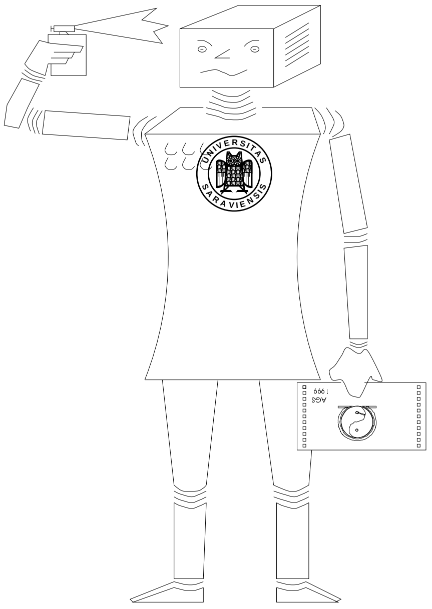
UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY
WWW: http://www.ags.uni-sb.de/

SEKI Report
ISSN 1437-4447



An Approach to Assertion Application via Generalized Resolution

Quoc Bao Vo, Christoph Benzmüller and Serge Autexier

SEKI Report SR-03-01

An Approach to Assertion Application via Generalized Resolution

Quoc Bao Vo[1], Christoph Benzmüller[1], and Serge Autexier[2]
[1]FR Informatik, Universität des Saarlandes, 66041 Saarbrücken, Germany
{bao|chris}@ags.uni-sb.de
[2]DFKI GmbH, 66123 Saarbrücken, Germany
autexier@dfki.de

Abstract

In this paper we address assertion retrieval and application in theorem proving systems or proof planning systems for classical first-order logic. Due to Huang the notion of assertion comprises mathematical knowledge from a mathematical knowledge base KB such as definitions, theorems, and axioms. We propose a distributed mediator module between KB and a theorem proving system S which is independent of the particular proof representation format of S and which applies generalized resolution in order to analyze the logical consequences of arbitrary assertions for a proof context at hand.

Our approach is applicable without any extension also to the assumptions that are dynamically created during a proof search process. It therefore realizes a crucial first step towards full automation of assertion level reasoning. We discuss the benefits and connection of our approach to proof planning and motivate an application in a project aiming at a tutorial dialogue system for mathematics.

1 Introduction

Mathematical assistant systems provide users with tools that support them in constructing proofs. Thereby it is an important issue that the constructed proof fragments are presented in a format that is easy to understand for human users. That is the central problem of proof presentation and explanation. On the other hand, sophisticated proofs for real mathematical problems have now been constructed in mathematical assistant systems (either automatically or interactively) at different levels of granularity[1]. In some of the employed approaches, 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. [9] and [16, 20, 19] and the references therein).

The assertion level introduced by Huang [15] is argued to be just the right level for machine generated proofs to be transformed into before being presented to (human) users. In this paper we argue further that the assertion level can also serve as one of those levels of

[1]There is a ongoing debate whether *level of abstraction*, *level of detail*, or *level of granularity* is the most appropriate notion to use. In this paper we will use the term *level of granularity*, since we address abstraction on logical derivations and proof operators and not, for instance, abstraction of the object representation language.

granularity where knowledge-based proof planning should be based upon. This viewpoint is also taken in [3], which provides a proof representation framework subsuming the assertion level. However, proof automation of the framework is still a challenge.

Huang [15] achieves proofs at the assertion level by reconstructing machine generated proofs constructed by theorem provers or proof planners. This process basically involves clustering several proof steps of the input proof into segments. From the outside, each segment can be viewed as one (assertion level) proof step. While this partly solves the problem of proof presentation as each assertion level proof step can be mapped naturally to single statements of the natural language presentation of the proof, it has not been investigated yet how the proof planning process may directly benefit from such a representation level. We are convinced that by planning directly on the assertion level it will be possible to overcome at least some of the identified limitations and problems of proof planning as discussed in [5, 10] — in particular, those, that are caused by an unfortunate intertwining of proof planning and calculus level theorem proving. The perspective we therefore motivate in this paper is to consider the assertion level as a well chosen borderline between proof planning and machine oriented methods. Determining the logical consequences of assertions in a proof context is the task of machine oriented methods (in our case generalized resolution). The tasks on top of this level — for instance, an domain dependent containment of the initial assertions to be considered, the heuristic selection of the most promising among the computed logical consequences, the introduction, constraining and handling of *meta-variables*, etc. — belong to the scope of domain specific proof planning.

In summary, instead of reconstructing natural deduction (ND) proofs to obtain assertion level proofs as suggested by Huang, we propose to directly plan for proofs at the assertion level. This should improve the quality of the resulted proof plans and also facilitate better user interaction.

The development of our ideas revolves around the mathematical assistant system OMEGA [25] and the current initiative in this project to rebuild the system on top of the proof representation framework in [3]. We furthermore employ OMEGA's agent-based search mechanism OANTS [7] for a distributed modeling of our framework and we motivate an application of the approach in a project aiming at a tutorial dialogue system for mathematics.

2 Proof planning at the assertion level

2.0.1 A brief review on the assertion level

This part is reproduced from [15] for completeness. Consider the problem of presenting machine generated proofs. Assume that the target formalism is natural deduction which is a common practice for many existing proof transformation systems. In contrast to proofs found in mathematical textbooks, proofs constructed by these 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 ND proofs that allow atomic justifications at a higher level of abstraction. Huang then went on to introduce the following three levels of justifications:

- *Logic level* justifications are simply verbalizations of the ND inference rules, such as the rule of Modus Ponens.

