Proof Planning: A Fresh Start?

Christoph Benzmüller Andreas Meier Erica Melis Martin Pollet Volker Sorge
Universität des Saarlandes, Fachbereich Informatik
66041 Saarbrücken, Germany
{chris|ameier|melis|pollet|sorge}@ags.uni-sb.de

No Institute Given

1 Introduction

Proof Planning [5, 17] is a technique for automated (and interactive) theorem proving that has been originally conceived as an extension of tactical theorem proving. It is motivated by the fact that human mathematicians do not construct proofs at pure logical calculus level but employ abstract inferences instead. Proof methods therefore abbreviate mathematically motivated and recurring patterns of calculus level inferences. Additional pre- and post-conditions model their applicability conditions. Proof planning searches for a plan of a theorem which consists of applications of several proof methods. Thereby the search space for a proof of a theorem can often be dramatically reduced. Proof methods should be transparent and intuitive and the hope is that proof planning is a mathematically adequate approach to perform mathematics with a computer. Implementations of proof planning are the systems CLAM [6], λCLAM [20], and ΩMEGA [2]. These systems have demonstrated the feasibility and usefulness of proof planning in different case studies. CLAM and λCLAM, for instance, have been used for inductive theorem proving both in mathematical and software verification. ΩMEGA's planner has been successfully used for non-inductive theorem proving in various mathematical domains such as analysis and finite algebra.

Despite its partial success proof planning has still various shortcomings. In [4], for instance, Alan Bundy discusses problems of proof planning from the perspective of the CLAM and λCLAM projects. His main critique concerns the lack of generality of proof methods (i.e. methods are only applicable in a very limited problem domain and proof plans can usually only be executed in one particular calculus), the brittleness of the planning process (i.e. to restrict the search space there are hardly calculus level methods available that can bridge the gap between the application of abstract, specialized methods), the problem to create adequate human oriented proof methods, and the impossibility to separate plan formation and execution (i.e. proof methods have to be executed in order to generate the next planning state).

In this paper we want to address the aspect whether proof planning indeed provides a suitable basis for the modeling of human mathematical reasoning from the perspective of our experience with the ΩMEGA system.

In ΩMEGA we have solved the problem of separation of plan formation and execution by specifying both pre- and postconditions of methods schematically in ΩMEGA's simply typed higher order lambda calculus [8] together with an expansion schema or tactic that transforms the method into a calculus level subproof. Thus, proof plans can be assembled on a high level without the need for intermediate execution. Proof methods can also incorporate unreliable computations in their proof steps. In order to ensure correctness, however, a complete plan has to be expanded into ΩMEGA's basic calculus, a natural deduction (ND) calculus [10]. The expansion of a proof plan to a calculus-level proof is executed locally, that is, each method is expanded separately. In this process, the abstract step is replaced by a sub-proof of its conclusions from its premises using less abstract steps (tactics, or basic calculus level steps).

Despite its advantage for abstract plan formation the requirement of local expandability has some severe drawbacks: The necessary tied coupling of abstract methods with underlying calculus has the effect that certain constraints the underlying ND calculus imposes have to be respected also on the planning level. For instance, the order of variable elimination and hypotheses introduction can be crucial for the success of a later expansion. The intermediate subgoals constructed during the plan formation correspond essentially to proof lines on the calculus level. The method execution provides only more detailed subproofs to justify the single derivations. This deprives us of the possibility to use a more intuitive and maybe mathematically more adequate language on the planning layer.

In the following we shall discuss these problems in more detail and speculate a possible solution which consists of the total separation of the planning and the calculus level. We give up the local expandability of methods as well as the direct correspondence of a method to a sequence of calculus level steps. We instead shift the expansion to a general transformation problem from a proof plan to a calculus level proof. This enables us to adopt a more constraint based approach to proof planning and, moreover, to choose a more appropriate and mathematically more intuitive representation of our problem.

2 Some Speculations and the Modeling of Mathematical Reasoning

There are two possible ways in modeling mathematical reasoning. The machine-oriented approach, that exploits the capabilities of computers to do millions of simple computations. And the human-oriented approach, which tries to imitate the reasoning done by mathematicians.

Without doubt, the machine-oriented approach was quite successful. In some domains, like chess, computers reached and excel human capabilites with a methodology that is surely different from the way humans play chess. Also in automated theorem proving systems have reached an impressive power. Nevertheless, ATPs are so far away from mathematical problem solving, that their existence isn't even noticed by most of the mathematicians.

