Problem-Oriented Applications of Automated Theorem Proving

W. Bibel, D. Korn, C. Kreitz, and S. Schmitt

Fachgebiet Intellektik, Fachbereich Informatik
Technische Hochschule Darmstadt
Alexanderstr. 10, 64283 Darmstadt, Germany
{bibel,korn,kreitz,steph}@intellektik.informatik.th-darmstadt.de

Abstract. This paper provides an overall view of an approach to developing a coherent ATP-system which can deal with a variety of logics and with different applications in a tailored way. The paper also summarizes research results achieved in the course of this development.

1 Introduction

Logical reasoning is an inherent feature of human problem solving. Mathematical reasoning is a particularly precise form of human problem solving which is used especially in scientific applications. Because of its precision it lends itself to automation more easily than reasoning in general. The field of Automated Theorem Proving (ATP) is developing deductive systems which simulate mathematical reasoning. In consequence these systems are of relevance for any application which requires mathematical reasoning.

To mention a few of these applications, finding proofs for mathematical theorems as done in mathematics is of course one of them. A slightly less obvious one is the use of ATP to automatically synthesize computer programs from a given specification. Following the "proofs-as-programs" paradigm [BC85] this task amounts to finding a constructive proof for the existence of a function which maps input elements to output elements of the specified program. A related but more general task is the automated control of the behavior of intelligent agents within a given environment. This application involves the need to automatically generate plans for their actions in that environment. Such plans are a series of actions leading from the initial situation to the desired goal situation. Again this can be modelled as a proof-finding task if these situations as well as the possible actions are modelled in a resource-sensitive logic.

In general, ATP becomes useful whenever the problem to be solved can be formalized in some logical language for which a proof calculus and an efficient implementation is available. In a number of cases, however, we face the problem that such implementations are not available. This is the case in particular if the logic is different from classical first order logic (FOL). Many applications, however, lend themselves to formalizations in logics other than FOL. For instance, trying to formalize the above-mentioned concept of constructive proof within the language of FOL results in a very unnatural formalism which by no means suits to the environments of program development used in concrete systems.

In J. Calmet & C. Limongelli, ed. 4^{th} *International Symposium on Design and Implementation of Symbolic Computation Systems (DISCO'96)*, LNCS 1128, pp. 1–21, ©Springer Verlag, 1996.

A second problem arises for the use of ATP in applications like those mentioned above. The actual proofs generated by ATP-systems tend to have a very technical look. Before they can be understood by experts of the envisaged application they need to be transformed into a more readable form.

The work done at the Intellectics department of the Technical University in Darmstadt during the last few years has tackled these two problems in the development of ATP-systems. On the one hand we have developed systems, or rather a combined system, which deals with problem formalizations in classical as well as in various non-classical logics. On the other hand we have enhanced their interface towards presenting the generated proofs in a way tailored to the particular application. They are *application-oriented* in the sense that they deal with the logic most convenient for the particular application and also provide an application-specific proof interface. The applications envisaged so far in our approach are the three mentioned above.

Our approach is illustrated in Figure 1. It shows at the top and bottom levels the three areas of application, viz. mathematics, programming, and planning. For each area we mention a particular well-known system of possible use in the respective area (Mathematica, NuPRL, and ISABELLE). The vertical arrows at the top level represent the extraction of proof tasks by the user to be solved by the ATP-system. As far as systems such as Mathematica are concerned one may additionally interpret these arrows as applications of these systems within the proof task. The vertical arrows at the bottom level represent the use of generated proofs within each of these applications. For each of the three areas a particular logic is selected (FOL, intuitionistic logic, modal logics). Their applicability may be characterized more generally as follows.

- **classical logic**: covers all applications which dispense in their proofs with epistemological concepts (e.g. proving mathematical theorems). The availability of a normal form (NF) for classical formulae provides a greater flexibility of problem-specific ATP approaches in terms of the possibility to either work directly on the non-normal form input or to transform it first into normal form. Postprocessing of proofs mainly aims at the readability of the proof steps.
- **intuitionistic logic**: covers applications which require constructive proofs (e.g. program-synthesis). No normal form exists which limits the straightforward application of classical ATP-systems. However, as described later in the paper, the non-classical features in the logical formulae may be regarded as semantic information in addition to the formulae's logical contents in the classical sense. The latter can then again be treated similarly as the classical case. Various approaches to deal with this semantical meta-knowledge have been made as discussed in the paper. Postprocessing of proofs needs also to account for the re-integration of the reasoning about this meta-knowledge into the classical line of reasoning.
- **modal logics**: cover applications which require epistemological concepts (e.g. time, belief). Again, no normal form exists but semantical meta-knowledge can be handled in various ways. Postprocessing of proofs again has

to re-integrate the reasoning about this meta-knowledge into the resulting proof.

This paper provides an overall view of our approach to developing a coherent ATP-system which can deal with each of these logics and with the three areas of applications in a way suitable for each of them. Along with this view the paper summarizes research results achieved in the attempt of mechanizing (classical and) non-classical logics.

In the remaining paper we first review the basic structure of the proof process of ATP-systems in order to be able to describe the need for and the location and details of the extensions which enable the handling of different logics and of application-oriented proof presentations. In the main Section 3 we describe two different ways of dealing with semantical meta-knowledge originating from non-classical logics. For each of these two ways we discuss a number of technical alternatives. We conclude with an outlook to future work.

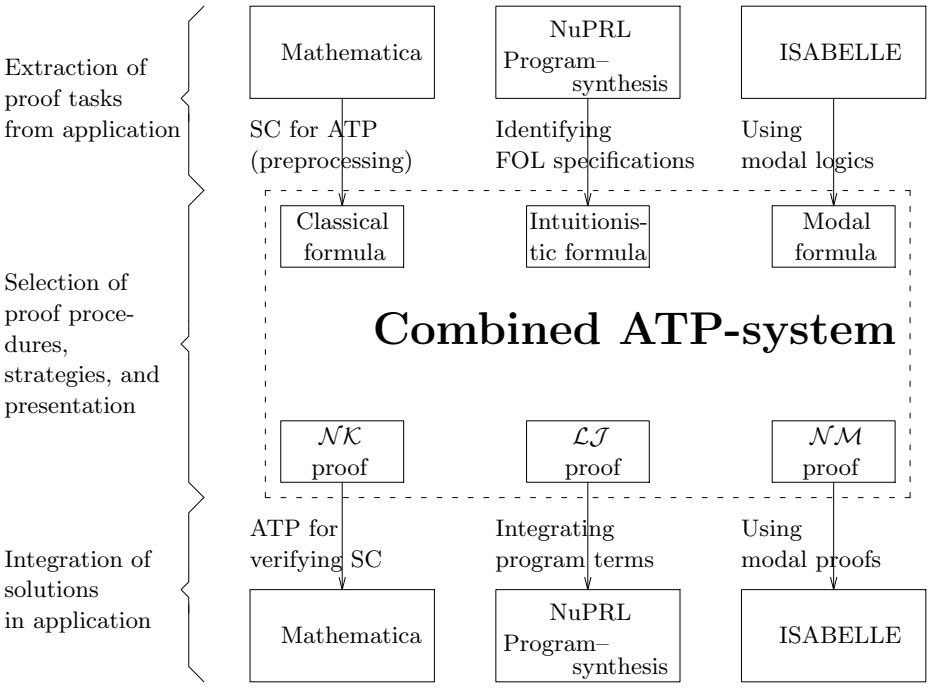


Fig. 1. An application-oriented ATP-system

2 Structuring the Process of Theorem Proving

The core of each ATP-system is the *inference machine* which amounts to sort of a "microprocessor" for theorem proving [Ohl91]. A formula to be proved usually is preprocessed by some input layer in order to transform it into the "machine language" of the inference machine. At the output end the generated proof is possibly postprocessed in order to suit the original input language and to make the result readable for a potential user. Hence, we face the problem of providing mechanisms which deal with three different tasks:

1. **input transformations**: this is the interface between the application-oriented and the inference machine oriented languages. A popular example is the normal form transformation (NFT) which serves to translate arbitrary first order formulae into clause-form which is the preferred language of most inference machines. Another important field of translation is formed by the so-called "logic morphisms" [Ohl91]. They basically aim at encoding semantical meta-knowledge for non-classical logics within the language of classical logic.
2. **inference machines**: they do the actual exploration of the search space which is spanned by the given formula w.r.t. the underlying logic. Usually they consist of a small set of inference rules which are applied to the given formula according to a certain strategy. Popular examples are the various resolution-based strategies [Rob65] as well as the connection-based methods like the extension-procedure [Bib87].
3. **proof presentation**: this is the interface between the logical calculus of the inference machine and the one of the given application. The main goal here is to put the machine-oriented result into a readable form so that the generated proof can be read in terms of the application's own language. Special care has to be taken in order to re-incorporate the necessary information which has been processed during possible input transformations. Examples for such techniques can be found in [And80,Wos90,And91,Pfe87] and [Lin89,Lin90,Gen95]. They try to present resolution- or connection-based proofs within natural calculi such as $\mathcal{LK}$ or $\mathcal{NK}$ [Gen35]. Other approaches try to present them even in natural language [DGH$^+$94]. Similar approaches have been developed for non-classical logics as well [SK95,SK96].

Structuring the proof process in this way provides flexibility for the design of problem-specific ATP-systems. Up to a certain extent, one is given the freedom to distribute the proof process between input transformation and the inference machine. Take, for instance, a classical first order formula which contains equality predicates. Then on the one hand we can deal with the theory of equality by a preprocessing step which generates lemmata that follow from the given formula w.r.t. equality theory [Bra75]. Thus, we would do part of the proof work already within the input transformation layer. On the other hand, we may reason about the equalities by equipping our inference machine with an additional mechanism which allows for reasoning about equality. This amounts to leaving all of the proof work to the inference machine [RW69].

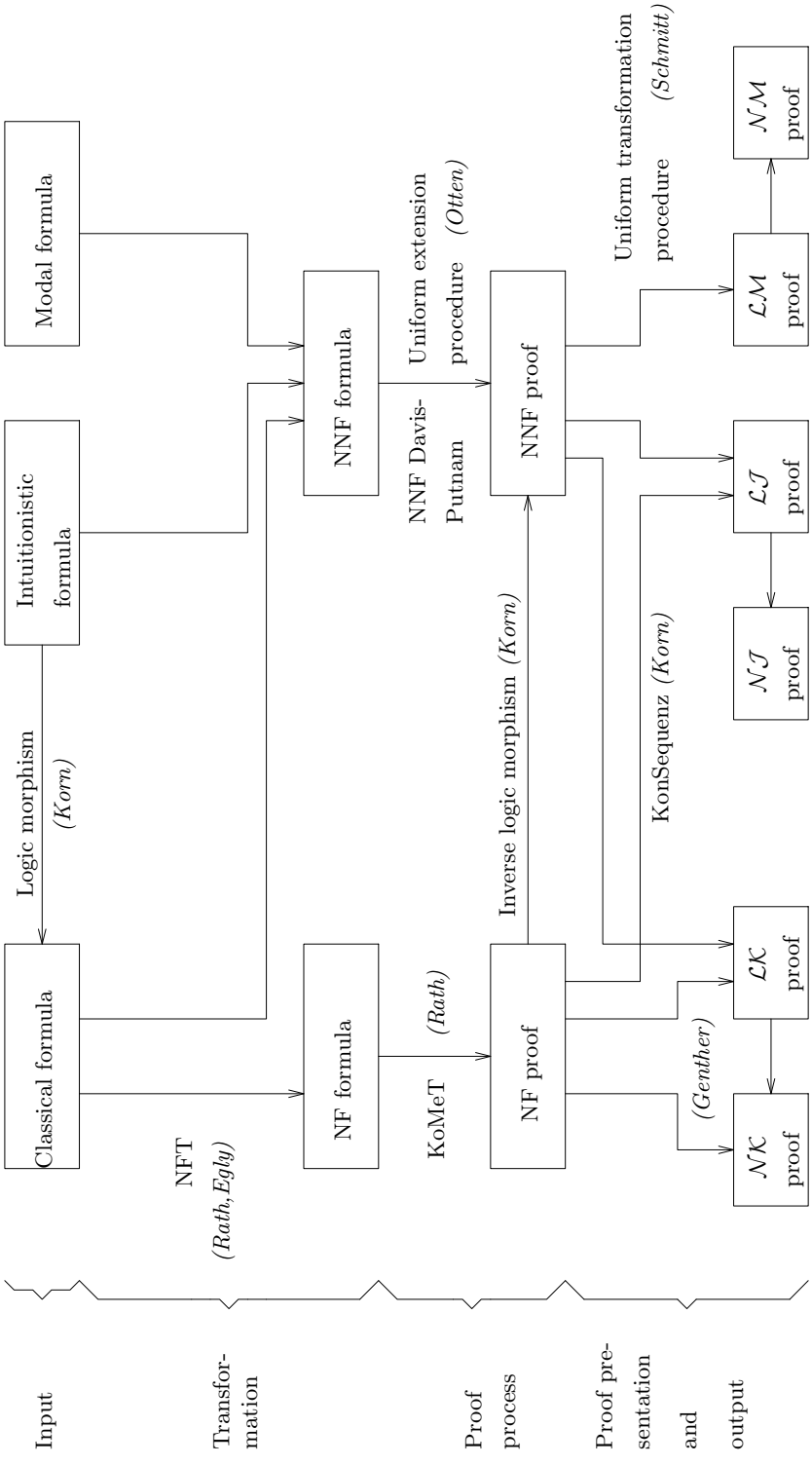


Fig. 2. The concept of combining ATP-systems

Likewise, we might wish to deal with the semantical meta-knowledge for non-classical logics at the input transformations-layer in order to have the transformed formula proved by an inference machine for classical logic. On the other hand we can try to add a logic-specific "co-processor" to the classical inference machine thus obtaining a non-classical inference machine which will directly prove the non-classical formula.

We would like to emphasize that it is not yet clear whether it is more efficient to have a rather simple inference machine which is preceded by a sophisticated input transformation or vice versa. Both approaches have their specific advantages and disadvantages and we have been developing them in parallel. As soon as they both have reached a comparable level of development we will be able to attempt a fair practical comparative evaluation.

Figure 2 provides an overview of the basic techniques which have been developed so far in order to cover the variety of possible approaches. The three above-mentioned layers of the proof process are arranged vertically whereas the three different logics are shown in horizontal order. Beginning at the top we find formulae of the three logics as input objects. From there we can apply various mechanisms which generate suitable inputs for the given inference machines. Among these mechanisms we find the NFT-module providing different techniques for normal form transformation. Practical evaluations have shown that the duration of the proof search strongly depends on the appropriate choice of such a NFT-technique [ER96]. Another example for input transformation is the logic morphism for propositional intuitionistic logic [Kor96] which will be described in detail later on. Below this layer we have three inference machines: the KoMeT-system [BBE$^+$94] which implements various strategic variants of the extension procedure, a non-normal form Davis-Putnam prover for classical propositional logic as well as an inference machine which implements the non-normal form version of the extension procedure thereby allowing for the use of various co-processors for first order non-classical logics [OK96b]. The output layer consists of various techniques which present the generated proofs within natural and logistic calculi for the respective input logic. Obviously their application depends on the way the input-formula was processed. If, for instance, the formula was transformed by a logic morphism then the encoding of the semantical meta-knowledge has to be re-compiled in order to obtain a suitable input for the uniform proof construction mechanism [SK96] shown at the right hand side of the bottom layer. Another approach, the KonSequenz-system [Kor93], was designed to present a classical connection proof for the given formula within an intuitionistic sequent calculus *whenever this is possible*.