- *Assertion level* justifications account for derivations in terms of the application of an axiom, a definition or a theorem (collectively called an assertion). 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".

- *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.

2.0.2 Assertion level and proof planning

Scrutinizing the above example for the assertion level justifications, it seems just the right level for proof plans to be carried out. Rather than having to work around with ND inference rules, the proof planner simply goes straight to the intended conclusion, *viz.* e is a member of B, from the given premises, *viz.* e is a member of A and $A \subseteq B$, and the assertion, *viz.* the definition of subset. This would in the first place save the proof planner from exploding its own search space. Furthermore, it should fit better to the structural mechanisms of other high level proof methods such as analogy, induction, diagonalization, etc. and in particular island proof planning, cf. [18]. It should also fit well to the framework of *incremental proof planning* proposed by Gerberding and Pientka [14] in which assertion application can be considered as a special form of meta-rules.

Now let's consider an assertion $\mathcal{A}$. As pointed out by Huang, 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 our running example:

$$\forall S_1, S_2 : Set.(S_1 \subseteq S_2 \Leftrightarrow \forall x : Element.(x \in S_1 \Rightarrow x \in S_2))$$

This assertion allows us to derive: (1) $a \in V$ from $a \in U$ and $U \subseteq V$, (2) $U \nsubseteq V$ from $a \in U$ and $a \notin V$, (3) $\forall x : Element.(x \in U \Rightarrow x \in V)$ from $U \subseteq V$, (4) etc.

A theorem prover/proof planner 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.

Huang [15] overcomes this problem by introducing different inference rules to allow all possible applications of such an assertion. These inference rules are called *assertion level inference rules* or *macro-rules* by their nature of being "pseudo inference rules" which actually stand for sequences of more elementary inference rules, called *compound proof segments.* While the more elementary rules (of ND) are chunked into macro-rules (at the assertion level), this in turn potentially introduces an exploded set of inference rules. Therefore, it may not work too well for a proof search mechanism that is required to be efficient. For instance, for the above assertion from our running example, at least seven (7) different macro-rules can be generated (cf. [15].)

2.0.3 Formalization of the problem

We take as the starting point for our approach the proof development environment OMEGA [25] whose core consists of a proof planner together with a hierarchical plan data structure ($\mathcal{PDS}$). The proof format in OMEGA is based on the natural deduction calculus. A linearized version of ND proofs as introduced by Andrews [1] is employed. In this formalism [16], an ND proof is a sequence of proof lines , each of them is of the form:

`Label` $\Delta \vdash$ *Derived-formula* (`Rule` *premiss-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 *premiss-lines*. `Rule` and *premiss-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 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. 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.

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 OMEGA, a proof planning process starts with a task, a data structure designed to encapsulate a complete (sub-)problem. Formally, a *task* is a pair $\langle Sp_{L_{open}}, L_{open}\rangle$ consisting of an open line of the $\mathcal{PDS}$, L_{open}, and a set of lines from the $\mathcal{PDS}$, $Sp_{L_{open}}$. L_{open} is called the *task line* whose formula is called *task formula*. Members of $Sp_{L_{open}}$ 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 OMEGA 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 to the current proof situation to (i) either obtain further closed lines serving as intermediate steps for solving one of the tasks; (ii) or reduce a goal task (on some open line) to some subtasks which can be resolved by further proof steps.

It now boils down to the question of how tasks and assertions are to be handled without having to generate all the possible macro-rules as suggested by Huang. And we, of course don't want to sacrifice any possible outcome from applications of assertions.

We follow a representation framework employed by Wallen [27] and Autexier [3] using the concept of polarities and uniform notation (cf. [12, 13, 27]) to represent the formulae the proof planner has to deal with.

2.0.4 Signed formulae:

Each formula is assigned a polarity which is either positive ($\oplus$) or negative ($\ominus$). A set of formulae Γ is assigned a polarity p iff each member of Γ is assigned p.

If a deduction line $\Delta \vdash \varphi$ is marked closed (in a $\mathcal{PDS}$) then Δ is assigned $\oplus$ and φ $\ominus$. If $\Delta \vdash \varphi$ is marked open then Δ is assigned $\ominus$ and φ $\oplus$.

Intuitively, a signed formula $\varphi^{\oplus}$ is something the prover wishes to prove while $\varphi^{\ominus}$ is something given.

NOTATIONS:

1. $form(\varphi^p) \overset{\text{def}}{=} \varphi$ and $polarity(\varphi^p) \overset{\text{def}}{=} p$. We will also employ the following operator: $- : \{\ominus, \oplus\} \to \{\ominus, \oplus\}$ such that $-\ominus = \oplus$ and $-\oplus = \ominus$.

α	α_1	α_2
$(\varphi \vee \psi)^{\oplus}$	$\varphi^{\oplus}$	$\psi^{\oplus}$
$(\varphi \Rightarrow \psi)^{\oplus}$	$\varphi^{\ominus}$	$\psi^{\oplus}$
$(\varphi \wedge \psi)^{\ominus}$	$\varphi^{\ominus}$	$\psi^{\ominus}$
$(\neg\varphi)^{\oplus}$	$\varphi^{\ominus}$	$-$
$(\neg\varphi)^{\ominus}$	$\varphi^{\oplus}$	$-$

β	β_1	β_2
$(\varphi \wedge \psi)^{\oplus}$	$\varphi^{\oplus}$	$\psi^{\oplus}$
$(\varphi \vee \psi)^{\ominus}$	$\varphi^{\ominus}$	$\psi^{\ominus}$
$(\varphi \Rightarrow \psi)^{\ominus}$	$\varphi^{\oplus}$	$\psi^{\ominus}$

(γ, x)	$\gamma_0(x)$
$(\forall_x\varphi)^{\ominus}$	$\varphi^{\ominus}$
$(\exists_x\varphi)^{\oplus}$	$\varphi^{\oplus}$

(δ, x)	$\delta_0(sko)^{(*)}$
$(\forall_x\varphi)^{\oplus}$	$\varphi[sko/x]^{\oplus}$
$(\exists_x\varphi)^{\ominus}$	$\varphi[sko/x]^{\ominus}$

$(*) sko \in D$ is a *Skolem term* generated by Skolemization.

