current proof situation as some other proof fragments may be relevant to realize such constraints. Furthermore, there may be different proof techniques that are successful (possibly on different problems) with or without a certain constraint. Proof planning, however, has developed more sophisticated ways to guide and constrain possible instantiations and applications of assertions (see [27]). The investigation on how some of these techniques can optimally be employed on top of our assertion application module M is further work.

3.1.3 Example

We present a proof constructed in the ΩMEGA system using assertion agents. The initial $\mathcal{PDS}$ formulating the problem is as follows:

1.	1;	$\vdash$ *symmetric*)	Hyp ✓		A
2.	2;	$\vdash$ *symmetric*)	Hyp ✓		B
THM	1,2;	$\vdash$ *symmetric*))			$A \cap B$?

The following proof for the above problem uses *the definition of the symmetric relation, viz.* (Sym-Def): *symmetric*) $_{x,y}\langle x\ \ y\rangle \in R$ $\langle y\ \ x\rangle \in R$, and *the definition of intersection (on sets), viz.* (∩-Def): R $\forall_{S_1,S_2}\forall_x x \in$ $\cap$ $\wedge\, x \in$. $x \in$

1.	1;	$\vdash$ *symmetric*)	Hyp		A
2.	2;	$\vdash$ *symmetric*)	Hyp		B
3.	1,2;	$\vdash$ $_{x,y}\langle x\ \ y\rangle \in A$ $\langle y\ \ x\rangle \in A$	[Sym-Def]	1	
4.	1,2;	$\vdash$ $_{y}\langle x\ \ y\rangle \in B$ $\langle y\ \ x\rangle \in B$	[Sym-Def]	2	
5.	5;	$\vdash \langle c\ \ c\ \rangle \in A \wedge \langle c\ \ c\ \rangle \in B$	Hyp		
6.	1,2,5;	$\vdash \langle c\ \ c\ \rangle \in A$	[3]	5	
7.	1,2,5;	$\vdash \langle c\ \ c\ \rangle \in B$	[4]	5	
8.	1,2,5;	$\vdash \langle c\ \ c$ $A \wedge \langle c\ \ c$ B	And-I	6,7	$\rangle \in$ $\rangle \in$
9.	1,2;	$\vdash$ $_{y}\langle x\ \ y\rangle \in A \cap B$ $\langle y\ \ x\rangle \in A \cap B$	[∩-Def]	5,8	
THM	1,2;	$\vdash$ *symmetric*)	[Sym-Def]	9	$A \cap B$

In the above example, the only rule needs explanation is And-I which is an inference rule of the ND calculus performing Conjunction Introduction. The interpretation is: The proof line (8.) can be obtained from (6.) and (7.) by taking conjunction of the two formulas in the succedents. Hyp indicates that the given proof line is a given hypothesis. The scopes of the hypotheses are rendered by the antecedents of the proof lines, i.e. those on the left of $\vdash$). Rules denoted by [assertion] indicate the application of `assertion`. In particular, on line (6.), the `assertion` is `(3.)` which indicates that the succedent of the sequent on line (3.) has been added to the current mathematical database and now serves as a normal assertion. Similar remark can be made for line (7.) (and the assertion introduced on line (4.)).

3.2 Advanced features

In this section we discuss several interesting features of our agent-based mechanism for assertion application.

3.2.1 Dynamically updated assertion databases

In order to preserve the completeness of algorithm Assertion-Application ([40, 41]), the mathematical knowledge base KB described in the preceding section should not be kept static during a proving session. Firstly, since ΩMEGA employs integrated reasoning systems (possibly distributed over the Internet) through the MATHWEB agent architecture [13], new information including additional assertions must be dynamically updated to KB. Secondly, as a proof proceeds, new derivatives or constructs established by the earlier proof steps can be considered as (additional) knowledge-based assertions which in turn may be applied to establish further conclusions. For instance, many proofs, e.g. completeness proofs for logical systems, center around the construction of certain mathematical structures, e.g. particular models for a certain set of formulas. Such constructions/definitions are first introduced in the proof before being used intensively throughout the rest of the proof.

As an illustration, the assertion level proof presented in the example in subsection 3.1.3 dynamically augments the assertion database with the formulas on proof lines (3.) and (4.), obtaining the two new assertions [3] and [4], respectively. They are subsequently applied to the formula on proof line (5.) to obtain (6.) and (7.).

The decision of what formulas in the current proof state are to be considered as an assertion is beyond the scope of assertion agents. Such decisions must be made by the module that carries out the proof, i.e. the theorem prover or the proof planner. The prover then updates the knowledge base KB with these additional assertions. Once new assertions have been added, the associated assertion agents will be created to take care of these assertions.

3.2.2 Agents for combined assertions

As described in section 3.1, each (instance of an) assertion agent is assigned to a single assertion. This restriction is not essential as our formalism for assertion application in principle allows finitely many assertions to be taken into account. It however is related to one of the open questions of the intelligent agent community: Should the agents be deliberative or reactive? Taking too many assertions into consideration with too many ways assertions can be applied and/or too many orders of application would almost definitely lead to combinatorial problems and explode the search space. Furthermore, collapsing too many assertion applications into one single proof step potentially introduces very obscure proof steps without sufficient justifications.

