

Proof Planning: A Fresh Start?

Christoph Benzmüller Andreas Meier Erica Melis
Martin Pollet Jörg Siekmann Volker Sorge

Universität des Saarlandes, Fachbereich Informatik
66041 Saarbrücken, Germany
{chris|ameier|melis|pollet|siekmann|sorge}@ags.uni-sb.de

Abstract. Proof Planning is a technique for automated (and interactive) theorem proving that searches for proof plans at the level of abstract methods. Proof methods consist of a chunk of mathematically motivated, recurring patterns of calculus level inferences with additional pre- and post-conditions that model their applicability conditions.

Despite its partial success, proof planning has various shortcomings. In this paper we discuss some shortcomings of proof planning in the Ω MEGA system. They result mainly from the strong coupling of the search for a proof plan at the method level with the underlying logic-calculus level. We present some ideas to overcome these shortcomings by separating the plan level and the calculus level completely.

1 Introduction

Proof Planning [5, 17] is a technique for automated (and interactive) theorem proving that has been conceived originally as an extension of tactical theorem proving. Among others, it is motivated by the fact that mathematicians do not construct proofs at a logical calculus level but employ more abstract inferences instead. Therefore, the proof methods used in proof planning consist of a chunk of mathematically motivated, recurring patterns of calculus level inferences with additional application conditions. Proof planning searches then for a *plan* of a proof which consists of a sequence of methods deriving the conclusion of a theorem from its assumptions. The search space for a proof of a theorem can be dramatically reduced this way, since the system first searches for a proof plan at a more abstract level and a successfully compiled plan can then be expanded into a proof at the calculus level which may be one or two orders of magnitude longer (in its number of inference rule application) than the original plan (in its number of methods). Moreover, (mathematical) control knowledge essentially contributes to the restriction of the search space. Proof planning is implemented in the systems CLAM [6], λ CLAM [20], and Ω MEGA [2]. These systems have demonstrated the feasibility and usefulness of proof planning in many case studies. CLAM and λ CLAM, for instance, have been used for inductive theorem proving both, in mathematical and software verification. Ω MEGA's planner has been successfully used for non-inductive theorem proving in various mathematical domains such as analysis and finite algebra.

In order to take advantage of mathematical methods (rather than logical ones) and heuristics that are often domain-specific, Ω MEGA implements *knowledge based proof planning* [17]. These mathematically adequate methods and control knowledge can also support a user in interactive proof planning because they are better understandable. Ω MEGA's proof methods can incorporate unreliable computations e.g. from Computer Algebra Systems, a constraint solver, or mathematical heuristics into their proof steps. In order to ensure final correctness, however, a complete plan has to be expanded into Ω MEGA's basic calculus, a Natural Deduction (ND) calculus [10]. The methods are also specified in Ω MEGA's simply typed higher-order lambda calculus [8]. In Ω MEGA we have addressed the problem of plan formation and execution by specifying for each method an expansion, i.e., a tactic or a schema, that transforms a method application into a lower level subproof. Thus, proof plans can be assembled at a high level but if there is a need for intermediate execution, the expansion can be called and executed during the proof planning process already. The recursive expansion of a proof plan into a calculus-level proof is executed locally, that is, each method is expanded separately. In this process, the abstract step is replaced by a subproof of its conclusions from its premises that uses less abstract steps (tactics, or basic calculus level steps).

Despite its partial success, proof planning has various shortcomings. In [4], for instance, Alan Bundy discusses problems of proof planning from the perspective of the CLAM and λ CLAM project. 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 be executed in one particular calculus only), the brittleness of the planning process (e.g., the need to bridge the gap between the application of abstract, specialized methods), the problem to create proof plans that can be easily understood by humans, and the impossibility to separate plan formation and execution.

This paper takes the experience with knowledge-based proof planning into account and again addresses the problem whether proof planning indeed provides a suitable basis for modeling mathematical reasoning. The perspective is our experience with the Ω MEGA system. The necessary tied coupling of abstract methods with the underlying calculus has the effect that certain constraints of the underlying ND-calculus have to be respected also at 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 at the calculus level and 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 at the planning layer.