Figure 1: Analyzing signed formulae

2. we use the special signed formulae $(\Box)^{\ominus}$ and $(\Box)^{\oplus}$ which are essentially equivalent to $(\bot)^{\ominus}$ and $(\top)^{\oplus}$, respectively. They both represent a refutation (i.e., a contradiction at the succedent of the sequent).

Signed formula trees: Using the rules from Fig. 1, a signed formula can be represented in the form of a tree whose sub-trees represent its signed sub-formula, these trees are called *signed formula trees*, or *SFT*s. In such a tree, each node has a natural interpretation as a signed formula, and we shall speak of nodes and signed formulae as if they were one and the same (e.g., a "node" $(\varphi \vee \psi)^{\oplus}$ is one labelled with $(\varphi \vee \psi)^{\oplus}$ and having arcs to nodes $\varphi^{\oplus}$ and $\psi^{\oplus}$.) A more economic representation would be to allow structure sharing among sub-formulae. This technique is well known in unification theory as *term dags* and facilitates efficient algorithms (cf. [4]). However, there is less structure sharing in formulae (in comparison to terms), therefore we don't allow any kind of sharing of sub-formulae to retain clarity.

Example:[2]

The following is the signed formula tree of the formula $\mathcal{A}$ given as $\forall_{S_1,S_2}(S_1 \subseteq S_2 \Leftrightarrow \forall_x(x \in S_1 \Rightarrow x \in S_2))$. Thereby we assume that assumptions are initially assigned with the polarity $\ominus$. (In our tree F abbreviates the formula $\forall_x(x \in S_1 \Rightarrow x \in S_2)$).

[2]We choose to suppress all references to sorts for the sake of readability.

$$\begin{tikzcd}
 & & (\forall_{S_1,S_2}(S_1 \subseteq S_2 \Leftrightarrow \forall_x(x \in S_1 \Rightarrow x \in S_2)))^{\ominus}_{(\gamma,S_1,S_2)} \arrow[d] & & \\
 & & (S_1 \subseteq S_2 \Leftrightarrow F)^{\ominus}_{\alpha} \arrow[dl] \arrow[dr] & & \\
 & (S_1 \subseteq S_{\Rightarrow}F)^{\ominus}_{\beta} \arrow[dl] \arrow[d] & & (F \Rightarrow S_1 \subseteq S_2)^{\ominus}_{\beta} \arrow[d] \arrow[dr] & \\
(S_1 \subseteq S_2)^{\oplus} & (F)^{\ominus}_{(\gamma,x)} \arrow[d] & & (F)^{\oplus}_{(\delta,x)} \arrow[d] & (S_1 \subseteq S_2)^{\ominus} \\
 & (x \in S_1 \Rightarrow x \in S_2)^{\ominus}_{\beta} \arrow[dl] \arrow[d] & & (sko(S_1,S_2) \in S_1 \Rightarrow sko(S_1,S_2) \in S_2)^{\oplus}_{\alpha} \arrow[d] \arrow[dr] & \\
(x \in S_1)^{\oplus} & (x \in S_2)^{\ominus} & & (sko(S_1,S_2) \in S_1)^{\ominus} & (sko(S_1,S_2) \in S_2)^{\oplus}
\end{tikzcd}$$

Now we can go back to our problem. Assume that a task $\langle Sp_{L_{open}}, L_{open}\rangle$ together with an assertion $\mathcal{A}$ are currently under consideration. As the goal is to come up with a desirable conclusion derivable from the assertion $\mathcal{A}$ (relative to $\langle Sp_{L_{open}}, L_{open}\rangle$), the following computational process revolves around the signed formula tree of $\mathcal{A}$, called $\mathcal{T}_{\mathcal{A}}$. Using unification-based algorithms we unify the terms at the leaf nodes of the signed formula trees other than $\mathcal{T}_{\mathcal{A}}$ with the terms at the leaf nodes of $\mathcal{T}_{\mathcal{A}}$.

Secondly, we will need the following notation to talk about the interrelations between formulae: Given a signed formula φ^p,

- if an α-rule is applicable to φ^p then its children are said to be *directly α-related*, e.g. the two sub-formulae $\varphi^{\oplus}$ and $\psi^{\oplus}$ of $(\varphi \vee \psi)^{\oplus}$ are α-related;

- if a β-rule is applicable to φ^p then its children are said to be *directly β-related*, e.g. the two sub-formulae $\varphi^{\oplus}$ and $\psi^{\oplus}$ of $(\varphi \wedge \psi)^{\oplus}$ are β-related;

A signed formula φ is α (resp. β) *-related* to ψ if and only if either φ or one of its ancestors is directly α (resp. β) -related to ψ.[3]

2.0.5 s-unifiable and complementary:

Two nodes φ^p and ψ^q are *s-unifiable* iff $p \neq q$ and φ and ψ are unifiable. φ^p and φ^q are *complementary* iff $p \neq q$.

