Distributed Assertion Retrieval

Christoph Benzmüller, Andreas Meier and Volker Sorge
Fachbereich Informatik
Universität des Saarlandes
D-66041 Saarbrücken, Germany
{chris|ameier|sorge}@ags.uni-sb.de
http://www.ags.uni-sb.de/~{chris|ameier|sorge}

1 Motivation

HUANG has identified the *assertion level* as a well defined abstraction level for natural deduction proofs [5, 6]. Proofs at assertion level are composed of the direct application of assertions, like theorems, axioms, and definitions.

To clarify the notion of assertion application we pick one of HUANG's examples as given in [6]. An assertion application is for instance the application of the *SubsetProperty*

$$\forall S_1 . \forall S_2 . S_1 \subset S_2 \equiv \forall x . x \in S_1 \Rightarrow x \in S_2$$

in the following way:

$$\frac{a \in U \quad U \subset F}{a \in F}\ \mathit{Assertion}(\mathit{SubsetProperty})$$

The direct application of the assertion is thus an abbreviation for a more detailed reasoning process on the calculus level; that is, the explicit derivation of the goal $a \in F$ from the two premises by appropriately instantiating and splitting the *SubsetProperty* assertion.

In the ΩMEGA system [1] assertions are applied using a specialized *Assertion* tactic. Its purpose is to derive a goal from a set of premises with respect to a theorem or axiom. It thus enables the more abstract reasoning with respect to given assumptions at the assertion level. We can depict the assertion tactic as a general inference rule in the following way

$$\frac{\mathit{Prems}}{\mathit{Goal}}\ \mathit{Assertion}(\mathit{Ass})$$

where *Prems* is a list of premises, *Goal* is the conclusion and *Ass* is the assertion that is applied.

Determining possible assertion applications for subsequent subgoals in a proof attempt can easily become a very difficult task and a direct, sequential interleaving of assertion applicability tests with the main theorem proving loop is a rather ineligible option.

Firstly, there might be too many assertions in the database to be checked sequentially in each proof step. This motivates a concurrent mechanism; optimally one with any-time behavior, that allows to continue the proving process regardless of termination of applicability checks for assertions but also to resume those checks if necessary.

Secondly, each applicability check might call for costly, complex, and probably even undecidable algorithms. One problem is that an assertion or a succedent of an assertion might not directly match with a focused open subgoal such that some deduction steps are needed to establish the applicability. For instance, if a mathematical database entails a theorem of the form $\mathbf{A}_1 \wedge \ldots \wedge \mathbf{A}_n \Rightarrow (\mathbf{B} \Leftrightarrow \mathbf{C})$ while the focused subgoal is $\mathbf{C} \Leftrightarrow \mathbf{B}$, then symmetry of $\Leftrightarrow$ is crucial for the applicability of the theorem. Thus, in its most general form the task of determining applicable assertions entails some, more or less restricted, theorem proving as an undecidable side condition. In a higher-order theorem proving environment (e.g., ΩMEGA is based on a higher order natural deduction variant, and thus higher order assertions and proof goals do occur) even matching is already complex and generally undecidable[1]. On the other hand many applicable assertions can already be detected by much simpler algorithms such as first-order matching. This motivates a flexible and parameterizable assertion retrieval mechanism, where simple, efficient algorithms as well as powerful, complex ones can be concurrently employed in a resource adapted manner.

Thirdly, assertion retrieval should not rely on database dependend specifics such as theory names or even assertion names. For instance, in some ΩMEGA proof methods such specific knowledge is employed to precisely search for respective assertions. However, this database dependend knowledge contrasts robustness of assertion retrieval as slight modification of the database may compromise the success of these methods. It also contrasts the possibility of interoperatibility with alternative databases.

