Journal of Information and Computation 162(1–2):226–254 ©Academic Press, 2000.

A Uniform Procedure for Converting Matrix Proofs into Sequent-Style Systems

Christoph Kreitz Stephan Schmitt*

Department of Computer Science, Cornell University
Ithaca, NY 14853, USA
{kreitz,steph}@cs.cornell.edu

Abstract. We present a uniform algorithm for transforming machine-found matrix proofs in classical, constructive, and modal logics into sequent proofs. It is based on unified representations of matrix characterizations, of sequent calculi, and of *prefixed* sequent systems for various logics. The peculiarities of an individual logic are described by certain parameters of these representations, which are summarized in tables to be consulted by the conversion algorithm.

1 Introduction

Logical reasoning is a particularly precise form of human problem solving that is used especially in scientific applications. Because of its precision it lends itself to automation more easily than reasoning in general. In classical predicate logic, theorem provers based on resolution [23, 31] and the connection method [5, 6, 14, 4] have demonstrated that logical deduction can be simulated sufficiently well on a computer. Recently the characterizations of logical validity, which underly these systems, have been extended to intuitionistic logic and the modal logics $K, K4, D, D4, T, S4$, and $S5$ [17, 28, 29, 27]. On this basis the existing proof *methods* have been extended accordingly [18, 19] in order to develop a coherent theorem prover that can deal with a variety of logics and many applications requiring mathematical reasoning.

The use of automated theorem proving (ATP) in practical applications, however, causes another problem. As the efficiency of proof search strongly depends on a compact representation of a proof, 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 must be transformed into a more readable form.

Techniques for a humanly comprehensible presentation of machine-found proofs have been described in [1, 32, 2, 21, 15, 16]. They transform classical resolution- or matrix proofs into natural calculi such as $\mathcal{LK}$ or $\mathcal{NK}$ [11]. Other approaches try to present them even in natural language [8]. In [24] we have developed an algorithm for converting *intuitionistic* matrix proofs into the sequent calculus $\mathcal{LJ}$ to support program development in interactive proof assistants [13].

In this paper we present a transformation procedure that allows a *uniform* treatment of classical, constructive, and modal logics. It will support the integration of automated theorem proving into a rich variety of application systems for mathematics, program development, planning, and other areas of logical reasoning, as illustrated in Figure 1. A proof task to be solved by the ATP system will be extracted by the user or a strategy of the application system (e.g. Mathematica [30], NuPRL [7], and Isabelle [20]). After selecting the particular logic (classical, intuitionistic, or modal) a uniform proof procedure will create a machine-proof (e.g. a matrix proof in non-normal form). This proof will be converted by a uniform procedure into a sequent or natural deduction proof and can then be interpreted immediately by the application system.

* This work is supported by the German Academic Exchange Service DAAD with a fellowship to the second author.

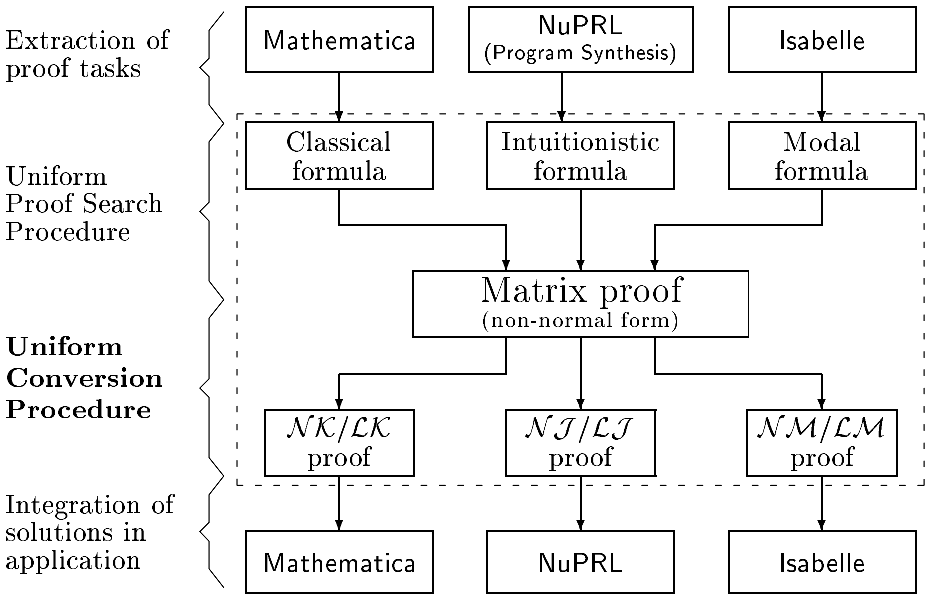


Fig. 1. Integrating Automated Theorem Proving into Application Systems

Since currently only matrix-based proof methods are able to handle different logics in a uniform way [19] our starting point will be a proof according to Wallen's matrix characterizations [29] of logical validity. Here a formula F is valid if every path through a matrix representation of F contains at least one *complementary* pair of atomic formulae. In classical logic, complementarity means that the two atomic formulae have different *polarity* and that their subterms can be unified. In non-classical logics, also the so-called *prefixes* of the two atoms, i.e. strings describing their position in the formula tree, must be unifiable. The tree ordering of F, together with the two unifiers, induces an ordering $\propto^\star$ on the nodes of the formulae tree, which can be seen as an encoding of the order of rule applications within a sequent proof for F – the goal of our transformation procedure.

The basic idea of our algorithm is simple. We traverse the ordering $\propto^\star$, select sequent rules according to the subformula represented by each node and its polarity, and keep track of all the subgoals of the sequent proof that are already solved. There are, however, several subtle details that need to be dealt with.

To achieve uniformity we need unified representations of both the matrix characterizations and the sequent calculi for classical, intuitionistic, and modal logics. For the latter, we had to develop schematic inference rules whose parameters encode a specific logic and have to be consulted by our algorithm when it determines the sequent rule that "reduces" a given node. Secondly, positions of tableau type β cause a sequent proof to split into two independent subproofs. In each subproof the algorithm must determine which subrelations of $\propto^\star$ will still be relevant. Finally, the ordering $\propto^\star$ does not uniquely determine the order of sequent rules to be applied. In some cases the algorithm has to identify proof-relevant positions that have priority over others and insert *wait*-labels at nodes that should not yet be reduced. These *wait*-labels again depend on the underlying logic.

Methodically we proceed in two steps. Because of a similarity between matrix calculi and Fitting's prefixed tableau systems [10] we first show how to convert a matrix proof into a *prefixed* sequent proof. This algorithm, which we present in Section 3, only requires unified representations of the calculi. Afterwards we extend our algorithm into one that creates conventional sequent proofs (Section 4). Here the absence of prefixes makes it necessary to deal with proof-relevant positions and the effects of β-splits. A prerequisite for both algorithms are unified representations of matrix characterizations, sequent calculi, and prefixed sequent systems, which we will develop in Section 2.

α	$\langle A \wedge B, 1\rangle$	$\langle A \vee B, 0\rangle$	$\langle A \Rightarrow B, 0\rangle$	$\langle \neg A, 1\rangle$	$\langle \neg A, 0\rangle$
α_1	$\langle A, 1\rangle$	$\langle A, 0\rangle$	$\langle A, 1\rangle$	$\langle A, 0\rangle$	$\langle A, 1\rangle$
α_2	$\langle B, 1\rangle$	$\langle B, 0\rangle$	$\langle B, 0\rangle$	–	–

β	$\langle A \Rightarrow B, 1\rangle$	$\langle A \vee B, 1\rangle$	$\langle A \wedge B, 0\rangle$
β_1	$\langle A, 0\rangle$	$\langle A, 1\rangle$	$\langle A, 0\rangle$
β_2	$\langle B, 1\rangle$	$\langle B, 1\rangle$	$\langle B, 0\rangle$

ϕ	$\langle \forall x.A, 1\rangle$	$\langle \neg A, 1\rangle$	$\langle A \Rightarrow B, 1\rangle$	$\langle P, 1\rangle$
ϕ_0	$\langle \forall x.A, 1\rangle$	$\langle \neg A, 1\rangle$	$\langle A \Rightarrow B, 1\rangle$	$\langle P, 1\rangle$
ψ	$\langle \forall x.A, 0\rangle$	$\langle \neg A, 0\rangle$	$\langle A \Rightarrow B, 0\rangle$	$\langle P, 0\rangle$
ψ_0	$\langle \forall x.A, 0\rangle$	$\langle \neg A, 0\rangle$	$\langle A \Rightarrow B, 0\rangle$	$\langle P, 0\rangle$

γ	$\langle \forall x.A, 1\rangle$	$\langle \exists x.A, 0\rangle$
$\gamma_0(t)$	$\langle A[x\backslash t], 1\rangle$	$\langle A[x\backslash t], 0\rangle$
δ	$\langle \forall x.A, 0\rangle$	$\langle \exists x.A, 1\rangle$
$\delta_0(a)$	$\langle A[x\backslash a], 0\rangle$	$\langle A[x\backslash a], 1\rangle$
ν	$\langle \Box A, 1\rangle$	$\langle \Diamond A, 0\rangle$
ν_0	$\langle A, 1\rangle$	$\langle A, 0\rangle$
π	$\langle \Box A, 0\rangle$	$\langle \Diamond A, 1\rangle$
π_0	$\langle A, 0\rangle$	$\langle A, 1\rangle$

Table 1. Primary and secondary types of formulae

2 Unified Representations of Proof Calculi

We present unified representations of matrix calculi, sequent calculi, and prefixed sequent systems for classical logic (C), intuitionistic logic (J), and the modal logics $K, K4, D, D4, T, S4, S5$ with their cumulative, varying and constant domain variants concerning the Kripke-semantics of these logics [9, 10]. In our representations we shall separate the aspects common to all these logics from those that depend on a particular logic. This separation allows a compact presentation of each calculus and is crucial for the development of a uniform algorithm that translates between different calculi. The structure of the algorithm will depend only on the *invariant* parts whereas the *variant* parts will be stored in tables expressing the values of certain parameters in the uniform description.

2.1 Matrix Calculi

Matrix characterizations of logical validity are the basis for an efficient proof search in classical and non-classical logics. They avoid notational redundancies contained in mathematical languages or sequent calculi and allow a very compact representation of a formal proof. Originally developed as foundation of Bibel's connection method for classical logic [5, 6], they have later been extended to non-classical logics by Wallen [29] and serve as a basis for a uniform proof method for a rich variety of logics [19]. Since our starting point will be a *given* matrix proof we will present only the basic ideas and syntactical concepts and refer to [29] or [19] for details, semantical justifications, and aspects of proof search.

Position trees, Types, and Prefixes. The basic structure for representing matrix proofs is a *tree ordering* $\ll$, which will be constructed from a formula tree. We classify a formula A and its subformulae according to the tableau scheme in Table 1. We use the concept of *signed formulae* where each subformula B of A receives a *polarity* $k \in \{0, 1\}$ depending on a positive (0) or negative (1) occurrence of B in A (starting with $\langle A, 0\rangle$). Each signed (sub)formula $\langle B, k\rangle$ has *primary* type *Ptype* according to its tableau class and a *secondary* type *Stype* according to its immediate parent formula.

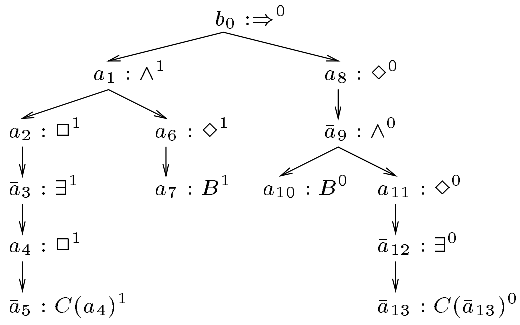
We associate each subformula $\langle B, k\rangle$ of $\langle A, 0\rangle$ with a *position* x in $\ll$. B is called the *label* $lab(x)$ of x, k its *polarity* $pol(x)$, and $Ptype(x)$, $Stype(x)$ its *Ptype* and *Stype*. We denote a signed formula at position x by $sform(x) = \langle lab(x), pol(x)\rangle$. At a γ- or δ-position x the actual variable in $lab(x)$ will be replaced in the successor formula by the name of the corresponding γ_0- or δ_0-position. As a result the positions occur directly as variables in all formulae and we can use a uniform mechanism to define substitutions on terms, positions, and the reduction ordering. Within intuitionistic logic we additionally have to insert ϕ and ψ positions into $\ll$ *before* all positions that represent so-called *special* formulae (see Table 1, where $\langle P, k\rangle$ denotes an atom). These *special* positions do not encode reductions and are necessary only for proof search.