NOTATION: We will liberally use the notions applicable to logical formulae and apply them to signed formulae by absorbing them through the polarities of signed formulae. E.g., given a signed formula τ and a substitution θ, we write $\tau\theta$ to denote the signed formula $(form(\tau)\theta)^{polarity(\tau)}$.

We also observe that it is possible to reconstruct formulae from an SFT in such a way that the semantics is preserved. And because all information is conveyed from a node to its

[3]Note that the notions of α and β-related formulae are more general than the notions of conjunction and disjunction as the α (resp. β)-relations can refer to both conjunctions and disjunctions depending on whether they occur in the antecedent or the succedent of a sequent.

children, it is possible to reconstruct a formula from the leaf nodes of its SFT together with the α and β -relations between the subtrees without having to trace back through all of the internal nodes.

Now we come up with the algorithm for manipulating SFTs in order to obtain desirable conclusions from an assertion (in relation to other given assumptions or open goal tasks). The idea generalizes Robinson's resolution principle [23] and in particular reminds to path resolution [21] and the connection graph principle [17, 2, 8]. In much the same way that an arbitrary formula can be reduced to the normal form which contains a set of clauses, i.e. disjunctions of literals, we observe that a given signed formula tree can be reduced to a set of SFTs whose members don't have any α-related pair of sub-formulae.

As with resolution, the idea is to replace a literal in a clause (correspondingly, a leaf node of an SFT) by the information entailed by it that comes from another clause (correspondingly, another SFT). However, unlike resolution theorem proving formalism where the focus is on a refutation procedure in order to establish the unsatisfiability of a set of formulae, our focus of attention is on the derivation of new formulae (be it new assumptions or new subgoals) from the existing ones. Therefore, we won't transform every signed formula (or equivalently, SFT) to the normal form which contains only β-related subtrees as in resolution theorem proving, but rather we delay that until as late as possible. The idea is: instead of normalizing all formulae to the normal form, we just need to take out from the formulae to be resolved the *relevant parts* when resolution is carried out. For instance, consider a set of formulae $S = \{(A \wedge B) \vee (C \wedge D), \neg A \vee E\}$, we can just take out the relevant formula $A \vee (C \wedge D)$ from $(A \wedge B) \vee (C \wedge D)$ and resolve it using $\neg A \vee E$. This result in the following new set of formulae: $S' = \{(A \wedge B) \vee (C \wedge D), \neg A \vee E, (C \wedge D) \vee E\}$.

Now we can go back to the more sophisticated representation of SFTs. From the above example with resolution, it is important to know how the so-called relevant parts can be computed. While it is not so clear how that can be done with logical formulae, it is interesting that SFTs contain structures supporting such operations: Instead of resolving complementary pairs of literals from clauses, we resolve s-unifiable pairs of leaf nodes (or, atomic signed formulae) from SFTs. We therefore term our inference principle *generalized resolution*, or *GR*.

We observe that in addition to the leaf nodes of SFTs on which we perform GR, the structure of SFTs, i.e. the internal nodes and links between nodes, also plays an important part in GR inferences.[4]

2.0.6 The algorithm:

The following algorithm takes as input two SFTs τ_1 and τ_2 with τ_1 being updated by the information from τ_2 by performing resolution on the two leaf nodes η_1 and η_2 from τ_1 and τ_2, resp., under the substitution θ.

```
*GR*($\tau_1$: SFT, $\eta_1$: leaf_node; $\tau_2$: SFT, $\eta_2$: leaf_node; $\theta$: substitution)[5]
**begin**
**If** $\eta_1\theta$ and $\eta_2\theta$ are complementary **then**
    1. Instantiate all variables in $\tau_1$ and $\tau_2$ with the substitution $\theta$;
```

[4]Given a pair of s-unifiable leaf nodes φ^p and ψ^{-p} on two SFTs τ_1 and τ_2, respectively: the results of resolving φ^p on τ_1 by ψ^{-p} from τ_2 and resolving ψ^{-p} on τ_2 by φ^p from τ_1, in general lead to two syntactically different, albeit logically equivalent, SFTs.

[5]The correctness of this algorithm and other properties are established in [26].

```
    2. For $i = 1, 2$, prune all $\alpha$-related subtrees wrt. $\eta_i$ on $\tau_i$;
    3. Replace the leaf node $\eta_2$ in $\tau_2$ by $(\Box)^{polarity(\eta_2)}$;
    4. Replace the formula at each ancestor of $\eta_2$ in $\tau_2$ with a question mark (?) while retaining the
       polarity and the rule originally applied there;
    5. Reconstruct the formulae at the internal nodes of $\tau_2$ currently marked (?) using the formula(e)
       from their child(ren) and the polarity/rule at those nodes;
    6. Replace the formula at each ancestor of $\eta_1$ in $\tau_1$ with a question mark (?) while retaining the
       polarity and the rule originally applied there;
    7. If $polarity(\tau_2) = polarity(\eta_1)$ then: Replace the leaf node $\eta_1$ on $\tau_1$ by $\tau_2$
       else: Replace the leaf node $\eta_1$ on $\tau_1$ by $\neg form(\tau_2)^{polarity(\eta_1)}$;
    8. Reconstruct the formulae at the internal nodes of $\tau_1$ currently marked (?) using the formula(e)
       from their child(ren) and the polarity/rule at those nodes;
    9. Return the new $\tau_1$;
else
    { Do nothing }
end;
```