However there are cases in which two or more assertions can be combined naturally into reasonable single proof steps with comprehensible justifications. For instance, the proof-based assertion that a set A is a subset of $A \cup B$ for some set B which can easily be justified (possibly by a Venn diagram) is the result of applying two (knowledge-based) assertions which are *the definition of subset* (i.e. $\subseteq$) and *the definition of union* (i.e. $\cup$). However, such flexible combinations of assertions in assertion agents need to be guided by domain-specific knowledge. We are currently investigating an extension of the Ω-ANTS agent architecture to allow encoding of such meta-information to the specification of agents.

3.2.3 Proof search distribution

So far in this paper, we have described assertion agents as information mediators that are specially designed so that the retrieved information can contribute directly to the potential solutions of the problem at hand. However, the decision of what technique (i.e. assertion/method/rule/etc.) to be used and under which strategy still remains to be made by the prover. That is, the proof search process is distributed among different components of the mathematical assistant system such as the proof planner/user, the assertion agents, the suggestion mechanism, etc. but only in a weak sense. In order to really achieve a full proof search distribution mechanism, it's necessary to distribute the decision making process as well, at least to a certain extent.

As we don't want assertion agents to be involved in very long proof search processes, we introduce *assertion-based task agents* to carry out particular proving strategies for the assertion level. Amongst these strategies, the two most successful we have discovered so far are Looking-Ahead and Island-Planning:

Looking-Ahead: is a simple and efficient strategy. In this technique, assertions are not required to be fully instantiated and applied to a current proof state. Rather, the algorithm just speculates the proposed assertions to see if, by putting these assertions together, there is a way to resolve all the outstanding subgoals without generating new subgoals, i.e. all newly generated subgoals are resolved within these assertions agents. Since, under this strategy, the assertions may not be completely instantiated, there may be potential solutions discovered by Looking-Ahead that do not guarantee any actual solution, i.e. no consistent instantiations exist to witness the potential solutions.

This strategy works quite well with the assertion agents due to several reasons: (a) it doesn't impose constraints or restrictions on the order of the speculated assertions and thus will not interfere the communication between assertion agents and the suggestion mechanism; (b) it is independent of any strategy or method pursued by the prover and thus suits best to the philosophy behind our framework of assertion agents; and (c) the outcome of the algorithm can be updated on-the-fly: if the prover has committed to further proof steps which generate new outstanding subgoals under new instantiations of variables then they can be used to further constrain the potential solutions generated by algorithm Looking-Ahead, e.g. some previously acceptable solutions can be ruled out because they violate the new instantiations.

Island-Planning: Melis [28] proposes a powerful technique for proof planning and theorem proving, *viz. island planning and refinement.* Informally, the key idea is as follows: during a problem solving process, when there is no clear indication of how to proceed, e.g. which branch of the search tree to be pursued among a very large number of possible branches, problem solvers sometimes make extra assumptions to figure out a path to the goals before attempting to satisfy these assumptions with the current premises. Such extra assumptions are called *islands* as, at the time they are introduced, they are completely isolated from other objects in the domain reasoner's possession. This is particularly common in mathematics when mathematicians attempt to speculate a lemma (or lemmata) that is needed to narrow the gap between the goal and the given premises. The remaining question is how to come up with the right lemmata (most of the time). Melis [28] suggests that meta-reasoning (using meta knowledge about the problem domain) is necessary to produce the right lemmata for the problem.

Whereas a working version of algorithm Island-Planning has not been realized, substantial efforts within our research group have been devoted to this proving strategy. Island planning is, without doubt, best suited to assertion agent mechanism introduced in this paper since its objects of reasoning are the assertions themselves which can be inserted directly to the current proof situation and further reasoning (with the assistance of assertion agents) can be immediately carried out without any disruption. Issues and applications of island planning to assertion agent mechanism will be revisited and further discussed in the next section.