In the following section we shall present an example for both of the above-mentioned ways to treat meta-knowledge: a logic morphism mapping propositional intuitionistic logic to propositional classical logic will demonstrate how to handle the meta-knowledge at transformation time. In contrast to that a generic inference machine for dealing with non-normal form formulae will be used as a uniform framework in order to add co-processors for various non-classical logics.

3 Combining Classical and Non-Classical Theorem Proving

ATP in non-classical logics is usually seen as a combination between classical reasoning and reasoning about the semantical meta-knowledge for the given logic. The semantical properties for the logics we have dealt with can be formalized in terms of Kripke-Semantics [Kri63]. The basic idea behind these semantics is the notion of *possible worlds*. Roughly spoken, they represent different interpretations of the predicate symbols and terms within the logical language. Between these worlds an *accessibility relation* $\mathbf{R}$ is defined in order to describe how the possible worlds interact. The modal operators $\Box$ and $\Diamond$ reflect quantifications over possible worlds which are accessible from a given one via $\mathbf{R}$. Thus, given a possible world w and a first order formula F then F is said to be *forced* at w (denoted $w\|\!\vdash F$) if F it is classically satisfied by the (classical) interpretation associated to w. A modal formula $\Box F$ is forced at w if $v\|\!\vdash F$ for every v with $w\mathbf{R}v$. A modal formula $\Diamond F$ is forced at w if $v\|\!\vdash F$ for at least one v with $w\mathbf{R}v$. Various modal logics are determined by combining different properties of $\mathbf{R}$ (e.g. reflexivity, symmetry, transitivity, etc.).

Proof-theoretically, this world-dependent behavior is reflected in the reduction rules for modal operators within a sequent calculus for the given logic. Applying these rules may cause the deletion of certain formulae within the current sequent. This behavior results in the non-permutability of the respective reduction rules.

From this perspective, reasoning in non-classical logics amounts to classical reasoning about the first order fragment of the non-classical language within single possible worlds combined with reasoning about the modal operators w.r.t. the whole set of possible worlds and the accessibility relation between them. While the former part is usually processed within the inference machine, the latter is not necessarily so. The following two sections present two different approaches for the treatment of this additional reasoning. The first one takes place at preprocessing time. It basically formalizes the Kripke-semantical notions within the language of classical logic. The second one extends the classical inference machine by a co-processor which does the additional reasoning. It basically checks for each classical inference-step whether it is admissible w.r.t. the given modal context.

3.1 Using Classical Theorem Proving for Constructive Logics

In contrast to explicit modal logics, intuitionistic logic does not involve special operators to denote epistemological notions. Rather, the usual logical connectives are interpreted in an epistemological way. For instance, an implicative formula of the form $A \Rightarrow B$ is read as "*we possess a construction which gives us a construction for B, provided we possess a construction for A*" [Hey71]. This notion of already "possessing" a construction can be viewed as being independent of any additional information that might be available somewhere in the future. In

this sense $A \Rightarrow B$ is intuitionistically valid w.r.t. some current knowledge, if B is contained within any possible extension of the current knowledge (including no extension at all) which contains A. Likewise, a negated formula $\neg A$ (intuitionistically read as "*there cannot be any construction for* A") is intuitionistically valid w.r.t. some current knowledge, if A is not contained within any possible extension of the current knowledge (again including no extension at all).

Now if we replace "*to possess a construction* A" (i.e. "*A is part of our current knowledge*") with "*A is forced at our current possible world*" then we end up with a modal concept. Possible knowledge extensions become possible worlds and the fact that a set of knowledge entities is a possible extension of another set is denoted by the accessibility relation. The concept of knowledge-extension is ensured by the so-called *heredity condition*. It demands that any atomic formula A which is forced in the current world w is so in any possible world v accessible from w:

$$w\|\!\vdash A \text{ and } w\mathbf{R}v \quad \Rightarrow \quad v\|\!\vdash A$$

Obviously, the accessibility relation must be reflexive and transitive because every possible set of knowledge entities is a (trivial) extension of itself and any extension of an extension of the current set in particular is an extension for the current set as well.

With these insights we are able to define a propositional intuitionistic interpretation I_j as a triple $\langle \mathbf{W}, \mathbf{R}, \iota_j \rangle$ where $\mathbf{W}$ is a (non-empty) set of worlds, $\mathbf{R}$ is a transitive and reflexive relation on $\mathbf{W}$ and ι_j is a function from $\mathbf{W}$ to subsets of the set of atomic propositional formulae such that $w\mathbf{R}v \Rightarrow \iota_j(w) \subseteq \iota_j(v)$ (i.e. the heredity condition holds) for any two $w, v \in \mathbf{R}$. We then call a formula F to be forced at some $w \in \mathbf{W}$ (denoted $w\|\!\vdash F$) iff,

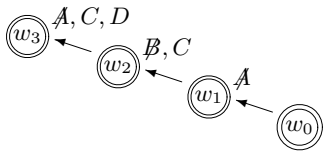
1. F is atomic and $F \in \iota_j(w)$
2. $F = F_1 \vee F_2$ and $w\|\!\vdash F_1$ or $w\|\!\vdash F_2$
3. $F = F_1 \wedge F_2$ and $w\|\!\vdash F_1$ and $w\|\!\vdash F_2$
4. $F = F_1 \Rightarrow F_2$ and $v\|\!\vdash F_2$ provided $v\|\!\vdash F_1$ for any v with $w\mathbf{R}v$
5. $F = \neg F_1$ and $v \not\|\!\vdash F_1$ for any v with $w\mathbf{R}v$

A formula F is satisfied by an intuitionistic interpretation $I_j = \langle \mathbf{W}, \mathbf{R}, \iota_j \rangle$ (denoted $I_j \models_j F$) iff it is forced at every $w \in \mathbf{W}$. A formula F is intuitionistically valid (denoted $\models_j F$) iff it is satisfied by any intuitionistic interpretation.

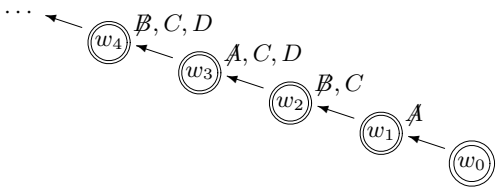
A very natural approach to deal with intuitionistic validity within classical logic would be to just formalize the above definition within the language of classical first order logic. An atomic formula A forced at some world w, for instance, would then simply become $\forall v.(wRv \Rightarrow A(v))$, where $A(v)$ encodes $v\|\!\vdash A$ and R encodes the accessibility relation $\mathbf{R}$. Likewise, $w\|\!\vdash A \Rightarrow B$ would become $\forall v.(wRv \Rightarrow A(v) \Rightarrow B(v))$. This technique is known as the *relational translation* [Moo77,Ohl91,vBe84]. Of course, to obtain completeness one has to add some axiom-formulae which encode the properties of $\mathbf{R}$ like, for instance, reflexivity ($\forall w.wRw$). The conjunction of these axiom-formulae then must classically imply the above-mentioned translation of the input formula.

A major disadvantage of this approach lies in the potential undecidability of the resulting formula since it is formulated in full first order logic with non-monadic predicate symbols (R) involving arbitrarily nested combinations of universal and existential quantifications. If we could, however, predict a finite set of accessible worlds $R_F(w)$ for a given formula F and a possible world w, then we could use finite conjunctions over the elements of these sets instead of universally quantifying over them. $w\|\!\vdash A \Rightarrow B$, for example, would become $\bigwedge_{v\in R_F(w)}(A(v) \Rightarrow B(v))$. So the main question to be solved is whether it is always possible to construct a finite countermodel (i.e. a countermodel with a finite set of possible worlds) for every propositional intuitionistic non-theorem. In principle this must be the case, since otherwise propositional intuitionistic logic would not be decidable. Our concern, however, was to find a sufficiently effective construction mechanism for such finite countermodels.