All the above operations should be clear except the resconstruction of the SFTs. Consider an internal node $\tau = (\varphi)^p_r$ of an SFT. After a subtree of τ is modified during the execution of $GR()$, the formula at τ, *viz.* φ, is replaced by a question mark (?). Once all children of τ have been reconstructed, we can proceed to reconstruct the formula at τ:

1. Special treatment is due when at least one of τ's children is of the form: $(\Box)^q$:
   (a) Case $r = \beta$: let v be the other child of τ. If *polarity*(τ) =*polarityv* then τ and v are collapsed into one node; otherwise, (?) is replaced by $\neg form(v)$.
   (b) otherwise: the question mark (?) is replaced by $\Box$.
2. Case $r = \alpha$:
   - Case $\alpha_2 = nil$: the question mark (?) is replaced by $\neg form(\alpha_1)$.
   - Case $\alpha_2 \neq nil$: the question mark (?) is replaced by $form(\alpha_1) \wedge form(\alpha_2)$ if $p = \ominus$, and by $form(\alpha_1) \vee form(\alpha_2)$, otherwise.
3. Case $r = \beta$: similar to the second case above and according to the table for β-rule.
4. Case $r = (\gamma, x)$: Attention must be taken to avoid clash of variables. If x is a bound variable in $form(\gamma_0(x))$ then it must be renamed to a variable name that does not clash with any other variable in $form(\gamma_0(x))$. Then, (?) is replaced by $\forall_x form(\gamma_0(x))$ if $p = \ominus$, and by $\exists_x form(\gamma_0(x))$, otherwise.
5. Case $r = (\delta, x)$: the question mark (?) is replaced by $form(\delta_0(\tau, sko))$.

Now we are in the position to apply GR on the assertion $\mathcal{A}$ and all the relevant formulae coming from an agenda $\mathcal{D}$. We are fortunate not having to perform proof search, we only have to find out derivable information (which can be new assumptions, i.e. closed lines, or new subgoals, i.e. open lines) from $\mathcal{A}$ using the information provided in $\mathcal{D}$. That is, we have to resolve the leaf nodes of $\mathcal{T_A}$ by the s-unifiable leaf nodes on other SFTs. To simplify the presentation, we assume that our algorithm takes as input the SFT of $\mathcal{A}$, *viz.* $\mathcal{T_A}$, and a set of SFTs representing the current reasoning context $\mathcal{RC}$ so that we don't have to worry about other details such as closed/open lines, assumptions, goals, etc. The algorithm also employs the on-the-fly *skolemization* whenever generalized resolution is performed.

```
Assertion Application($\mathcal{T}_A$: SFT; $\mathcal{RC}$: set_of_SFTs)
begin
  1. $All\_results \leftarrow \emptyset$;
  2. $Processed \leftarrow \emptyset$;
  3. For each leaf node $s$ of $\mathcal{T}_A$ do:
       While there is a leaf node $t$ of an SFT $\tau \in \mathcal{RC} \cup \{\mathcal{T}_A\}$ such that $s$ and $t$ are s-unifiable
       and $(s, t, \theta) \notin Processed$ do:
       (a) $C_{\mathcal{T}_A} \leftarrow \mathcal{T}_A$; { Create a working copy of $\mathcal{T}_A$ }
       (b) $p \leftarrow polarity(root(\tau))$;
       (c) Repeat:
              i. $current\_resolved \leftarrow s$;
             ii. $current\_resolver \leftarrow t$;
            iii. $\theta \leftarrow mgu(form(s), form(t))$;
             iv. $Processed \leftarrow Processed \cup \{(s, t, \sigma)\}$;
              v. Repeat:
                   A. $\tau \leftarrow GR(C_{\mathcal{T}_A}$, $current\_resolved$, $\tau$, $current\_resolver$, $\theta)$;
                   B. If there is a s-unifiable pair of leaf nodes $s'$ and $t'$ from $C_{\mathcal{T}_A}$ and $\tau$, respectively,
                      then
                      – $current\_resolved \leftarrow s'$;
                      – $current\_resolver \leftarrow t'$;
                      – $\sigma \leftarrow mgu(form(s'), form(t'))$;
                      – $\theta \leftarrow \theta\sigma$;
                      – $Processed \leftarrow Processed \cup \{(s', t', \sigma)\}$;
                      else
                      – $current\_resolved \leftarrow nil$;
                      – $current\_resolver \leftarrow nil$;
                 until $current\_resolved = nil$;
             vi. If $p = \oplus$ and $polarity(root(\tau)) = \ominus$ then
                   $C_{\mathcal{T}_A} \leftarrow Negate(\tau)$;
                 else
                   $C_{\mathcal{T}_A} \leftarrow \tau$;
           until there is no any further s-unifiable pair of leaf nodes from $C_{\mathcal{T}_A}$ and any SFT in
           $\mathcal{RC} \cup \{\mathcal{T}_A\}$;
       (d) $All\_results \leftarrow All\_results \cup \{form(root(C_{\mathcal{T}_A}))\}$;
  4. Return $All\_results$;
end;
```

In the above algorithm, the function $Negate(\tau)$ adds a parent to the root node of the SFT τ. The formula and the polarity at the root node of the new tree is the negations of the formula and the polarity at the root node of τ. We have to carry out that operation because we do not want the outcome of applying an assertion to an open line, i.e. a goal, to be a closed line. For instance, applying the assertion $A \Rightarrow B$ to a goal $B^{\oplus}$ should result in a new subgoal $A^{\oplus}$ rather than a conclusion $(\neg A)^{\ominus}$.