To represent *all* the formulae that are necessary for proving $\langle A, 0\rangle$ we extend $\ll$ by copies of subformulae used more than once in the matrix proof. These *generative* formulae have *Ptype* γ for all logics and, in addition, ν for modal logics and ϕ for J. A *multiplicity* $\mu(y) \in \mathbb{N}$ is assigned to all positions y in $\ll$, where $\mu(y) \geq 1$ if $Stype(y) \in \{\gamma_0, \nu_0, \phi_0\}$ and $\mu(y)=1$ otherwise. A *generative* position x with $Ptype(x) \in \{\gamma, \nu, \phi\}$ will receive n distinct copies of successor trees with root y where $x \ll y$, $Stype(y) \in \{\gamma_0, \nu_0, \phi_0\}$, and $\mu(y) = n$.

In the resulting ordering the sets of γ_0-, ν_0-, and ϕ_0-positions (Γ_0, $\mathcal{V}_0$, Φ_0) are called *variables* whereas δ_0-, π_0-, and ψ_0-positions (Δ_0, Π_0, Ψ_0) are *constants*. For integrating the Kripke-semantics of modal logics and J we associate each position x in $\ll$ with a *prefix pre*(x), which is the string of positions $y \in \mathcal{V}_0 \cup \Pi_0$ (for modal logics except $S5$) or $y \in \Phi_0 \cup \Psi_0$ (for J) between the root of $\ll$ and x. The root of $\ll$ is counted as element of Π_0 or Ψ_0. In $S5$ a prefix *pre*(x) is the greatest ancestor $y \leqslant x$ with $y \in \mathcal{V}_0 \cup \Pi_0$.

Example 1. Consider the formula $F \equiv \Box\exists x.\Box C(x) \wedge \Diamond B \Rightarrow \Diamond(B \wedge \Diamond\exists x.C(x))$ *whose tree ordering with is depicted below.*

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	$b_0 a_7$
$\bar{a}_9, a_{10}, a_{11}$	$b_0\bar{a}_9$
$\bar{a}_{12}, \bar{a}_{13}$	$b_0\bar{a}_9\bar{a}_{12}$



Each position has multiplicity 1. In $\ll$ *we have associated each position* x *with the main operator of* $lab(x)$ *and* $pol(x)$. *Variable positions are marked with an overbar. Ptype*(x) *and Stype*(x) *can be derived using Table 1. The prefixes of the positions are given in the table on the left. For* $S5$ *the prefixes consist only of the last position in these strings.*

Paths, Unification, and Complementarity. Complementary paths through a formula tree are the key concept for characterizing logical validity. We define paths via a process starting at the root b_0 of $\ll$ and successively replace positions in $\ll$ by their successors. At a β-position we split into two paths, one containing β_1, the other β_2. If no reducible positions are left we obtain a collection of sets of leaves in $\ll$. These sets are called *paths* through the formula $\langle A, 0\rangle$. If A is valid, then every path through $\langle A, 0\rangle$ must contain a *connection* that is *complementary* under a *global substitution* σ. The substitution σ consists of two parts: a *quantifier* substitution σ_Q replacing quantified variables by terms and a *prefix* substitution σ_L ($L \in \{J, M\}$) for intuitionistic and modal logics, which replaces variables in a prefix by strings of positions. These substitutions are defined as follows:

- Let $T_Q = \Gamma_0 \cup \Delta_0$, C_0 be a set of constants, and $\mathcal{T}$ be a set of terms over $C_0 \cup T_Q$ and a given signature of function symbols. A *first order substitution* is a mapping $\sigma_Q : \Gamma_0 \mapsto \mathcal{T}$. It induces a relation $\sqsubset_Q$ on $\Delta_0 \times \Gamma_0$, which is defined by the condition: *If* $\sigma_Q(u)=t$, *then* $v \sqsubset_Q u$ *for all* $v \in \Delta_0$ *that are subterms of* t
- Let $T_M = \Pi_0 \cup \mathcal{V}_0$, $T_J = \Psi_0 \cup \Phi_0$, $V_M = \mathcal{V}_0$, $V_J = \Phi_0$, and $L \in \{M, J\}$. Furthermore let T_L^+ be the set of strings over T_L and $T_L^* = T_L^+ \cup \{\varepsilon\}$. A *prefix-substitution* (or *L-substitution*) is a mapping $\sigma_L : V_L \mapsto T_L^*$ that fulfills certain restrictions depending on the logic $\mathcal{L}$ (see [29] for details). σ_L induces a relation $\sqsubset_L$ on $T_L \times T_L$, which is defined by the condition: *If* $\sigma_L(u)=p$ *and* $p \notin V_L$, *then for all* v *occurring in* $p, v \sqsubset_L u$.

Let P be a path through $\langle A, 0\rangle$ and $\sigma = \langle \sigma_Q, \sigma_L\rangle$ be a combined substitution. A *connection* is a subpath $\{u, v\} \subseteq P$, where u, v have the same predicate symbol in their labels but $pol(u) \neq pol(v)$. $\{u, v\}$ is called σ*-complementary*, iff (i) $\sigma_Q^{\#}(lab(u)) = \sigma_Q^{\#}(lab(v))$ and (ii) $\sigma_L^{\#}(pre(u)) = \sigma_L^{\#}(pre(v))$, where $\sigma_Q^{\#}, \sigma_L^{\#}$ are homeomorphic extensions of σ_Q, σ_L. P is σ-

complementary if it contains a σ-complementary connection. A set $\mathcal{C}$ of σ-complementary connections *spans* a formula $\langle A,0\rangle$ if every path through it contains an element of $\mathcal{C}$.

Example 2. *Consider again* $F \equiv \Box\exists x.\Box C(x) \wedge \Diamond B \Rightarrow \Diamond(B \wedge \Diamond\exists x.C(x))$. *Its two paths* $\{\bar{a}_5, a_7, a_{10}\}$ *and* $\{\bar{a}_5, a_7, \bar{a}_{13}\}$ *contain the connections* $\{a_7, a_{10}\}$ *and* $\{\bar{a}_5, \bar{a}_{13}\}$. $\sigma_M(\bar{a}_9){=}a_7$ *unifies the prefixes of* a_7 ($b_0 a_7$) *and* a_{10} ($b_0\bar{a}_9$). $\bar{a}_5$ *and* $\bar{a}_{13}$ *(prefixes* $b_0\bar{a}_3\bar{a}_5$, $b_0\bar{a}_9\bar{a}_{12}$*; labels* $C(a_4)$, $C(\bar{a}_{13})$ *) are complementary for* $\sigma_M(\bar{a}_3){=}a_7$, $\sigma_M(\bar{a}_{12}){=}\bar{a}_5$, *and* $\sigma_Q(\bar{a}_{13}){=}a_4$. *Thus* $\langle\sigma_Q,\sigma_M\rangle$ *makes both connections complementary and the connections span* $\langle F,0\rangle$.

The combination of the induced relations $\sqsubset_L, \sqsubset_Q$, and $\ll$ defines a *reduction ordering* $\lhd$, which encodes the non-permutabilities of rules in sequent systems for the logic $\mathcal{L}$. In addition to the complementarity requirement there are certain *admissibility* conditions on $\lhd$ that involve the interaction between σ_L and σ_Q when integrating the *domain* conditions.

Definition 1. The $\mathcal{L}$-accessibility relation R_0 on $T_L^* \times T_L^*$ is defined by
pR_0q iff $p,q \in T_L^*$ and (i) $q{=}pu$, $u \in T_L$ (*general*),
(ii) $q{=}p$ (*reflexivity*),
(iii) $q{=}pp'$, $p' \in T_L^+$ (*transitivity*),
or (iv) $p,q \in T_M$ (*equivalence*, for $S5$ only).

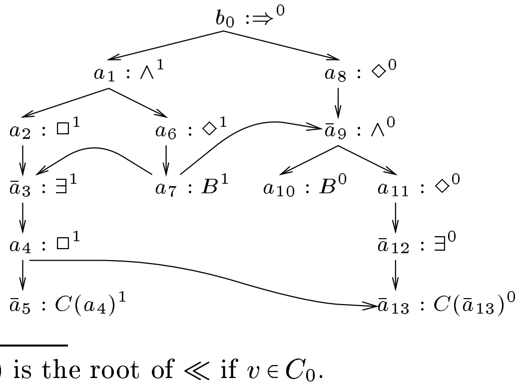
R_0 reflects the conditions on the accessibility relation R for modal logics and J according to their Kripke semantics.

Definition 2. A *combined substitution* $\sigma{=}\langle\sigma_Q,\sigma_L\rangle$ is $\mathcal{L}$-*admissible* provided:
1. For all $p,q \in T_L^*$: pR_0q implies $\sigma_L^{\#}(p)R_0\sigma_L^{\#}(q)$.
2. $\lhd := (\ll \cup \sqsubset)^+$ is irreflexive, where $\sqsubset := \sqsubset_L \cup \sqsubset_Q$.
3. If $\sigma_Q(u){=}t$ then for all $pre(v) \in P(t){=}\{pre(v) \mid v \in T_Q \cup C_0,\ v$ subterm of $t\}$[1]
   (i) *varying domains*: $\sigma_M^{\#}(pre(v)) = \sigma_M^{\#}(pre(u))$
   (ii) *cumulative domains*: if $v \notin \Gamma_0$ either $\sigma_L^{\#}(pre(v)){=}\sigma_L^{\#}(pre(u))$
   or $\sigma_L^{\#}(pre(v))R_0\sigma_L^{\#}(pre(u))$
   (iii) *constant domains*: no conditions

The condition $v \notin \Gamma_0$ in *3.-(ii)* ensures that no priorities between variables will be possible, especially if they occur within k-ary function symbols f^k, $k \geq 1$. This requirement preserves completeness, assuming non-empty domains in every world w as well as the *closed domain condition*: $f^k(a_1,\ldots,a_k)$ is in the domain of w if $a_1,\ldots,a_k$ are. For $K, K4$ there are additional conditions, which are discussed in detail in [29]. Using the above definitions the following theorem has been proven in [29].

Theorem 1. *A formula* A *is* $\mathcal{L}$*-valid iff there is a multiplicity* μ*, an* $\mathcal{L}$*-admissible combined substitution* $\sigma{=}\langle\sigma_Q,\sigma_L\rangle$, $L \in \{M,J\}$*, and a set of* σ*-complementary connections* $\mathcal{C}$ *that spans the signed formula* $\langle A,0\rangle$.

Example 3. *Consider again* $F \equiv \Box\exists x.\Box C(x) \wedge \Diamond B \Rightarrow \Diamond(B \wedge \Diamond\exists x.C(x))$ *from Example 1 and the combined substitution* $\sigma{=}\langle\sigma_Q,\sigma_M\rangle$ *from Example 2.*



$\sigma_M(\bar{a}_3){=}\sigma_M(\bar{a}_9){=}a_7$
$\sigma_M(\bar{a}_{12}){=}\bar{a}_5$
$\sqsubset_M \ =\{(a_7,\bar{a}_3),(a_7,\bar{a}_9)\}$

$\sigma_Q(\bar{a}_{13}){=}a_4$
$\sqsubset_Q \ =\{(a_4,\bar{a}_{13})\}$

$\sigma_M^{\#}(pre(a_4)){=}b_0a_7$
$\sigma_M^{\#}(pre(\bar{a}_{13})){=}b_0a_7\bar{a}_5$

---
[1] $pre(v)$ is the root of $\ll$ if $v \in C_0$.

σ induces relations $\sqsubset_Q$ and $\sqsubset_M$ (curved arrows in the diagram), which together with $\ll$ yield an irreflexive relation $\lhd$. σ_M respects R_0 for $D, D4, T, S4, S5$ and satisfies the cumulative and constant domain conditions. Thus σ is $\mathcal{L}$-admissible in these logics and F is $\mathcal{L}$-valid. F can also be proven $\mathcal{L}$-valid in $T, S4$ and $S5$ under varying domains by extending σ_M: $\sigma_M(\bar{a}_5)=\varepsilon$ for $T, S4$ or $\sigma_M(\bar{a}_5)=a_7$ for $S5$.

2.2 Sequent Calculi

Proofs according to the matrix-characterization in Theorem 1 and the induced reduction orderings are the starting point of our transformation. Sequent proofs – which are used by many application systems requiring user interaction – will be the target of such a transformation. For systems based on natural deduction these proofs can be translated in a straightforward manner. Cut-free sequent calculi are known for classical logic, intuitionistic logic, and for the *cumulative* and *varying domain* modal logics $K, K4, D, D4, T, S4$. For *constant domains* and $S5$ such sequent calculi do not always exist (see [10]).

In our unified presentation we shall provide schematic rules containing parameters whose values describe the individual logics. To achieve a compact representation we adopt a notation based on tableau systems [3, 10] and encode a sequent $\Gamma \vdash \Delta$ by a set S of signed formulae, which we call the *associated set*. Formally, the associated set of $\Gamma \vdash \Delta$ is defined as $S{=}S_\Gamma {\cup} S_\Delta$ where $S_\Gamma{=}\{\langle F,1\rangle \mid F \in \Gamma\}$ and $S_\Delta{=}\{\langle F,0\rangle \mid F \in \Delta\}$.