To this end we have been carefully studying the main mechanisms which may yield infinite countermodels: imagine, we would like to find a countermodel for the following intuitionistic non-theorem: $((D \Rightarrow A) \Rightarrow B) \wedge ((C \Rightarrow B) \Rightarrow A) \Rightarrow A$. Then this countermodel must contain a world w_0 where the above formula is not forced. By the definition of forcing for implicative formulae this amounts to the existence of another world w_1 accessible from w_0 with $w_1\|\!\vdash((D \Rightarrow A) \Rightarrow B) \wedge ((C \Rightarrow B) \Rightarrow A)$, i.e. $w_1\|\!\vdash(D \Rightarrow A) \Rightarrow B$ as well as $w_1\|\!\vdash(C \Rightarrow B) \Rightarrow A$, but $w_1 \not\|\!\vdash A$. Now, if $w_1\|\!\vdash(C \Rightarrow B) \Rightarrow A$ then any v accessible from there forces A provided it forces $C \Rightarrow B$. In particular this is the case for w_1 itself by the reflexivity of $\mathbf{R}$. On the other hand we know that A is not forced at w_1 so neither can be $C \Rightarrow B$. Therefore there has to be another world w_2 accessible from w_1 where C is forced but B is not. We call w_2 a world which *refutes* the implicative subformula $C \Rightarrow B$. Recall, however, that $(D \Rightarrow A) \Rightarrow B$ was forced at w_1 as well. Hence, any v accessible from w_1 forces B if it forces $D \Rightarrow A$. But then in particular this holds for w_2, so $D \Rightarrow A$ cannot be forced at w_2 since B is not forced there. This, in turn yields the existence of another world w_3 accessible from w_2 which refutes $D \Rightarrow A$, i.e. $w_3\|\!\vdash D$ but $w_3 \not\|\!\vdash A$. The following picture summarizes this situation:



Note that the above-mentioned heredity condition homomorphically extends to arbitrary formulae, i.e. any formula forced at some world remains so for all worlds accessible from there. Hence, $w_3\|\!\vdash(C \Rightarrow B) \Rightarrow A$ as well, since this was the case for w_1. Now, we face about the same situation for w_3 which we had for w_1: A is not forced at w_3, thus there must be a w_4 accessible from w_3 forcing C but not B, hence neither $D \Rightarrow A$, again yielding another w_5 that forces D but not A, and so on:



Although this model is potentially infinite, we can replace it by the finite one in the previous figure. The reason for this again stems from the heredity condition. At w_2 we find C forced for the first time. By the heredity condition this remains so for any world accessible from there. But then $C \Rightarrow B$ is forced if and only if B is. Thus, in any world accessible from w_2 $C \Rightarrow B$ becomes logically equivalent to B. Hence, in the above example we would not have to consider w_4 anymore, since knowing $w_3 \not\Vdash A$ we only need to infer $w_3 \not\Vdash B$ instead of $w_3 \not\Vdash C \Rightarrow B$. From this example we may conclude that we do not need to add any worlds refuting a certain implication as soon as our current world is accessible from one which already refutes this implication.

Besides implicative formulae also for negated formulae possible worlds have to be added to a potential countermodel. Imagine that during the attempt to construct a countermodel for a propositional formula F we need to show the existence of a possible world w_0 which does not force a specific negated subformula of F, say $\neg A$. According to the definition of intuitionistic forcing for negated formulae this amounts to the existence of at least one possible world w_1 accessible from w_0 such that $w_1 \| \!\vdash A$. But by the heredity condition A will remain forced at every world accessible from w_1. So let us advance through an arbitrary chain of possible worlds following w_1 via the accessibility relation. Again by the heredity condition we know that every formula which becomes forced along our way will remain so from there on. But then sooner or later we must have reached a world $\max_F(w_1)$ where no other world accessible from there forces more subformulae of F than $max_F(w_1)$ itself (recall that the set of subformulae of F is finite). We shall call such a world $\max_F(w_1)$ *F-maximal*. Now the crucial feature of $\max_F(w_1)$ is that for any subformula F' of F we have $\max_F(w_1) \| \!\vdash F'$ if and only if F' is classically satisfied by the set $\iota_j(\max_F(w_1))$. Hence, when we wish to argue that a negated subformula $\neg A$ of our input-formula F is not forced at some world w_0 we simply check if there is an F-maximal world $\max_F(w_1)$ forcing A, i.e. if A is classically satisfied by the model $\iota_j(\max_F(w_1))$. Thus, it becomes unnecessary to consider any further possible world accessible from $\max_F(w_1)$.

In order to obtain a suitable encoding of a potential intuitionistic countermodel for a given input formula F we first of all associate a unique function symbol $\mathtt{w}_i$ to each implicative or negated subformula of F which has polarity 0 (i.e. is a *positive part* in the terminology of [Sch77]). Then, beginning with a "root" possible world $\mathtt{w}_0$, we construct a set W of terms that are obtained by applying a proper choice of concatenations of the $\mathtt{w}_i$ to $\mathtt{w}_0$. Roughly spoken, the order of the $\mathtt{w}_i$ within these concatenations is determined by the tree ordering between the associated subformulae. No function symbol will occur more than once within a single such concatenation (reflecting the above-mentioned insights

on implicative subformulae). Function symbols associated to negated subformulae will only occur as the outermost symbol of a concatenation (reflecting the above-mentioned insights on negated formulae).

Given a term $t \in W$ (that denotes a particular possible world within a potential countermodel for the given formula) the set $R_F(t)$ of accessible worlds will then be the set of those terms $t' \in W$ that contain t as a subterm. The translation procedure itself will come in the form $\Psi(t, F')$ meaning to encode that the subformula F' is forced at the possible world denoted by the term t. It basically reflects the definition of intuitionistic forcing as mentioned above. If, for instance, $F' = F'_1 \Rightarrow F'_2$ and F' has polarity 1 (i.e. F' is possibly forced at t) then $\Psi(t, F') = \bigwedge_{t' \in R_F(t)} \Psi(t', F'_1) \Rightarrow \Psi(t', F'_2)$, meaning that if F' is forced at t then F'_2 is so at any t' accessible from t where F'_1 is forced. Conversely if F' is of polarity 0 (i.e. F' is possibly not forced at t) then $\Psi(t, F') = \Psi(\mathtt{w}_i(t), F'_1) \Rightarrow \Psi(\mathtt{w}_i(t), F'_2)$ where $\mathtt{w}_i$ is the unique function symbol associated to F'. This reflects that if F' is not forced at t then there is a possible world $\mathtt{w}_i(t)$ accessible from t where F'_1 is forced but F'_2 is not. Special care has to be taken if $\mathtt{w}_i$ already occurs somewhere in t. According to our above expositions we then know that F'_1 is already forced at t. Thus the translation will evaluate to $\Psi(t, F'_2)$ only in this case, indicating that F'_2 is not forced at t. Hence t is a possible world accessible from itself by the reflexivity of $\mathbf{R}$ where F'_1 is forced but F'_2 is not. Likewise, we do not have to extend t via $\mathtt{w}_i$ if the outermost function symbol in t is associated to a negated subformula (hence denoting a F-maximal world).

To sum up we have achieved a morphism from intuitionistic to classical logic that maps propositional input formulae to propositional output formulae (note that no quantifiers or uninstantiated terms occur in the output formula). The translation result can be directly used as an input for any classical theorem prover. No theory reasoning is required at proof time anymore — neither implicitly (through an encoding within the object-language) nor explicitly (through a modification of the inference machine).