Example (Forward Reasoning):

Let the following deduction lines be marked closed in the input task: (1) $\vdash A \subseteq B$ and (2) $\vdash e \in A$.

By applying the first tree $A \subseteq B^{\ominus}$ to $C_{\mathcal{T}_{\mathcal{A}}}$ which is the same as $\mathcal{T}_{\mathcal{A}}$ for the moment, we obtain the updated SFT for $C_{\mathcal{T}_{\mathcal{A}}}$ displayed to the right (under the substitution $\theta = \{S_1 \mapsto A, S_2 \mapsto B\}$).

$$\begin{tikzcd}[column sep=small]
 & (\forall_x(x \in A \Rightarrow x \in B))^{\ominus}_{\gamma} \arrow[d] & \\
 & (x \in A \Rightarrow x \in B)^{\ominus}_{\beta} \arrow[dl] \arrow[dr] & \\
(x \in A)^{\oplus} & & (x \in B)^{\ominus}
\end{tikzcd}$$

By applying the second tree $e \in A^{\ominus}$ to the resulting $C_{\mathcal{T}_{\mathcal{A}}}$, we obtain as updated SFT for $C_{\mathcal{T}_{\mathcal{A}}}$ (under the substitution $\theta_1 = \{x \mapsto e\}$) the tree $e \in B^{\ominus}$ which is translated back to the OMEGA-proof format as a (closed) deduction line (3) $\vdash e \in B$. This is also is also the desirable logical consequence of assertion $\mathcal{A}$ for lines (1) and (2).

Example (Resolve internal nodes):

To see the necessity of the inner iteration, we re-run our example with another task consisting of the following closed deduction line: (4) $\vdash \forall_x(x \in A \Rightarrow x \in B)$. This assumption results in the SFT presented to the right. Applying this tree to $\mathcal{T}_{\mathcal{A}}$ and assuming that we unify first the two nodes $(sko(S_1, S_2) \in S_1)^{\ominus}$ (on $\mathcal{T}_{\mathcal{A}}$) and $(x \in A)^{\oplus}$ on the above SFT we optain the updated $C_{\mathcal{T}_{\mathcal{A}}}$ below (under the substitution $\theta = \{x \mapsto sko(S_1, S_2), S_1 \mapsto A\}$).

$$\begin{tikzcd}[column sep=small]
 & (\forall_x(x \in A \Rightarrow x \in B))^{\ominus}_{\gamma} \arrow[d] & \\
 & (x \in A \Rightarrow x \in B)^{\ominus}_{\beta} \arrow[dl] \arrow[dr] & \\
(x \in A)^{\oplus} & & (x \in B)^{\ominus}
\end{tikzcd}$$

$$\begin{tikzcd}[column sep=small]
 & (\neg sko(S_1, S_2) \in B \Rightarrow A \subseteq S_2)^{\ominus}_{\beta} \arrow[dl] \arrow[dr] & \\
(\neg sko(S_1, S_2) \in B)^{\oplus}_{\alpha} \arrow[d] & & (A \subseteq S_2)^{\ominus} \\
(sko(S_1, S_2) \in B)^{\ominus} & &
\end{tikzcd}$$

Notice how the α-related branches on the SFT and the nodes labelled with the α and γ and δ-rules are pruned away by $GR()$.

Now we try to unify a leaf node on this tree to some leaf node on $\mathcal{T}_{\mathcal{A}}$ again which is achievable through the unification on the two nodes $(sko(S_1, S_2) \in B)^{\ominus}$ (on the tree to the left) and $(sko(S_1, S_2) \in S_2)^{\oplus}$ (on $\mathcal{T}_{\mathcal{A}}$).

The updated $C_{\mathcal{T}_{\mathcal{A}}}$ now becomes the tree displayed to the right (under the substitution $\theta_1 = \{S_2 \mapsto B\}$)[6].

$$\begin{tikzcd}[column sep=small]
 & (\neg A \subseteq B \Rightarrow A \subseteq B)^{\ominus}_{\beta} \arrow[dl] \arrow[dr] & \\
(\neg A \subseteq B)^{\oplus}_{\alpha} \arrow[d] & & (A \subseteq B)^{\ominus} \\
(A \subseteq B)^{\ominus} & &
\end{tikzcd}$$

This tree translates into the desirable logical consequence of assertion $\mathcal{A}$ for line (4) and the closed line (5) $\vdash A \subseteq B$ is obtained.

2.0.7 Example (Sidewards Reasoning):

We now consider an example in which a goal line (i.e. an open line from the input task) and a closed line is applied. Assume that our current task consists of an open line (6) $\vdash \epsilon \in B$ and a closed line (7) $\vdash A \subseteq B$, which correspond to the two singleton trees $(\epsilon \in B)^{\oplus}$ and $(A \subseteq B)^{\ominus}$ respectively.

[6]Note that the root node of the computed SFT is: $(A \subseteq B)^{\ominus}$ has a different polarity from that of the node it is going to replace, *viz.* $(sko(S_1, S_2) \in S_2)^{\oplus}$, on the SFT $\mathcal{T}_{\mathcal{A}}$. Therefore, we replace that node with $(\neg A \subseteq B)^{\oplus}$ instead.