The human-oriented approach has to face the question which paradigm, that is feasible to implement comes closest to human reasoning. A look into Polya's famous "How to Solve It" could suggest an approach based on the detection of similarities and different levels of analogy together with 'auxiliary subproblems' [19], where each of them is still challenging in AI for their own. Another aspect is, that the 'data structure' of humans and computers are to different to reach a full imitation. When Minsky says that a CD-ROM would contain enough commonsense knowledge [21], because a computer can easily remember ten books by heart what is rather difficult for humans, we see the difference. Humans can remember the content of more than ten books easily while they get not stuck in the details of thousands of words. We think that this kind of compression of knowledge also appears in mathematics. Since the solution for this sort of questions is surely outside of todays AI technology, we have to look for a compromise: proof planning based on the well-explored AI planning paradigm.

The two basic arguments supporting the thesis that proof planning can model the reasoning of human mathematician adequately are (cf. [9])

- Mathematicians use abstract proving methods, which can be domain specific and whose application is triggered not only by the problem at hand but also by the experience of the mathematician.
- Mathematicians use a plan based approach when proving theorems starting with an abstract sketch, which is successively refined.

3 Problems with Logic-Oriented Proof Planning

In this section we present examples to illustrate problems caused by the logic-biased setting in which proof planning in ΩMEGA is done. The logic-biased setting affects both, the search for proofs and the modeling of reasoning techniques used by mathematicians. First we take a look at principles of the ND calculus that are inherited by methods. Then we present problems caused by the formal representation of mathematical objects.

3.1 Order of Steps

As the word 'natural' indicates, Gentzen's intention was to set up a calculus that comes close to the actual human reasoning. ND therefore is a good choice for the base calculus. Unfortunately proof in ND depends on the order of forward and backward steps. Since in the current ΩMEGA approach proof plans can be expanded to ND proofs locally, this drawback also affects the method-level. This will be illustrated with the following examples.

A Simple Example Consider the proof situation given in Fig. 1. In this example an existentially quantified goal $\exists x P[x] \wedge Q[x]$ (line L_{10}) should be using the hypothesis $\exists x P'[x] \wedge Q'[x]$ (line L_1) where P, P', Q, Q' represent arbitrary

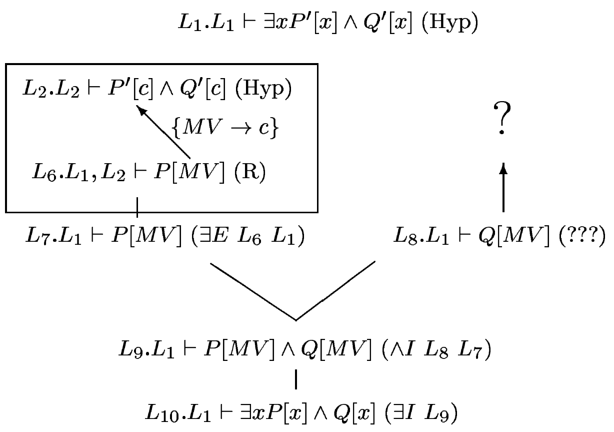
$L_1.L_1 \vdash \exists x P'[x] \wedge Q'[x]$ (Hyp)
$L_2.L_2 \vdash P'[c] \wedge Q'[c]$ (Hyp)
$\{MV \to c\}$
$L_6.L_1, L_2 \vdash P[MV]$ (R)
?
$L_7.L_1 \vdash P[MV]$ ($\exists E$ L_6 L_1)
$L_8.L_1 \vdash Q[MV]$ (???)
$L_9.L_1 \vdash P[MV] \wedge Q[MV]$ ($\wedge I$ L_8 L_7)
$L_{10}.L_1 \vdash \exists x P[x] \wedge Q[x]$ ($\exists I$ L_9)

Fig. 1. Problems with explicit hypotheses treatment

complex formulas. Moreover, P,P' and Q,Q' are chosen such that $P[t]$ follows from $P'[t]$ and $Q[t]$ follows from $Q'[t]$ for each term t.

