Converting Non-Classical Matrix Proofs into Sequent-Style Systems

Stephan Schmitt Christoph Kreitz

FG Intellektik, FB Informatik, TH Darmstadt
Alexanderstr. 10, 64283 Darmstadt, Germany
{steph,kreitz}@intellektik.informatik.th-darmstadt.de

Abstract. We present a uniform algorithm for transforming matrix proofs in classical, constructive, and modal logics into sequent style proofs. Making use of a similarity between matrix methods and Fitting's *prefixed tableaus* we first develop a procedure for extracting a prefixed sequent proof from a given matrix proof. By considering the additional restrictions on the order of rule applications we then extend this procedure into an algorithm which generates a conventional sequent proof.
Our algorithm is based on unified representations of matrix characterizations for various logics as well as of prefixed and usual sequent calculi. The peculiarities of a logic are encoded by certain parameters which are summarized in tables to be consulted by the algorithm.

1 Introduction

Non-classical logics such as modal, intuitionistic, and linear logic are used extensively in various branches of AI and Computer Science. In many of these applications there is a need for automated proof methods. In general, however, calculi for these logics are of Gentzen or Tableaux style and not very efficient for proof search. On the other hand, theorem provers based on resolution [14, 19] and the connection method [3, 4, 8, 2] have demonstrated that formal reasoning in *classical* predicate logic (C) can be automated sufficiently well if less redundant calculi are being used. Recently Wallen [17, 18] and Ohlbach [11] have extended the classical characterizations of logical validity into characterizations of validity for intuitionistic logic (J) and the modal logics $K, K4, D, D4, T, S4$, and $S5$. Since then attempts have been undertaken to extend the existing proof *methods* accordingly in order to create efficient proof procedures for intuitionistic logic [12] and modal logics [13].

The efficiency of automated proof methods strongly depends on a compact representation of a proof. The characterizations of logical validity, on which these methods are based, avoid the notational redundancies contained in mathematical languages or sequent calculi. As a result, an automatically generated proof is almost impossible to comprehend. A user of a theorem prover will get the information *that* a theorem is valid but he will hardly understand *why* this is the case.

Since the development of automated theorem provers attempts have been made to convert machine proofs into a humanly comprehensible form. Lingenfelder [9, 10] has (partially) succeeded in transforming *classical* resolution proofs into natural deduction proofs. The ILF System [5] can create classical proofs in a semi-natural mathematical language. In earlier work [15] we have developed a method for transforming *intuitionistic* matrix proofs into sequent proofs. For other non-classical logics, however, such a conversion procedure does not yet exist.

In M. McRobbie and J. Slaney, editors, *13th International Conference on Automated Deduction*, LNAI 1104, pp. 418–432, ©Springer Verlag, 1996.

In this paper we present an algorithm for transforming matrix proofs into sequent style proofs which allows a *uniform* treatment of classical, constructive, and modal logics. For this purpose we have developed a unified representation of Wallen's matrix characterizations [18] for these logics. A matrix proof for a given formula F essentially shows that every path through a matrix representation of F contains at least one pair of atomic formulae which are *complementary* under some substitution σ, which means that they have different *polarity* and their subterms are made identical by σ. For non-classical logics, σ also has to make the *prefixes* of the two atomic formulae (descriptions of their position in the formula tree) identical. σ, together with the tree ordering of F, induces an ordering $\propto^\star$ on the nodes of the formulae tree. This ordering efficiently encodes that within a top-down sequent proof there are certain restrictions on the order of rule applications which shall 'reduce' F in order to reach the axioms of the calculus.

Thus the basic idea of our algorithm is very simple. We have to traverse $\propto^\star$, select sequent rules according to the sub-formula 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. There are, however, several subtle details. First we need a unified representation of sequent calculi for classical, intuitionistic, and modal logics and a set of tables to be consulted by the algorithm when determining an appropriate sequent rule corresponding to a given node. Secondly positions of type β cause a sequent proof to split into two independent subproofs and we have to determine which sub-relations of $\propto^\star$ will be relevant in each subproof. Finally, since $\propto^\star$ 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 at certain nodes in order to keep them from being reduced too early.

Because of a strong similarity between matrix characterizations of logical validity and Fitting's *prefixed tableaux systems* [7] we present the algorithm in two phases. First we show how to convert a matrix proof into a *prefixed* sequent proof. Such an algorithm requires 'only' a unified representation of prefixed sequent systems and tables for determining appropriate sequent rules. We then extend the algorithm into one which creates conventional sequent proofs. Now the absence of prefixes makes it necessary to deal with β-splits and to consider the additional priorities of proof-relevant positions. The resulting algorithm allows a uniform treatment of all the logics under consideration and can be viewed as the key to an efficient conversion of automatically generated proofs into a comprehensible form.