In practice, this translation technique has turned out to be quite successful. Not only it preserves decidability, hence allowing the use of a Davis-Putnam-style prover. Compared to other techniques it reduces the search space considerably and does so the more negated subformulae occur within the given input formula. Moreover, we are quite confident to be able to lift the underlying ideas to full first order intuitionistic logic. Once we have achieved this goal we then would have to provide a technique that reconstructs the meta-knowledge reasoning from a given classical proof of a transformed intuitionistic formula in order to eventually complete our translation approach. This technique, shown in figure 2 as "Inverse logic morphism", is currently under development.

3.2 A Uniform Framework for Non-Normal Form Theorem Proving

In the previous section we have presented the first approach for dealing with additional reasoning which is caused by non-classical semantics of the logical connectives. Its basic idea was to encode these semantics into a classical first

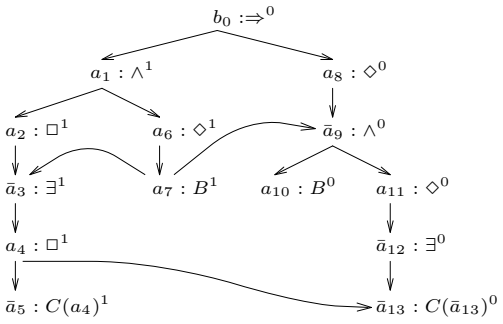
order formula at preprocessing time. In the second approach which will be described below this additional reasoning is achieved by an extension of the classical inference machine, i.e. by using a specialized co-processor.

The difference to classical normal form theorem proving essentially consists of proving theorems within a non-normal form environment. On the one hand the input formula can be used in a direct way without any translations and normal form transformations. On the other hand when considering non-classical logics the semantical information (depending on the selected logic) may be processed during the proof procedure itself which makes a single inference step more complicated. Finally this additional information is contained in the resulting proof and makes it easier to construct a readable output, i.e. to present the proof within sequent or natural deduction calculi.

For this purpose we have developed a uniform framework for non-normal form theorem proving. The resulting method basically consists of a two step algorithm, i.e. a uniform procedure finding the proofs [OK96b] and a uniform transformation procedure converting these proofs into sequent-style systems [SK96]. During the development of the algorithm we have put particular emphasis on the uniformity of our environment. First of all we have to unify all logics under consideration into one system of matrix- and sequent-style calculi, i.e. fixing the *invariant* and *variant* parts of these logics when searching for matrix proofs and transforming into sequent proofs. From this we obtain the basic structure of the corresponding procedures using the invariant parts as uniform steps. The variant parts are designed in a modular way to define the special properties depending on the selected logics. This leads to generalized invariant proof and transformation procedures where the variant parts are included as a kind of *drawer–technique* for specializing to the single logics. Furthermore this approach realizes a great degree of flexibility and can be extended to other logics in an easy way.

Non-normal form Automated Theorem Proving

The theoretical foundation for our proof procedure is a matrix characterization of logical validity developed in [Wal90] where a given formula F is logically $\mathcal{L}$–valid iff each *path* through F has at least one *complementary connection* according to the selected logic $\mathcal{L}$. This characterization can be seen as a generalization of Bibel's connection method [Bib87] to capture also non-classical logics which do not have a normal form. This is achieved by introducing the syntactical concept of *prefixes* to encode the Kripke semantics of the logics explicitly into the proof search. As a result we obtain an extension of the notion of *complementarity* depending on the logic under consideration. Whereas in the classical case two atomic formulae are complementary if they have different polarity but can be unified by some substitution on their sub-terms, for non-classical logics also the prefixes of the two atomic formulae (i.e. descriptions of their modal context in the formula tree) have to be unifiable. Thus in addition to the conventional unification algorithms our procedure requires a special *string-unification* procedure [OK96a] for making two prefixes identical. Moreover, the absence of normal



x	$pre(x)$
b_0, a_1, a_2, a_6, a_8	b_0
$\bar{a}_3, a_4$	$b_0\bar{a}_3$
$\bar{a}_5$	$b_0\bar{a}_3\bar{a}_5$
a_7	b_0a_7
$\bar{a}_9, a_{10}, a_{11}$	$b_0\bar{a}_9$
$\bar{a}_{12}, \bar{a}_{13}$	$b_0\bar{a}_9\bar{a}_{12}$

$\sigma_L(\bar{a}_3) = \sigma_L(\bar{a}_9) = a_7$
$\sigma_L(\bar{a}_{12}) = \bar{a}_5$
$\sqsubset_L = \{(a_7, \bar{a}_3), (a_7, \bar{a}_9)\}$
$\sigma_Q(\bar{a}_{13}) = a_4$
$\sqsubset_Q = \{(a_4, \bar{a}_{13})\}$
$\sigma_L^{\#}(pre(a_4)) = b_0a_7$
$\sigma_L^{\#}(pre(\bar{a}_{13})) = b_0a_7\bar{a}_5$

Fig. 3. Formula tree, prefix table, and substitutions of the formula from Example 1

forms requires a proof procedure which operates on the formula tree instead of a clause set. The notions of *current clauses* and *active paths* which are used to guide the development of a classical matrix proof in normal form have to be extended as well. The following example illustrates the appropriately extended characterization of logical validity based on non-classical connections.

Example 1. (Non-classical validity) Consider the modal formula $F \equiv \Box\exists x.\Box C(x) \wedge \Diamond B \Rightarrow \Diamond(B \wedge \Diamond\exists x.C(x))$. Its formula tree (denoted by an ordering $\ll$) is shown in figure 3 where we use *positions* for a unique indexing of all subformulae. Additionally we have associated each position x with the main operator of the corresponding subformula and its polarity. For a uniform treatment of quantifier unification σ_Q and the special string unification σ_L (specifying the logic $\mathcal{L}$) we distinguish variable positions (marked with an overbar) and constants. The table next to the tree shows the prefixes corresponding to the positions of $\ll$. Basically a prefix $pre(x)$ assigned to a position x is a string consisting of the root b_0 and all positions which dominate x in $\ll$ and are successors of positions associated to a modal main operator. For $S5$ the prefixes consist only of the last positions in these strings.

$$\left[\left[b_0\bar{a}_3\bar{a}_5 : C(a_4)^1 \quad b_0a_7 : B^1 \right] \quad \begin{bmatrix} b_0\bar{a}_9 : B^0 \\ b_0\bar{a}_9\bar{a}_{12} : C(\bar{a}_{13})^0 \end{bmatrix} \right]$$

Fig. 4. Matrix representation of the formula from Example 1

Using the well–known tableaux classification of subformulae [Wal90,Fit83] we obtain the corresponding *matrix* (two-dimensional representation) shown in figure 4 where each atom is represented next to its prefix. The vertically arranged atoms correspond to a branching at some β–formula in an analytic tableau. The horizontally written atoms are of formula-type α which does not cause a split

in a tableau proof. The matrix has two paths (defined on the set of positions associated with the atoms) $\{\bar{a}_5, a_7, a_{10}\}$ and $\{\bar{a}_5, a_7, \bar{a}_{13}\}$. The corresponding connections are $\{a_7, a_{10}\}$ and $\{\bar{a}_5, \bar{a}_{13}\}$. A *combined substitution* $\sigma = \langle \sigma_Q, \sigma_L \rangle$ can be computed as shown below the prefix table. The induced relations $\sqsubset_Q$ and $\sqsubset_L$ have already been integrated into the tree ordering $\ll$ and result in a relation $\lhd \; = \ll \cup \sqsubset_Q \cup \sqsubset_L$. The irreflexivity of $\lhd$ is *one* required condition for σ to be $\mathcal{L}$–*admissible*. Furthermore σ_L respects semantical conditions w.r.t. the $\mathcal{L}$–accessibility relation R_0 for the modal logics $D, D4, T, S4, S5$. The interaction between σ_Q and σ_L satisfies the cumulative and constant domain conditions for these logics, too (see [Wal90] for detailed definitions). Summarizing *all* these conditions we obtain σ being $\mathcal{L}$–*admissible* for these cases. It is easy to verify that σ makes the two connections σ–complementary and hence the corresponding paths as well. Thus F is shown to be $\mathcal{L}$–valid.