In order to meet the above requirements we suggest a technique to separate the search for applicable assertions from the the main proving process and to supersede database dependencies by general search criterions. Our approach, which employs the Ω-ANTS blackboard mechanism [2, 3], serves for both automated (e.g., with ΩMEGA's proof planner) and interactive theorem proving. The idea is to view the applicability tests as little independent and resource bounded processes that run in parallel in the background. This especially enables us to employ powerful and even undecidable algorithms without directly affecting the main theorem proving process in the foreground. Applicable theorems are dynamically signaled to the prover including information which particular subgoal is addressed. Thus, whenever the prover is determining which step to perform next it can take these options into account – this holds for new open subgoals as well as for re-opened, backtracked subgoals in the proof process.

[1]Decidability of full higher-order matching is still an open question.

2 Modeling Assertion Retrieval in Ω-Ants

Ω-Ants [2] is a blackboard architecture that was originally developed to compute applicable inference rules in interactive and automated theorem proving. In this context inference rules (e.g., calculus rule, tactics, or planning methods) are uniformly regarded as sets of premises, conclusions and additional parameters. The elements of these three sets comprise the arguments of an inference rule and have generally certain dependencies amongst each other. In order for an inference rule to be applicable at least some of its arguments have to be instantiated with respect to the given proof context. The task of the Ω-Ants architecture is now to determine the applicability of inference rules by computing instantiations for their arguments. The architecture consists of two layers: The upper layer collects and evaluates data on instantiations of arguments for the given inference rules and thereby compiles a set of applicable rules. This information is computed on the lower layer of the architecture concurrently for each inference rule. The lower layer consists of single societies of agents, one society for each inference rules. These agents are independent processes that cooperatively compute argument instantiations with respect to the current proof state. Ω-Ants provides facilities to define and modify its distributed processes at run-time and also facilities to employ resource reasoning to guide the search.

We currently examine an approach to model assertion retrieval in Ω-Ants, which distributes the applicability checks for assertions to several agents. These agents are arranged in agent societies realizing the following two principles: (1) Each agent society is able to check a certain cluster of assertions. (2) Within an agent society the checks are done in two phases: first a filter agent performs a simple check, then further retrieval agents employ more fine grained, but probably more expensive applicability checks. The cluster of assertions associated with an agent society consists of related assertions applicable to subgoals that share a certain property which is simple to test. This test is performed by the filter agent of the society. The retrieval agents of a society employ then different further applicability checks such as first order matching, higher order matching, or even full higher order theorem proving. This separation of preselection and main check enables the exclusion of many available assertions already by rather simple tests without having to carry out all possible matchings and unifications.

During the proof process, for an open subgoal first the filter agents identify promising theorem clusters. For societies whose filter agent succeeds the retrieval agents become active. If successfull, these agents provide suggestions about assertions that are applicable to the open subgoals. When there is a suggestion about an applicable assertion a further agent, called premise agent, is triggered which tries to identify whether the proof context already contains support lines that can be employed to justify the premises of the assertion. This premise agent is also part of every agent society. With the information computed for the single agent societies Ω-Ants can compile a set of applicable assertions on the top level and suggest them to the prover. Each individual suggestion contains information on the investigated subgoal, an identified applicable assertion, and available support lines that can be used immediately to weaken premises of the assertion.

Note, that the cluster of assertions associated with an agent society is formed dynamically at run-time. Technically, this is realized by equipping the retrieval agents with specifications, about which assertions the agent can process. Then at run-time the retrieval agents first look-up the data-base of assertions and form the clusters of assertions associated with the different agent societies, respectively. Selecting the assertions via specifications enables a more refined selection of assertions and makes this selection independent of explicit references to assertions or a particular data-base. Furthermore, new assertions can be dynamically added and fitted into the existing clusters.