Our paper is organized as follows. In section 2 we develop unified representations for matrix characterizations, sequent calculi, and prefixed sequent systems. In section 3 we introduce the uniform algorithm for converting matrix proofs into prefixed sequent proofs and discuss the effects of β-splits. Section 4 shows how to specialize the basic algorithm into one that creates conventional sequent proofs.

2 Unifying Logics in Matrix– and Sequent Calculi

2.1 Matrix Characterizations for Various Logics

We shall now introduce matrix characterizations for the logics C, 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 [6, 7, 18]. These characterizations have been adopted from [18] which again can be viewed as an extension of Bibel's matrix method [4]. Within this paper we focus on the basic ideas and syntactical concepts and refer to [18] or [13] for details.

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 sub-formulae according to the tableaux scheme in table 1. We use the concept of *signed formulae* where each sub-formula B of A gets a *polarity* $k \in \{0,1\}$ depending on a positive (0) or negative (1) occurrence of B in A (starting with $\langle A, 0\rangle$). From this classification each signed (sub)formula $\langle B, k\rangle$ has *primary* type *Ptype* according to its tableaux class and a *secondary* type *Stype* according to its immediate parent formula.

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

We associate each sub-formula $\langle B, k\rangle$ of $\langle A, 0\rangle$ uniquely with a *position* x in $\ll$, respecting the order in the formula tree of A. B is called the *label* $lab(x)$ of x, k its *polarity* $pol(x)$, and $Ptype(x)$, $Stype(x)$ its *Ptype* and *Stype* corresponding to B. 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 $lab(x)$ which allows to use a uniform mechanism to define substitutions on both terms and positions and hence the reduction ordering. When dealing with intuitionistic logic we additionally have to insert ϕ and ψ positions into $\ll$ *before* all positions representing so-called *special* formulae (see table 1, where $\langle P, k\rangle$ denotes an atom).

Finally, to represent within $\ll$ *all* the formulae which will be necessary in a matrix proof for $\langle A, 0\rangle$, we extend $\ll$ by copies of sub-formulae needed more than once in the proof. These *generative* formulae have *Ptype* γ for all logics and, in addition, ν for modal logics and ϕ for J. A *multiplicity* $\mu \in \mathbb{N}$ is assigned to all positions in $\ll$, where $\mu(x){\geq}1$ if $Ptype(x) \in \{\gamma, \nu, \phi\}$ and $\mu(x){=}1$ otherwise. Furthermore in $\ll$ all *generative* positions x will receive $\mu(x)$ successors by inserting $\mu(x)$ distinct copies of its successor trees (having *Stypes* γ_0, ν_0 or ϕ_0 at their root).

In the resulting ordering $\ll$ the positions uniquely correspond to the sub-formulae of $\langle A, 0\rangle$. The sets of γ_0–, ν_0-, and ϕ_0-positions (Γ_0, $\mathcal{V}_0$, Φ_0) are called *variables*. The δ_0–, π_0–, and ψ_0-positions (Δ_0, Π_0, Ψ_0) are called *constants*. For integrating the Kripke semantics of modal logics and J we also need the concept of *prefixes*. Each position x in $\ll$ will be associated with a prefix $pre(x)$, i.e. the string of positions $y \in \mathcal{V}_0 \cup \Pi_0$ (modal logics except $S5$) or $y \in \Phi_0 \cup \Psi_0$ (J) between the root of $\ll$ and x where the root of $\ll$ is counted as element of Π_0 or Ψ_0. For $S5$ a prefix $pre(x)$ consists of the greatest ancestor $y \leqslant x$ with $y \in \mathcal{V}_0 \cup \Pi_0$.

Paths, Unification, and Complementarity For defining paths we start from 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 set of sets, each consisting of some leaves in $\ll$. These sets are called *paths* through a formula $\langle A, 0\rangle$. In order to prove this formula one has to show that each path through $\langle A, 0\rangle$ contains at least one *complementary connection* under a *global substitution* σ. Within σ we distinguish a *quantifier* substitution σ_Q and a modal (σ_M) or intuitionistic (σ_J) substitution. For unifying the definitions for modal logics and J we shall use the abbreviation σ_L and T_L for $L \in \{M, J\}$, where $T_M{=}\Pi_0 \cup \mathcal{V}_0$ and $T_J{=}\Psi_0 \cup \Phi_0$.