4 The DIALOG project: an application of agent-based assertion level proof planning

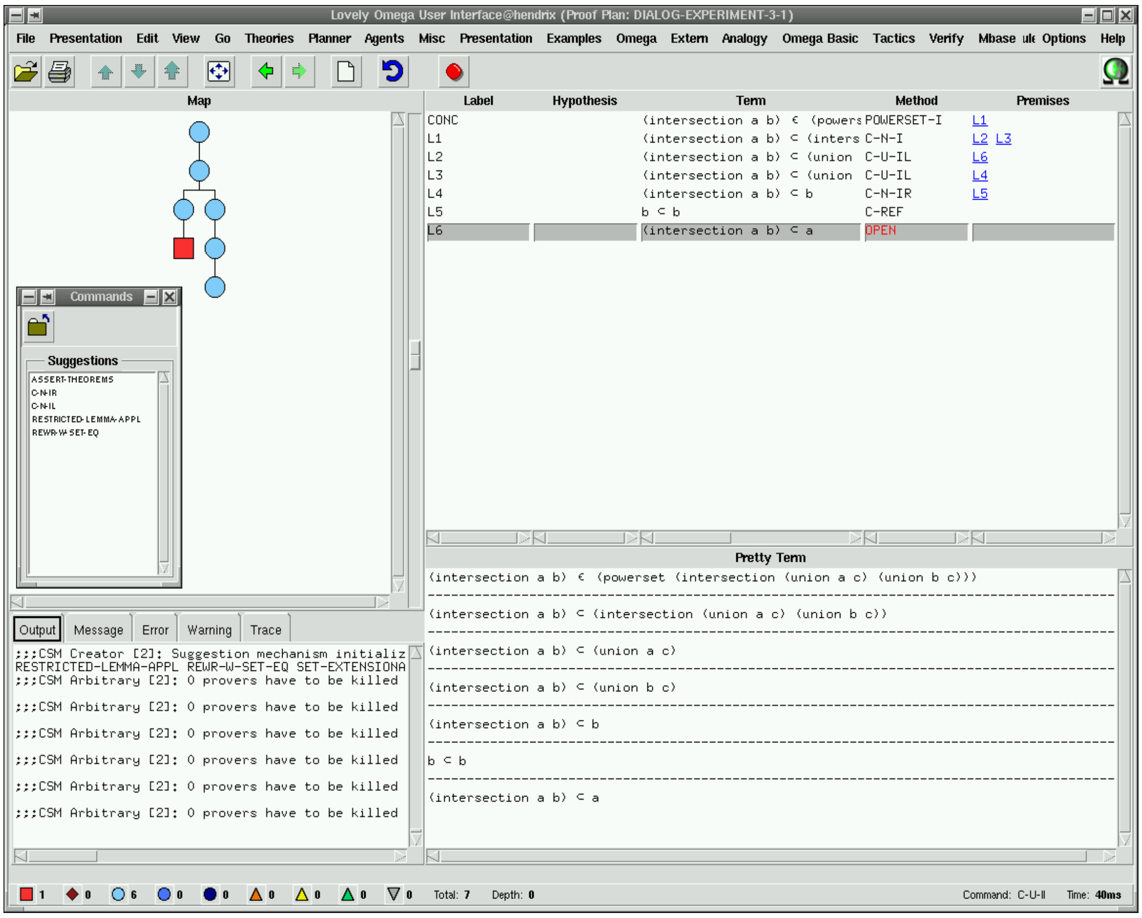
4.1 Overview

Our approach to agent-based assertion level proof planning is motivated by an application in the DIALOG project [7] as part of the Collaborative Research Center on *Resource adaptive cognitive processes* at Saarland University. The goal of this research project is (i) to empirically investigate flexible dialogue management strategies in complex mathematical tutoring dialogues, and (ii) to develop an experimental prototype system gradually embodying the empirical findings. The experimental system will engage in a dialogue in natural language (and perhaps other modes of communication) and help a student to understand and produce mathematical proofs. It is important that such a system be supported by a human oriented mathematical proof development environment and the ΩMEGA system with its advanced proof presentation and proof planning facilities is a suitable answer to this requirement.

The overall scenario in the DIALOG project is as follows: A student user is first taking an interactive course on some mathematical domain (e.g., naive set theory) within a learning environment such as ACTIVEMATH [29]. When finishing some sections the student may be asked to test his learning process by actively applying the studied lesson material by performing an interactive proof exercise. Since the learning environment is equipped with user monitoring and modeling facilities a user model is maintained and dynamically updated containing information on the axioms, definitions, and theorems (hence the assertions) the student has studied so far. Also a teaching model is available for each exercise containing information on the mathematical material that should be employed and tested.

In this scenario we expect the mathematical assistant system to be capable of (i) stepwise-interactive and/or (ii) automated proof construction at a human oriented level of granularity for the proof exercise at hand using exactly the mathematical information specified in the (a) teaching model or (b) user model. The proofs constructed for (a) reflect what we want to teach and the proofs for (b) what the system expect the user to be capable of. For interactive tutorial dialogue the support for a stepwise proof construction with the mathematical assistant system is of course important, while fully automatically generated proofs are needed to be able to also give away complete solutions or to initially generate a discourse structure for the dialogue on the chosen exercise. We want to stress that the user model may be updated also during an exercise, hence the set of relevant assertions may dynamically change during an interactive session.

It is easy to motivate the design of our assertion application module for this scenario. Its capabilities for assertion application for a dynamically varying set of assertions are crucial for the project. It is also essential that reasoning is facilitated at a human oriented level of granularity, since we do not want the user to puzzle around with the peculiarities of, for instance, logical derivations in sequent or natural deduction calculus. Figure 3 shows a screen shot of the ΩMEGA system illustrating a proof of one of the exercises used in the experiment described below. The proof was found with the assistance of the assertion agents (and rewriting agents) who offer their ranked suggestions through the command agents shown in the little **Commands** window at middle left.


Lovely Omega User Interface@hendrix (Proof Plan: DIALOG-EXPERIMENT-3-1)
File Presentation Edit View Go Theories Planner Agents Misc Presentation Examples Omega Extern Analogy Omega Basic Tactics Verify Mbase ule Options Help
Map
Commands
Suggestions
Output Message Error Warning Trace
Pretty Term