The main findings are that while [17] was a step in the right direction problems remain, because this step is by far not radical enough in its departure from the logical calculus level paradigm as it still assumes the (recursive) expandability of methods into calculus-level subproofs which was inherited from tactical

theorem proving. We shall instead propose and discuss a view, where the usual logic oriented proof theory is taken for granted — and is now to be completely separated from the “psychology of mathematical invention” (Hadamard), which should be dominant while searching for a proof plan.

Technically speaking, we shall propose to separate completely the search for a proof plan from its execution and take both as the two major efforts to be undertaken in finding a proof. This has far reaching consequences in how we view a proof plan possibly even without expansion as a valid argument similar to current mathematical practice, where an expansion of a human proof into a calculus-level proof is hardly ever provided. Then new problems arise, e.g.: what is the proof-theoretic status of a proof plan at such a high human-oriented level of abstraction? How can we cope with the problem of intermediate execution?

Of course we do not have the final answer to these questions, but shall point into this direction on the basis of our current experience. We propose to 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 less constrained approach to proof planning and to choose a more appropriate and mathematically more intuitive representation of our problem.

2 Problems with Logic-Oriented Proof Planning

In this section we present examples to illustrate the problems caused by the logic-biased setting of current proof planning in Ω MEGA. 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.

2.1 Order of Steps

As the word ‘natural’ in *Natural Deduction* indicates, Gentzen’s intention was to set up a calculus that comes close to actual human reasoning and it is therefore a good choice for our purpose as the basis calculus. However, ND-proofs depend on the order of forward and backward steps. Since currently proof plans in Ω MEGA can be expanded to ND-proofs locally this drawback also affects the method-level. This will be illustrated with the following examples.

First Example Consider the proof situation given in Fig. 1. In this example the goal is existentially quantified: $\exists xP[x] \wedge Q[x]$ (line L_{10}). A proof should use the hypothesis $\exists xP'[x] \wedge Q'[x]$ (line L_1) where P, P', Q, Q' represent arbitrarily complex formulas for which holds that $P[t]$ follows from $P'[t]$ and $Q[t]$ follows from $Q'[t]$ for each term t .

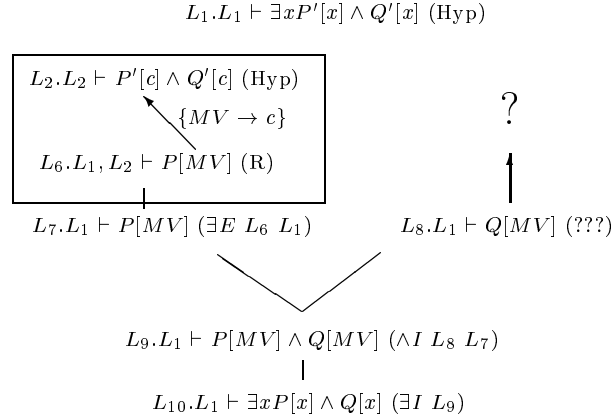


Fig. 1. Problems with explicit hypotheses treatment

A backward search proceeds in the following way. For the existentially quantified variable in L_{10} a meta-variable MV is inserted and the formula in L_{10} is decomposed in two steps to the subgoals $P[MV]$ (line L_7) and $Q[MV]$ (line L_8). The meta-variable is introduced to postpone the instantiation for the quantified variable until a witness term is found during the planning process (such a postponement is also called a middle-out-reasoning process [14]). After splitting $P[MV] \wedge Q[MV]$ into the two subgoals $P[MV]$ and $Q[MV]$ the planner focuses on $P[MV]$ first. A meta-reasoning process could detect that the hypothesis L_1 contains in $P'[x]$ a part relevant for this subgoal. Hence, steps are employed to unwrap these parts. For this the existential quantifier in L_1 is eliminated. Thereby a new constant c is introduced (for the existentially quantified variable) as well as the new hypothesis in line L_2 . Since the unwrap process is 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 Eigenvariable condition The next steps to justify $P[MV]$ instantiate MV by c and derive $P[MV \rightarrow c]$ from $P'[c]$. Unfortunately, it is not possible to instantiate MV by c , since this would violate the *Eigenvariable condition* of the application of the ND-rule $\exists E$. The Eigenvariable condition prohibits that the new constant c occurs in the conclusion or in one of the hypotheses of the conclusion. When MV would be instantiated by c the $\exists E$ step becomes incorrect because c occurs in the hypothesis L_2 . During the proof planning process such Eigenvariable conditions are collected, therefore the instantiation $MV \rightarrow c$ is prohibited and the planner fails to justify $P[MV]$.