A backward search proceeds in the following way. The formula in L_{10} is decomposed in two steps to the subgoals $P[MV]$ (line L_7) and $Q[MV]$ (line L_8). For the existentially quantified variable in L_{10} a meta-variable MV is inserted. This invokes a middle-out-reasoning [14] process to postpone the instantiation for the quantified variable until a witness term is found during the planning process, that can be inserted for the meta-variable. After splitting $P[MV] \wedge Q[MV]$ into the two subgoals $P[MV]$ and $Q[MV]$ the planner focuses first on $P[MV]$. A meta-reasoning process could detect that the hypothesis L_1 contains in $P'[x]$ parts relevant for this subgoal. Hence, steps are employed to unwrap these parts. For this the existential quantifier in L_1 is eliminated, introducing a new constant c and a new hypothesis as line L_2. Since the unwrap process was done for the open line L_7, only this line is in the scope of the hypothesis L_2 (indicated by the box in Fig. 1).

Problems with the Eigenvarable condition The next steps to justify $P[MV]$ should instantiate MV by c and derive $P[MV \to c]$ from $P'[c]$. Unfortunately, it is not possible to instantiate MV by c, since this would violate the *Eigenvariable condition* for the application of the ND-rule $\exists E$. The Eigenvariable condition that forbids that the new constant occurs in the conclusion of the step or in the hypotheses of the conclusion. When MV would be instantiated by c the $\exists E$ step becomes incorrect. During the proof planning process such Eigenvariable conditions are collected, therefore the instantiation $MV \to c$ is forbidden and the planner fails to justify $P[MV]$.

Problems with hypothesis scope Let us assume that there is a some kind of repair mechanism for the Eigenvariable condition of the last paragraph, and the planner was able to instantiate MV by c and to justify line L_7. Then the proof planner would focus on the remaining open goal $Q[MV]$ in L_8. Since MV is instantiated to c the goal has become $Q[c]$. A meta-reasoning process could detect that the hypothesis L_1 contains in $Q'[x]$ parts relevant for this subgoal. But it is not possible to use $Q'[c]$ in L_2 to derive $Q[c]$ since L_8 is outside of the scope of L_2. Also the application of $\exists E$ to L_6 and L_1 would not result in a successful proof because $\exists E$ would introduce a new constant c' such that $Q[c']$ but not $Q[c]$ would be inferable.

The only possibility to construct a correct proof is to perform $\exists E$ as first step. So that neither the later instantiation of the meta-variable MV violates the Eigenvariable condition nor the scope of the introduced hypothesis is too restricted. We used a simple example where rules of the ND calculus were used as methods. The same problems may occur whenever the expansion of a complex method contains rules that introduce new constants or hypotheses. In order to enable the local expansion of methods, the introduction of hypotheses and the Eigenvariable conditions have to be respected at the method level. Therefore the order of steps becomes important for proof planning.

The *Cont-If-Deriv* Theorem Another example which is closer to mathematical reality is the *Cont-If-Deriv* theorem. It says that a function f is continuous at point a if it has a derivative F at point a. More formally, from the assumption $\lim_{x \to a} \frac{f(x)-f(a)}{x-a} = F$ follows $\lim_{x \to a} f(x) = f(a)$. In the proof planning process the definition of *lim* in the goal and the assumption is replaced first by the ϵ-δ-criterion. Further decomposition of the goal formula results in the new goal $|f(MV_x) - f(a)| < c_\epsilon$ for an arbitrary constant c_ϵ. The decomposition of the assumption results in $|\frac{f(c_x)-f(a)}{c_x-a} - F| < MV_\epsilon$. Indeed, the goal can then be closed under this assumption (thereby the mapping $\{MV_x \mapsto c_x, MV_\epsilon \mapsto c_\epsilon\}$ is applied). Unfortunately, a further subgoal $c_x \neq a$ is created during the application of the assumption and there is no assumption to close this goal. To deal with this goal a case split has to be introduced. That is, the goal $|f(MV_x \mapsto c_x) - f(a)| < c_\epsilon$ has to be proven twice: one time assuming $c_x = a$ — then the goal follows trivially since $|f(MV_x \mapsto c_x) - f(a)| = 0 < c_\epsilon$ holds for every positive c_ϵ — and one time assuming $c_x \neq a$ — then the goal follows from the initial assumption as explained above. The subgoal $c_x \neq a$ can now be closed with by the hypothesis $c_x \neq a$ of the case split. The crucial point is: When should this case split be introduced? From mathematical intuition it should be introduced when the goal $c_x \neq a$ is created and cannot be closed, but logical calculus requires the introduction before the assumption is decomposed, such that the goals created during the decomposition of the assumption depend on the hypothesis of the case split. Such an introduction is completely counter-intuitive. The only possibility to deal in ΩMEGA with such a problem is to realize the need for a case split when the unsolvable goal is reached, then to intelligent backtrack as much as necessary and to introduce the case split at the logical suitable position. Afterwards, the backtracked proof parts have to be performed again.