F can also be proven $\mathcal{L}$–valid in $T, S4$ and $S5$ under varying domains by extending σ_L. For $T, S4$ we must add $\sigma_L(\bar{a}_5) = \emptyset$, and for $S5$ $\sigma_L(\bar{a}_5) = a_7$ satisfying the corresponding condition.

A uniform and compact presentation of Wallen's matrix characterizations for classical, intuitionistic, and various modal logics can be found in [OK96b] or [SK96]. The proof procedure working on these compactly represented matrix characterizations has been developed in [OK96b] involving extended concepts of paths, (open) subgoals, and multiplicities for non-normal form matrices. The basic mechanism for proof search is similar to Bibel's extension procedure [Bib87]. It consists of a short and uniform (invariant) algorithm for connection driven path checking of a formula F. By following connections we avoid the notational redundancies occurring in sequent or tableaux-based proof procedures (driven by the logical connectives) whereas by considering non-normal form we reduce the size of the matrix to be searched through by a connection based proof search. The variant component of the procedure determining the complementarity of the connections strongly depends on the underlying logic $\mathcal{L}$. It has been successfully realized using the string-unification procedure [OK96a,OK96b] on the corresponding prefixes of the connected atoms. The modular design of this unification procedure and its integration into the search algorithm allows us to treat a rich variety of logics in a uniform, efficient, and simple way. We could even extend it to logics not yet considered simply by providing a component for checking complementarity according to a matrix characterization of validity in this logic.

Converting non-normal form matrix proofs into sequent-style systems

Uniform methods for proof search in a non-normal form environment are only the first step in solving problems from various application domains via ATP-methods. In a second step we also have to develop a uniform algorithm which transforms non-normal form matrix proofs into a comprehensible form. This step is necessary because the efficiency of automated proof methods based on matrix characterizations strongly depends on a compact representation of a proof. This makes it very difficult to use an automated generated proof as guideline for solving application problems.

Since the development of automated theorem provers attempts have been made to convert machine proofs into a humanly comprehensible form. Since a classical non-normal form proof explicitly preserves structural information (in contrast to a matrix proof in clause form) a corresponding sequent proof can be extracted in a direct way. For non-classical logics this easy approach becomes rather complicated since semantical information (also contained in the output "proof code") has to be integrated. In [SK96] unified and compact representations of the matrix characterizations [Wal90], *conventional* sequent calculi, and *prefixed* sequent calculi have been presented for all logics under consideration. For the representation of the sequent calculi we use a uniform system of tables to specify the variant parts of the logics. The tables reflect conditions on rule applications depending on different Kripke-semantics of the logics. For this reason these tables can be seen as the counterpart to the string unification procedure, i.e. variant parts in the matrix characterizations. The invariant parts of the sequent systems are given by one basic system of inference rules for the logical connectives.

In [SK96] we have developed a transformation procedure working on this uniform framework of logical calculi. Starting with a matrix proof of a formula F we obtain a combined substitution $\sigma = \langle \sigma_Q, \sigma_L \rangle$, which induces, together with the tree ordering of F, an ordering $\lhd$ on the nodes of the formula tree. This ordering efficiently encodes the fact that within a (top-down) sequent or tableaux proof there are certain restrictions on the order of rule applications which are necessary to "reduce" the formula F in order to reach the axioms of the calculus. Thus the basic idea of our algorithm is very simple. Essentially we have to traverse $\lhd$, to select appropriate sequent rules according to the subformula represented by each node and its polarity, and to keep track of all the subgoals already solved. No search will be involved in the transformation. From the uniform representation of the sequent calculi we obtain a set of tables to be consulted by the algorithm when determining an appropriate sequent rule corresponding to a given node. Positions of type β cause a sequent proof to split into two independent subproofs and force us to determine which sub-relations of $\lhd$ will be relevant in each subproof. Finally, since $\lhd$ does not uniquely determine the order of rule applications, we have to identify proof-relevant positions in the formula tree which have priority over others. These may again depend on the underlying logic and force us to insert *wait*-labels to certain nodes in order to keep them from being reduced too early.

Prefixed sequent systems. Because of a strong similarity between the matrix characterizations and Fitting's *prefixed tableaux systems* [Fit69] we have developed our algorithm in a first phase to convert a matrix proof into a *prefixed* sequent proof. From the prefixes computed during proof search via string unification we obtain the prefixes for the sequent proof in a direct way. Hence, the semantical information is preserved for prefix construction in the prefixed sequent proof. The input of the uniform transformation algorithm $TOTAL$ is given by a reduction ordering $\lhd$ and a logic $\mathcal{L}$ such that $\lhd$ represents a matrix proof in $\mathcal{L}$. $TOTAL$ traverses $\lhd$ to compute a *totalization* of the partial ordering

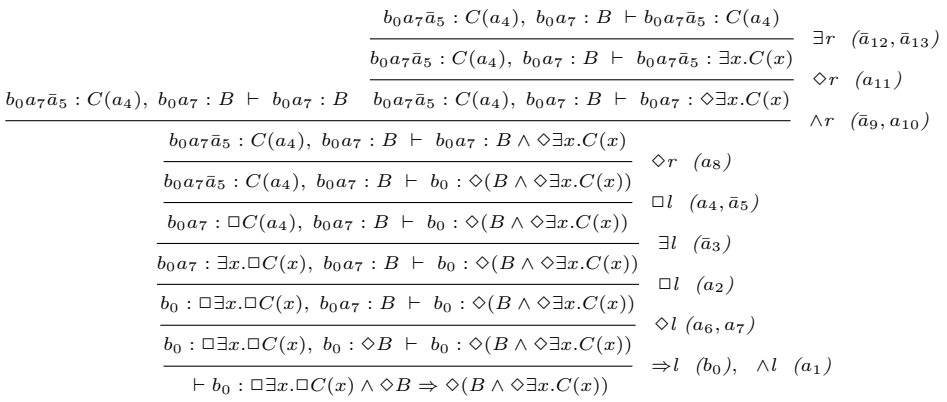


Fig. 5. Prefixed sequent proof of the formula from Example 2

$\lhd$. From this we obtain a linear sequence of sequent rules which forms a proof in the prefixed sequent system corresponding to $\mathcal{L}$. The specific sequent rules and their parameters (i.e. prefixes and terms) are determined by the main operator of the currently visited node in $\lhd$ and by the substitutions σ_L and σ_Q. The *correct* application of sequent rules and construction of prefixes will be guaranteed by using our table system according to the selected logic $\mathcal{L}$. For the details we refer again to [SK96].