Example 4. In modal logics the rules dealing with $\Box$ in the succedent or $\Diamond$ in the antecedent are very similar. In D, for instance, they are presented as

$$\frac{\{F \mid \Box F \in \Gamma\} \;\vdash\; A, \{F \mid \Diamond F \in \Delta\}}{\Gamma \;\vdash\; \Box A, \Delta}\;\Box r \qquad \frac{\{F \mid \Box F \in \Gamma\}, A \;\vdash\; \{F \mid \Diamond F \in \Delta\}}{\Gamma, \Diamond A \;\vdash\; \Delta}\;\Diamond l$$

If we encode sequents by their associated sets, then $\Box r$ reads as follows

$$\frac{\{\langle F,1\rangle \mid \langle \Box F,1\rangle \in S_\Gamma\},\; \langle A,0\rangle,\; \{\langle F,0\rangle \mid \langle \Diamond F,0\rangle \in S_\Delta\}}{S_\Gamma,\; \langle \Box A,0\rangle,\; S_\Delta}\;\Box r$$

According to the tableau classification in Table 1 the formulae $\langle \Box F,1\rangle$ and $\langle \Diamond F,0\rangle$ are of type ν. $\langle F,1\rangle$ and $\langle F,0\rangle$ are the corresponding ν_0-subformulae. Furthermore $\langle \Box A,0\rangle$ is of type π and $\langle A,0\rangle$ is its π_0 subformula. Similarly for $\Diamond l$ we have $\langle \Diamond A,1\rangle$ of type π in the conclusion and $\langle A,1\rangle$ as its π_0 subformula in the premise. Thus both *rules can be represented as*

$$\frac{\{\nu_0 \mid \nu \in S\}, \pi_0}{S, \pi}\;\{\Box r,\; \Diamond l\}$$

We get the same result for K and T and slightly different rules for other logics.

$$\frac{\{\nu \mid \nu \in S\}, \pi_0}{S, \pi}\;\{\Box r,\; \Diamond l\}\;(S4) \qquad \frac{\{\nu_0 \mid \nu \in S\}, \{\nu \mid \nu \in S\}, \pi_0}{S, \pi}\;\{\Box r,\; \Diamond l\}\;(K4, D4)$$

Thus the scheme $\frac{S^*, \pi_0}{S, \pi}$ is the common form of $\Box r$ and $\Diamond l$ in all modal logics but the value of the parameter S^* in the premise depends on the particular logic.

Table 2 uniformly describes the rules of all sequent calculi. The rules are arranged according to the tableau classification and directly usable for *cumulative domains*. We apply them from bottom to top in order to *reduce* a certain formula, which is determined by the name of the rule, i.e. by the logical operator and its polarity. We further abbreviate these *reduction-formulae* by their *Ptypes* (e.g. as α in $\wedge l$ and $\vee r$). No *structural rules* are necessary since we use associated sets in our formulation. To retain completeness we have to copy some reduction-formulae explicitly into the rule's premises, as in the case of γ-formulae for all logics and ν-formulae for modal logics $T, S4$. Further modal multiplicities will be handled by forming the sets described below (recall that there are no ν-rules for $K, K4$). When applying δ-rules an eigenvariable-condition has to be respected: the constant a has to be new in the premises.

$\mathcal{L}$:	C	J	K, D, T	$K4, D4$	S4
$S^{\#}$	S	$\{\langle A,1\rangle \mid \langle A,1\rangle \in S\}$	S	S	S
S^{*}	–	–	$\{\nu_0 \mid \nu \in S\}$	$\{\nu_0 \mid \nu \in S\} \cup \{\nu \mid \nu \in S\}$	$\{\nu \mid \nu \in S\}$
S^{+}	S	$S \cup \{\langle C,1\rangle\}$	S	S	S

$$\textit{axiom}: \quad \frac{}{S, \langle P,1\rangle, \langle P,0\rangle}\ \textit{axiom}$$

$$\alpha: \quad \frac{S^{+}, \langle A,0\rangle}{S, \langle \neg A,1\rangle}\ \neg l \qquad \frac{S^{\#}, \langle A,1\rangle}{S, \langle \neg A,0\rangle}\ \neg r \qquad \frac{S^{\#}, \langle A,1\rangle, \langle B,0\rangle}{S, \langle A \Rightarrow B,0\rangle} \Rightarrow r \qquad \frac{S, \alpha_1, \alpha_2}{S, \alpha}\ \{\wedge l, \vee r\}$$

$$\beta: \quad \frac{S^{+}, \langle A,0\rangle \quad S, \langle B,1\rangle}{S, \langle A \Rightarrow B,1\rangle} \Rightarrow l \qquad \frac{S, \beta_1 \quad S, \beta_2}{S, \beta}\ \{\wedge r, \vee l\}$$

$$\delta: \quad \frac{S, \langle A[x\backslash a],1\rangle}{S, \langle \exists x.A,1\rangle}\ \exists l \qquad \frac{S^{\#}, \langle A[x\backslash a],0\rangle}{S, \langle \forall x.A,0\rangle}\ \forall r \qquad \gamma: \quad \frac{S, \gamma, \gamma_0(t)}{S, \gamma}\ \{\forall l, \exists r\}$$

$$\pi: \quad \frac{S^{*}, \pi_0}{S, \pi}\ \{\Box r, \Diamond l\} \qquad \nu: \quad \{D, D4\}: \ \frac{S^{*}}{S} \qquad \{T, S4\}: \ \frac{S, \nu, \nu_0}{S, \nu}\ \{\Box l, \Diamond r\}$$

Table 2. Sequent calculi and their conditions depending on the selected logic

The table on top of the rules describes the logic $\mathcal{L}$ by providing conditions for forming the sets $S^{\#}$, S^{*}, and S^{+}. S^{*} (for π- and ν-rules) encodes which of the conclusion's formulae from S will occur in the premise. $S^{\#}$ and S^{+} occur in sequent rules of *all* logics but cause changes only in J. The set $S^{\#}$ plays the role of S^{*} in modal logics whereas S^{+} encodes the duplication of the actual reduction-formulae (denoted by $\langle C,1\rangle$) within $\Rightarrow l$ or $\neg l$.

If we consider *varying domains* we have to check an additional condition (see [10]): Let C_B be the set of *alive* constants on a branch B in the sequent proof. Only ground terms t over C_B are allowed to be introduced with $\gamma_0(t)$ when applying a γ-rule on B. For this, the set C_B is defined as follows: (i) Starting with $\langle A,0\rangle$, C_B consists of all constants occurring in A. If there exists no constant, then $C_B := \{c\}$ where c is new since we deal with non-empty domains. (ii) When applying a δ-rule, then $C_B := C_B \cup \{a\}$. (iii) When applying a modal, i.e. a π-rule for all modal logics or a π/ν-rule for $D, D4$, again $C_B := \{c'\}$ for a new c'. (iv) For α- and β-rules C_B will not be modified.

A sequent proof for a formula A is constructed by successively applying reductions starting with $\langle A,0\rangle$. The reductions form a *derivation* tree, splitting into two independent branches at β-reductions. A branch is *closed* if its leaf is marked with *axiom*. A derivation is a *sequent proof* if all branches are closed.

*Example 5. In the modal logics $D, D4$ (*cumulative domains*) the following series of reductions proves the formula $F \equiv \Box\exists x.\Box C(x) \wedge \Diamond B \Rightarrow \Diamond(B \wedge \Diamond\exists x.C(x))$ from Example 1. The positions mentioned after a rule indicate the positions in the formula tree that will be reduced.*

$$\dfrac{\dfrac{\dfrac{\dfrac{\Box C(a_4),\ B \ \vdash\ B \qquad \dfrac{\dfrac{C(a_4)\ \vdash\ C(a_4)}{C(a_4)\ \vdash\ \exists x.C(x)}\ \exists r\ (\bar{a}_{12}, \bar{a}_{13})}{\Box C(a_4),\ B\ \vdash\ \Diamond\exists x.C(x)}\ \Diamond r\ (a_{11})\ \ [a_4, \bar{a}_5]}{\Box C(a_4),\ B\ \vdash\ B \wedge \Diamond\exists x.C(x)}\ \wedge r\ (\bar{a}_9, a_{10})}{\exists x.\Box C(x),\ B\ \vdash\ B \wedge \Diamond\exists x.C(x)}\ \exists l\ (\bar{a}_3)}{\Box\exists x.\Box C(x),\ \Diamond B\ \vdash\ \Diamond(B \wedge \Diamond\exists x.C(x))}\ \Diamond l\ (a_6, a_7)\ \ [a_2, a_8]}{\vdash\ \Box\exists x.\Box C(x) \wedge \Diamond B \Rightarrow \Diamond(B \wedge \Diamond\exists x.C(x))}\ \Rightarrow r\ (b_0),\ \ \wedge l\ (a_1)$$

Recall that the ν-positions $[a_2, a_8]$ are reduced when applying the π-rule $\Diamond l$ at a_6. Similarly $[a_4, \bar{a}_5]$ are reduced using the ν-rule $\Diamond r$ at a_{11}.

If we consider varying domains *instead we must begin with $C_B = \{c_1\}$ since F contains no constants. After the π-rule $\Diamond l$ we obtain a new set $C_B = \{c_2\}$ which will be extended to $C_B = \{c_2, a_4\}$ when introducing the eigenvariable ($\exists l$). After splitting we obtain $C_{B_2} = \{c_3\}$ applying the ν-rule $\Diamond r$. Since c_3 is the only constant alive on branch B_2 we reach $C(a_4) \vdash C(c_3)$ after applying $\exists r$. Hence F is no theorem in varying domains for $D, D4$.*

$\mathcal{L}$:	C	J			$K, K4$		$D, D4, T, S4, S5$	
S^+	S	$S \cup \{r : \langle C, 1\rangle\}$			S		S	
r, s		*axiom*	ϕ	ψ				
	ε	$rR_0 s$	$rR_0 s$ and $used(s)$ or $usext(r, s)$	$usext(r, s)$	$r = s$		$r = s$	
p, q					ν	π	ν	π
	–	–			$pR_0 q$ and $used(q)$	$usext(p, q)$	$pR_0 q$ and $used(q)$ or $usext(p, q)$	$usext(p, q)$

Stype	*cumulative*	*varying*	*constant domains*
$q : \gamma_0(t)$	t *over* $\bigcup\{C_0^p \mid pR_0 q \text{ or } p = q\}$	t *over* C_0^q	t *over* C_0
$q : \delta_0(a)$	$a \in C_0^q, new(a)$		$a \in C_0, new(a)$

$\mathcal{L}$	Properties of R
C	no relation R
K, D	general
T	general, reflexive
$K4, D4$	general, transitive
$S4, J$	gen., refl., transitive
$S5$	wRv for all w, v

$$ax : \quad \frac{}{S, r : \langle P, 1\rangle, s : \langle P, 0\rangle}\ axiom$$

$$\alpha : \quad \frac{S^+, s : \langle A, 0\rangle}{S, r : \langle \neg A, 1\rangle}\ \neg l \qquad \frac{S, s : \langle A, 1\rangle}{S, r : \langle \neg A, 0\rangle}\ \neg r \qquad \frac{S, s : \langle A, 1\rangle, s : \langle B, 0\rangle}{S, r : \langle A \Rightarrow B, 0\rangle} \Rightarrow r \qquad \frac{S, r : \alpha_1, r : \alpha_2}{S, r : \alpha}\ \{\wedge l, \vee r\}$$

$$\beta : \quad \frac{S^+, s : \langle A, 0\rangle \quad S, s : \langle B, 1\rangle}{S, r : \langle A \Rightarrow B, 1\rangle} \Rightarrow l \qquad \frac{S, r : \beta_1 \quad S, r : \beta_2}{S, r : \beta}\ \{\wedge r, \vee l\}$$

$$\delta : \quad \frac{S, s : \langle A[x\backslash a], 0\rangle}{S, r : \langle \forall x.A, 0\rangle}\ \forall r \qquad \frac{S, r : \delta_0(a)}{S, r : \delta}\ \exists l \qquad \gamma : \quad \frac{S, r : \langle \forall x.A, 1\rangle, s : \langle A[x\backslash t], 1\rangle}{S, r : \langle \forall x.A, 1\rangle}\ \forall l \qquad \frac{S, r : \gamma, r : \gamma_0(t)}{S, r : \gamma}\ \exists r$$

$$\pi : \quad \frac{S, q : \pi_0}{S, p : \pi}\ \{\Box r, \Diamond l\} \qquad \nu : \quad \frac{S, p : \nu, q : \nu_0}{S, p : \nu}\ \{\Box l, \Diamond r\}$$

Table 3. Prefixed sequent systems with conditions on prefixes and domains

2.3 Prefixed Sequent Systems

Prefixed sequent systems are an extension of sequent calculi that is closer to tableau and matrix calculi. They are also important for dealing with logics which do not have a cut-free sequent calculus such as modal logics with *constant domains*. We have constructed them from *prefixed tableau systems* [10] for $K, K4, D, D4, T, S4$ in all domain variants and for $S5$ in varying and constant domains. We have also developed a prefixed sequent system for intuitionistic logic. Prefixes may be ignored in classical logic. In our presentation we shall again use schematic rules containing parameters and tables listing their values.