Analysis Let us analyze the situation:

- Proof planning is goal centered, and therefore backward reasoning is preferred. The proof construction in ND consist of an interplay of forward and backward steps, the order of steps is important for a successful proof.
- Because of the expansion of methods to ND-level, the method-level inherits the dependency on the order of steps at the ND-level. This leads to counter-intuitive introductions of order-relevant steps. In particular, the necessity of promising forward steps is detected when the goal is already decomposed.
- The requirements of the ND calculus affect the design of methods and the heuristics to control the proof search. Hence, the danger is that the motivation to capture the reasoning of mathematicians becomes secondary.

We will discuss possible solutions in Sec. 4.

3.2 Representation of Mathematical Objects

One aspect of knowledge-based proof planning is the imitation of mathematical proof construction by encoding typical reasoning steps as methods. In this section the representation of mathematical objects by a logic-biased language such as in ΩMEGA is addressed. We use logic-biased formulations at the plan level since with the current expansion mechanism im ΩMEGA the formulas are inherited from one abstraction level to the next; that is, the formulas at the plan level are also in the calculus level. Hence, the formalization at the plan level are the same as at the calculus level. However, our thesis is that although methods allow to incorporate mathematical knowledge into the proof construction process, the logic-oriented language in which the methods and the formula of the proof plan are expressed is not knowledge-based enough such that important mathematical information is lost. In the following we give evidence for this thesis.

Knowledge-Based Notations in Mathematics Mathematicians have spent a lot of time and thoughts in the development of adequate representations of mathematical objects. Representation of mathematical objects starts with notational aspects. For example, the use of the letters x, y, z denoting the 'unknown' is today common practice. Historians credit the first use of symbols for variables in algebraic equations to François Vieta. This invention influenced the understanding of the concept of variables and allows a simple treatment for formulas containing different 'unknown' terms like polynomials with multiple variables.

Mathematical notations allow to store a lot of available knowledge about the expressed concepts in a compact and intuitive way. For instance, a mapping $\circ$ that is a binary operation on finite structures can be expressed by a multiplication table:

$$\begin{array}{c|ccc}
\circ & d_1 & \cdots & d_n \\
\hline
d_1 & c_{11} & \cdots & c_{1n} \\
\vdots & \vdots & \ddots & \vdots \\
d_n & c_{n1} & \cdots & c_{nn}
\end{array}$$

What kind of information does this representation provide?

1. $\circ$ is a well-defined function if all d_i are different,
2. $\circ$ has domain $\{d_1, \ldots, d_n\} \times \{d_1, \ldots, d_n\}$ and codomain $\{c_{11}, \ldots, c_{nn}\}$,
3. $\circ$ is a mapping with only finitely many values and a finite domain.
4. the value of $d_i \circ d_j$ is c_{ij},
5. $\circ$ is closed if $\{c_{11}, \ldots, c_{nn}\} \subset \{d_1, \ldots, d_n\}$.

All these facts about the mapping $\circ$ can be looked up from the multiplication table directly. As opposed thereto, how can we formalize the same mapping $\circ$ in a logical language?

– Axiomatic: extend the signature with a function constant $\circ$ and add the hypothesis $d_1 \circ d_1 = c_{11} \wedge \ldots \wedge d_n \circ d_n = c_{nn}$.
– With description operator:[1] the function is given by
$\circ \equiv \lambda x.\iota y.(x = (d_1, d_1) \wedge y = c_{11}) \vee \ldots \vee (x = (d_n, d_n) \wedge y = c_{nn})$.

In both formalizations information is lost that is available in the multiplication table. In the axiomatic formalization, well-definedness cannot be checked in the object-level (a not well-defined function leads to a contradiction). Moreover, domain and codomain are hard to retrieve. In the formalization with the description operator, each application of this function results in a proof nontrivial obligation[2].