Label	Hypothesis	Term	Method	Premises
CONC		(intersection a b) ∈ (powers	POWERSET-I	L1
L1		(intersection a b) ⊂ (inters	C-N-I	L2 L3
L2		(intersection a b) ⊂ (union	C-U-IL	L6
L3		(intersection a b) ⊂ (union	C-U-IL	L4
L4		(intersection a b) ⊂ b	C-N-IR	L5
L5		b ⊂ b	C-REF	
L6		(intersection a b) ⊂ a	OPEN	

```
ASSERT-THEOREMS
C-N-IR
C-N-IL
RESTRICTED-LEMMA-APPL
REWR-W-SET-EQ
```

```
;;;CSM Creator [2]: Suggestion mechanism initializ
RESTRICTED-LEMMA-APPL REWR-W-SET-EQ SET-EXTENSIONA
;;;CSM Arbitrary [2]: 0 provers have to be killed

;;;CSM Arbitrary [2]: 0 provers have to be killed

;;;CSM Arbitrary [2]: 0 provers have to be killed

;;;CSM Arbitrary [2]: 0 provers have to be killed

;;;CSM Arbitrary [2]: 0 provers have to be killed

;;;CSM Arbitrary [2]: 0 provers have to be killed
```

```
(intersection a b) ∈ (powerset (intersection (union a c) (union b c)))
--------------------------------------------------------------------------------
(intersection a b) ⊂ (intersection (union a c) (union b c))
--------------------------------------------------------------------------------
(intersection a b) ⊂ (union a c)
--------------------------------------------------------------------------------
(intersection a b) ⊂ (union b c)
--------------------------------------------------------------------------------
(intersection a b) ⊂ b
--------------------------------------------------------------------------------
b ⊂ b
--------------------------------------------------------------------------------
(intersection a b) ⊂ a
```

🟥 1 ◆ 0 ○ 6 ● 0 ● 0 ▲ 0 ▲ 0 ▲ 0 ▽ 0 Total: 7 Depth: 0 Command: C-U-II Time: 40ms

Figure 3: Screen shot of the ΩMEGA system.

4.2 Empirical experiments and preliminary results

The first of a series of empirical experiments for the project was carried out in February 2003.[4] In these studies, 24 students with varying background were interacting with an intended tutorial natural language dialogue system in a *Wizard-of-Oz* (WOz) study (see [10]), where the dialogue system was mimicked by a human mathematician; the joint task was to construct proofs in naive set theory. Neither the tutor system nor the subjects were constrained in this experiments with respect to particular proof formats. The obtained corpus has now been analyzed and provides valuable insights in the notion and nature of proofs constructed by humans in this field. Some of the relevant phenomena are (a) the level of granularity of these proofs, (b) their style, and (c) the aspect of under-specification; e.g. missing references to premise assertions, rule and instantiation specifications.

Findings from the experiment provide us with substantial insights into the following particular aspects (in addition to other findings, e.g. about linguistic phenomena):

- *Human-oriented interactive proof:* Many research papers in the field of deduction systems refer to the notions of human-oriented proofs. However, rather few empirical data are available in the sense of our corpus in which proofs were constructed interactively by students and teachers. Many scientist and system developers in the theorem proving community still take for granted that sequent- or natural-deduction-based proof representation formats are already a nearly optimal solution for human-oriented proofs. To the contrary, the corpus we collected indicates that neither the natural-deduction nor the sequent calculus nor a pure rewriting based approach are sufficient here, whereas assertion-level proofs come much closer. We are also able to pinpoint an important aspect of human-oriented proofs which has been neglected thus far in the literature of theorem proving and proof presentation, namely *under-specification* (see [5]).

- *User-interfaces for theorem provers:* The clarification of the notion of human-oriented proofs is very relevant for the design of user-interfaces for theorem provers. Our viewpoint coincides with those by Théry [37], who investigates paper proofs instead of interactively constructed proofs with under-specification, in the sense that a clear separation between the optimally user-oriented proof representation in the user interface and the usually machine-oriented proof representation in the theorem prover appears increasingly important.

It turns out that both aspects above are closely related to the way that proof search is carried out by mathematical assistant systems. As discussed in [5], in relation to human-oriented proofs, it is reasonable to differentiate between cleaned-up textbook proofs (which have been investigated in the literature) and interactive dialogues on proofs, for instance, between students and tutors, as in our setting. In interactive dialogues (on proofs) between a human user and a mathematical assistant system, the system must deal with the user's utterances which contain natural language, formal and informal mathematical expressions among other things. For the system to be able to judge several key criteria about the user's utterances such as: *Is the proof step proposed by the user relevant to the current proof context?* or, *How much detail has the user ignored by coming up with this proof step?,* it is essential that the system reason at the same levels of abstraction as those of the user and they share common, or at least similar, proof objects.

On the other hand, the approaches used by the system and the user to solve the problem can be completely different: they pursue different strategies, they proceed along the proof in different directions, e.g. the user may choose to apply forward proof steps whereas the system proceeds backwardly, etc. More importantly, as has been found by studies in cognitive science, humans often switch to different levels of

[4]The complete corpus from the experiment is available at the DIALOG homepage `http://www.ags.uni-sb.de/~dialog/`

abstraction when reasoning. For instance, rather than performing exhaustive search over the current proof situation to find possible instantiations of variables or possible ways to manipulate the current proof objects, human mathematicians often sit back to try to look at the problem as a whole, concentrate on the essential concepts and the central part of the problem, or speculate some lemmata that could be useful to provide a link between disconnected parts, etc. As such, the user's utterances to the system are usually very similar to the *islands* discussed in the preceding section. Provided with such an island proof step, there are a number of challenges the system must be up to: (i) the user might not provide any connection between her proposed proof step and the current proof state, (ii) there are often no immediate connections between the user's proof step and the current proof state, (iii) the proposed proof step is invalid but, as it is a remote island relative to the current proof state, disproving such false conjectures could be a very daunting task as shown by Autexier and Schümann [6], etc.