We define a *prefix* p to be a finite sequence of positive integers like p=1121. We extend signed formulae to *prefixed signed formulae* of the form '$p : \langle A, k\rangle$' where $\langle A, k\rangle$ is a signed formula and p a prefix. The conditions on the accessibility relation R (see Table 3) will now be encoded into the use of the prefixes. For this we have to define an *accessibility relation* R_0 *on prefixes* and two conditions for *manipulating prefixes* in sequent systems, denoted by $used(q)$ and $usext(p, q)$.

Definition 3. Let p, q be two prefixes. q is *accessible* from p, pR_0q,
if $\quad q{=}pn$ for some $n \in \mathbb{N}$ $\quad$ (*general*),
$\quad q{=}p$ $\quad$ (*reflexivity*),
or $q{=}pt$ for some non-empty sequence t $\quad$ (*transitivity*).
q is called *used*, $used(q)$, if there exists some $r : \langle A, k\rangle$ in the associated set S where q is an initial sequence of r ($q \preceq r$). q is an *unrestricted simple extension* of p, $usext(p, q)$, if $q{=}pn$ for some $n \in \mathbb{N}$ and $q \not\preceq r$ for any prefix r in S.

By combining the properties of the accessibility relation R_0 with the conditions *used* and *usext* we have developed a mechanism for constructing prefixes while applying a reduction rule. In the resulting rule-system presented in Table 3 the prefixes occurring in the premises of a rule are constructed from those in the conclusions according to the conditions given in the table on top.[2]

[2] We abbreviate $S5$-prefixes by integers instead of sequences. Then nR_0m for all $n, m \in \mathbb{N}$, $used(m)$ means that m exists in S, and $usext(n, m)$ stands for m is new in S.

We also had to give conditions for introducing terms in γ- and δ-rules depending on the domain variants. For this purpose we have divided a set C_0 of constants into countably many disjoint classes such that each prefix p has an associated countable set of constants C_0^p. The prefixed *Stype*-formula stands for term introduction at prefix q and is part of the premise in a γ- or δ-rule. For γ-reductions the introduction of a ground term t over sets of constants C_0^p has to respect the prefixes p or q. For the δ-rule the constant a is only related to the actual prefix q, where $new(a)$ indicates the eigenvariable condition.

Example 6. In the modal logics $D, D4, T, S4$ the following prefixed sequent proof proves the formula $F \equiv \Box\exists x.\Box C(x) \wedge \Diamond B \Rightarrow \Diamond(B \wedge \Diamond\exists x.C(x))$ from Example 1.

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
075 : C(a_4),\ 07 : B \ \vdash\ 07 : B
\qquad
\dfrac{
\dfrac{075 : C(a_4),\ 07 : B \ \vdash 075 : C(a_4)}{075 : C(a_4),\ 07 : B \ \vdash\ 075 : \exists x.C(x)}\ \exists r\ (\bar{a}_{12}, \bar{a}_{13})
}{075 : C(a_4),\ 07 : B \ \vdash\ 07 : \Diamond\exists x.C(x)}\ \Diamond r\ (a_{11})
}{075 : C(a_4),\ 07 : B \ \vdash\ 07 : B \wedge \Diamond\exists x.C(x)}\ \wedge r\ (\bar{a}_9, a_{10})
}{075 : C(a_4),\ 07 : B \ \vdash\ 0 : \Diamond(B \wedge \Diamond\exists x.C(x))}\ \Diamond r\ (a_8)
}{07 : \Box C(a_4),\ 07 : B \ \vdash\ 0 : \Diamond(B \wedge \Diamond\exists x.C(x))}\ \Box l\ (a_4, \bar{a}_5)
}{07 : \exists x.\Box C(x),\ 07 : B \ \vdash\ 0 : \Diamond(B \wedge \Diamond\exists x.C(x))}\ \exists l\ (\bar{a}_3)
}{0 : \Box\exists x.\Box C(x),\ 07 : B \ \vdash\ 0 : \Diamond(B \wedge \Diamond\exists x.C(x))}\ \Box l\ (a_2)
}{0 : \Box\exists x.\Box C(x),\ 0 : \Diamond B \ \vdash\ 0 : \Diamond(B \wedge \Diamond\exists x.C(x))}\ \Diamond l\ (a_6, a_7)
}{\vdash 0 : \Box\exists x.\Box C(x) \wedge \Diamond B \Rightarrow \Diamond(B \wedge \Diamond\exists x.C(x))}\ \Rightarrow r\ (b_0),\ \ \wedge l\ (a_1)
$$

We assume $a_4 \in C_0^{07}$ when introducing the eigenvariable ($\exists l$ at $\bar{a}_3$). To reduce $075 : \exists x.C(x)$ *we can again instantiate x with a_4 ($\exists r$ at $\bar{a}_{12}$) since* $07R_0075$.

When considering varying domains *the proof fails since $a_4 \notin C_0^{075}$. For the reflexive logics $T, S4$, however, the proof can be revised by constructing prefixes with the* used (07) *rule instead of applying* usext (07, 075), *since* $07R_007$ *holds. Then no prefix* 075 *occurs in the whole proof and* $07 : \exists x.C(x)$ *is now reducible at $\bar{a}_{12}$ with $a_4 \in C_0^{07}$. This results in a proof for $T, S4$ also with varying domains.*

3 A Uniform Transformation Procedure

We now develop the basic transformation algorithm. It takes a logic $\mathcal{L}$ and a matrix proof in $\mathcal{L}$ and generates a proof in the corresponding prefixed sequent system *without any additional search*. The presence of prefixes in our target calculi makes it possible to focus on uniformity and to postpone issues that are only relevant for conventional sequent calculi.

3.1 Relating Matrix Proofs and Prefixed Sequent Proofs

Our algorithm has to traverse the reduction ordering $\propto^\star$ – created from $\lhd$ (Definition 2) by adding a new root w – and to mark all the visited positions x with solved. The specific sequent rules and their parameters (i.e. prefixes and terms) are determined by the nodes of the formula tree and the substitutions σ_L and σ_Q. Because of their admissibility these substitutions reflect exactly the semantics of the logic $\mathcal{L}$ and are all we need for constructing prefixed sequent proofs.

For a precise description of this process we introduce a few notations. The set of immediate successors of a position x in $\ll$ is denoted by $succ(x) = \{x_1, \ldots x_m\}$ while $pred(x)$ results in the immediate predecessor of x. The successors $succ(x)$ are assumed to be ordered in $\ll$ and can be chosen by a selection function $succ_i(x) = x_i$, $1 \leq i \leq m$. With $succ^+(x)$ we denote the set of *all* successors of x in $\ll$. Moreover, we abbreviate $succ_j^+(x) = \{succ_j(x)\} \cup succ^+(succ_j(x))$. Traversal selects an *open* position x in $\propto^\star$ to be visited next, provided it is not blocked by $\sqsubset$, i.e. if there exists no $(y, x) \in \sqsubset$. In this case, x can be marked solved and the sequent rule can be constructed at x. Otherwise, this step must be delayed until y has been solved and hence, x became unblocked.

Definition 4. Let $\propto^\star$ be a reduction ordering and x be a position. We define

1. the set of elements blocking x as $W_x = \{(v_1, v_2) \mid (v_1, v_2) \in \sqsubset \text{ and } v_2 = x\}$.
2. the set of open position as $P_o = \{x \mid pred(x) \text{ solved and } x \text{ not solved in } \propto^\star\}$.
3. the selection function '$x :=$ select_unblocked P_o' $\Leftrightarrow$ $x \in P_o \wedge W_x = \emptyset$.

Besides traversing $\propto^\star$ we have to map the prefixes from of the matrix proof to prefixes in the sequent proof. In almost all cases, we can directly use a prefix $\sigma_L^{\#}(pre(x))$ in $\propto^\star$ without violating the *construction principle* for prefixes in the sequent systems.

Definition 5. *(prefix mapping)* Let $\propto^\star$ be a reduction ordering and T_L be the set of all prefix positions in $\ll$. Moreover, let $\mathcal{P}_{\sigma_L}$ be the image set of all prefixes in $\propto^\star$ under σ_L. Then we provide an *injective* mapping $n : T_L \mapsto \mathbb{N}$ and a *prefix mapping* $f_p : \mathcal{P}_{\sigma_L} \mapsto \mathbb{N}^*$ such that $f_p(x_1 x_2 \cdots x_m) = n(x_1)n(x_2)\cdots n(x_m)$ for each prefix $\sigma_L^{\#}(p) \in \mathcal{P}_{\sigma_L}$ with $\sigma_L^{\#}(p) = x_1 x_2 \cdots x_m$. We abbreviate $f_p(\sigma_L^{\#}(pre(x)))$ by $f_p^{\sigma_L}(x)$ wrt. a position x.

The basic justification for such a simple mapping arises from the admissibility conditions on the combined substitution σ (Definition 2): all substituted prefixes have to respect the relation R_0 on prefixes in $\propto^\star$. Secondly, $\sqsubset_L$ ensures that a new prefix $f_p^{\sigma_L}(x)$ at a position x is either an unrestricted simple extension of $f_p^{\sigma_L}(pred(x))$, or $W_x \neq \emptyset$ holds. In the latter case all positions leading to the same prefix under $\sigma_L^{\#}$ have to be visited before x while traversing $\propto^\star$. After this the prefix $f_p^{\sigma_L}(x)$ at x can be introduced with length $|f_p^{\sigma_L}(x)| \geq 2$ satisfying the *used* condition in the actual sequent. Thus the reduction ordering $\propto^\star$ together with the prefix mapping f_p guarantees that a prefix q in the premise of a sequent rule will be accessible via the relation R_0 from a prefix p in the conclusion and respects the conditions $used(q)$ or $usext(p,q)$ in Table 3.

Remark 1. As mentioned above, this concept is not complete for all cases. In the transitive logics $D4, S4, J$, we may obtain $W_x = \emptyset$ although $|f_p^{\sigma_L}(x)| - |f_p^{\sigma_L}(pred(x))| \geq 2$, which would violate the construction principle. This phenomenon requires a slightly modification of the prefixed sequent calculi for $D4, S4, J$ in order to provide proof reconstruction for *all* matrix proofs. The $usext(p,q)$ condition from Definition 3 has to be replaced by a *transitive* extension as follows: q is an *unrestricted extension* of p, $uext(p,q)$, if $q{=}pp'$ for some $|p'| \geq 1$, and $q \not\preceq r$ for any prefix r in S. The complete treatise of this problem as well as correctness proofs for the modified prefixed sequent calculi can be found in [26].

3.2 The Algorithm *TOTAL*

The uniform transformation algorithm *TOTAL* is presented in Figure 2. It takes as input a logic $\mathcal{L}$ and a partial reduction ordering $\propto^\star$ that represents a matrix proof in $\mathcal{L}$. The result is a (*total*ly ordered) list of reduction rules $\mathcal{S}$-*list* representing a proof in the prefixed sequent system for $\mathcal{L}$. $TOTAL(\propto^\star, \mathcal{L})$ begins with *global* initializations of the solved-marks and *local* initializations within the subprocedure $TOT(\propto^\star, \mathcal{L})$. Then it steps into the main loop by selecting an unblocked position x that shall be solved next. This is done within the procedure $SOLVE(x, \propto^\star, \mathcal{L})$ by forming appropriate sequent rules at x.