Let T_L^+ be the set of strings over T_L and $T_L^* := T_L^+ \cup \{\emptyset\}$. We define an L–substitution σ_L to be a mapping $\sigma_L : \mathcal{V}_0 \mapsto T_L^*$. The conditions on this *prefix–substitution* uniquely determine the logic $\mathcal{L}$ to be considered (see [18] and [13]). σ_L induces a relation $\sqsubset_L$ on $T_L \times T_L$ satisfying the following condition: *If* $\sigma_L(u){=}p$ *and* $p \notin \mathcal{V}_0$*, then for all* v *occurring in* $p, v \sqsubset_L u$.

Let $T_Q := \Gamma_0 \cup \Delta_0$ and $\mathcal{T}$ be a set of terms defined over $C_0 \cup T_Q$ (with C_0 set of constants). A *first order substitution* is a mapping $\sigma_Q : \Gamma_0 \mapsto \mathcal{T}$ inducing a relation $\sqsubset_Q$ on $\Delta_0 \times \Gamma_0$ by satisfying the following condition: *If* $\sigma_L(u){=}t$*, then* $v \sqsubset_Q u$ *for all* $v \in \Delta_0$ *which are sub-terms of* t.

The combination of $\sqsubset_L, \sqsubset_Q$, and $\ll$ defines a *reduction ordering* which encodes non-permutabilities of rules in sequent systems for L. In addition, there are certain *admissibility* conditions on the substitutions which 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).*

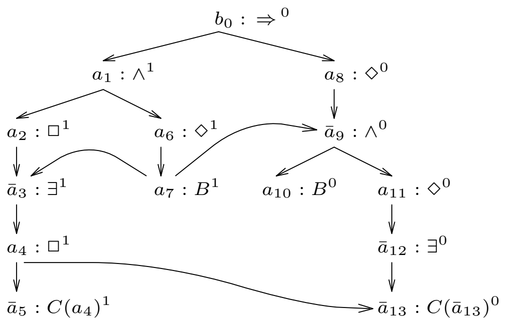
Definition 2. *A* combined substitution $\sigma := \langle \sigma_L, \sigma_Q\rangle$ *is* $\mathcal{L}$–admissible *provided:*

1. *For all* $p, q \in T_L^*$*:* pR_0q *implies* $\sigma_L^{\#}(p)R_0\sigma_L^{\#}(q)$*, where* $\sigma_L^{\#} : T_L^* \mapsto T_L^*$.
2. $\lhd := (\ll \cup \sqsubset)^+$ *is irreflexive, where* $\sqsubset := \sqsubset_L \cup \sqsubset_Q$.
3. *If* $\sigma_Q(u){=}t$ *then for all* $pre(v) \in P(t)$
   (i) varying domains*:* $\sigma_M^{\#}(pre(v)) = \sigma_M^{\#}(pre(u))$
   (ii) cumulative domains*: if* $v \notin \Gamma_0$ *then 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 (with* $pre(v)$ *is the root of* $\ll$ *if* $v \in C_0$ *and* $P(t){=}\{pre(v) \mid v \in T_Q \cup C_0,\ v$ *sub-term of* $t\}$*)*

Let $\langle A, 0\rangle$ be a signed formula, $\ll$ its position tree, p a path through $\langle A, 0\rangle$, and $\sigma := \langle \sigma_L, \sigma_Q\rangle$ an $\mathcal{L}$–admissible combined substitution. A *connection* is a subpath $\{u, v\} \subseteq p$, where $pol(u){\neq}pol(v)$ but u, v have the same predicate symbol in their labels. $\{u, v\}$ is called σ*-complementary*, iff (i) $\sigma_L^{\#}(pre(u)){=}\sigma_L^{\#}(pre(v))$, and (ii) $\sigma_Q^{\#}(lab(u)){=}\sigma_Q^{\#}(lab(v))$, where $\sigma_Q^{\#}$ is the homomorphic extension of σ_Q. A path is σ-complementary iff it contains a σ-complementary connection. A set $\mathcal{C}$ of σ-complementary connections *spans* a formula $\langle A, 0\rangle$ if each path through $\langle A, 0\rangle$ contains an element $\mathcal{C}$. The following characterization theorem can be proven (see [18]).