Of course, there exist more possible formalizations of $\circ$ than the two formalizations mentioned. However, in all logic-oriented formalizations it seems to be difficult to reconstruct simple properties of $\circ$ such as: $\circ$ is a function, the domain of $\circ$ is $\{d_1, \ldots, d_n\} \times \{d_1, \ldots, d_n\}$ and the codomain of $\circ$ is $\{c_{11}, \ldots, c_{nn}\}$. These formalizations lack to allow information retrieval because they are logic-biased and not knowledge-based as mathematical notations.

Object Orientation in Mathematics In mathematics usually an elegant and minimalistic definition of concepts is preferred. However, the basic properties are introduced right afterwards and 'attached' to the object. For instance, a group can be defined as a structure that consists of a non-empty set G and an associative operation $\circ : G \times G \to G$ such that there exists a right unit in G and for every element of G exists the right inverse in G. Additional properties such as the existence of a left unit and the identity of left and right unit do not have to be part of the definition but can be introduced as lemmas. However, such 'additional' properties become part of the concept *group*. So mathematicians structure their knowledge object centered.

In contrast, logical formalizations are fact-based. That is, additional properties are stated as lemmas or theorems and become part of a general database.

[1] The description operator ι returns the element of a singleton. $\iota y.P[y]$ denotes the unique element c such that $P[c]$ holds.

[2] When applying the function to an element (d_i, d_j) the existence of a *unique* c_{ij} has to be proved such that the description operator ι can return this c_{ij}.

Hence they are not closely related to the concept anymore. To a certain degree the retrieval and collection of needed and suitable facts is just a database connection problem (when requesting the database for all available information relevant for a certain situation, does it return too much information, too less information, the right information?). However, in mathematics some facts are closer related to concepts than others (thereby the grade of relationship can vary from different points of view) and the difference in the grade of relationship is also part of the mathematical reasoning. That is, facts closely related to a concept will be tried before other facts, moreover when applying a fact closely related to a concept result often in justifications like *follows by definition*, whereas the application of other facts is handled more explicitly.

To store logical facts just in a database and to express no differences in their relationship to certain concepts is another source of loss of mathematical information.

Example: Representation Shifts One important mathematical aspect typically closely related with a concept are different representations for the same concept. Mathematicians are able to switch their viewpoint whenever this seems to be useful. That is, *the* formalization of a problem does not exists. Rather mathematical problem solving exploits flexible switching between obviously equal representations. In the large these shifts are for instance the shift from solving equations to merely structural investigations in algebra, or the arithmetic representation of geometry (due to Descartes). In the small, these shifts consist, for instance, in the choice of suitable representations for terms.

How important such representation shifts are in mathematics is demonstrated on the proof of the following theorem:

Given two normal subgroups A and B of a group G with $G = A \cdot B$ and $A \cup B$ contains only the unit of G. Then G is isomorphic to the direct product $A \times B$.

For the proof of this theorem, a mapping h is needed that maps elements $x \in G$ to element $y \in A \times B$ such that h is bijective and a homomorphism. A possible candidate for h (which is indeed the right choice) becomes obvious when the available information about G and A, B is used to represent x, y in a more suitable way: $x = a \cdot b$ and $y = (a, b)$. With this representation it is obvious to choose $h(a \cdot b) = (a, b)$. For this h it is easy to prove that it is bijective and a homomorphism.

The logic-biased proof planning in ΩMEGA that exploits no strong relationship between different representations for x, y would tackle the subproblems directly. That is for a not specified h (represented by a meta-variable) the proof planner would try to prove the properties injectivity, surjectivity, and homomorphism. Thereby, the proof planner would try to collect constraints about h that would allow at the end to determine a suitable h. Unfortunately, the subproblems injectivity, surjectivity, and that it is a homomorphism do not provide any structural information that would allow to synthesize the needed h. The proof plan attempt would fail.

Analysis Let us analyze the situation:

- Mathematical notations are knowledge-based. They allow to represent many informations about the denoted concept in a compact and intuitive manner.
- Mathematics is object oriented. Facts that belong together are centered around concepts.
- Logical formalizations of mathematical concepts cover typically only a part of the information contained in the mathematical notations. Moreover, the object centering is lost.
- Because of the loss of mathematical information, operations that are trivial for mathematicians are difficult to perform in proof planning.

Possible solutions are discussed in the next section.

4 Mathematically-Biased Proof Planning: First Ideas

In the last section we described some problems we encountered in the logic-biased proof planning in ΩMEGA. In particular we described ordering problems and formalization problems that are inconsistent with the aim of ΩMEGA's proof planning to imitate mathematical theorem proving. In the following we first point to work and approaches that partially address our problems. Then we present our first ideas how we intend to make the proof planning in ΩMEGA more mathematically-biased and less logic-biased.