The sequent rules will be constructed according to $Ptype(x)$ and $Stype(x)$ using Table 1. If $pred(x) \in \{\gamma, \nu\}$ is a generative position then its reduction rule r_1 has to be constructed before working on x. This special treatment of generative positions is necessary because the blocking elements from $\sqsubset$ are related to their *immediate successors*, i.e. within the sets W_x. Thus an actual generative position must be *skipped* (see case $\{\gamma, \nu\}$) and the corresponding rule construction r_1 will be done when its successor x becomes the actual position. We refer to this step as a *look-back* reduction at x. Afterwards x will be marked solved and all elements in $\sqsubset$ caused by x are removed from $\propto^\star$ using

function $TOTAL(\propto^\star, \mathcal{L}) : \mathcal{S}\text{-}list$

(1) $\boxed{\textbf{for all } \text{positions } x \text{ in } \propto^\star \textbf{ do } \text{set } x \text{ not solved}}$

  set the root w of $\propto^\star$ solved;

  return $TOT(\propto^\star, \mathcal{L})$

function $TOT(\propto^\star, \mathcal{L}) : \mathcal{S}\text{-}list$

  set $\propto^\star$ not proven; $\mathcal{S}_{\propto^\star} := []$;

  compute the set P_o;

  for all positions x in $\propto^\star$ **do** compute the set W_x

  while not proven $\propto^\star$ **do**

(2) $\boxed{x := \text{select\_unblocked } P_o;}$

    $\mathcal{S}_{\propto^\star} := append(\mathcal{S}_{\propto^\star}, SOLVE(x, \propto^\star, \mathcal{L}))$

  return $\mathcal{S}_{\propto^\star}$

function $SOLVE(x, \propto^\star, \mathcal{L}) : \mathcal{S}\text{-}list$

(3) $\boxed{\textbf{if } Stype(x) \in \{\gamma_0, \nu_0\} \textbf{ then } r_1 := \mathsf{Rule}(Stype, x, \mathcal{L}) \textbf{ else } r_1 := \square}$

  set x solved;

  if $Ptype(u) \notin \{\beta, \pi\}$ **then** $P_o := (P_o \setminus \{x\}) \cup succ(x)$

  if $Stype(x) \in \{\nu_0, \psi_0, \phi_0\}$ **then** $\propto^\star := update(x, \propto^\star)$

  if $Ptype(x) \in \{\pi, \delta\}$ **then** $\propto^\star := update(succ_1(x), \propto^\star)$

  case $Ptype(x)$ **of**

    $\{\gamma, \nu, \phi, \psi\}$: **return** $[r_1]$

    $\{-\}$: **if** exists $\{x, y\} \in \mathcal{C}$ with y solved **then**

        set $\propto^\star$ proven;

        return $[r_1, \mathsf{Rule}(axiom, \{x, y\}, \mathcal{L})]$

      else return $[r_1]$

    $\{\delta, \alpha\}$: **return** $[r_1, \mathsf{Rule}(Ptype, x, \mathcal{L})]$

(4) $\boxed{\{\pi\} : \quad P_o := (P_o \setminus \{x\}) \cup succ(x); \quad \textbf{return } [r_1, \mathsf{Rule}(\pi, x, \mathcal{L})]}$

(5) $\boxed{\{\beta\} : \quad [\propto_1^\star, \propto_2^\star] := \beta\text{-}split(\propto^\star, x);}$

      $p_0 := [r_1, \mathsf{Rule}(\beta, x, \mathcal{L})]$;

      $p_1 := TOT(\propto_1^\star, \mathcal{L})$;

      $p_2 := TOT(\propto_2^\star, \mathcal{L})$;

      set $\propto^\star$ proven;

      return $append(p_0, append(p_1, p_2))$

function $update(x, \propto^\star) =$

  for all $\{(v_1, v_2) \mid (v_1, v_2) \in \sqsubset \text{ and } v_1 = x\}$ **do**

    $\sqsubset := \sqsubset \setminus \{(v_1, v_2)\}$; $W_{v_2} := W_{v_2} \setminus \{(v_1, v_2)\}$

Fig. 2. The uniform transformation procedure

the function *update*. Moreover, the set P_o is updated, except if $Ptype(x) \in \{\beta, \pi\}$: the first exception will be covered after the split-operation below, the second one becomes relevant in Section 4. If x has no *Ptype* it must be part of a connection $\{x, y\} \in \mathcal{C}$. We can terminate with setting $\propto^\star$ proven if y has already been solved. At a β-position x we have to split $\propto^\star$ into two independent suborderings $\propto_1^\star$ and $\propto_2^\star$. We then recursively call the local initializations (recomputing the sets P_o) and subproof-transformations for each of these two suborderings. Afterwards $\propto^\star$ is set proven. The following definitions introduce the mechanism for *splitting at β-positions*.

Definition 6. Let x be a position of $\propto^\star$ and $\ll^x$ the subtree ordering with root x and position set $pos(x)$ (including x). [3] The *restriction* of $\propto^\star$ involving positions from $pos(x)$ is defined as $t^x := \ll^x \cup \sqsubset^x$, where $\sqsubset^x := \{(x_1, x_2) \in \sqsubset \mid x_1 \in pos(x) \vee x_2 \in pos(x)\}$. Moreover, $\mathcal{C}^x := \{\{c_1, c_2\} \in \mathcal{C} \mid c_1 \in pos(x)\}$.

[3] For this we assume $(pred(x), x) \in \ll^x$ although $pred(x) \notin pos(x)$.

Rule $(Stype, x, \mathcal{L})$		Rule $(Ptype, x, \mathcal{L})$		Rule $(axiom, \{x, y\}, \mathcal{L})$
$Stype$	sequent rule	$Ptype$	sequent rule	sequent rule
γ_0	$\gamma(sform(pred(x)), \sigma_Q(x))$	$\{\alpha, \beta, \pi\}$	$Ptype(sform(x))$	$axiom(sform(x), sform(y))$
ν_0	$\nu(sform(pred(x)))$	δ	$\delta(sform(x), succ_1(x))$	

J			Modal Logics		
Reduction	old prefix	new prefix	Reduction	old prefix	new prefix
special $\{\alpha, \beta, \delta\}$	$f_p^{\sigma_L}(pred(x))$	$f_p^{\sigma_L}(x)$	π	$f_p^{\sigma_L}(x)$	$f_p^{\sigma_L}(succ_1(x))$
special γ	$f_p^{\sigma_L}(pred(pred(x)))$	$f_p^{\sigma_L}(pred(x))$	ν	$f_p^{\sigma_L}(pred(x))$	$f_p^{\sigma_L}(x)$
other	$f_p^{\sigma_L}(x)$	$f_p^{\sigma_L}(x)$	*other*	$f_p^{\sigma_L}(x)$	$f_p^{\sigma_L}(x)$

Table 4. Rule instantiations for various logics

Definition 7. *(β-split)* Let x be a β-position in $\propto^\star$ with $succ(x)=\{x_1, x_2\}$. The β-split at x is defined by $\beta\text{-}split(\propto^\star, x) := [\propto_1^\star, \propto_2^\star]$, where $\propto_1^\star=\propto^\star \setminus t^{x_2}$ and $\propto_2^\star=\propto^\star \setminus t^{x_1}$. For the subrelations and connections we have $\sqsubset_i := \sqsubset \setminus \sqsubset^{x_j}$, $\ll_i := \ll \setminus \ll^{x_j}$, and $\mathcal{C}_i := \mathcal{C} \setminus \mathcal{C}^{x_j}$ where $i, j \in \{1, 2\}, j \neq i$.

The prefixed sequent proof constructed by our algorithm starts with $0 : \langle A, 0\rangle$, provided $n(b_0) = 0$ for the original root b_0 in $\ll$ and $sform(b_0)=\langle A, 0\rangle$. In order to solve an actual position x in $\propto^\star$ we first construct a look-back reduction r_1 if $Stype(x) \in \{\gamma_0, \nu_0\}$. We instantiate Rule $(Stype, x, \mathcal{L})$ according to Table 4: The sequent rule will be constructed using the secondary type of x and the signed formula $sform(pred(x))$ (upper table). For instance, at a γ_0-position x we uniquely construct a γ-reduction for $sform(pred(x))$ using $\sigma_Q(x)$ for term instantiation. In addition, the prefix in the rule's conclusion is derived from the entry 'old prefix' (lower table) via the prefix mapping f_p. From the classification Table 1 we get the subformulae that have been processed by rule application. The new prefix belonging to these subformulae is determined from the entry 'new prefix'.

After possible look-back reductions, the rule at x itself has to be constructed. For this, we instantiate Rule $(Ptype, x, \mathcal{L})$ and extract the required prefixes in a similar way. The eigenvariable to be introduced at a δ-position x is uniquely determined by $succ_1(x)$.

In matrix proofs for J the prefixes will be constructed between ϕ and ϕ_0, or between ψ and ψ_0 positions. Therefore, the reduction of a *special* formula at a position x requires the 'old prefix' from $pred(x)$, i.e. the ϕ- or ψ-position, whereas the 'new prefix' has already been developed at x. Recall, that this process iterates for a special γ-reduction, which takes place as a look-back reduction at a γ_0-position x. Hence, the ϕ-position for 'old prefix' is given by $pred(pred(x))$.

Because of the admissible interactions between σ_Q and σ_L, the instantiated terms and eigenvariables directly satisfy the domain conditions for the prefixed sequent systems (Table 3). Considering all the above insights as well as the remark on p. 10, we can prove our transformation procedure to be correct and complete.

Theorem 2. (correctness and completeness) *The algorithm TOTAL for converting matrix proofs into prefixed sequent proofs is correct and complete for the logics C, J and for $K, K4, D, D4, T, S4, S5$ in constant, cumulative, and varying domains.*

The proof is similar to the one of Theorem 3, which we will comment on in Section 4.3.

Example 7. We take the formula $F \equiv \Box\exists x.\Box C(x) \land \Diamond B \Rightarrow \Diamond(B \land \Diamond\exists x.C(x))$ and the substitutions σ_M, σ_Q from Example 3. The reduction ordering $\propto^\star$ is generated from $\lhd$ by adding a new root w. We choose the mapping $n(b_0)=0$ and $n(a_i)=i, i \in \{1, \ldots, 13\}$, and obtain $f_p^{\sigma_L}$ for all *positions as follows:*

x	b_0, a_1, a_2, a_6, a_8	$\bar{a}_3, a_4, a_7, \bar{a}_9, a_{10}, a_{11}$	$\bar{a}_5, \bar{a}_{12}, \bar{a}_{13}$
$f_p^{\sigma_L}(x)$	0	07	075

We start traversing the reduction ordering $\propto^\star$ for $D, D4, T, S4$ with cumulative domains. The prefixed sequent proof begins with $0 : \langle F, 0\rangle$. For the α-positions b_0, a_1 we construct the rules $\Rightarrow r : 0$ and $\wedge l : 0$ (we write the 'new prefix' and parameters next to the rule names) using the above table and Table 4.

After skipping the ν-position a_8 we are blocked at $\bar{a}_9$ because $W_{\bar{a}_9} = \{(a_7, \bar{a}_9)\}$. We select a_6, create $\Diamond l : 07$, and update $\sqsubset_L$ at a_7. Solving a_7 yields the atom $B^1 : 07$. We skip a_2 and reach $\bar{a}_3$: We construct the look-back reduction $\Box l : 07$ for a_2 as well as $\exists l : 07; a_4$, introducing the eigenvariable $succ_1(\bar{a}_3) = a_4$ and updating $\sqsubset$ at a_4. Visiting a_4 and $\bar{a}_5$ constructs the look-back reduction $\Box l : 075$ and isolates $C(a_4)^1 : 075$ in the sequent proof.

Solving $\bar{a}_9$ we first construct $\Diamond r : 07$ as look-back reduction for a_8. The reduction $\wedge r : 07$ at $\bar{a}_9$ forces a split of $\propto^\star$ according to Definition 7. For $\propto_1^\star$ we solve a_{10} having the atom $B^0 : 07$ and hence apply an axiom rule with $B^1 : 07$ (a_7 already solved*). For reducing $\propto_2^\star$ we visit $a_{11}, \bar{a}_{12}$ and $\bar{a}_{13}$ obtaining $\Diamond r : 075$ and $\exists r : 075; a_4$, where $\sigma_Q(\bar{a}_{13}) = a_4$ has been used at the γ_0-position $\bar{a}_{13}$. Finally, we solve $\bar{a}_{13}$ and apply the axiom rule with $C(a_4)^0 : 075$. The resulting prefixed sequent proof is presented in Example 6 (Section 2.3).*

The eigenvariable a_4 is associated with 07 and the prefix at $\bar{a}_{13}$ is 075. We use a_4 for the quantifier reduction $\exists r$ satisfying the cumulative domain condition $07\, R_0\, 075$ (Table 3). The sequent proof can be extended to $T, S4$ for varying domains by integrating $\sigma_L(\bar{a}_5){=}\varepsilon$ into the prefix mapping $f_p^{\sigma_L}$ above.

4 Transformation into Conventional Sequent Calculi

While prefixed sequent systems are rather artificial constructs, conventional sequent calculi are the basis of many application systems. In order to transform matrix proofs into these calculi we have extended our basic conversion procedure into one which explicitly takes care of the non-permutabilities of inference rules. This makes it necessary to identify additional positions whose reduction has priority over others, and to eliminate redundant subrelations from $\propto^\star$ after *β-splits* . The latter requires some subtle refinements for the logics $K, K4$, which are beyond the scope of this article. An appropriate extension of the presented concepts to $K, K4$ can be found in [26].

4.1 Dealing with Proof Relevant Positions

The reduction ordering $\propto^\star$ only proves the *existence* of a rule ordering but does not uniquely determine it. In fact, the non-permutabilities of sequent rules are *not* completely encoded by $\propto^\star$. The missing ordering constraints depend on the application of certain reduction rules which cause the deletion of sequent formulae in intuitionistic and modal logics. In addition for some modal logics, these rules reduce all ν-formulae in the actual sequent and cause global changes of the current ordering conditions. Both kinds of reduction rules will be controlled by extending our algorithm with the following concepts. First, we denote as *ν_0-unclosed* those ν_0-positions in $\propto^\star$ that are not necessarily open but whose corresponding sequent formulae would be reduced when applying such a global ν-reduction. Second, we provide an operation *reduced_marks* for distributing so-called reduced-marks to every ν_0-position that has been completely reduced by global ν-reductions. Thus, when reaching a reduced position during further traversal no look-back ν-reduction must be constructed. Obviously, both concepts depend on each other, e.g. positions already marked reduced cannot be *ν_0-unclosed*.