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

Example 1. Consider the formula $F \equiv \Box\exists x.\Box C(x) \wedge \Diamond B \Rightarrow \Diamond(B \wedge \Diamond\exists x.C(x))$. Its formula tree is shown below where $\mu(x){=}1$ for all positions x. 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 are shown on the right side table.



$$\begin{tikzcd}
 & & b_0 : \Rightarrow^0 \arrow[dl] \arrow[drr] & & & \\
 & a_1 : \wedge^1 \arrow[dl] \arrow[dr] & & & a_8 : \Diamond^0 \arrow[d] & \\
a_2 : \Box^1 \arrow[d] & & a_6 : \Diamond^1 \arrow[d] & & \bar{a}_9 : \wedge^0 \arrow[dl] \arrow[dr] & \\
\bar{a}_3 : \exists^1 \arrow[d] & & a_7 : B^1 \arrow[ll, bend right] \arrow[urr, bend left] & a_{10} : B^0 & & \bar{a}_{11} : \Diamond^0 \arrow[d] \\
a_4 : \Box^1 \arrow[d] \arrow[drrrrr, bend left] & & & & & \bar{a}_{12} : \exists^0 \arrow[d] \\
\bar{a}_5 : C(a_4)^1 & & & & & \bar{a}_{13} : C(\bar{a}_{13})^0
\end{tikzcd}$$

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

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

The formula has two paths $\{\{\bar{a}_5, a_7, a_{10}\}, \{\bar{a}_5, a_7, \bar{a}_{13}\}\}$. The corresponding connections are $\{a_7, a_{10}\}$ and $\{\bar{a}_5, \bar{a}_{13}\}$. The combined substitution $\sigma{=}\langle\sigma_Q, \sigma_M\rangle$ induces the relations $\sqsubset_Q$ and $\sqsubset_M$ which integrated into $\ll$ result in an irreflexive relation $\lhd$. σ_M respects the relation R_0 for $D, D4, T, S4, S5$ and satisfies the cumulative and constant domain conditions. So σ is $\mathcal{L}$–admissible for these cases and additionally makes the two connections (and hence the corresponding paths) σ-complementary. Thus the connections span the formula $\langle F, 0\rangle$ and 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_M, i.e. $\sigma_M(\bar{a}_5){=}\emptyset$ (for $T, S4$) or $\sigma_M(\bar{a}_5){=}a_7$ (for $S5$).

2.2 Sequent Calculi

We present a system of cut-free sequent calculi for C, J and for the *cumulative* and *varying domain*[1] modal logics $K, K4, D, D4, T, S4$ in a uniform and compact notation based on classical tableaux systems [1, 7]. According to the *Kripke–semantics* [6, 7] the domain conditions mean that either *wRv implies* $P(w) \subseteq P(v)$ ($P(x) \subseteq C_0$ is a set of constants known at world x) or that $P(w)$ and $P(v)$ are independent domains of their own for all w, v. To achieve uniformity in our presentation we shall use the concept of associated sets. Let Γ and Δ be the sets of antecedent formulae and succedent formulae respectively. We define $S_\Gamma{=}\{\langle F, 1\rangle \mid F \in \Gamma\}$, $S_\Delta{=}\{\langle F, 0\rangle \mid F \in \Delta\}$, and call $S{=}S_\Gamma \cup S_\Delta$ the *associative set* of $\Gamma \vdash \Delta$.

The rules of all sequent calculi and their conditions are uniformly described in table 2 and arranged according to the tableaux classification. They are directly usable for *cumulative domains*. We apply rules from bottom to top in order to *reduce* a formula. This *reduction-formula* is determined by the name of the rule, i.e. by the logical operator and its polarity. We further abbreviate the reduction-

[1] For constant domains such sequent calculi do not exist in all cases (see [7]).

formulae by their *Ptypes* (as e.g. in the α–reductions $\wedge l$ and $\vee r$). Since we do not need *structural rules*, we have to copy some reduction-formulae explicitly into the rule's premises to retain completeness. These are the γ-formulae and the ν-formulae for all modal logics except $K, K4$. For the δ-rules the constant a has to be new in their premises (eigenvariable-condition).

$\mathcal{L}$:	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

$\mathcal{L}$	VD
$D, D4$	π, ν
$T, S4,$ $(K), (K4)$	π

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

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

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

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

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

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

The table on top of the rules decides which logic $\mathcal{L}$ will be modeled 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 $\neg l$.

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* iff all branches are closed.

If we consider *varying domains* (VD) we have to check an additional condition when applying a γ-rule (see [7]) which says that only constants which are *alive*on a γ-branch β.are allowed to be introduced with $\gamma_0(t)$. The set C_β of all constants alive on β is defined as follows: (i) Starting with $\langle A,0\rangle$ C_β consists of all constants occurring in A. (ii) When applying a δ-rule $C_\beta := C_\beta \cup \{a\}$. (iii) When applying a modal rule $C_\beta := \{c\}$ on the actual branch β (where c is new since we deal with nonempty domains). (iv) For α– and β–rules C_β will not be modified.

2.3 Prefixed Sequent Systems

Prefixed sequent systems are an extension of the usual sequent calculi. Additionally they can also deal with *constant domains*. We construct them in a uniform notation from the *prefixed tableaux systems* developed in [7] for $K, K4, D, D4, T, S4$ in all domain variants, for $S5$ in varying and constant domains, and we present a prefixed sequent system for J. For classical logic C the prefixes should be ignored.