3 Example

Our example is taken from a case study on the proofs of properties of residue classes. In this case study we apply ΩMEGA's proof planner to classify residue class sets over the integers together with given binary operations in terms of their basic algebraic properties. The case study is described in detail in [7]. We concentrate here on how Ω-ANTS determines the applicability of assertions in this context. We consider the first step in the proof of the theorem

$$\mathit{Conc.} \vdash \mathit{Closed}(\mathbb{Z}_5, \lambda x_\bullet \lambda y_\bullet (x \bar{*} y) \bar{+} \bar{3}_5).$$

It states that the given residue class set $\mathbb{Z}_5$ is closed with respect to the operation $\lambda x_\bullet \lambda y_\bullet (x \bar{*} y) \bar{+} \bar{3}_5$. Here $\mathbb{Z}_5$ is the set of all congruence classes modulo 5, i.e., $\{\bar{0}_5, \bar{1}_5, \bar{2}_5, \bar{3}_5, \bar{4}_5\}$. $\bar{*}$ and $\bar{+}$ are the multiplication and addition on residue classes.

Among the theorems we have for the domain of residue classes there are some that are concerned with statements on the closure property. In particular, we have the following six theorems:

$$\begin{array}{ll}
\mathit{ClosedConst}: & \forall n{:}\mathbb{Z}_\bullet \forall c{:}\mathbb{Z}_{n\bullet} \mathit{Closed}(\mathbb{Z}_n, \lambda x_\bullet \lambda y_\bullet c) \\
\mathit{ClosedFV}: & \forall n{:}\mathbb{Z}_\bullet \mathit{Closed}(\mathbb{Z}_n, \lambda x_\bullet \lambda y_\bullet x) \\
\mathit{ClosedSV}: & \forall n{:}\mathbb{Z}_\bullet \mathit{Closed}(\mathbb{Z}_n, \lambda x_\bullet \lambda y_\bullet y) \\
\mathit{ClComp}\bar{+}: & \forall n{:}\mathbb{Z}_\bullet \forall op_{1\bullet} \forall op_{2\bullet} (\mathit{Closed}(\mathbb{Z}_n, op_1) \wedge \mathit{Closed}(\mathbb{Z}_n, op_2)) \Rightarrow \\
 & \qquad \mathit{Closed}(\mathbb{Z}_n, \lambda x_\bullet \lambda y_\bullet (x\ op_1\ y) \bar{+} (x\ op_2\ y)) \\
\mathit{ClComp}\bar{-}: & \forall n{:}\mathbb{Z}_\bullet \forall op_{1\bullet} \forall op_{2\bullet} (\mathit{Closed}(\mathbb{Z}_n, op_1) \wedge \mathit{Closed}(\mathbb{Z}_n, op_2)) \Rightarrow \\
 & \qquad \mathit{Closed}(\mathbb{Z}_n, \lambda x_\bullet \lambda y_\bullet (x\ op_1\ y) \bar{-} (x\ op_2\ y)) \\
\mathit{ClComp}\bar{*}: & \forall n{:}\mathbb{Z}_\bullet \forall op_{1\bullet} \forall op_{2\bullet} (\mathit{Closed}(\mathbb{Z}_n, op_1) \wedge \mathit{Closed}(\mathbb{Z}_n, op_2)) \Rightarrow \\
 & \qquad \mathit{Closed}(\mathbb{Z}_n, \lambda x_\bullet \lambda y_\bullet (x\ op_1\ y) \bar{*} (x\ op_2\ y))
\end{array}$$

The theorems *ClosedConst*, *ClosedFV*, and *ClosedSV* talk about residue class sets with simple operations whereas $\mathit{ClComp}\bar{+}$, $\mathit{ClComp}\bar{-}$, and $\mathit{ClComp}\bar{*}$ are concerned with combinations of complex operations. The difference between the groups of theorems is that the applicability of former can be checked with slightly adapted first order matching whereas for the latter we need higher order matching. For example, when applying the theorem $\mathit{ClComp}\bar{+}$ to our problem at hand the required instantiations are $op_1 \leftarrow \lambda x_\bullet \lambda y_\bullet x \bar{*} y$ and $op_2 \leftarrow \lambda x_\bullet \lambda y_\bullet \bar{3}_5$, which cannot be found by first order matching. However, since we are concerned with only a distinct set of binary operations and their combinations, we can keep things decidable by using a special, decidable algorithm, which matches the