Finally, we introduce so-called *wait*-labels which avoid the deletion of relevant sequent formulae and ensure a correct distribution of reduced-marks. These labels will be assigned dynamically to several positions in $\propto^\star$ according to a table of conditions that depend on the different logics.

Definition 8. *(ν_0-unclosed)* A position x in $\propto^\star$ is called ν_0*-unclosed*, provided

- *Stype*$(x) = \nu_0$ and
- x is the smallest position wrt. $\ll$ such that
  a. either $x \in P_o$ and x is not marked reduced,
  b. or *pred*$(x) \in P_o$ and *Stype*$($*pred*$(x)) \neq \gamma_0$.

By U_{ν_0} we denote the set of all ν_0-unclosed positions in $\propto^\star$.

The ν_0-*unclosed* property captures all possibilities in which a ν-formula, already isolated in the actual sequent S, can be represented in $\propto^\star$. Especially we have to consider a ν_0-position y even if only *pred*(y) is open and not hidden behind a look-back γ-reduction. In this case *pred*(y) will be skipped by further traversing $\propto^\star$ and the corresponding ν-rule will be applied as a look-back reduction when reaching y. But in the sequent proof the ν-formula corresponding to *pred*(y) is already isolated and hence reducible in S. Thus y must be considered *before* solving *pred*(y), which is achieved by calling it ν_0-unclosed.

Several reductions cause the deletion of sequent formulae in intuitionistic and modal logics resulting in the premise sets $S^\#$ or S^*. Moreover, applications of π-reductions in $D, D4, T$ and ν-reductions in $D, D4$ force the reduction of *all* ν-formulae in the actual sequent S. For T, D only the resulting ν_0-formulae, for $D4$ the ν_0- *and* ν-formulae will be saved in S^*. We refer to this operation as *macro reduction* in the sequent proof.

In order to control these macro reduction during proof reconstruction, reduced-marks will be assigned to ν_0-positions, depending on the peculiarities of the different logics. Let $U'_{\nu_0} = \{y \mid y \in U_{\nu_0} \wedge \exists y'.\,(y, y') \in \sqsubset_L\}$ and $P'_o = \{y \mid y \in P_o \wedge succ(y) \subseteq U_{\nu_0}\}$. For $D4$ we define a *reduction counter* as a mapping $c : \mathcal{V}_0 \mapsto \mathbb{N}$. It is globally initialized for every ν_0-position y in $\propto^\star$ by $c(y) = |\sigma_L(y)|$. During proof reconstruction in $D4$, the assignment of reduced-marks is guided by updating this counter.

Definition 9. Let x be the currently visited position, encoding either a look-back ν-reduction in $D, D4$ or a π-reduction in $D, D4, T$. We define *reduced_marks*$(x, \propto^\star, \mathcal{L})$ as:

- For all $y \in P'_o$, set y solved.
- Let $P_o := (P_o \setminus P'_o) \cup U_{\nu_0}$.
- For D, T: Set the reduced-mark to all $y \in U_{\nu_0}$.
- For $D4$: Let $c_{min} = \min\{c(x) \mid x \in U_{\nu_0}\}$. For all $y \in U_{\nu_0}$:
  (i) update $c(y) := c(y) - c_{min}$, if x encodes a look-back ν-reduction, or update $c(y) := c(y) - 1$, if x encodes a π-reduction.
  (ii) If $c(y) = 0$, then set the reduced-mark to y.
  For all $x \in U'_{\nu_0}$, call *update*$(x, \propto^\star)$.

1. Obviously, *Ptype*$(y) = \nu$ for all $y \in P'_o$. Hence, all positions P'_o can be set solved since they only cause *skip*-operations in the sequent proof. Consequently, the set of open positions P_o is updated by all elements of U_{ν_0}.
2. For D, T, no generative ν-formula in S will be saved into the premise S^* after a π-reduction. Similarly for a ν-reduction in D. Hence, each ν_0-unclosed position y receives the reduced-mark, indicating that no look-back ν-reduction must be constructed for the ν-position *pred*(y) when solving y by further traversal of $\propto^\star$.
3. For $D4$, each ν-formula is also copied into S^* since it can be re-used several times (transitivity). The number of re-uses is initially given by $|\sigma_L(y)|$ for each ν_0-position y in $\propto^\star$. Updating the reduction counter $c(y)$ wrt. the actual set U_{ν_0} eventually sets the reduced-mark to y after the last re-use of *pred*(y), i.e. if $c(y) = 0$. Recall, that $c(y) > |W_y|$ may hold during the reconstruction process due to global substitutions σ_L. A similar problem occurred for prefixed sequent calculi (see remark 1).

Finally, we have to guarantee correct applications of the operation *reduced_marks* wrt. the ordering $\sqsubset$. First, consider modal reductions at the actual position x. All blocking elements $(z, y) \in \sqsubset_L$, $z = succ_1(x)$ for a π-reduction in $T, D, D4$, or $z = x$ for a look-back ν-reduction in $D4$, will be deleted via *update* in the current step. Second, interaction with the assignment of additional *wait*-labels during the reconstruction process provides correct conditions for respecting $\sqsubset$ (see [26] for details). Besides guiding the distribution of reduced-marks, *wait*-labels avoid the deletion of *relevant* sequent formulae in all modal logics and J. Depending on the traversal order of $\propto^\star$, they will be *dynamically* assigned to π-positions in $S4, T, D, D4$, to ν_0-positions in $D, D4$, and to ψ_0-positions in J.

For positions x, y let $W_x^y = W_x \setminus \{(y, x)\}$. For $y \in P_o$ we introduce the predicate $U_{\nu_0}(y) \Leftrightarrow y \in U_{\nu_0} \vee succ(y) \subseteq U_{\nu_0}$, and associate either $\hat{y} = y$ or $\hat{y} \in succ(y)$ such that $\hat{y} \in U_{\nu_0}$. Let P_a be the set of atomic positions already solved in $\propto^\star$, and $P_u = P_o \cup P_a$ be the set of *usable* positions. Dynamic *wait*-labels are then uniformly defined as follows:

Definition 10. *(wait-labels)* Let $x \in P_o$ in $\propto^\star$ with the set U_{ν_0} of ν_0-unclosed positions. We define a relation $wait \subseteq P_u \times P_o$ dynamically as

$$wait = \{(y, x) \mid y \neq x \ \wedge \ x \in P_o \ \wedge \ y \in P_u \ \wedge \ pcond(x) \ \wedge \ scond(y)\},$$

where $pcond(x)$ denotes the *primary* conditions and $scond(y)$ the *secondary* conditions, depending on the different logics:

	J	T	$S4$	$D, D4$
$pcond(x)$	$Stype(x) = \psi_0$ (except atom)	$Ptype(x) = \pi$		$Stype(x) = \nu_0$ (not reduced), or $Ptype(x) = \pi$
$scond(y)$	$pol(y) = 0$	$\neg U_{\nu_0}(y)$, or $\exists \hat{y}.\, W_{\hat{y}}^{succ_1(x)} \neq \emptyset$	$\neg U_{\nu_0}(y)$	

A position x is blocked by a *wait*-label, denoted by $wait_x$, iff there exists a $(v_1, v_2) \in wait$ such that $v_2 = x$. Similarly, $\neg wait_x$ denotes that x is not blocked by a *wait*-label.

Observe that $wait_x$ may hold in all modal logics if $Ptype(x) = \pi$ and $Stype(x) = \gamma_0$. Although the look-back γ-reduction could be applied without any difficulties, it will be postponed together with the succeeding π-reduction at x. This is justified by the fact that the γ-reduction cannot change the *wait*-label at x and must not be applied separately. A similar treatment is given in $S4, T$ if $Ptype(x) = \pi$ and $Stype(x) = \nu_0$ (where x not reduced in T), i.e. the look-back ν-reduction will also be postponed if $wait_x$ holds. Finally, the integration of *wait*-labels causes a slightly modification of the selection of open positions: '$x :=$ select_unblocked P_o' $\Leftrightarrow$ $x \in P_o \ \wedge \ W_x = \emptyset \ \wedge \ \neg wait_x$.

4.2 Redundancy deletion and β-splits

While reducing β-positions our transformation algorithm has to split the reduction ordering $\propto^\star$ into two independent suborderings $\propto_1^\star$ and $\propto_2^\star$. These were constructed by eliminating the subrelations involving positions that do not occur in the corresponding subtree (Definition 7). This simple technique was sufficient for creating prefixed sequent proofs. When dealing with conventional sequent calculi, however, the presence of *wait*-labels requires the deletion of redundant subrelations from each $\propto_i^\star$ after a β-split. Otherwise the algorithm might run into a deadlock and become incomplete.

For this purpose we refine our algorithm by two reductions: First, the split-operation *β-split* is extended by a non-normal form *(β, Θ)-purity*; second, a so-called *$[t^z]$-reduction* provides the selection of relevant subrelations from $\propto^\star$, requiring "little" search when integrated into the algorithm. These refinements are necessary for ensuring completeness of the transformation into conventional sequent calculi. They also optimize the algorithm *TOTAL* when reconstructing prefixed sequent proofs.

Definition 11. The $\ll$-greatest predecessor y of x with $succ(y) \geq 2$ is called the *associated β-node* of x if $Ptype(y) = \beta$, and the *associated Θ-node* of x if $Ptype(y) \neq \beta$. We write $y \ll^{\beta} x$ and $y \ll^{\Theta} x$, respectively.

For the purity reduction we use the fact that an unconnected leaf b of $\propto^{\star}$ cannot be relevant for the remaining subproof. We distinguish two sorts of purity. If $y \ll^{\beta} b$ then y has an unconnected β-related subgoal b and hence, cannot contribute to the subproof. Therefore, the whole subtree with root y can be eliminated from $\propto^{\star}$ and the predecessor position of y inherits the purity property. If $y \ll^{\Theta} b$ then then the unconnected position b is α-related to y. Hence, only the branch s of y containing b must be deleted.

Definition 12. *((β,Θ)-purity)* Let P be the actual set of positions in $\propto^{\star}$. Moreover, let the set of pure leaf positions in $\propto^{\star}$ be given by $P_r = \{\, b \mid b \in P \wedge succ(b) = \emptyset \wedge \forall c \in \mathcal{C}.\, b \notin c\}$. Then the *$(\beta,\Theta)$-purity* reduction is defined as:

function (β,Θ)-*purity* $(\propto^{\star}, \mathcal{C})$: *reduction-ordering* =
  while $P_r \neq \emptyset$ **do**
    select $b \in P_r$; $P_r := P_r \setminus \{b\}$; let y be the associated node of b
    if $y \ll^{\beta} b$ **then** % *β-purity*
      $\propto^{\star} := \propto^{\star} \setminus t^{y}$; $\mathcal{C} := \mathcal{C} \setminus \mathcal{C}^{y}$; $\sqsubset := \sqsubset \setminus \sqsubset^{y}$; $P := P \setminus pos(y)$
      recompute the set P_r
    else % $y \ll^{\Theta} b$ % *Θ-purity*
        compute s such that $b \in succ_s^{+}(y)$;
        $\propto^{\star} := \propto^{\star} \setminus t^{succ_s(y)}$; $\sqsubset := \sqsubset \setminus \sqsubset^{succ_s(y)}$; $P := P \setminus pos(succ_s(y))$

The combined function (β,Θ)-*purity* will be applied to each subrelation $\propto_i^{\star}$ after β-*split* , which yields the following *split*-operation.

Definition 13. *(split)* At a β-position x we define $[\propto_1^{\star}, \propto_2^{\star}] := split(\propto^{\star}, x)$, provided $\propto_i^{\star} = (\beta,\Theta)\text{-}purity(\propto_{i'}^{\star}, \mathcal{C}_i)$, for $i = 1, 2$, and $[\propto_{1'}^{\star}, \propto_{2'}^{\star}] = \beta\text{-}split(\propto^{\star}, x)$.

Besides the deletion of branches corresponding to redundant leaves we have to apply a more subtle reduction, the *$[t^z]$-reduction*, to $\propto^{\star}$ in order to avoid any deadlocks that might occur during the reconstruction process.

Definition 14. *(deadlock)* A reduction ordering $\propto^{\star}$ is called a *deadlock* iff $P_r = \emptyset$ and for all $x \in P_o$ there exists some position y such that $(y, x) \in \sqsubset \cup wait$.

The required reduction makes sure that all remaining subrelations in $\propto^{\star}$ have root positions y from the set of usable positions P_u and are transitively connected via the actual set $\mathcal{C}$ of connections. For this purpose we group all restrictions of $\propto^{\star}$ with roots $y \in P_u$ into equivalence classes $[t^z]$ such that exactly one class is relevant for completing the sequent proof. This comes from the fact that all equivalence classes are α-related in $\propto^{\star}$, i.e. each class corresponds to a set of sequent formulae in the actual sequent, and no connection from $\mathcal{C}$ occurs *between* two of those classes. Finally, the *$[t^z]$-reduction* selects one class $[t^z]$ and eliminates all elements from $\propto^{\star}$ that do not belong to $[t^z]$. Proof reconstruction then tries to finish the sequent proof with the selected class.