We define a *prefix* p to be a finite sequence of positive integers, for example p=1121. We extend signed formulae to *prefixed signed formulae* '$p : \langle A,k\rangle$' 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*(p) and *ext*(p, q).

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

Combining the properties of R_0 with the conditions *used* and *ext* we obtain a complete construction principle for prefixes when applying reduction rules. In the resulting rule–system (table 3) the prefixes in the premises are constructed from the ones in the conclusions using the conditions in the upper part of the table.[2]

$\mathcal{L}$:	C	J			$K, K4$		$D, D4, T, S4, S5$	
S^+	S	$S \cup \{r : \langle C, 1\rangle\}$			S		S	
		axiom	ϕ	ψ				
r, s	ε	$r \preceq s$	rR_0s and *used*(s) or *ext*(r, s)	*ext*(r, s)	$r = s$		$r = s$	
					ν	π	ν	π
p, q	–	–			pR_0q and *used*(q)	*ext*(p, q)	pR_0q and *used*(q) or *ext*(p, q)	*ext*(p, q)

Stype	*cumulative domains*	*varying domains*	*constant domains*
$q : \gamma_0(t)$	$t \in C_0^p, pR_0q \; or \; p = q$	$t \in C_0^q$	$t \in C_0$
$q : \delta_0(a)$	$a \in C_0^q, new(a)$		$a \in C_0, new(a)$

$\mathcal{L}$	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

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

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

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

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

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

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

Finally we have to give conditions for introducing terms depending on the domain variants. We divide a set of constants C_0 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 term t 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.

3 A Uniform Transformation Procedure

3.1 Relating Matrix Proofs and Prefixed Sequent Systems

Our procedure 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

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

basic idea is to traverse the reduction ordering $\propto^\star$ (created from $\lhd$ by removing redundant sub-formulae and adding a new root w) and to mark all the visited positions x with *solved*(x)=$\top$. The specific sequent rules and their parameters (i.e. prefixes and terms) are determined by 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 our prefixed sequent proofs.

Definition 4. *The set of immediate successors/predecessors of a position x in $\ll$ is denoted by $suc(x)/pred(x)$ while $suc(x)^+/pred(x)^+$ denotes* all *successors/predecessors of x. For $suc(x)=\{x_1,\ldots,x_m\}$ we assume a unique ordering on the successors and a selection function $suc_j(x)$ resulting the j^{th} successor of x in $\ll$ ($1\leq j\leq m$). Additionally we write $suc_j^+(x) := \{suc_j(x)\} \cup suc^+(suc_j(x))$.*

Definition 5. *Let $\propto^\star$ be a reduction ordering and x a position. $wait_1(x)$=$\top$ iff there exists $(y,x)\in\ \sqsubset_Q\ \cup\ \sqsubset_L$. x is called open,* open(z,x), *if $pred(x)$=z and $solved(z)$=$\top$, $solved(x)$=$\bot$. $free(x,\propto^\star)$ computes the pair ancerstor/rank (k,r) with the $\ll$-greatest ancestor k such that $x\in suc_r^+(k)$ and $open(k, suc_r(k))$.*

Besides traversing the ordering $\propto^\star$ we have to map the prefixes of the matrix proof to prefixes of the prefixed sequent systems. For using a prefix $\sigma_L^{\#}(pre(x))$ in $\propto^\star$ in a direct way we have to respect the *construction principle* for prefixes in the sequent systems. This principle means that prefixes are accessible via the relation R_0 together with the two conditions *used*(p) or *ext*(p,q).

The basic justification for doing this results from the admissibility conditions on the combined substitution σ. First we have to respect the relation R_0 on prefixes in $\propto^\star$. Second the relation $\sqsubset_L$ ensures that at each position x its prefix $\sigma_L^{\#}(pre(x))$ either has increased by a string of maximal length 1 (satisfying *ext*(p,q)), or $wait_1(x)$=$\top$ otherwise. The latter case means that all positions leading to the same prefix under $\sigma_L^{\#}$ have to be visited first by further traversing $\propto^\star$. After this $wait_1(x)$=$\bot$ and the prefix of x can be extended by a string of length $\geq$2 corresponding to the the *used*(p) condition. Form this we get:

Definition 6. (Prefix mapping) *Let T_L be the set of prefix-positions in $\propto^\star$ and $n : T_L \mapsto \mathbb{N}$ be an* injective *mapping with $n(w)$=0 for the new root w. Let P^{σ_L} be the collection of substituted prefixes $\sigma_L^{\#}(pre(x))$ of positions x in $\ll$. The* prefix mapping *$f_p : P^{\sigma_L} \mapsto \mathbb{N}^*$ is defined by $f_p(q^{\sigma_L})$=$f_p(a_1a_2\ldots a_n)$:=$n(a_1)n(a_2)\ldots n(a_n)$ for each $q^{\sigma_L}\in P^{\sigma_L}$.*

The reduction ordering $\propto^\star$ together with the prefix mapping f_p gives us a *construction* ordering on the prefixes in the sequent proof and thus respects the construction principle on prefixes.