4.1 Discussion of Some (Partial) Solutions

How to deal with ordering problems inherited from the ND calculus? The first possibility is to accept the influence of the calculus on the planning level in favor of a simple and local expansion of methods. In order to assure a finitary and successful planning process, the methods have to be specified according to a normal form that respects the requirements of the base calculus. For pure ND calculus itself a normal form that allows for proof search has been given by Byrnes [7]. In the setting of an extensible repertoire of methods, this problem will need to be addressed with every new method. Human mathematical problem solving, however, is independent of such technical details.

The next consideration could be the use of another calculus without the shortcomings of ND. Clause normal form together with Skolemization is out of the question as it destroys structural information we want to employ in proof planning. There are good chances to establish a calculus with the desired properties. For the sequence calculus exist already results that overcome the order dependency of the standard Skolemization rule (cf. the simultaneous quantifier elimination rule [1]). But generally, by choosing another calculus we still cannot catch the conceptional difference between methods and calculus rules. Even with a calculus that allows for more freedom, the calculus-level will have an impact on the planning-level.

How to deal with formalization problems? We could try to employ a more expressive formal system. By using sorts, for instance, we get a representation that is more natural with respect to the mathematical notation. And clearly additional sort information has a positive effect on the proof search. This seems to be the right direction, but unfortunately, a sort system that is strong enough to express sorts of every definable function, requires a mechanism for the well-formedness of terms that has to be as powerful as the theorem proving mechanism itself. So it seems that an approach that tries to fix a hierarchy between trivialities and hard problems into syntax is not in accordance with the mathematical reality.

A contrasting idea is not to use any feature of the language at all. The richness of mathematical representations could be encoded instead into the object-level of the formal language itself. For each representation (like a multiplication table) a constant with the appropriate arguments (like domain, codomain, a list of lists) could be added to the signature together with a formal definition. Now that the object is made explicit in this way, a set of methods that model the use of this construct can be added. On one hand, we can retrieve all informations for our method-level, on the other hand this seems to misuse the formal system just for data storage, because basic features, like evaluation of an application (that is built-in in λ-calculus) and equality have to be made explicit.

Between these extreme positions lies the idea of an enhancement of the usual formalizations by additional information in form of annotations (for terms and their sub-terms) and the application of annotated reasoning techniques [11].

4.2 Our Proposal

Currently, methods and tactics in ΩMEGA are expanded locally. That is, a complex step such as a method or tactic application can be expanded to a proof tree with the conclusion of the complex step as root and the premises of the complex step as leaves. Hence, the conclusion and the premises of the complex step are not changed, only the complex justification between them is replaced by a more fine-grained justification in form of a whole proof tree. The expansion is done either by interpreting a pattern or calling a small procedure. A proof plan is expanded to a ND proof by successively expanding all complex steps until ND steps are reached.

The expansion of proof plans to ND proofs is important because complex steps can be incorrect. Only at the ND calculus level the correctness of a proof object can be checked. However, the analysis of the problems described in Sec. 3 shows that, indeed all problems rest more or less on the direct connection of the plan level with the calculus level. Firstly, the ordering problems at the plan level are directly inherited from the ordering mechanisms at the calculus level. Secondly, since all formulas and term used and introduced at the plan level are kept during the expansions, the concepts used at the plan level have to be the same as at the calculus level.

Hence, we suggest to separate the plan level and the calculus level completely and to give up the local expandability of plan steps. We do not want to lose the possibility to expand our proof plans to calculus level proofs. But instead of the strong connection of a local expandability, we suggest to replace the expansion problem by a more general transformation problem in its own right. This separation provides at the plan level the freedom to perform more flexible planning as well as to have own not logic-biased formalizations. In the following we discuss our first ideas about how to exploit this freedom at the plan level and how a more general transformation-expansion can work.

Towards more flexible Proof Planning In AI planning almost every planner employs the idea of successively accumulating and validating constraints. For instance, nonlinear planner (see [15]) use causal links or protection intervals to constrain the temporal orderings of plan steps. Most planner perform just *passive constraint postponement*; that is, constraints are used to postpone decisions, but they are not employed in the further planning process. As opposed thereto, in *active postponement* the posted constraints are used in the subsequent planning process to guide the search. For instance, the Descartes system [12] is a planner that performs active postponement by representing every planning decision with constrained variables, and posting constraints that represent the correctness criteria for the plan. The variables and constraints, taken by themselves, describe a constraint satisfaction problem (CSP). Thus, a problem can be viewed as either a planning problem or a CSP. This duality paradigm is called *dual-representation planning* in [12].