Applying the former to $\mathcal{T}_{\mathcal{A}}$ by unifying it to the leaf node $(x \in S_2)^{\ominus}$, we obtain the SFT to the right.

$$\begin{tikzcd}
& (\forall_{S_1}(S_1 \subseteq B \Rightarrow \neg\epsilon \in S_1))^{\ominus}_{\gamma} \arrow[d] & \\
& (S_1 \subseteq B \Rightarrow \neg\epsilon \in S_1)^{\ominus}_{\beta} \arrow[dl] \arrow[dr] & \\
(S_1 \subseteq B)^{\oplus} & & (\neg\epsilon \in S_1)^{\ominus}_{\alpha} \arrow[d] \\
& & (\epsilon \in S_1)^{\oplus}
\end{tikzcd}$$

Then we can apply $(A \subseteq B)^{\ominus}$ to this SFT by unifying it to the node $(S_1 \subseteq B)^{\oplus}$, the resulting tree is:

$$\begin{tikzcd}
(\neg\epsilon \in A)^{\ominus}_{\alpha} \arrow[d] \\
(\epsilon \in A)^{\oplus}
\end{tikzcd}$$

This corresponds to a new open line (8) $\vdash \epsilon \in A$, which is the desirable consequence of assertion $\mathcal{A}$ for lines (6) and (7).

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 OANTS approach [7]. This agent based formalism is the driving force behind a distributed proof search approach in OMEGA. It enables the distribution of proof search among groups of reasoning agents.

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 mathmetical knowledge base KB. TP is currently focusing a proof task $\mathcal{T} = \langle Sp_{L_{open}}, L_{open} \rangle$ and candidate assertions $\{\mathcal{B}_i\}$ are determined in KB and handed 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 this paper. Note that this algorithm only depends on the SFT of the focused assertion and a further set of SFTs for the proof context, and both are specified as parameters of *AssertionApplication*. Each assertion agent instance $AG_{\mathcal{B}_i}$ computes and suggest the logical consequences of $\mathcal{B}_i$ in proof context $\mathcal{T}$ to our module M which passes them further to TP.

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 (recall the remark above about the number of possible ND inference rules associated with the assertion of our running example) to be handled in practice. For instance, going back to our running example again, let our current task consist of the open 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 SFT displayed next.

$$\begin{tikzcd}
& (\forall_{S_1}(S_1 \subseteq U \Rightarrow \neg e \in S_1))^{\ominus}_{\gamma} \arrow[d] & \\
& (S_1 \subseteq U \Rightarrow \neg e \in S_1)^{\ominus}_{\beta} \arrow[dl] \arrow[dr] & \\
(S_1 \subseteq U)^{\oplus} & & (\neg e \in S_1)^{\ominus}_{\alpha} \arrow[d] \\
& & (e \in S_1)^{\oplus}
\end{tikzcd}$$

As the root node is non-empty and the input trees include an open line, we have to reproduce the root node to have polarity $\oplus$, which is $(\neg\forall_{S_1}(S_1 \subseteq U \Rightarrow \neg e \in S_1))^{\oplus}$. This corresponds to a new open line: $\vdash \exists_{S_1}(S_1 \subseteq U \wedge e \in S_1)$ which is, even though logically correct, not a very useful subgoal to be pursued.[7]

We sum up the above argument by claiming that restricted application of assertion is necessary. One possible and simple

[7]In fact, Huang's [15] framework fails to produce the above inference step. This poses a serious question about the completeness of his approach.

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. This simple kind of specific constraints are of course trivially to realize. A simple but useful heuristic to prove the membership of an object wrt. a set S using the subset definition is that the task include a closed line stating that S is a superset of some other set. It is this restricted version of assertion application that has been implemented in the agent-based mechanism OANTS in the OMEGA system.

Proof planning, however, has developed more sophisticated ways to guide and constrain possible the possible instantions and applications of assertions. The investigation how some of these techniques can optimally be employed on top of our assertion application module M are further work.

4 An application: The DIALOG project

Our approach to assertion application is motivated by an application in the DIALOG project as part of the Collaborative Research Center on *Ressource 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 is supported by a human oriented mathematical proof development evironment and the OMEGA 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 folows: 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. When finishing some sections the student may be asked to test his learning process by actively applying the studied lesson material by peforming 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 tutor model is available for each exercise containing information on the mathematical material that should be employed and tested by it.

In this szenario 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) tutor 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 dialog 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 dialog 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.[8]

5 Related Work

[3], which itself employs ideas from [27, 24] suggest a proof representation and manipulation framework based on the uniform notation idea, while we here only employ it at the heart of an assertion mediator module which can be coupled with arbitary proof systems. The generalised resolution algorithm we propose also reminds to ideas employed in the connection graph method [17, 2, 8] and the *clause graph method* in [22]. However, since it is a crucial concern for us not to break down the structure of the assertions we operate with tree or graph structures genereated for arbitary formulas instead of just clause sets.

[6] motivates and also presents a mediator between mathematical knowledge bases and theorem provers. A main difference of the approach here is that a assertion agents are uniformly modeled; their individual behaviour is only determind by the particular assertion and the proof context they are instantiated with. Further related work is described in [11].

6 Conclusion and Future Work

We argued that the assertion level introduced by Huang [15] is one of the more interesting abstract levels where proof planning should be carried out. Therefore, it is necessary that the proof planner at the assertion level be equipped with an adequate infrastructure to be able to take full advantage of the inferences offered by the assertions. We went on to develop a framework for extracting important information from assertions in accordance to the reasoning context (i.e. the proof situation) in which the assertion is invoked. We describe an agent-based mechanism for our framework which serves as an assistant for proof search in Omega. This mechanism also reflects the cognitive process human agents employ when performing theorem proving: A skilled mathematician would in general accept conclusions emerged from algorithm *AssertionApplication*() as a trivial or obvious step and normally not require a detailed (i.e. logic level) explanation of such steps. The research bears immediate fruit through our application in the Dialog project aiming at building an intelligent mathematical tutoring system. The realization of natural language dialog in such a project should take full advantage of the assertion level proofs developed by our formalism.