3.2 The Algorithm *TOTAL*

The uniform transformation algorithm *TOTAL* is presented in figure 1. It takes as input a reduction ordering $\propto^\star$ and a logic $\mathcal{L}$ such that $\propto^\star$ represents a matrix proof in $\mathcal{L}$. The result is a list of reduction rules $\mathcal{S}$*-list* representing a proof in the prefixed sequent system for $\mathcal{L}$. With the following definitions we introduce the mechanism for splitting at β-positions.

```
function $TOTAL(\propto^\star, \mathcal{L}) : \mathcal{S}\text{-}list$
  for all $x \in positions(\propto^\star)$  do $solved(x) := \bot$ od
  let $w := root(\propto^\star)$ and $solved(w) := \top$
  return $TOT(\propto^\star, \mathcal{L})$
function $TOT(\propto^\star, \mathcal{L}) : \mathcal{S}\text{-}list$
  let $proven(\propto^\star) := \bot$ and $\mathcal{S}(\propto^\star) := []$
  for all $x \in positions(\propto^\star)$  do $compute\ wait_1(x)$ od
  while $not\ proven(\propto^\star)$ do
    $select\ open(z, x)$
    $\mathcal{S}(\propto^\star) := append(\mathcal{S}(\propto^\star), SOLVE((z, x), \propto^\star, \mathcal{L}))$
  od
  return $\mathcal{S}(\propto^\star)$
function $SOLVE((z, x), \propto^\star, \mathcal{L}) : \mathcal{S}\text{-}list$
(1)   if $wait_1(x)$ then
        $(k, r) := free(y, \propto^\star)$, where $(y, x) \in \sqsubset$
        return $SOLVE((k, suc_r(k)), \propto^\star, \mathcal{L})$
      else
(2)     if $Ptype(z) \in \{\gamma, \nu\}$ then $r_1 :=$ Rule $(Ptype(z), (z, x), \mathcal{L})$ else $r_1 := empty$ fi
        $solved(x) := \top$
        if $Stype(x) \in \{\psi_0, \nu_0\}$ then $\propto^\star := update(x, \propto^\star)$ fi
        if $Ptype(x) \in \{\pi, \delta\}$ then $\propto^\star := update(suc_1(x), \propto^\star)$ fi
        case $Ptype(x)$ of
          $\{\gamma, \nu, \phi, \psi\}$ :  return $[r_1]$
          $\{-\}$ :           let $\{x, y\} \in Connections(\propto^\star)$
                          if $solved(y)$ then
                            $proven(\propto^\star) := \top$
                            return $[r_1,$ Rule $(axiom, \{x, y\}, \mathcal{L})]$
                          else
                            $(k, r) := free(y, \propto^\star)$
                            return $[r_1 \mid SOLVE((k, suc_r(k)), \propto^\star, \mathcal{L})]$
                          fi
          $\{\delta, \alpha\}$ :        return $[r_1,$ Rule $(Ptype(x), x, \mathcal{L})]$
(3)       $\{\pi\}$ :            return $[r_1,$ Rule $(\pi, x, \mathcal{L})]$
          $\{\beta\}$ :           $p_0 := [r_1,$ Rule $(\beta, x, \mathcal{L})]$
                          $[\propto_1^\star, \propto_2^\star] := \beta\text{-}split(x)$
                          for $i = 1, 2$ do $p_i := TOT(\propto_i^\star, \mathcal{L})$ od
                          $proven(\propto^\star) := \top$
                          return $append(p_0, append(p_1, p_2))$
        esac
(4)
      fi
function $update(x, \propto^\star) : reduction\text{-}ordering$
  for all $\{y \mid (x, y) \in \sqsubset\}$ do
    $\sqsubset := \sqsubset \setminus \{(x, y)\}$
    $wait_1(y) := \bot$
  od
```