As assertion agents and assertion-based task agents facilitate distribution of proof search with varying proving strategies and abstraction levels, these challenging tasks can be significantly smoothened. In order to overcome the under-specification issue and to facilitate a smooth communication interface between the system and the user, the ΩMEGA system is wrapped up by a Proof Manager which provides value-added services such as: providing an interface proof language to facilitate communication between the ΩMEGA system and the tutoring system, providing data structures that allow informal and under-specified expressions from the user to be analyzed and reformatted before passing them down to the mathematical assistant system, etc. For details about this formalism and its realization, the reader is referred to [5].

5 Mediators and Digital Libraries (DLs)

5.1 The problems ...

As discussed by Wiederhold and Genesereth [42, 43], information agents generally connect to DLs and obtain information from these rich and highly specified data resources. Since most DLs are general-purpose, the format of data stored in these DLs is generally *not* semantics driven.[5] In general, DLs currently store data in formats such as HTML and its variants, Adobe's Portable Document Format (PDF) or Adobe's PostScript which are readable to humans but can hardly be used by computer systems such as theorem provers. The assertion agents described in the preceding section are not able to operate directly on such data as they assume the input assertions to be (logical) well-formed formulae.

Before we proceed further, we briefly discuss about mathematical knowledge modeling within the mathematical assistant system ΩMEGA. Mathematical knowledge modeling deals with several general aspects such as *representation*, *storage* and *multiple reuse* of collected pieces of mathematical knowledge, creating tools for combining knowledge from different sources and interactively exchanging it with numerous existing mathematical systems. The collected mathematical knowledge is stored in MBASE — a web-based distributed database of mathematical knowledge (see [23]). The goal of MBASE is to serve as a storage of mathematical knowledge and provide tools for content oriented search and retrieval of the stored information. Highly structured and content-oriented, the semantic XML-based language OMDOC[6] allows multiple reuse of pieces of knowledge. MBASE uses OMDOC as a main data exchange format. Therefore, it can contain formalized mathematical knowledge embodied as structured mathematical objects such as assertions, proofs, theories, etc. formulated in natural language.

[5]The problem is currently being remedied by research in the Semantic Web community.

[6]OMDOC [22] is an extension of OPENMATH [32], a standard framework for transmitting mathematical objects over the Internet. OMDOC extends OPENMATH by introducing the concept of Content Dictionaries to allow the semantics of mathematical objects to be defined and integrated.

5.2 ... and the proposed approaches

The above discussion exposes two major obstacles for information mediators that provide support to mathematical assistant systems in a general setting: (i) information obtained from Digital Libraries are in general not directly usable by MathAS unless it is transformed to the specified formats; and (ii) despite efforts to achieve a consensus over a standardized format for mathematical knowledge modeling, several formats exist, e.g. MATHML [25] and OMDOC for mathematical modeling. Any information mediator providing the general infrastructure for MathAS necessarily overcomes these two problems.

The second problem, albeit non-trivial, is apparently easier to solve than the first one. The *facilitator approach* (also called *federation multi-agent architecture*) was proposed by the SHADE project [26] and demonstrated in the PACT project [12]. In this approach, agents interact through facilitators that translate system-specific knowledge into and out of a standard knowledge interchange language. Each agent can therefore reason in its own syntax and formalization, asking other agents for information and providing other agents with information as needed through the facilitators. Given several existing knowledge interchange formats (KIFs), meta-facilitators can be constructed to allow messages in one language to be translated to other languages and vice-versa. This is made possible by sharing of the Content Dictionaries used by different KIFs.

The approach we are currently pursuing to solve the first problem is based on the information mediator architecture proposed by Wiederhold and Genesereth [43]. The basic technologies towards a solution to this problem have been carefully researched by other research groups working in the area of information agents and mediators mention in Section 1 including the *value-added services* proposed by Wiederhold, the hybrid approach to information integration by *pre-compiling the source descriptions* into a minimal set of integration axioms proposed by Ambite *et al.* [2], etc. These technologies include important techniques such as: (i) resource locator, i.e. locating the relevant data by systematic profiling and indexing. Wiederhold and Genesereth [43] propose that this module can be achieved through *facilitation.* Facilitation also provides additional information about resource capabilities; (ii) resolution (including mismatch resolution and conflict resolution), i.e. data obtained from remote and autonomous sources will often not match in terms of naming, scope, granularity of abstraction, temporal bases and semantics (including domain semantics, value semantics). Such mismatches, without an adequate resolution scheme, can at best lead to the mediators' failure to locate the data sources and at worst cause the mediators to return the incorrect information. This again emphasizes the significance of semantics-driven representation of data and information. Wiederhold and Genesereth [43] propose that resolution is accomplished by *automation.*

Due to the availability of the above basic technologies, we will focus in stead on the domain specific features of (mathematical) information mediators in the following. The idea, which is still being under development, is to treat mathematical assistant systems together with information mediators as a whole, i.e. a society of agents, rather than MathAS being the clients who request and receive the information services from mediators.