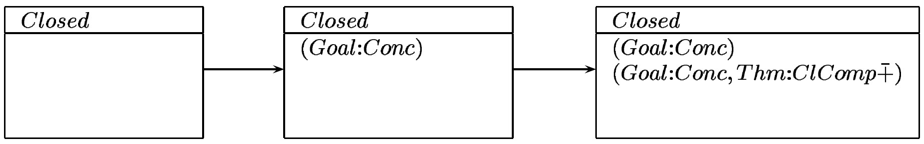
$$
\begin{array}{lcl}
\mathfrak{G} & = & \{\textit{Goal}\text{: } \textit{Goal} \text{ contains the } \textit{Closed} \text{ predicate}\} \\
\mathfrak{F}_1 & = & \{\textit{Thm}\text{: Conclusion matches } \textit{Goal} \text{ with first order matching}\} \\
 & & \left\{\begin{array}{l}\textit{Acquisition}\text{: Conclusion contains } \textit{Closed} \text{ as outermost} \\ \qquad\qquad \text{predicate and a constant operation}\end{array}\right\} \\
\mathfrak{F}_2 & = & \{\textit{Thm}\text{: Conclusion matches } \textit{Goal} \text{ with special algorithm}\} \\
 & & \left\{\begin{array}{l}\textit{Acquisition}\text{: Conclusion contains } \textit{Closed} \text{ as outermost} \\ \qquad\qquad \text{predicate and a binary operation}\end{array}\right\} \\
\mathfrak{S} & = & \{\textit{Prem}\text{: The nodes matching the premises of } \textit{Thm}\}
\end{array}
$$

Figure 1: Agent society for the *Closed* theorem cluster.

statements of the theorems $ClComp\bar{+}$, $ClComp\bar{-}$, and $ClComp\bar{*}$ with nested operations on congruence classes.

In Ω-ANTS we have the agent society as depicted in Figure 1 for the cluster comprising the theorems given above. The filter agent $\mathfrak{G}$ searches for possible conclusions that contain an occurrence of the *Closed* predicate. $\mathfrak{G}$ writes respective suggestions of goals to the blackboard. We then have two retrieval agents, $\mathfrak{F}_1$ and $\mathfrak{F}_2$, that try to match the theorems. $\mathfrak{F}_1$ tries to match the theorems *ClosedConst*, *ClosedFV*, and *ClosedSV* to the formulas suggested by $\mathfrak{G}$ using first order matching. $\mathfrak{F}_2$ uses the special algorithm instead of matching the theorems $ClComp\bar{+}$, $ClComp\bar{-}$, and $ClComp\bar{*}$ conventionally. $\mathfrak{F}_1$ and $\mathfrak{F}_2$ have additional acquisition predicates specifying that the agents can acquire theorems whose conclusions have *Closed* as the outermost predicate. $\mathfrak{F}_1$ furthermore requires that the theorem conclusion contains a simple, constant operation while $\mathfrak{F}_2$ expects a complex operation. The acquisition predicate serves to retrieve appropriate theorems from the knowledge base initially and dynamically at run-time if new theorems are added. For each applicable theorem they detect $\mathfrak{F}_1$ and $\mathfrak{F}_2$ place new extended suggestions on the blackboard. The last agent is the premise agent $\mathfrak{S}$, which has an algorithm to extract the necessary premises from a theorem suggested by $\mathfrak{F}_1$ or $\mathfrak{F}_2$ and, if there are any, tries to find appropriate proof lines containing them.

For our concrete example theorem the information that accumulates on the command blackboard for the *Closed* theorem cluster is as follows:



First $\mathfrak{G}$ detects an occurrence of the *Closed* predicate in the given goal *Conc* and adds an entry suggesting it as instantiation for *Goal* to the blackboard. With this entry the $\mathfrak{F}_1$ and $\mathfrak{F}_2$ start matching their respective theorems to *Conc*. $\mathfrak{F}_2$ is successful with the $ClComp\bar{+}$ theorem and adds the matched theorem as suggestion. Then $\mathfrak{S}$ starts its search; for our example it is looking for premises of the form $Closed(\mathbb{Z}_5, \lambda x.\lambda y.\bar{3}_5)$ and $Closed(\mathbb{Z}_5, \lambda x.\lambda y.x\bar{*}y)$.