Fig. 1. The uniform transformation procedure

Definition 7. *Let x be a position of $\propto^\star$ and $\ll^x$ the subtree ordering with root x and position set $pos(x)$. The subrelation of $\propto^\star$ involving positions from $pos(x)$ is defined as $t^x := \ll^x \cup \sqsubset^x$, where $\sqsubset^x := \sqsubset \setminus \{(x_1, x_2) \mid x_1 \in pos(x)$ or $x_2 \in pos(x)\}$. If C is the connection set of $\propto^\star$, we define $C^x := C \setminus \{\{c_1, c_2\} \mid c_1 \in pos(x)\}$.*

Definition 8. *Let x be a β–node with $suc(x)=\{x_1, x_2\}$.
We define β–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 connections we have $C_i := C \setminus C^{x_j}$ where $j \in \{1, 2\}, j \neq i$.*

$TOTAL(\propto^\star, \mathcal{L})$ begins with *global* initializations concerning the *solved*-marks and *local* initializations within the sub-procedure $TOT(\propto^\star, \mathcal{L})$. Then we step into the main loop by selecting a position x that shall be ‘solved’. $SOLVE((z, x), \propto^\star, \mathcal{L})$ tries to do this by forming appropriate sequent rules, provided that there are no $wait_1$-labels. Otherwise some unsolved ancestor position $free(y, \propto^\star)$ of a position y with $(y, x) \in \sqsubset$ has to be considered first.

The sequent rules will be constructed according to $\mathit{Ptype}\,(x)$ and table 1. Before working on x the reduction rule r_1 of some generative position $\mathit{pred}(x) \in \{\gamma, \nu\}$ has to be made up. x will be marked as *solved* and all wait_1-labels caused by x are removed from $\propto^\star$ using *update*. If x has no *Ptype* it must be part of a connection $\{x, y\}$. We can terminate if y has already been solved. Otherwise we start with a new position using the "goal"–directed selection function $\mathit{free}(y, \propto^\star)$ again. 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 and subproof-transformations for each of these two suborderings and terminate with $\mathit{proven}\,(\propto^\star) := \top$ afterwards.

$\mathsf{Rule}\,(T, x, \mathcal{L})$		$\mathsf{Rule}\,(T, (z, x), \mathcal{L})$		$\mathsf{Rule}\,(\mathit{axiom}, \{x, y\}, \mathcal{L})$
T	*sequent rule*	T	*sequent rule*	*sequent rule*
$\{\alpha, \beta\}$	$T(\mathit{sform}\,(x))$			$\mathit{axiom}\,(\mathit{sform}\,(x), \mathit{sform}\,(y))$
δ	$\delta(\mathit{sform}\,(x), \mathit{suc}_1(x))$	γ	$\gamma(\mathit{sform}\,(z), \sigma_Q(x))$	
π	$\pi(\mathit{sform}\,(x))$	ν	$\nu(\mathit{sform}\,(z))$	

J			Modal Logics		
Ptype	*old prefix*	*new prefix*	*Ptype*	*old prefix*	*new prefix*
special $\{\alpha, \beta, \delta\}$	$f_p^{\sigma_L}(\mathit{pred}(x))$	$f_p^{\sigma_L}(x)$	π	$f_p^{\sigma_L}(x)$	$f_p^{\sigma_L}(\mathit{suc}_1(x))$
special γ	$f_p^{\sigma_L}(\mathit{pred}(z))$	$f_p^{\sigma_L}(z)$	ν	$f_p^{\sigma_L}(z)$	$f_p^{\sigma_L}(x)$
other	$f_p^{\sigma_L}(y)$	$f_p^{\sigma_L}(y)$	*other*	$f_p^{\sigma_L}(y)$	$f_p^{\sigma_L}(y)$

Table 4. Rule instantiations for various logics

To construct the actual prefixed sequent rules from some position $y \in \{x, z\}$ we instantiate the statements $\mathsf{Rule}\,(\mathit{Ptype}\,(x), x, \mathcal{L})$ or $\mathsf{Rule}\,(\mathit{Ptype}\,(z), (z, x), \mathcal{L})$ according to table 4. The prefixed sequent proof starts with $0 : \langle A, 0\rangle$ since $n(w)=0$ and $\mathit{sform}\,(w)=\langle A, 0\rangle$ for the new root w in $\propto^\star$. We assume an associated set S in the prefixed sequent system to which the constructed rule will be applied. From the primary type T of a position $y \in \{x, z\}$, the signed formula $\mathit{sform}\,(y)$ (upper table), and the corresponding entry *old prefix* (lower table via mapping f_p on the prefix $\sigma_L^{\#}(\mathit{pre}(y)))$ we uniquely obtain the prefixed signed formula and rule name for reduction in S.[3] From the classification table 1 we get the sub-formulae which have been processed by rule application. The new prefix belonging to these sub-formulae can be determined using the entry *new prefix*.