Formally, this operation depends on the concept of a *connection relation* in $\propto^{\star}$. Let $T_o = \{t^y \mid y \in P_o\}$ and $T_u = T_o \cup P_a$ corresponding to the set P_u.

Definition 15. (connection relation) Let $T_u = \{t^{x_1}, \ldots, t^{x_n}\}$ be the restrictions of the usable positions P_u in $\propto^{\star}$. The *connection relation* $R_{\mathcal{C}} \subseteq T_u \times T_u$ is defined as:

$$R_{\mathcal{C}} = \{\ (t^{x_i}, t^{x_j}) \mid 1 \leq i, j \leq n \ \wedge\ \exists \{c_1, c_2\} \in \mathcal{C}.\ c_1 \in pos(x_i) \ \wedge\ c_2 \in pos(x_j)\ \}$$

By $R_{\mathcal{C}}^{*}$ we denote the transitive closure of $R_{\mathcal{C}}$.

It is easy to see that $R^*_{\mathcal{C}}$ defines an equivalence relation on T_u if $P_r = \emptyset$. In this case we will write $\sim_{\mathcal{C}}$ instead of $R^*_{\mathcal{C}}$. An equivalence class $[t^x] \in T_u/_{\sim_{\mathcal{C}}}$ is defined by $[t^x] := \{t^y \mid t^y \sim_{\mathcal{C}} t^x\}$. In the following we use the fact, that all $t^x \in T_u$ are α-related in $\propto^\star$ and hence, only one $[t^x] \in T_u/_{\sim_{\mathcal{C}}}$ might be relevant to represent a proof.

Definition 16. Let $P_r = \emptyset$ and $T_u/_{\sim_{\mathcal{C}}} = \{[t^{x_1}], \ldots, [t^{x_n}]\}$. The *decomposition problem* in $\propto^\star$ is the problem of selecting the proof relevant class $[t^{x_i}] \in T_u/_{\sim_{\mathcal{C}}}$.

If $\propto^\star$ is a deadlock, then '$x :=$ select_unblocked P_o' does not terminate, since $W_x \neq \emptyset$ or $wait_x$ holds for all $x \in P_o$. In this case, a solution for a decomposition problem is necessary in order to guarantee completeness of proof reconstruction. Such a solution is characterized by the following key-lemma, which uniformly relates deadlocks to decomposition problems in $\propto^\star$ for all non-classical logics under consideration.

Lemma 1. *Let $P_r = \emptyset$. If $\propto^\star$ is a deadlock then there exist two constant positions $\{w_1, w_2\} \subseteq P_o$ such that $wait_{w_1}$, $wait_{w_2}$, and $[t^{w_1}] \neq [t^{w_2}]$.*

The proof proceeds by induction on the *distance* between t^{w_1} and t^{w_2}, i.e. the number of connections establishing $t^{w_1} \sim_{\mathcal{C}} t^{w_2}$. The detailed proof can be found in [26].

Lemma 1 says that deadlocks can only occur if there is also a decomposition problem $T_u/_{\sim_{\mathcal{C}}} = \{[t^{w_1}], [t^{w_2}]\}$ in $\propto^\star$. Consequently, completeness of proof reconstruction can be guaranteed if the decomposition problem can either be avoided or solved. Otherwise, proof relevant formulae might be deleted in the sequent proof. By contraposition of Lemma 1 we obtain the basic property for completeness of the reconstruction approach.

Corollary 1. *If $|T_u/_{\sim_{\mathcal{C}}}| = 1$ then '$x :=$ select_unblocked P_o' always terminates.*

A complete solution of the decomposition problem consists of establishing a selection function $f_{\sim_{\mathcal{C}}}$ that chooses the only *relevant* class $[t^z]$ from $T_u/_{\sim_{\mathcal{C}}}$ whenever a decomposition problem (not necessarily a deadlock) occurs in $\propto^\star$. The following reduction operation in $\propto^\star$ computes such a function $f_{\sim_{\mathcal{C}}}$ *after* the relevant class $[t^z]$ has been decided.

Definition 17. *($[t^z]$-reduction)* Let $P_r = \emptyset$ and $[t^z] \in T_u/_{\sim_{\mathcal{C}}}$ be an equivalence class. Moreover, let $D_u = \{y \mid y \in P_u \;\wedge\; [t^y] \neq [t^z]\ \}$. Then the $[t^z]$-*reduction* of $\propto^\star$ wrt. the connection set $\mathcal{C}$ is defined as:

function $[t^z]$-*reduction* $(\propto^\star, \mathcal{C})$: *reduction-ordering* $=$
  for all $y \in D_u$ **do** $\propto^\star := \propto^\star \setminus [t^y]$; $\mathcal{C} := \mathcal{C} \setminus \mathcal{C}^y$; $\sqsubset := \sqsubset \setminus \sqsubset^y$
  $\propto^\star_z := (\beta, \Theta)$-*purity* $(\propto^\star, \mathcal{C})$

Recall, that only Θ-*purity* is finally applied since all predecessors of the pure leaves *pred*(y), $y \in D_u$, have already been solved. Unfortunately, "little" search is needed in order to find the relevant class $[t^z]$ during the reconstruction process. Application of the $[t^z]$-*reduction* has to be applied whenever a decomposition problem might occur, that is (i) after α-reductions and *skip*-operations at a position x if *succ*$(x) \geq 2$, (ii) after distributing reduced-marks since ν-positions P'_o will be solved, and (ii) after β-*splits* . In the worst case, the function $f_{\sim_{\mathcal{C}}}$ has to try all classes $[t^y] \in T_u/_{\sim_{\mathcal{C}}}$ for every decomposition problem in $\propto^\star$ until a subproof can be reconstructed with the relevant class $[t^z]$.

We will neither present the realization of the search function $f_{\sim_{\mathcal{C}}}$ nor the integration of the $[t^z]$-*reduction* when developing the algorithm for proof reconstruction in conventional sequent claculi. Searching through all equivalence classes is a straight-forward extension of the resulting algorithm given in Section 4.3. Instead, we assume that $|T_u/_{\sim_{\mathcal{C}}}| = 1$ (provided by some selection function $f_{\sim_{\mathcal{C}}}$) in order to focus on the cacluli specific properties within the algorithm. A complete integration of the selection function as well as the realization of the search during proof reconstruction is presented in [26].

It should be noted that the above reduction concepts may fail for the logics $K, K4$. In these logics, there are theorems that have *pure* relevant subformulae not involved in any connection of the matrix proof. In a sequent proof of the K-theorem $\Box A \wedge \Diamond B \Rightarrow \Diamond A$, for instance, the π-reduction of the *pure* subformula $\Diamond B$ is relevant since there are no ν-rules for K. For all other logics we were able to show that this effect cannot occur. Thus our transformation algorithm will be applicable to matrix proofs in C, J and $D, D4, T, S4$ for cumulative and varying domains. Apart from the above restriction, the deletion of redundancies should also be applied when creating prefixed sequent proofs. An extension of the purity concept to $K, K4$ can be found in [26], which generalizes the algorithm for reconstructing conventional sequent proofs even in $K, K4$.

4.3 Adapting the Transformation Algorithm

Using the above considerations we lift *TOTAL* to an algorithm *TOTAL** that converts matrix proofs into conventional sequent proofs for C, J and $S4, T, D, D4$ with cumulative and varying domains. For this purpose we present extensions of our algorithm in Figure 2 by replacing the boxed areas (1) – (5). Thereby, we omit the integration of search behaviour, which becomes necessary for a complete solution of decomposition problems.

(1) Within global initializations, we additionally set all positions to not reduced. This has no impact in C, J, and $S4$ where no macro reductions take place. Furthermore, we apply (β, Θ)-*purity* in order to remove initial redundancies from $\propto^\star$.

for all positions x in $\propto^\star$ **do**
  set x not solved; set x not reduced
$\propto^\star := (\beta, \Theta)$-*purity* $(\propto^\star, \mathcal{C})$;

(2) The relation *wait* is computed dynamically before every selection of an open position $x \in P_o$. In order to analyze x to be unblocked wrt. $wait_x$ we assume that the sets U_{ν_0} and $W_{\hat{y}}^{succ_1(x)}$, $\hat{y} \in U_{\nu_0}$ (for T only), have already been computed.

compute the relation *wait*;
$x :=$ select_unblocked P_o;

(3) The construction of the look-back ν-reduction r_1 at a ν_0-position x has to be modified, depending on distributed reduced-marks in $T, D, D4$. If x is already marked reduced then r_1 is set to the empty rule (observe that this will never be the case in $S4$ since no reduced-marks are distributed).

In addition for $D, D4$, a look-back ν-reduction itself causes macro reductions when forming the premise S^* and hence, the operation *reduced_marks* $(x, \propto^\star, \mathcal{L})$ has to be applied (Definition 9). Then we have to re-compute the relation *wait*: Although $\neg wait_x$ holds *before* the look-back ν-reduction, it may change to $wait_x$ afterwards if *Ptype* $(x) = \pi$. For this we assume a new set U_{ν_0} wrt. the updated set of open positions P_o. Moreover, $c(x) \neq 0$ is possible for the reduction counter at x, i.e. the actual position x itself must not receive the reduced-mark. Finally, x is locally set to blocked if either $wait_x$ or not reduced x after the operation *reduced_marks*. If x is blocked in $D, D4$ after construction of the look-back ν-reduction, only r_1 will be returned. Otherwise we proceed with the rule construction at x itself.

```
if Stype(x) = γ0 then r1 := Rule(γ0, x, ℒ)
else
  if Stype(x) = ν0 and not reduced x then
    case ℒ of
      {T, S4} :  r1 := Rule(ν0, x, ℒ)
      {D, D4} :  r1 := Rule(ν0, x, ℒ);
                 perform reduced_marks(x, ∝⋆, ℒ);
                 compute the relation wait;
                 if not reduced x or wait_x then set blocked x
  else r1 := □
if ℒ ∈ {D, D4} and blocked x then return [r1]
else
```

(4) When solving a π-position x we have to distribute reduced-marks in $D, D4, T$ via the operation *reduced_marks* $(x, \propto^\star, \mathcal{L})$. In order to use the correct set U_{ν_0}, the set of open positions P_o has to be updated *after* this operation has been applied. In contrast to that, the relation $\sqsubset_L$ has already been updated wrt. $succ_1(x)$.

```
{π} : if ℒ ∈ {D, D4, T} then
        perform reduced_marks(x, ∝⋆, ℒ)
      P_o := (P_o \ {x}) ∪ {succ1(x)};
      return [r1, Rule(π, x, ℒ)]
```

(5) At a β-position x the new operation *split* $(\propto^\star, x)$ is used, comprising (β, Θ)-*purity* .

```
{β} : [∝⋆1, ∝⋆2] := split(∝⋆, x);
```

For instantiating the sequent rules we use again the upper table of Table 4. We start with $sform(b_0) = \langle A, 0\rangle$ where b_0 is the original root of $\ll$. When transforming modal matrix proofs with varying domains, the admissible interaction between σ_Q and σ_M (Definition 2) ensures correct instantiations during the quantifier reductions.

Assuming that a complete search function $f_{\sim_C}$ on decomposition problems in $\propto^\star$ is provided, we obtain:

Theorem 3. (correctness and completeness) *The algorithm* TOTAL* *for converting matrix proofs into conventional sequent proofs is correct and complete for the logics* C, J *and* $D, D4, T, S4$ *in cumulative and varying domains.*

Proof. (Sketch) We prove the following facts:

(1) Every expansion of the output $\mathcal{S}$-*list* at a position x is correctly constructed respecting $\sqsubset_L, \sqsubset_Q$ and using either $sform(x)$ or $sform(pred(x))$, the substitutions σ_Q, σ_L, as well as the required prefixes from the matrix proof. Then, $\mathcal{S}$-*list*, forms a sequence of correct reduction steps in the sequent calculus for the logic $\mathcal{L}$.
(2) The *split*-operation at a β-position x, containing β-*split* and (β, Θ)-*purity* , is correct. That is, all positions deleted from $\propto_1^\star, \propto_2^\star$ are no longer relevant for the corresponding subproofs. (Proof reconstruction in prefixed sequent systems prohibits (β, Θ)-*purity* after β-*splits* for $K, K4$.)
(3) Correctness of the operation *reduced_marks* while solving x is provided by interacting with the unblocking conditions on *wait*-labels at x.
(4) The deletion of sequent formulae is correct due to the secondary conditions on dynamic *wait*-labels: The positions corresponding to deleted sequent formulae either do no longer exist in $\propto^\star$, or they are ν_0-positions in $D4$ which are not reduced in $\propto^\star$. In the latter case, the preceding ν-positions will be reused and thus, the deleted ν_0-formulae will be reproduced.