Example 2. (Conversion into prefixed sequent systems) From Example 1 we take the formula $F \equiv \Box\exists x.\Box C(x) \wedge \Diamond B \Rightarrow \Diamond(B \wedge \Diamond\exists x.C(x))$, the substitutions σ_L, σ_Q, and the reduction ordering $\lhd$ generated from the matrix proof. For *all* positions we obtain the prefixes under σ_L shown in the following table:

x	b_0, a_1, a_2, a_6, a_8	$\bar{a}_3, a_4, a_7, \bar{a}_9, a_{10}, a_{11}$	$\bar{a}_5, \bar{a}_{12}, \bar{a}_{13}$
$\sigma_L^{\#}(pre(x))$	b_0	b_0a_7	$b_0a_7\bar{a}_5$

We start our algorithm *TOTAL* traversing the reduction ordering $\propto^\star$ for the logics $D, D4, T, S4$ with cumulative domains. The total ordering of positions encoding the sequent rules is given by

$$[b_0, a_1, a_6, a_7, a_2, \bar{a}_3, a_4, \bar{a}_5, a_8, \bar{a}_9, a_{10}, a_{11}, \bar{a}_{12}, \bar{a}_{13}]$$

The corresponding prefixed sequent proof respecting this ordering looks like shown in figure 5. The goal sequent of the prefixed sequent proof is given by $\vdash b_0 : F$. For explanation of the traversing process we use the picture of $\lhd$ from Example 1. The conversion algorithm starts with positions b_0, a_1 where we construct the the rules $\Rightarrow r : b_0$ and $\wedge l : b_0$ (in the following we write the constructed prefix and quantifier parameter behind the rule name). After skipping a_8 we are blocked at $\bar{a}_9$ because of the reduction ordering $(a_7, \bar{a}_9) \in \sqsubset_L$. We continue the conversion at a_6 which can be seen as a goal-oriented selection

in order to remove the effect of blocking at $\bar{a}_9$. We obtain the rule $\Diamond l : b_0a_7$ and the atom $B^1 : b_0a_7$ solving a_7. Now we can delete the blocking arrow pointing to $\bar{a}_9$. Next we skip a_2 and, reaching $\bar{a}_3$, construct the two rules $\Box l : b_0a_7$ and $\exists l : b_0a_7; a_4$ (introducing the eigenvariable a_4 and deleting the arrow blocking $\bar{a}_{13}$). Visiting a_4 and $\bar{a}_5$ the atom $C(a_4)^1 : b_0a_7\bar{a}_5$ can be isolated in the sequent proof. We make up the reduction $\Diamond r : b_0a_7$ at a_8 and afterwards, the application of $\wedge l : b_0a_7$ at β–position $\bar{a}_9$ forces a split of the reduction ordering into two independent suborderings $\lhd_1, \lhd_2$ (detailed definitions given in [SK96]). For $\lhd_1$, we solve a_{10} having the atom $B^0 : b_0a_7$, and hence an axiom rule with $B^1 : b_0a_7$ (a_7 already solved). For reducing $\lhd_2$ we visit $a_{11}, \bar{a}_{12}$ and $\bar{a}_{13}$ obtaining $\Box r : b_0a_7\bar{a}_5$ and $\exists r : b_0a_7\bar{a}_5; a_4$, where $\sigma_Q(\bar{a}_{13}) = a_4$ has been used at $\bar{a}_{13}$. Finally, the sequent proof will be finished solving $\bar{a}_{13}$, i.e. applying the axiom rule with $C(a_4)^0 : b_0a_7\bar{a}_5$.

The eigenvariable a_4 is associated with the prefix b_0a_7 and the prefix at $\bar{a}_{13}$ is given by $b_0a_7\bar{a}_5$. We use a_4 for the quantifier reduction $\exists r$ with $\sigma_Q(\bar{a}_{13}) = a_4$ satisfying the cumulative domain condition on the semantics of the considered logics, i.e. $b_0a_7\, R_0\, b_0a_7\bar{a}_5$ (where R_0 denotes the accessibility relation on prefixes [Fit83]). The sequent proof can be extended to $T, S4$ for varying domains by extending the modal substitution $\sigma_L(\bar{a}_5) = \emptyset$.

Conventional sequent calculi. The transformation algorithm $TOTAL$ developed in the first phase realizes conversion into prefixed sequent systems only. The advantage lies in an easy re-integration of semantical meta-knowledge by using the prefixes from the matrix proof in a direct way. Furthermore we are able to capture modal logics for which there are no cut-free conventional sequent calculi (for example $S5$ and all modal logics under consideration with constant domains). The disadvantage of converting into prefixed sequent systems is given by little practical use of these calculi in real application systems.

The latter fact leads to the second phase of developing a uniform conversion procedure, i.e. to extend the algorithm $TOTAL$ into $TOTAL^\star$ which transforms non-normal form matrix proofs into conventional sequent calculi [Gen35,Wal90]. For non-classical logics we have to consider the fact that prefixes cannot be used explicitly. The non-permutabilities of inference rules are now encoded by the structure of the rules themselves which may delete sequent formulae during a reduction. In this case the absence of prefixes makes it necessary to deal with β-splits and to consider the additional priorities of proof-relevant positions while the essential algorithm remains unchanged.

Because of the compact representation of matrix proofs it is not trivial to extract the sequent proofs in a direct way. We have to develop a concept for re-building the notational redundancies which were eliminated in the matrix proof but still have to occur in a sequent proof. The main problem occurs when splitting at β–positions in the transformation process. In this case one has to decide which sub-relations of $\lhd_1, \lhd_2$ are still relevant in the matrix (sub-)proof and which subformulae are necessary for further reductions closing the corresponding branch in the sequent proof.

In our general transformation algorithm a more detailed consideration of β–nodes can have two different effects. When creating prefixed sequent systems (denoting a *lower* transformation level) all semantical informations concerning the specific logic are encoded by the prefixes of the sequent formulae. In this case deleting *irrelevant sub-relations* after a β–split can be viewed as an optimization which eliminates redundancies. In contrast to that a conversion into conventional sequent calculi without explicit prefixes has to take into account that the semantics of a logic, in particular the non-permutabilities of rule applications, is now completely contained in the structure of the inference rules. Thus the reduction ordering $\lhd$ has to be extended by additional ordering constraints (called *wait*–labels) which cause deletion of sub-relations after β–splits to be more than an optimization feature. Now these operations will be essential for preserving the completeness of our transformation procedure.

In [Sch95] we have developed a concept of reductions on non-normal form matrix proofs for intuitionistic logic which respects these optimization- and completeness features. In [SK96] we have extended this concept to modal logics, especially the definitions of required *wait*–labels for making the reduction ordering $\lhd$ complete. These *wait*–labels depend on the selected logic $\mathcal{L}$ and, hence can be seen as an additional variant part for conversion in conventional sequent calculi. The whole concept of β–splits is one of the theoretical foundations for the uniform transformation algorithm $TOTAL^{\star}$. One basic principle is the concept of proof relevant positions, where a position x is said to be *relevant* if some successor leaf position c is part of a connection $\{c, c_2\}$. However, we were able to show that for some of the logics $K, K4$ and $D4_{co}$ (*co* means in *constant domains*) this principle fails. There are theorems which have *pure* relevant sub-formulae not involved in any connection of the matrix proof [1]. For all other logics under consideration one should use the deletion of reductions after splitting even when creating prefixed sequent proofs.

Both parts, the proof search and the transformation procedure can be integrated into an application system (like a proof development system) by providing appropriate interfaces similar to the ones described in [KOS95]. We hope that this general methodology will make ATP useful in a rich variety of application domains.

4 Future Work

This paper has provided an overall view of the development of an ATP-system which is able to handle three different types of applications in a tailored way. In this last section a short outlook into future work will be given. We restrict this outlook mainly to the application of automated theorem proving techniques for the development of correct programs from formal specifications which is one of the three areas of applications discussed in the paper.

[1] $\Diamond B$ in the K-theorem $\Box A \wedge \Diamond B \Rightarrow \Diamond A$, and $\Diamond\Diamond B$ in the $D4_{co}$–theorem $\forall x.\Box\Box A(x) \wedge \Diamond\Diamond B \Rightarrow \Diamond\forall x.A(x)$.

Currently, we are developing a C-implementation of our connection-based proof procedure for intuitionistic logic [OK95,OK96b] and of our method for converting matrix proofs into intuitionistic sequent proofs [SK95,SK96]. We also implement an interface which integrates the resulting software with the interactive proof/program development system NuPRL [C$^+$86]. This will enable to use an efficient proof procedure interactively [KOS95] as a tactic within the NuPRL system for solving sub-problems from first order logic, the need for which arises during the process of deriving programs with the NuPRL system.