The future work includes:

1. We want to investigate whether our approach scales up to higher-order contexts. Additional problems have to be addresed, such as undecidability of higher-order unification, primitive substitution, etc. In higher-order contexts our distributed modeling will win on impact, since this will provide a flexible basis, for instance, to model iterative deepening over constraints such as a maximal unification depth.

[8]The unsuitability of proof representations in the latter calculi for maths tutoring purposes has already been indicated as a side result of a first (Wizard of Oz) experiment in the project. Even in the very low level mathematical domain of naive set theory the students found it very confusing when they got involved in too low level natural deduction steps. A more extensive experiment in which we analyse the phenomena of tutorial natural language dialog in the domain of naive set theory is at present carried out and confirms this thesis so far. The detailed results will be available by the end of February 2003.

2. Equational reasoning is not yet addressed in our framework and needs to be investigated. In particular adapting our framework to (extensional) equational reasoning within higher-order logic contexts is a challenge.

3. We propose to fully integrate automated proof planning with our approach to assertion level reasoning.

References

[1] P. Andrews. Transforming matings into natural deduction proofs. In W. Bibel and R. A. Kowalski, editors, *Proc. 5th CADE*, pages 281–292. Springer-Verlag, 1980.

[2] P. B. Andrews. Theorem proving via general matings. *J. ACM*, 28(2):193–214, 1981.

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

[4] F. Baader and W. Snyder. Unification theory. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 8, pages 445–532. Elsevier Science, 2001.

[5] C. Benzmüller, A. Meier, E. Melis, M. Pollet, and V. Sorge. Proof planning: A fresh start? In *Proceedings of the IJCAR 2001 Workshop: Future Directions in Automated Reasoning*, pages 25–37, Siena, Italy, 2001.

[6] C. Benzmüller, A. Meier, and V. Sorge. Bridging theorem proving and mathematical knowledge retrieval. In *Festschrift in Honour of Jörg Siekmann*, LNAI. Springer, 2002.

[7] C. Benzmüller and V. Sorge. Oants – an open approach at combining interactive and automated theorem proving. In M. Kerber and M. Kohlhase, editors, *Symbolic Computation and Automated Reasoning*, pages 81–97. A.K.Peters, 2000.

[8] W. Bibel. Matings in matrices. *Communications of the ACM*, 26:844–852, 1983.

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

[10] A. Bundy. A critique of proof planning. In A. C. Kakas and F. Sadri, editors, *Computational Logic. Logic Programming and Beyond*, volume 2408 of *Lecture Notes in Computer Science*. Springer, 2002.

[11] I. Dahn, A. Haida, T. Honigmann, and C. Wernhard. Using mathematica and automated theorem provers to access a mathematical library. In *Proceedings of the CADE-15 Workshop on Integration of Deductive Systems*, 1998.

[12] M. Fitting. Tableau methods of proof for modal logic. *Notre Dame Journal of Formal Logic*, 13:237–247, 1972.

[13] M. Fitting. *First Order Logic and Automated Theorem Proving.* Springer, 1990.

[14] S. Gerberding and B. Pientka. Structured incremental proof planning. In *KI - Kunstliche Intelligenz*, pages 63–74. Springer-Verlag, 1997.

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

[16] X. Huang, M. Kerber, J. Richts, and A. Sehn. Planning mathematical proofs with methods. *Journal of Information Processing and Cybernetics, EIK*, 30(5-6):277–291, 1994.

[17] R. Kowalski. A proof procedure using connection graphs. *J. ACM*, 22(4):572–595, 1975.

[18] E. Melis. Island planning and refinement. Technical report, University of Saarland, 1996.

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

[20] E. Melis and J. Siekmann. Knowledge-based proof planning. *Articial Intelligence Journal*, 115(1):65–105, 1999.

[21] N. V. Murray and E. Rosenthal. Inference with path resolution and semantic graphs. *Journal of the ACM (JACM)*, 34(2):225–254, 1987.

[22] H. J. Ohlbach and J. H. Siekmann. The Markgraf Karl refutation procedure. In J. L. Lassez and G. Plotkin, editors, *Computational Logic, Essays in Honor of Alan Robinson*, pages 41–112. MIT Press, 1991.

[23] J. A. Robinson. A machine-oriented logic based on resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.

[24] K. Schütte. *Proof Theory.* Springer Verlag, 1977.

[25] J. Siekmann, C. Benzmüller, V. Brezhnev, L. Cheikhrouhou, A. Fiedler, A. Franke, H. Horacek, M. Kohlhase, A. Meier, E. Melis, M. Moschner, I. Normann, M. Pollet, V. Sorge, C. Ullrich, C.-P. Wirth, and J. Zimmer. Proof development with omega. In A. Voronkov, editor, *Proc. 18th CADE*, number 2392 in LNAI, pages 144–149, Copenhagen, Denmark, 2002. Springer.

[26] Q. B. Vo. Generalized resolution: A tree-based approach, 2003. Submitted to this conference.

[27] L. Wallen. *Automated proof search in non-classical logics: effcient matrix proof methods for modal and intuitionistic logics.* MIT Press series in artificial intelligence, 1990.