Problems with hypothesis scope Let us assume that there is 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 a part 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 the hypothesis 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 $Q[c']$ rather than $Q[c]$ would be inferable.

The only possibility to construct a correct ND-proof is to perform $\exists E$ as the very first step. So that neither the subsequent 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 schematic 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 is important in the current proof planning.

Second Example Another example which is closer to mathematical reality is the *Cont-If-Deriv* theorem. It states that a function f is continuous at point a if it has a derivative F at point a . More formally, from the assumption $\lim_{x \rightarrow a} \frac{f(x)-f(a)}{x-a} = F$ follows $\lim_{x \rightarrow a} f(x) = f(a)$. 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 proved 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 proved for two cases: in the first case $c_x = a$ is assumed and the goal follows trivially since $|f(MV_x \mapsto c_x) - f(a)| = 0 < c_\epsilon$ holds for every positive c_ϵ ; in the second case $c_x \neq a$ is assumed and the goal follows from the initial assumption since the subgoal $c_x \neq a$ can now be closed by the assumption $c_x \neq a$ of the case split. When should the 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 the logical calculus requires the introduction before the assumption is decomposed, such that the goals created during the decomposition of the assumption depend on the assumption of each case. Such an introduction is counter-intuitive. The only possibility to deal with such a problem in the current Ω MEGA system is to realize the need for a case split when the unsolvable goal is reached, then to intelligently backtrack as much as necessary and to introduce the case split at the logically suitable position. Afterwards, the backtracked proof parts have to be done again.

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 local (recursive) expansion of methods to ND-level, the level of methods inherits the dependency on the order of steps at the ND-level. This leads to counter-intuitive sequences of steps. In particular, the necessity of promising forward steps may be detected when the goal is already decomposed.
- The requirements of the ND-calculus influence the design of methods and the heuristics to control the proof search. Hence, there is a danger that the motivation to capture the reasoning of mathematicians becomes secondary.

2.2 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 in a logical language is addressed. We use logic-biased formulations at the plan-level since with the current expansion mechanism in Ω MEGA the formulas are inherited from one abstraction level to the next. Hence, the language of the subgoals and assumptions at the plan-level is the same as at the calculus level. 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 subgoals and assumptions of the proof plan are expressed is not knowledge-based enough and important mathematical information is lost. In the following we provide evidence for this thesis.

Mathematical Notation Mathematicians spend a lot of time and thought on the development of adequate representations of mathematical objects and representation 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 store the available knowledge about the expressed concepts in a compact and standardized way. For instance, a mapping \circ that is a binary operation on finite structures can be expressed by a multiplication table:

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

This table contains the following information implicitly by common agreement:

1. \circ is a well-defined function if all d_i are different,
2. \circ has domain $\{d_1, \dots, d_n\} \times \{d_1, \dots, d_n\}$ and codomain $\{c_{11}, \dots, c_{nn}\}$,
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 (with respect to $\{d_1, \dots, d_n\}$) if $\{c_{11}, \dots, c_{nn}\} \subset \{d_1, \dots, d_n\}$.

All these facts about the mapping \circ can be looked up from the multiplication table directly. How can we formalize the same information about the mapping \circ in a logical language?

- Axiomatic method: extend the signature by a function constant \circ and add the assumptions $d_1 \circ d_1 = c_{11} \wedge \dots \wedge d_n \circ d_n = c_{nn}$.
- With a description operator:¹ the function is given by $\circ \equiv \lambda x. \iota y. (x = (d_1, d_1) \wedge y = c_{11}) \vee \dots \vee (x = (d_n, d_n) \wedge y = c_{nn})$.