Similar to this approach, we suggest to combine proof planning with constraint solving. First approaches have been already studied in the *limit domain* where constraints about meta-variables are collected and used later on to compute suitable instantiations for the meta-variables [18]. Also other 'side-conditions' or 'side-computations' should be postponed into constraints. For instance, sort informations about constants and meta-variables can be expressed in constraints; then suitable constraints solvers can be applied to check the sorts of terms (instead of holding these checks as subproblems in the proof plan). Moreover, ordering restrictions existing in corresponding ND proofs can be expressed at the planning level with according ordering constraints. The accumulated constraints can be used twofold:

1. During the planning process to guide the search according to the active constraint postponement. In particular, to preserve the planner from steps that result in inconsistent constraint states (preserve the planner to continue on points of the search space from whose no proof plan can be found).
2. During the global expansion of a proof plan to guide the construction of the calculus level proof by at least partially ordering the proof methods.

Towards Knowledge-Based/Object-Centered Representation In Sec. 3 we have discussed two kind of problems: Firstly that certain mathematical objects cannot be adequately represented without loosing their intuitive meaning. And secondly that mathematical concepts consist of more than just their pure definition.

For the latter, we have to assign more information to an object. In [13] Kerber suggests a frame representation of mathematical objects. He distinguishes frames for axioms, definitions and theorems. A frame contains besides the formalization itself slots for equivalent formulas, examples, superconcepts, simple properties etc. The frames allow for an object centered encoding of mathematical knowledge. Unfortunately the additional information has to be given manually. An automation is more than questionable, as also mathematicians have to invest efforts (like analysis of given examples and theorems and exercises) to establish a useful hierarchy of knowledge.

Nevertheless, we might reduce the effort of encoding methods by representing mathematical entities as frames. This enables to represent equivalent definition of the same concept in the same frame. Methods having pre- and postconditions implemented with a frame representation are then be applicable in problems that use different formalizations of the same mathematical concept. This leads to a more robust proof planning behavior.

Moreover, the frame representation enables to store important properties directly with the definition of certain mathematical concepts and entities. A method can thus contain a variety of additional knowledge on the concepts it is concerned with and, depending of the state and context of the planning process, might be able to inject some of this knowledge into the proof planning process.

For the first problem, we can at least give a specification of the desired properties:

- It should keep the information of the mathematical representation.
- Allow introduction of mathematical representations.
- The use of specialized representation should ease planning.

Towards a General Transformation-Expansion Today's theorem proving systems (automatic and interactive ones) have reached a considerable strength. However, it has become clear that no single system is capable of handling all sorts of deduction tasks. Therefore, it is a well-established approach to delegate subgoals to other (specialist) systems. To use the results of other systems mediators are needed that transform the proofs in the formalism of the one system into proofs in the formalism of the other system (i.e., [16] describes the TRAMP system that transforms the output of several first order ATPs into proof objects in ΩMEGA's ND calculus).

In this transformation context, we suggest to view the expansion problem as a transformation problem in its own right. More precisely, a proof plan is considered as an object that has to be transformed into a pure ND calculus level proof object[3]. The transformation from the plan level to a calculus level has to perform two things:

[3] With respect to the fact that we use the calculus level proofs only for verification issues, a transformation into a ND calculus level proof is not obligatory. When understanding the expansion problem as a transformation problem it is also possible to have several underlying calculi such that different transformation algorithms can produce the proof objects in the different calculi.

1. Mapping the formulas in the formalization of the plan level into formulas in formalization of the calculus level.
2. Mapping the steps/justifications at the plan level to steps/justifications at the calculus level.

Concretely, our idea is that first the formulas of the plan level are transformed into corresponding formulas at the calculus level. These formulas provide *isles*. Then the gaps between the islands are closed at the calculus level by other reasoning mechanisms that work at the calculus level. Moreover, the plan level could provide information to the reasoning mechanisms at the calculus level such as which inference rules are (probably) needed for closing a gap corresponding to a certain plan step as well as further control information. Hence, for closing the gaps we suggest a parameterizable mechanism (for instance, in ΩMEGA the ΩANTS mechanism (see [3] for a description of how the ΩANTS mechanism can be used as a parameterizable inference machine).