4 Outlook

Evaluation: An important aspect of our work will be to evaluate our current modeling of goal directed theorem retrieval in Ω-ANTS and also to relate them to both other possible modeling approaches in Ω-ANTS and standard indexing techniques.

A disadvantage of our current approach is that the predicates for forming clusters as well as some of the agents for matching assertions have to be explicitely specified. This could be avoided if each theorem is directly identified with an Ω-ANTS agent that analyzes the theorem's applicability. However, this approach essentially only distributes the complete testing of all assertions without effectively reducing the number and complexity of the tests. Nevertheless, a thorough comparison between the two different approaches should be made.

Compared with standard indexing techniques our viewpoint of theorem retrieval goes beyond their capabilities. Indexing techniques rely typically on decidable matching or unification contexts. Generally our mechanism aims at the retrieval of theorems modulo more powerful filters as well. Apart from standard matching or unification filters we also employ full theorem proving. Additionally the checks for different theorems are executed in parallel. These processes are controlled and bounded by the resource concepts of Ω-ANTS. Moreover we make use of Ω-ANTS anytime character that allows to apply suggested theorems without waiting for all applicability checks to have finished.

Integration: In the long run we aim to integrate our approach with the services provided by a mathematical database like Mizar [8] or MBASE [4]. MBASE, for example, already supports simple queries based on first order matching to retrieve theorems which are applicable to a subgoal at hand. More powerful query services are planned but it is clear that there will be limits. For instance, to employ full higher order theorem proving within the retrieval of theorems is out of reach for a mathematical knowledge base project. Here is where both approaches can be merged. Simple filter queries, for instance the test of cluster properties, can probably be successfully mapped to database services from within the Ω-ANTS agents. More powerful methods, such as theorem provers, should be employed in Ω-ANTS making use of the distribution and resource mechanisms.

References

[1] C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, E. Melis, A. Meier, W. Schaarschmidt, J. Siekmann, and V. Sorge. ΩMega: Towards a Mathematical Assistant. In *Proceedings of CADE–14*, volume 1249 of *LNAI*, pages 252–255. Springer Verlag, 1997.

[2] C. Benzmüller and V. Sorge. Critical Agents Supporting Interactive Theorem Proving. In *Proceedings of EPIA-99*, volume 1695 of *LNAI*, pages 208–221. Springer Verlag, 1999.

[3] C. Benzmüller and V. Sorge. Ω-ANTS – an open approach at combining interactive and automated theorem proving. In *Proceedings of the Calculemus Symposium 2000*. AK Peters, 2001.

[4] A. Franke and M. Kohlhase. MBase: Representing mathematical Knowledge in a Relational Data Base. In *CALCULEMUS 99, Systems for Integrated Computation and Deduction*, Electronic Notes in Theoretical Computer Science, pages 135–152. Elsevier, 1999.

[5] X. Huang. *Human Oriented Proof Presentation: A Reconstructive Approach.* PhD thesis, Computer Science Department, Universität des Saarlandes, Germany, 1994.

[6] X. Huang. Reconstructing Proofs at the Assertion Level. In *Proceedings of CADE–12*, volume 814 of *LNAI*, pages 738–752. Springer Verlag, 1994.

[7] A. Meier, M. Pollet, and V. Sorge. Classifying Isomorphic Residue Classes. In *Proceedings of EuroCAST 2001*, volume 2178 of *LNCS*. Springer Verlag, 2001.

[8] A. Trybulec and H. Blair. Computer assisted reasoning with MIZAR. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 26–28. Morgan Kaufmann, 1985.