In both formalizations information is lost that is available in the multiplication table. In the axiomatic formalization, well-definedness cannot be checked at 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 nontrivial proof obligation².

Of course, more formalizations of \circ than the two formalizations mentioned are possible. 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 $\{d_1, \dots, d_n\} \times \{d_1, \dots, d_n\}$ and the codomain of \circ is $\{c_{11}, \dots, c_{nn}\}$. These formalizations lack information.

Object Orientation in Mathematics In mathematics usually a minimalistic definition of concepts is preferred. In a math textbook the basic properties are introduced right after a definition 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 $\circ : G \times G \rightarrow G$ such that there exists a right unit in G and for every element of G there 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 also characterize the concept *group*. So mathematicians structure their knowledge in an object centered way.

In contrast, in logical formalizations additional properties are stated as lemmas or theorems and are stored in a general database. Hence they are not as closely related to the concept anymore. To a certain degree the retrieval and collection of needed and suitable facts is just a database retrieval problem (when

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} .

querying the database for all available information relevant to a certain situation, does it return too much information, too little information, the right amount of information?). However, in mathematics some facts are closer related to concepts than others (thereby the relationship can vary from different points of view) and the difference in the relationship is also part of the mathematical reasoning. That is, facts closely related to a concept will be considered before other facts. If a database design does not store some of the relationships of mathematical concepts, facts, properties and examples they cannot be used in proofs and mathematical information is lost.

Representational Shifts Important mathematical information that is closely related to a concept is concerned with the different representations for the same concept. Mathematicians are able to switch between these representations whenever this seems to be useful. That is, *the unique* formalization of a problem does not always exist. Rather, mathematical problem solving may switch between equivalent representations. Common examples for these switches are the shift from solving equations to merely structural investigations in algebra, or the shift to arithmetic representation of geometry (due to Descartes). At a lower level of representation, these shifts consist, for instance, in the choice of suitable representations for terms.

Example: *Given two normal subgroups A and B of a group G with $G = A \cdot B$ for which $A \cap B$ 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 an element $y \in A \times B$ such that h is bijective and a homomorphism. A candidate for h surfaces when the available information about G , A , and B is used to represent x and y in the following in a more suitable way: $x = a \cdot b$ and $y = (a, b)$. With this representation it is obvious to choose $h(a \cdot b) = (a, b)$ and to prove that h is bijective and a homomorphism.

The current logic-biased proof planning in Ω MEGA does not exploit the strong relationship between different representations for x and y . It tackles the subproblems directly. That is, for an unspecified h (represented by a meta-variable) the proof planner tries to prove the properties injectivity, surjectivity, and homomorphism. It collects constraints about h that allow to determine a suitable h eventually at the end. Unfortunately, the subproblems injectivity, surjectivity, and homomorphism do not provide any structural information that would allow to synthesize the needed h . Therefore, the proof plan attempt fails.

To summarize the conjectures:

- Mathematical notations are knowledge-based. They allow to represent information about the denoted concept in a compact and intuitive manner.
- Mathematical reasoning might be object oriented. Facts that belong together are centered around concepts.

- Typically, logical formalizations of mathematical concepts cover only some of the information contained in the mathematical notation. Moreover, the relationships are lost.
- Because of the loss of mathematical information, reasoning that is trivial for mathematicians is difficult to perform in proof planning.

3 Mathematically-Biased Proof Planning: First Ideas

In the last section we described some problems with logic-oriented proof planning in Ω MEGA. In particular, we described ordering problems and formalization problems that are inconsistent with the aim to imitate and support mathematical theorem proving. In the following, we first point to work and approaches that partially address our problems. Then we present first ideas how we intend to solve the problems with proof planning in Ω MEGA.

3.1 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 at 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 basic calculus. For the pure ND-calculus 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 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 sequent calculus there exist 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 overcome the conceptual 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.

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. 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.

A contrasting idea is not to use any feature of the language at all. Instead, the richness of mathematical representations could be encoded 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 concept can be added. On the one hand, we can retrieve all information 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 the 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].