5 Final Remarks

¿From our perspective the future of automated deduction lies in a further orientation towards the style of reasoning actually performed by mathematicians. Knowledge-based proof planning is one first step in this direction. In this paper we illustrated that proof planning is still too restricted because of its logic orientation. The idea is to liberate proof planning from logic oriented reasoning as far as possible, but to keep the transformability to logical arguments for proof checking purposes. This idea is rather new and its success is not guaranteed.

References

1. S. Autexier, H. Mantel, and W. Stephan. Simultaneous quantifier elimination. In *Proceedings of the 22. annual German Conference on AI.* Springer, Germany, 1998.
2. 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 W. McCune, editor, *Proceedings of CADE–14*, volume 1249 of *LNAI*, pages 252–255. Springer, Germany, 1997.
3. C. Benzmüller, A. Meier, M. Pollet, and V. Sorge. Proof transformation and expansion with a parameterizable inference machine. In *Proc. of Eighth Workshop on Automated Reasoning held at AISB'01*, 2001.
4. A. Bundy. A critique of proof planning. Submitted.
5. A. Bundy. The Use of Explicit Plans to Guide Inductive Proofs. In *Proceedings of CADE–9*, pages 111–120, 1988.
6. A. Bundy, F. van Harmelen, C. Horn, and A. Smaill. The OYSTER-CLAM system. In M. E. Stickel, editor, *Proceedings of the 10th International Conference on Automated Deduction (CADE–10)*, volume 449 of *LNCS*, pages 647–648, Kaiserslautern, Germany, 1990. Springer Verlag, Berlin, Germany.

7. J. Byrnes. *Proof Search and Normal Forms in Natural Deduction.* PhD thesis, Department of Philosophy, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 1999.
8. A. Church. A Formulation of the Simple Theory of Types. *The Journal of Symbolic Logic*, 5:56–68, 1940.
9. G. Faltings and U. Deker. Interview: Die Neugier, etwas ganz genau wissen zu wollen. *bild der wissenschaft*, 10:169–182, 1983.
10. G. Gentzen. Untersuchungen über das Logische Schließen I und II. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.
11. D. Hutter. Automated reasoning. *Annals of Mathematics and Artificial Intelligence. Special Issue on Strategies in Automated Deduction*, 2000.
12. D. Joslin and M. Pollack. Passive and active decision postponement in plan generation. In *New Directions in AI Planning*, pages 37–48. IOS Press, 1996.
13. M. Kerber. *On the Representation of Mathematical Concepts and their Translation into First Order Logic.* PhD thesis, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, Germany, 1992.
14. I. Kraan, D. Basin, and A. Bundy. Middle-Out Reasoning for Logic Program Synthesis. Technical Report MPI-I-93-214, Max-Planck-Institut, Im Stadtwald, Saarbrücken, Germany, 1993.
15. D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *Proceedings of 9th National Conference on Artificial Intellgence*, pages 634 – 639, 1991.
16. A. Meier. TRAMP: Transformation of Machine-Found Proofs into Natural Deduction Proofs at the Assertion Level. In *Proceedings of the 17th Conference on Automated Deduction (CADE–17)*, pages 460–464. Springer Verlag, Berlin, Germany, 2000.
17. E. Melis and J. Siekmann. Knowledge-based proof planning. *Artificial Intelligence*, 115(1):65–105, November 1999.
18. E. Melis, J. Zimmer, and T. Mueller. Extensions of constraint solving for proof planning. In *Proceedings of ECAI-2000*, 2000.
19. A. Newell. The Heuristic of George Polya and its Relation to Arti ficial Intelligence. Technical Report CMU-CS-81-133, Carnegie-Mellon-University, Dept. of Computer Science, Pittsburgh, Pennsylvania, U.S.A., 1981.
20. J. Richardson, A. Smaill, and I. Green. System description: Proof planning in higher-order logic with λ*Clam*. In C. and H. Kirchner, editor, *Proceedings of the 15th Conference on Automated Deduction (CADE–15)*, volume 1421 of *LNAI*, pages 129–133, Lindau, , Germany, July 1998. Springer Verlag, Berlin, Germany.
21. R. Sabbatini. The mind, artificial intelligence and emotions - interview with marvin minsky. *Brain & Mind Magazine*, 9, September/November 1998.