Furthermore we intend to investigate how inductive proof methods can be integrated into program synthesis systems by use of the same technology as just described. Further research focuses on providing a rigorous formalization of the meta-theory of programming in order to raise the level of formal reasoning in formal program synthesis from low level calculi to one comprehensible for programmers and to implement program development strategies on this level [Kre96]. These are considered to be steps on the way towards enabling a user of a program development system to focus on the key ideas in program design while being freed from formal details which in addition are needed to ensure correctness.

It seems very likely that our techniques can be generalized to some subset of linear logic and other calculi in use in artificial intelligence and computer science. In fact we are currently working on a matrix-characterization for validity in linear (and other) logics in order to provide the basis for extending our combined ATP system to a larger variety of application domains.

The last topic of future research mentioned here in the context of the material presented in this paper concerns the enhancement of efficiency of the resulting system. We plan to investigate how the techniques, used in systems like Setheo [LBB92] and KoMeT [BBE$^+$94] and resulting in the high performance of these systems, can be imported into our combined system. Examples of the kind of techniques we have in mind are preprocessing techniques and the use of typing information during unification. As mentioned in Section 3.1 we also aim at using these classical proof systems directly as inference machines via semantics-based translation techniques [Kor96]. We plan to lift the existing approach to full intuitionistic predicate logic. In order to re-integrate the generated proofs into the NuPRL-environment we are currently developing an appropriate postprocessing mechanism.

Experiments with examples from applications will eventually have to demonstrate whether after such provisions non-normal form proof techniques have an advantage over conventional clause-form theorem provers which is an open question of practical relevance up to this day.

References

[And80] P. Andrews. *Transforming matings into natural deduction proofs.* In *CADE–5*, LNCS 87, pp. 281–292. Springer, 1980.

[And91] P. Andrews. *More on the problem of finding a mapping between clause representation and natural-deduction representation. Journal of Automated Reasoning*, Vol. 7, pp 285–286, 1991.

[BC85] J. L. Bates, R. L. Constable. *Proofs as programs. ACM Transactions on Programming Languages and Systems*, 7(1):113–136, January 1985.

[Bib87] W. Bibel. *Automated Theorem Proving.* Vieweg Verlag, ^{2}1987.

[BBE^{+}94] W. Bibel, S. Brüning, U. Egly, T. Rath. *Komet.* In *CADE–12*, LNAI 814, pp. 783–787. Springer, 1994.

[Bra75] D. Brand *Proving theorems with the modification method.* In J. Hopcroft e. a., eds. *SIAM Journal of Computing Vol. 4*, pp. 412–430. Philadelphia, 1975.

[C^{+}86] R. L. Constable et. al. *Implementing Mathematics with the NuPRL proof development system.* Prentice Hall, 1986.

[DGH^{+}94] B. I. Dahn, J. Gehne, T. Honigmann, L. Walther, A. Wolf. *Integrating Logical Functions with ILF*, Preprint 94-10, Humboldt University Berlin, 1994.

[ER96] U. Egly, T. Rath. *On the practical value of different definitional translation to normal form.* To appear in *CADE–13*, Springer, 1996.

[Fit69] M. C. Fitting. *Intuitionistic logic, model theory and forcing.* Studies in logic and the foundations of mathematics. North–Holland, 1969.

[Fit83] M. C. Fitting. *Proof Methods for Modal and Intuitionistic Logic.* D. Reidel, 1983.

[Gen95] K. Genther. *Repräsentation von Konnektionsbeweisen in Gentzen-Kalkülen durch Transformation und Strukturierung.* Tech. Rep. AIDA-95-12, TH Darmstadt, 1995.

[Gen35] G. Gentzen. *Untersuchungen über das logische Schließen. Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

[Hey71] A. Heyting. *Intuitionism — An Introduction.* North Holland Publishing Company, Amsterdam, ^{3}1971.

[Kor93] D. Korn. *KonSequenz — ein Konnektionsmethoden-gesteuertes Sequenzenbeweisverfahren.* Master-thesis, TH Darmstadt, 1993.

[Kor96] D. Korn. *Efficiently Deciding Intuitionistic Propositional Logic via Translation into Classical Logic.* Tech. Rep. AIDA-96-09, TH Darmstadt, 1996.

[Kre96] C. Kreitz. *Formal Mathematics for Verifiably Correct Program Synthesis. Journal of the Interest Group in Pure and Applied Logics (IGPL)*, 4(1):75–94, 1996.

[KOS95] C. Kreitz, J. Otten, S. Schmitt. *Guiding Program Development Systems by a Connection Based Proof Strategy.* In M. Proietti, editor, 5^{th} *International Workshop on Logic Program Synthesis and Transformation*, LNCS 1048, pp. 137–151, Springer Verlag, 1996.

[Kri63] S. A. Kripke. *Semantical analysis of modal logic I. Normal modal propositional calculi.* Zeitschrift f. mathematische Logik u. Grundlagen d. Mathematik 9, pp 67–96, 1963.

[LBB92] R. Letz, J. Schumann, S. Bayerl, W. Bibel. Setheo: A high-performance theorem prover. *Journal of Automated Reasoning*, 8:183–212, 1992.

[Lin89] C. Lingenfelder. *Structuring computer generated proofs*. *IJCAI-89*, 1989.

[Lin90] C. Lingenfelder. *Transformation and Structuring of Computer Generated Proofs*. PhD thesis, Universität Kaiserslautern, 1990.

[Moo77] R. C. Moore. *Reasoning about Knowledge and Action* IJCAI-77, pp 223–227, Stanford, California 94305, 1977.

[Ohl91] H. J. Ohlbach. *Semantics–Based Translation Methods for Modal Logics. Journal of Logic and Computation*, Vol. 1, no. 6, pp 691–746, 1991.

[OK95] J. Otten, C. Kreitz. *A connection based proof method for intuitionistic logic. TABLEAUX-95*, LNAI 918, pp. 122–137, Springer, 1995.

[OK96a] J. Otten, C. Kreitz. *T-string-unification: unifying prefixes in non-classical proof methods. TABLEAUX–96*, LNAI 1071, pp. 244–260, Springer, 1996.

[OK96b] J. Otten, C. Kreitz. *A Uniform Proof Procedure for Classical and Non-Classical Logics.* To appear in *KI–96*, Springer, 1996.

[Pfe87] F. Pfenning. *Proof Transformations in Higher-Order Logic.* PhD thesis, Carnegie Mellon University, January 1987.

[Rob65] J. A. Robinson., *A machine-oriented logic based on the resolution principle Journal of ACM*, Vol. 12, pp 23–41, 1965.

[RW69] G. Robinson and L. Wos., *Paramodulation and theorem proving in first order theories with equality.* In B. Meltzer and D. Michie, eds., *Machine Intelligence 4*, pp. 135-150, Edinburgh University Press, 1969.

[Sch95] S. Schmitt. *Ein erweiterter intuitionistischer Sequenzenkalkül und dessen Anwendung im intuitionistischen Konnektionsbeweisen.* Tech. Rep. AIDA-95-01, TH Darmstadt, 1995.

[SK95] S. Schmitt, C. Kreitz. *On transforming intuitionistic matrix proofs into standard-sequent proofs*. *TABLEAUX–95*, LNAI 918, pp. 106–121, Springer, 1995.

[SK96] S. Schmitt, C. Kreitz. *Converting Non-Classical Matrix Proofs into Sequent-Style Systems*. To appear in *CADE–13*, Springer, 1996.

[Sch77] K. Schütte. *Proof theory*. Springer-Verlag, New York, 1977.

[vBe84] J. Van Benthem. *Correspondence Theory*. In: *D. Gabbay and F. Guenther (eds.): Handbook of Philosophical Logic*, Vol. II, pp. 167–247, D. Reidel Publishing Company, Dordrecht, 1984.

[Wal90] L. Wallen. *Automated deduction in non-classical logics.* MIT Press, 1990.

[Wos90] L. Wos. *The problem of finding a mapping between clause representation and natural-deduction representation. Journal of Automated Reasoning*, Vol. 6, pp 211–212, 1990.