3.2 Our Proposal

The expansion of proof plans to ND-proofs is important because complex steps can be incorrect. The correctness of a proof object can only be checked at the ND-calculus-level. Currently, methods and tactics in Ω MEGA 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 justifications between them are replaced by a more fine-grained justification in form of a proof tree. The expansion is done either by instantiating a schema or calling a procedure. A proof plan is expanded to a ND-proof by recursively expanding all complex steps until ND-steps are reached.

The analysis of the problems described in Sec. 2 shows that indeed many problems are caused by 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 requirements at the calculus-level. Secondly, since all formulae and terms 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 expansion of plan steps. Since we do not want to lose the possibility to expand proof plans into calculus-level proofs, we suggest to replace the local expansion by a more general transformation. The definition of proper transformations becomes now itself a problem in its own right, but this separation provides the freedom for more flexible planning. In the following, we discuss our first ideas about how to exploit this freedom at the plan-level and how a transformation-expansion could work.

Towards more Flexible Proof Planning Most AI-planners successively accumulate and validate constraints. For instance, nonlinear planners (see [15]) use causal links or protection intervals to constrain the temporal orderings of

plan steps. Most planners perform *passive constraint postponement*; that is, constraints are used to postpone decisions, but they are not employed in the further planning process. Differently, *active postponement* uses the constraints in the subsequent planning process to guide the search. For instance, the Descartes planner [12] performs active postponement by representing every planning decision with constrained variables and by posting constraints that represent the correctness criteria for the plan. The variables and constraints 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 will combine proof planning with constraint solving, which has already been studied and implemented in the *limit domain* where constraints of meta-variables are collected and used to compute suitable instantiations for the meta-variables [18]. Similarly, other ‘side-conditions’ or ‘side-computations’ can be represented and separated into constraints. For instance, sort information can be expressed by constraints and suitable constraint solvers can check the sorts of terms. Moreover, ordering restrictions can be stored at the planning-level by ordering constraints. The accumulated constraints can be used in the following way:

1. The planning process is guided by the active constraint postponement. In particular, to avoid steps that result in inconsistent constraint states.
2. The global expansion of a proof plan, i.e. the construction of the calculus-level proof, is guided by a partial ordering of the proof methods.

Towards Knowledge-Based/Object-Centered Representation In Sec. 2 we have discussed two kinds of problems: Firstly certain mathematical objects cannot be adequately represented without losing information. And secondly mathematical concepts consist of more than just their definition.

For the latter, we have to assign more information to an object. Kerber [13] suggests a frame representation of mathematical objects. He distinguishes frames for axioms, definitions and theorems. Besides the formalization itself a frame contains slots for equivalent formulas, examples, superconcepts, simple properties etc. A frame allows for an object-centered encoding of mathematical knowledge. Unfortunately the additional information has to be provided manually.

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

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

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 of one system into the proof formalism of the other system (e.g., [16] describes the TRAMP system that transforms the output of several first order ATPs into proof objects in Ω MEGA’s ND-calculus).

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 ND-calculus-level proof object.³ The transformation from the plan-level to a calculus-level has to

1. map the formulae in the formalization of the plan-level into formulae within the formalization of the calculus-level.
2. map the steps/justifications at the plan-level to steps/justifications and order restrictions at the calculus-level.

Our idea is that whenever possible the formulae of the plan-level are transformed into corresponding formulas at the calculus-level. These formulas provide *islands*. The gaps between these islands are closed at the calculus level by other reasoning mechanisms that may work only 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. For closing the gaps we suggest a parameterizable mechanism (for instance, the Ω ANTS mechanism (see [3] for a description of how the Ω ANTS mechanism can be used as a parameterizable inference machine).

4 Final Remarks

In this paper we illustrated that proof planning is still too restricted because of its logical orientation. The idea is to liberate proof planning from restrictions caused by the underlying logic as far as possible, and to keep the transformability to logical arguments for proof checking purposes.

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.

³ Since 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.

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. Ω Mega: 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. This was prepared as a book chapter for Bob Kowalski's festschrift.
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. Decker. 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 Intelligence*, 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 Artificial 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.