5.3 MathAS as an agent society

The underlying vision is to have all integrated components of MathAS, including information mediators, as *knowledge-based agents* working interactively with each other and helping improve each other. While the data sources that this society is able to access through the mediators are distributed all over the Internet, this MathAS agent society is equipped with its own mathematical knowledge base *KB*. Now, it is critical that the knowledge base *KB* of the society evolves, i.e. this knowledge is incrementally updated during MathAS activities including interactions with data resources which can be external mathematical data bases and/or Digital Libraries. It is also important that knowledge acquisition agents be built and incorporated to MathAS though this is a non-trivial research problem and relates to a whole research area of AI. Hence, we will adopt the following hypotheses:

1. Mediators have access to the knowledge base *KB* consisting of mathematical theories structured in a hierarchical taxonomy, i.e. a database contains basic mathematical concepts with their corresponding symbolic representation and the relations (at least partially) between these concepts, e.g. MBASE;

2. The agents of MathAS continually consolidate *KB* with new knowledge obtained in different ways. Such knowledge can be integrated information provided by mediators as they obtain data from external sources, or it can be acquired by the knowledge acquisition agents through interactions with human users, or it can also be a theorem successfully proved by the proof planner agent together with the constructed proof(s) as well as the invented mathematical concepts during the proof construction process.

3. Not only the mathematical knowledge base *KB* is improved but also the control information as well as the meta-information used to guide the agents' working, i.e. the agents' knowledge, must also be incrementally updated. That is, the proof planner learns new proving techniques and new methodologies or short-cuts, the knowledge acquisition agents have more ways to model the (human) user's knowledge, and the mediators have better ways to disambiguate vague expressions obtained from unstructured data sources, etc. Notice how human behavior is simulated by these features of MathAS: the system learns from other mathematicians (human or machine) and from (digital) libraries, then it possibly produces new results and publishes or communicates these new results to others, and so on.

In the ΩMEGA system, the first hypothesis has been realized with the representation language OMDOC for mathematical documents and the web-based distributed database of mathematical knowledge MBASE. On the other hand, the second and third hypotheses are long-term projects and non-trivial as it contains many artificial intelligence (AI) problems, in particular those of machine learning and knowledge acquisition. Only the problem of learning the proof methods for the proof planner has been partially addressed [20, 21].

6 Related work

Distributed proof search has been one of the fairly new research areas within the automated reasoning community as the task of proving a theorem can not be easily decomposed to subtasks that can be solved by different processes or agents. The problem largely is due to the interdependencies between different branches of the proof tree. These interdependencies in general can not be determined in advance and often require that the solutions for subproblems on different branches of the proof tree be sequentially ordered. For example, the instantiation of a meta-variable on one branch of the proof tree may determine how the proof for the subproblem on another branch will be unfolded. Recently, Harland and Pym [15] introduce a distributed proof search mechanism for linear logic. This is possible due to the fact that formulas in linear logic are considered as a kind of resources which can be "consumed" as the proof proceeds. Thus, there are very few or almost no constraints between different branches on the proof tree. Their work was further developed to formulate agent negotiation as proof search in linear logic [16].

On the other hand, distributed proof search with the ΩMEGA system [8, 9] takes a completely different perspective: The proof planner delegates the whole subtask to an external reasoner while working on its own proving strategies. If the external reasoner returns with a proof for the subtask that is consistent with the (partial) solution the proof planner has constructed thus far then a proof for the assigned subtask is taken as granted and the proof planner continues to pursues its solution. Nevertheless, in the future if an object-level proof of the whole problem is requested, the ΩMEGA system still has to re-prove the assigned subproblem.

With the mechanism of assertion agents, we take this idea one step further. Firstly, the assertion agents facilitate new problem solving methods: each applicable assertion offered by assertion agents can be considered as a new proof method which can be used by the proof planner in the current proof situation. Since assertion agents are independent of the proof planner, their services are offered declaratively (through the blackboard architecture). The proof planner then can have particular strategies or control rules to allow it to deploy the services of the assertion agents in the most optimal way.

Secondly, with the introduction of assertion-based task agents, the system is able to pursue several different proving strategies in parallel. Considering the proof trees of most non-trivial problems in mathematics of which most branches are infinite (due to infinitely many instantiations of the universally quantified variables), this mechanism provides us with an approach to this dilemma by allowing all proving strategies to be tried and tested. However, any solution to this problem certainly requires a more sophisticated multi-agent architecture based on powerful cooperative and distributive problem solving mechanisms. We observe that the RESINA Project of the Intelligent Software Agents Lab (CMU) [36] also proposes a distributed problem solving multi-agent framework in which task agents communicate tasks and (proposed) solutions to the users (through the interface agents). These task agents are therefore generic and should be able to solve any task requested by the users. Our assertion agents, on the other hand, are domain specific and implement a set of methods suitable for the theorem proving domain.Therefore, it would be interesting to see how the RETSINA multi-agent system infrastructure can be deployed to develop a more sophisticated framework for our assertion-based task agents.

With respect to task-oriented information mediator, a substantial amount of research has been carried out by the Information Agents Group at the University of Southern California, Information Sciences Institute [19]. Most notable are Ambite *et al.*'s [2] mediator-based approach to integrating information from heterogeneous data sources for the domain of logistic planning and Ambite and Knoblock's [1] approach to cost-based query planning in mediators. In their approach, data from heterogeneous sources is gathered and integrated before being passed to a query planner whose task is to generate a cost-efficient plan that computes a user query from the relevant information sources. In a sense, their information mediator plays a similar role to that of our assertion agents. However, it is not clear whether their information mediator has access to the state of the plan produced by the query planner in order to concentrate its effort on the more relevant information sources. Our approach therefore is much closer to a multi-agent system architecture based on a cooperative problem solving mechanism.

7 Conclusion and Future Work

In this paper, we proposed a task-oriented information mediator architecture. We presented our formalism in an integrated environment for mathematical assistant systems.