(5) Assume that '$x :=$ select_unblocked P_o' always terminates, using a function $f_{\sim c}$ that directly selects the relevant class $[t^z]$ from decomposition problems in $\propto^\star$. Then, termination of the recursion and basic loop is guaranteed since every step yields a decrease either of the set of unsolved positions (solving a position x), or of the set of unreduced positions (assigning a reduced-mark to x), or of the sum of all reduction counters in $D4$ (decreasing a counter $c(x)$). Moreover, the algorithm terminates with the construction of *axiom*-rules in every developed branch of the sequent proof. (For prefixed sequent systems the function $f_{\sim c}$ must not be considered.)
(6) Correctness of the algorithms follows from (1) – (5): Every output $\mathcal{S}$-*list* forms a correct (prefixed) sequent proof within the corresponding calculus for $\mathcal{L}$.
(7) Assume that we have a function $f_{\sim c}$ combined with the selection $[t^z]$-*reduction* which completely searches through every decomposition problem occurring in $\propto^\star$. This ensures $|T_u/_{\sim c}| = 1$ before selecting an open position. Due to Lemma 1 and Corollary 1 no deadlocks can arise and hence, '$x :=$ select_unblocked P_o' always terminates. (For prefixed sequent calculi, deadlocks are impossible since additional *wait*-labels do not exist). Consequently, the algorithms are complete since every input matrix proof is converted into a (prefixed) sequent proof within the corresponding calculus for $\mathcal{L}$.

Detailed correctness and completeness proofs for the reconstruction algorithms, comprising the integration of the search function $f_{\sim c}$, can be found in [26].

Complexity results. For describing the complexity of our algorithms, we use the following measures: The *length* of a matrix proof is the number of *inference steps* that are necessary to prove that a set $\mathcal{C}$ of connections is spanning for the input formula. (This number can grow exponentially in the number of connections $|\mathcal{C}|$.) The length of (cut-free) sequent proofs is given by the number of required *axiom*-rules. For all logics under consideration we have established a class of formulae F_n, $n \geq 1$, such that the following holds: Every (prefixed) sequent proof for F_n has an exponential length wrt. the length of a given matrix proof for F_n (see [26] for details and proofs). From this general result it follows that $TOTAL$ as well as $TOTAL^\star$ can only have an exponential worst case complexity in the length of the given matrix proofs.

Apart from that we achieve that both algorithms have the same reconstruction behavior, provided that (β, Θ)-*purity* is integrated into the split-operation for $TOTAL$. This is based on the fact that redundant reconstruction steps created by the search in $TOTAL^\star$ are also encoded into $TOTAL$: traversing redundant subrelations $[t^z]$ in $\propto^\star$ yields redundant proof steps in the prefixed sequent proof. Hence, the search behavior in $TOTAL^\star$, although essential for completeness, is no real deterioration wrt. additional reconstruction steps when comparing it with the search-free algorithm $TOTAL$. To the contrary, the search is completely provided within the reconstruction process and does not appear in the output, i.e. the resulting sequent proofs. Since the $[t^z]$-*reduction* is also correct for proof reconstruction in prefixed sequent calculi (extending the concepts for $K, K4$ appropriately), one could integrate a search function $f_{\sim c}$ on decomposition problems $T_u/_{\sim c}$ into $TOTAL$, similar as provided for $TOTAL^\star$. Hence, the reconstructed sequent proofs from both algorithms would only contain relevant inferences.

Example 8. We take the formula $F \equiv \Box\exists x.\Box C(x) \wedge \Diamond B \Rightarrow \Diamond(B \wedge \Diamond\exists x.C(x))$ from Example 3, the substitutions σ_M, σ_Q, and the reduction ordering $\propto^\star$ generated from $\lhd$ by adding a new root w. We develop sequent proofs for $D, D4$ with cumulative domains. We start with b_0, a_1 obtaining the rules $\Rightarrow r, \wedge l$, skip a_2, and become blocked because $W_{\bar{a}_3} = \{(a_7, \bar{a}_3)\}$. To select a_6 we have to check if $wait_{a_6}$ holds. With $U_{\nu_0} = \{\bar{a}_3, \bar{a}_9\}$ (where $pred(\bar{a}_9) = a_8$ is still open) we obtain $\neg wait_{a_6}$ and hence, solve a_6 constructing the π-rule $\Diamond l$. The operation reduced_marks $(a_6, \propto^\star, \mathcal{L})$ sets the ν-position a_8 to solved *and* *distributes the* reduced*-mark to the* ν_0*-positions* $\bar{a}_3, \bar{a}_9$. *Observe that* $\sqsubset_L$ *has already been updated wrt.* $succ_1(a_6) = a_7$ *when performing this operation. Solving* a_7 *then isolates the atom* B^1 *in the sequent proof.*

No look-back ν*-reduction has to be created when solving* $\bar{a}_3$ *because of its* reduced*-mark. We construct the* $\exists l$*-rule, introducing the eigenvariable* $succ_1(\bar{a}_3) = a_4$ *and deleting blocking elements from* $\sqsubset_Q$ *via* $update(a_4, \propto^\star)$. *Then the* ν*-position* a_4 *itself will be skipped. The selection of* $\bar{a}_5 \in P_o = \{\bar{a}_5, a_7, \bar{a}_9\}$ *fails because* $Stype(\bar{a}_5) = \nu_0$ *and* $y \notin U_{\nu_0} = \{\bar{a}_5\}$ *for* $y \in \{a_7, \bar{a}_9\}$. *Hence,* $wait_{\bar{a}_5}$ *holds. We select the* β*-position* $\bar{a}_9$*, which has the* reduced*-mark and thus, prevents us from constructing a look-back* ν*-reduction for* a_8. *The rule* $\wedge r$ *is constructed and the operation* $split(\propto^\star, \bar{a}_9)$ *is applied. In the resulting subrelations* $\propto_1^\star$ *and* $\propto_2^\star$ *we delete* t^{a_2} *and* t^{a_6} *via* (β, Θ)*-purity , respectively.*

The subproof $\propto_1^\star$ *will be finished solving* a_{10} *and providing the axiom-rule with position* a_7 *(already* solved*). For* $\propto_2^\star$ *we skip* a_{11} *and select* $\bar{a}_{12}$ *(since* $P_o = U_{\nu_0} = \{\bar{a}_{12}, \bar{a}_5\}$ *we obtain* $\neg wait_{\bar{a}_{12}}$*). We complete the look-back* ν*-reduction* $\Diamond r$ *for* a_{11} *while giving the* reduced*-mark to* $\bar{a}_5$. *Moreover, the* $\exists r$ *rule is constructed using* $\sigma_Q(\bar{a}_{13}) = a_4$. *Finally we solve* $\bar{a}_{13}$ *and* $\bar{a}_5$ *with the axiom rule. The resulting sequent proof is similar the one presented in Example 5. It is not a proof in* $D, D4$ *with varying domains since* $\sigma_M^\#(pre(a_4)) = b_0 a_7 \neq b_0 a_7 \bar{a}_5 = \sigma_M^\#(pre(\bar{a}_{13}))$.

5 Conclusion and Future Work

We have presented uniform algorithms for transforming non-classical and classical matrix proofs into sequent proofs within two different types of target calculi, i.e. prefixed sequent systems and conventional sequent calculi. The procedure is based on a unified representation of matrix characterizations of logical validity and of the sequent calculi for various logics. It relies on comparably small tables for encoding the peculiarities of a particular logic. Its modular design allows us to treat a rich variety of logics in a uniform and simple way. It would be easy to extend our algorithm to logics not yet considered by extending the tables appropriately. Therefore it is one of the most important steps in the development of a coherent ATP-system that can deal with a rich variety of logics and with different applications in a tailored way.

Both algorithms have a similar behavior when reconstructing sequent proofs, but only for conventional sequent calculi some search on decomposition problems is required in order to ensure completeness. In general, decomposition problems give evidence that redundancies are not completely removed form $\propto^\star$. However, avoiding the decomposition problem and hence, search behavior during the transformation demands the integration of additional knowledge from the proof search into the reconstruction process [25, 26].

Future work will involve combining our transformation algorithm with a proof procedure for various non-classical logics in order to guide the derivation of proofs in one of the existing generic tools for interactive proof development. We have already extended our procedure to the multiplicative fragment of linear logic [12, 26] and will investigate the integration of further fragments into our approach. Finally, we will consider the combination of induction techniques with logical reasoning (e.g. [22]) for a uniform representation within a matrix-based framework. This will eventually extend our proof reconstruction approach to ATP-systems comprising inductive prover components.

References

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

2. P. ANDREWS. *More on the problem of finding a mapping between clause representation and natural-deduction representation.* *JAR*, 7:285–286, 1991.
3. E. W. BETH. *The foundations of mathematics.* North–Holland, 1959.
4. W. BIBEL, S. BRÜNING, U. EGLY, T. RATH. KoMet. *CADE–12*, LNAI 814, pp. 783–787. Springer, 1994.
5. W. BIBEL. On matrices with connections. *JACM*, 28:633–645, 1981.
6. W. BIBEL. *Automated Theorem Proving.* Vieweg, 1987.
7. R. L. CONSTABLE ET. AL. *Implementing Mathematics with the NuPRL proof development system.* Prentice Hall, 1986.
8. B. I. DAHN, J. GEHNE, TH. HONIGMANN, L. WALTHER, A. WOLF. Integrating Logical Functions with *ILF*; Preprint 94-10, Humboldt University Berlin, 1994.
9. M. C. FITTING. *Intuitionistic logic, model theory and forcing.* Studies in logic and the foundations of mathematics. North–Holland, 1969.
10. M. C. FITTING. *Proof Methods for Modal and Intuitionistic Logic.* D. Reidel, 1983.
11. G. GENTZEN. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.
12. C. KREITZ, H. MANTEL, J. OTTEN, S. SCHMITT. Connection-Based Proof Construction in Linear Logic. *CADE–14*, LNAI 1249, pp. 207–221, Springer, 1997.
13. C. KREITZ, J. OTTEN, S. SCHMITT. Guiding Program Development Systems by a Connection Based Proof Strategy. *LoPSTR–95*, LNCS 1048, pp. 137–151, 1996.
14. R. LETZ, J. SCHUMANN, S. BAYERL, W. BIBEL. SETHEO: A high-performance theorem prover. *JAR*, 8:183–212, 1992.
15. C. LINGENFELDER. Structuring computer generated proofs. *IJCAI-89*, 1989.
16. C. LINGENFELDER. *Transformation and Structuring of Computer Generated Proofs.* PhD thesis, 1990.
17. H. J. OHLBACH. A resolution calculus for modal logics. Ph.D. Thesis, 1988.
18. J. OTTEN, C. KREITZ. A connection based proof method for intuitionistic logic. *TABLEAUX–95*, LNAI 918, pp. 122–137, Springer, 1995.
19. J. OTTEN, C. KREITZ. A Uniform Proof Procedure for Classical and Non-Classical Logics *KI-96*, LNAI 1137, pp. 307–319, Springer, 1996.
20. L. C. PAULSON. Isabelle: The next 700 theorem provers. *Logic and Computer Science*, pp. 361–386. Academic Press, 1990.
21. F. PFENNING. *Proof Transformations in Higher-Order Logic.* PhD thesis, 1987.
22. B. PIENTKA, C. KREITZ. Instantiation of existentially quantified variables in inductive specification proofs. *AISC–98*, LNAI 1476, pp. 247–258, Springer 1998.
23. J. A. ROBINSON. A machine-oriented logic based on the resolution principle. *JACM*, 12(1):23–41, 1965.
24. S. SCHMITT, C. KREITZ. On transforming intuitionistic matrix proofs into standard-sequent proofs. *TABLEAUX–95*, LNAI 918, pp. 106–121, Springer, 1995.
25. S. SCHMITT, C. KREITZ. Deleting redundancy in proof reconstruction. *TABLEAUX–98*, LNAI 1397, pp. 262–276, Springer, 1998.
26. S. SCHMITT. Uniform algorithms for proof reconstruction in classical and non-classical logics. Technical report, FG Intellektik, FB Informatik, TU Darmstadt, 1998.
27. T. TAMMET A Resolution Theorem Prover for Intuitionistic Logic *CADE–13*, LNAI 1104, pp. 2–16, 1996.
28. L. WALLEN. Matrix proof methods for modal logics. *IJCAI-87*, pp. 917–923. 1987.
29. L. WALLEN. *Automated deduction in nonclassical logic.* MIT Press, 1990.
30. S. WOLFRAM. *Mathematica: A System for Doing Mathematics by Computer.* Addison-Wesley, 1991.
31. L. WOS ET. AL. Automated reasoning contributes to mathematics and logic. *CADE–10*, LNCS 449, p. 485–499. Springer 1990.
32. L. WOS. *The problem of finding a mapping between clause representation and natural-deduction representation.* *JAR*, 6:211–212, 1990.