The assertion level introduced by Huang [17] is one of the more interesting abstract levels where theorem proving should be carried out. Therefore, it is necessary that the prover be equipped with an adequate infrastructure to be able to take full advantage of the inferences offered by the assertions. We developed an agent-based mechanism to realize the required infrastructure which serves as an assistant for proof search in ΩMEGA. The agents work as task-oriented information mediator between mathematical knowledge bases and the prover. The agents also offer further services such as look-ahead during the proof search process. The research bears immediate fruit through our application in the DIALOG project [7] aiming at building an intelligent mathematical tutoring system. The realization of natural language dialogue in such a project should take full advantage of the assertion level proofs developed by our formalism.

Task-oriented information mediators is suitable for problem solving applications with well-defined task structures that require access to (distributed) knowledge bases. In addition to theorem proving applications as described in the present paper, other problem solvers such as logistic planners, schedulers, etc. can also fit into this category. We also described the main features of information mediators connecting to general Digital Libraries. However, there are many challenging open problems which require further research. The future work also extends to fully automated proof search and proof construction based on the system's knowledge and/or external information sources. The realization of this project in the ΩMEGA system has been partially achieved by the introduction of assertion-based task agents which are built on top of the suggestion agent infrastructure and a blackboard architecture. However, as discussed at the end of section 3.1.2, the more complex heuristic-guiding search techniques developed in proof planning are still yet to be fully employed by task agents. A solution to this problem requires a more sophisticated multi-agent architecture based on powerful cooperative and distributive problem solving mechanisms. Furthermore, whether our approach developed mainly for theorem proving will eventually be applicable and extendable to other problem domains remains a long term research question.

References

[1] José Luis Ambite and Craig A. Knoblock. Flexible and scalable cost-based query planning in mediators: A transformation approach. *Artificial Intelligence Journal*, 118:115–161, 2000.

[2] José Luis Ambite, Craig A. Knoblock, Ion Muslea, and Andrew Philpot. Compiling source descriptions for efficient and flexible information integration. *Journal of Intelligent Information Systems*, 16(2):149–187, 2001.

[3] Peter Andrews. Transforming matings into natural deduction proofs. In Wolfgang Bibel and Robert A. Kowalski, editors, *Proc. 5th CADE*, pages 281–292. Springer-Verlag, 1980.

[4] Serge Autexier. A proof-planning framework with explicit abstractions based on indexed formulas. *Electronic Notes in Theoretical Computer Science*, 58(2), 2001.

[5] Serge Autexier, Christoph Benzmüller, Armin Fiedler, Helmut Horacek, and Quoc Bao Vo. Assertion-level proof representation with under-specification. *Electronic Notes in Theoretical Computer Science*. To appear.

[6] Serge Autexier and Carsten Schümann. Disproving false conjectures. In Moshe Vardi and Andrei Voronkov, editors, *Proceedings of Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'03)*. Springer-Verlag, 2003.

[7] Christoph Benzmüller, Armin Fiedler, Malte Gabsdil, Helmut Horacek, Ivana Kruijff-Korbayová, Manfred Pinkal, Jörg Siekmann, Dimitra Tsovaltzi, Bao Quoc Vo, , and Magdalena Wolska. Tutorial dialogs on mathematical proofs. In *Proceedings of IJCAI-03 Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems*, 2003.

[8] Christoph Benzmüller, Mateja Jamnik, Manfred Kerber, and Volker Sorge. Experiments with an agent-oriented reasoning system. In Franz Baader, Gerhard Brewka, and Thomas Eiter, editors, *KI 2001: Advances in Artificial Intelligence, Joint German/Austrian Conference on AI, Vienna, Austria, September 19-21, 2001, Proceedings*, number 2174 in LNAI, pages 409–424. Springer, 2001.

[9] Christoph Benzmüller and Volker Sorge. Oants – an open approach at combining interactive and automated theorem proving. In Manfred Kerber and Michael Kohlhase, editors, *Symbolic Computation and Automated Reasoning*, pages 81–97. A.K.Peters, 2000.

[10] N. O. Bernsen, H. Dybkjær, and L. Dybkjær. *Designing Interactive Speech Systems — From First Ideas to User Testing*. Springer, 1998.

[11] Alan Bundy. The use of explicit plans to guide inductive proofs. In *Proc. CADE*, pages 111–120, 1988.

[12] Mark R. Cutkosky, Robert S. Engelmore, Richard Fikes, Michael R. Genesereth, Thomas R. Gruber, William S. Mark, Jay M. Tenenbaum, and Jay C. Weber. PACT: An experiment in integrating concurrent engineering systems. *IEEE Computer*, 26(1):28–37, 1993.

[13] Andreas Franke, Stephan Hess, Christoph Jung, Michael Kohlhase, and Volker Sorge. Agent-oriented integration of distributed mathematical services. *Journal of Universal Computer Science*, 5(3):156–187, 1999.

[14] G. Gentzen. Untersuchungen über das logische schliessen. *Mathematische Zeitschrift*, 39(176–210):405–431, 1935.

[15] James Harland and David J. Pym. Resource-distribution via boolean constraints. *ACM Transactions on Computational Logic (TOCL)*, 4(1):568–90, 2003.

[16] James Harland and Michael Winikoff. Agent negotiation as proof search in linear logic. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002)*, pages 1938–939. ACM Press, 2002.

[17] Xiaorong Huang. Reconstructing proofs at the assertion level. In Alan Bundy, editor, *Proc. 12th Conference on Automated Deduction*, pages 738–752. Springer-Verlag, 1994.

[18] Xiaorong Huang, Manfred Kerber, Jörn Richts, and Arthur Sehn. Planning mathematical proofs with methods. *Journal of Information Processing and Cybernetics, EIK*, 30(5-6):277–291, 1994.

[19] Information Agents Group — University of Southern California, Information Sciences Institute. `http://www.isi.edu/info-agents/index.html`.

[20] Mateja Jamnik, Manfred Kerber, and Christoph Benzmüller. Towards learning new methods in proof planning. In Manfred Kerber and Michael Kohlhase, editors, *Proceedings of the Calculemus Symposium 2000*, pages 142–159. A.K.Peters, 2000.

[21] Mateja Jamnik, Manfred Kerber, Martin Pollet, and Christoph Benzmüller. Automatic learning of proof methods in proof planning. *Accepted for The Logic Journal of the IGPL*, 2003.

[22] Michael Kohlhase. OMDOC: Towards an internet standard for the administration, distribution, and teaching of mathematical knowledge. In John A. Campbell and Eugenio Roanes-Lozano, editors, *Proc. Artificial Intelligence and Symbolic Computation*, pages 32–52. Springer-Verlag, 2000.

[23] Michael Kohlhase and Andreas Franke. MBASE: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation*, 32(4), 2001.

[24] Large-scale Interoperation and Composition (LIC) Projects. `http://www-db.stanford.edu/LIC/`.

[25] MATHML. `http://www.w3.org/Math/`.

[26] J. McGuire, D. Kuokka, J. Webber, J. Tenenbaum, T. Gruber, and G. Olsen. SHADE: Technology for knowlege-based collaborative engineering. *Journal of Concurrent Engineering: Application and Research*, 1(3), 1993.

[27] Andreas Meier. *Proof planning with multiple strategies*. PhD thesis, Saarland University, 2003. forthcoming.

[28] Erica Melis. Island planning and refinement. SEKI SR-96-10, University of Saarland, 1996.

[29] Erica Melis, Eric Andrès, Jochen Büdenbender, Adrian Frischauf, George Goguadze, Paul Libbrecht, Martin Pollet, and Carsten Ullrich. ACTIVEMATH: A generic and adaptive web-based learning environment. *Artifical Intelligence in Education*, 12(4), 2001.

[30] Erica Melis and Andreas Meier. Proof planning with multiple strategies. In *Proc. CL-2000*, volume 1861, pages 644–653, 2000.

[31] Erica Melis and Jörg Siekmann. Knowledge-based proof planning. *Artificial Intelligence Journal*, 115(1):65–105, 1999.

[32] OPENMATH. `http://www.openmath.org/`.

[33] Jörg Siekmann, Christoph Benzmüller, Vladimir Brezhnev, Lassaad Cheikhrouhou, Armin Fiedler, Andreas Franke, Helmut Horacek, Michael Kohlhase, Andreas Meier, Erica Melis, Markus Moschner, Immanuel Normann, Martin Pollet, Volker Sorge, Carsten Ullrich, Claus-Peter Wirth, and Jürgen Zimmer. Proof development with OMEGA. In Andrei Voronkov, editor, *Proc. 18th CADE*, number 2392 in LNAI, pages 144–149, Copenhagen, Denmark, 2002. Springer.

[34] Jörg Siekmann, Helmut Horacek, Michael Kohlhase, Christoph Benzmüller, Lassaad Cheikhrouhou, Detlef Fehrer, Armin Fiedler, Stephan Hess, Karsten Konrad, Andreas Meier, Erica Melis, and Volker Sorge. An interactive proof development environment + anticipation = a mathematical assistant? *International Journal of Computing Anticipatory Systems (CASYS)*, 3:101–110, 1999.

[35] Software Agents group, MIT Media Laboratory.
`http://agents.media.mit.edu/index.html`.

[36] The RETSINA Project.
`http://www-2.cs.cmu.edu/ softagents/retsina/retsina.html`.

[37] Laurant Théry. Formal proof authoring: An experiment. In *Proceedings of the Workshop User Interfaces for Theorem Provers (UITP 2003)*, pages 143–160, Rome, Italy, 2003. MMIII ARACNE EDITRICE S.R.L. (ISBN 88-7999-545-6). Also available as: Technical Report No. 189, Institut für Informatik, Albert-Ludwig-Universität, Freiburg.

[38] University of Maryland Information Mediation Project.
`http://www.umiacs.umd.edu/labs/CLIP/im.html`.

[39] Quoc Bao Vo. A task-oriented agent-based mechanism for theorem proving. In *The 2003 IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003)*, pages 275–281. IEEE Computer Society/WIC, 2003.

[40] Quoc Bao Vo, Christoph Benzmüller, and Serge Autexier. An approach to assertion application via generalized resolution. In *International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1343–1344. IJCAI/Morgan Kaufmann, 2003.

[41] Quoc Bao Vo, Christoph Benzmüller, and Serge Autexier. An approach to assertion application via generalized resolution. Seki Report SR-03-01, Saarland University, February 2003.

[42] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.

[43] Gio Wiederhold and Michael R. Genesereth. The conceptual basis for mediation services. *IEEE Expert*, 12(5):38–47, 1997.