Since in matrix proofs for J the prefixes will be constructed between ϕ, ϕ_0 or ψ, ψ_0 positions the *new prefix* has already been developed at the *special* positions encoding a reduction rule in the sequent proof. For this reason we must extract the *old prefix* from the predecessor $\mathit{pred}(y)$ which is either a ϕ– or a ψ-position.

The eigenvariable introduced at a δ-position x is uniquely determined by $\mathit{suc}_1(x)$ for the sequent proof. For a γ-position z we take $\sigma_Q(x)$. Because of the admissible interactions between σ_Q and σ_L we can use these terms directly to satisfy the domain conditions for the prefixed sequent systems (table 3). For the *axiom*–rule we can conclude the conditions for J from the correctness of the prefix mapping. For classical logic C all considerations about prefixes should be ignored. Summarizing our considerations we obtain the following result about our transformation.

Theorem 2. *For the logics C, J, $S5$ in varying and constant domains, and $K, K4$, $D, D4, T, S4$ in constant, cumulative, and varying domains the transformation algorithm* TOTAL *is* complete *for conversion into prefixed sequent systems.*

[3] In the table we use the abbreviation $f_p^{\sigma_L}(y)$ for $f_p(\sigma_L^{\#}(\mathit{pre}(y)))$.

Example 2. 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 1, the substitutions σ_M, σ_Q, and the reduction ordering $\propto^\star$ 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 $f_p^{\sigma_M}$ for *all* positions as follows:

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

We start by traversing the reduction ordering $\propto^\star$ for $D, D4, T, S4$ with cumulative domains. The corresponding 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 quantifier parameter next to the rule name using table 4 and the table above). After skipping the ν-position a_8 we are blocked at $\bar{a}_9$ because $wait_1(\bar{a}_9)=\top$. We compute $free(\bar{a}_9, \propto^\star)=(a_1, 2)$ to reduce $suc_2(a_1)=a_6$ (creating $\Diamond l : 07$ and deleting $wait_1$-labels). Solving a_7 gives us the atom $B^1 : 07$. Next we skip a_2 and, reaching $\bar{a}_3$, construct the two rules $\Box l : 07$ and $\exists l : 07; a_4$ (introducing $suc_1(\bar{a}_3)=a_4$ and deleting the $wait_1$–label at a_4). Visiting a_4 and $\bar{a}_5$ the atom $C(a_4)^1 : 075$ can be isolated in the sequent proof. The reduction at $\bar{a}_9$ ($\wedge l : 07$) forces a split of the reduction ordering $\propto^\star$ due to definition 8. For $\propto_1^\star$, we solve a_{10} having the atom $B^1 : 07$, and hence an axiom rule with $B^0 : 07$ (a_7 already solved). For reducing $\propto_2^\star$ we visit $a_{11}, \bar{a}_{12}$ and $\bar{a}_{13}$ obtaining $\Box r : 075$ 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, solving $\bar{a}_{13}$ and applying the axiom rule with $C(a_4)^0 : 075$ will finish sequent proof. The eigenvariable a_4 is associated to 07 and the prefix at $\bar{a}_{13}$ is given by 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 $\sigma_M(\bar{a}_5)=\emptyset$ into the prefix mapping $f_p^{\sigma_M}$ above.

3.3 Considerations about β–splits

The extraction of sequent proofs from the compact representation of matrix proofs requires a concept for rebuilding the *notational redundancies* into a sequent proof. The main problem occurs when splitting at β-positions because one has to decide which subrelations of $\propto_i^\star$ are still relevant in this sub-proof 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 all semantic information concerning a specific logic are encoded by the rules for manipulating prefixes. Thus deleting *irrelevant subrelations* after a β–split is an optimization which eliminates redundancies. In contrast to that a conversion into convential sequent calculi (section 4) must take into account that the semantics of a logic is completely contained in the non-permutabilities of the rules. Thus the reduction ordering $\propto^\star$ has to be extended by additional *wait*-labels which make deleting subrelations after β-splits essential for preserving completeness.

In [16] we have developed a concept of reductions on non-normal matrix proofs which respects these optimization- and completeness features. This concept is one of the theoretical foundations of the transformation algorithm and is based on the principle of *proof relevant positions*. However, we were able to show that for some of the logics $K, K4$ and $D4_{co}$ (*co* $\hat{=}$ *constant domains*) this principle fails since there are theorems which have *pure* relevant sub-formulae not involved in any connection of the matrix proof. For all other logics one should delete redundant subrelations after splitting even when creating prefixed sequent proofs.

4 The Transformation for Conventional Sequent Calculi

4.1 Relating Sequent Calculi to the Reduction Ordering

When transforming matrix proofs into conventional sequent calculi we have to take into accountthat 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 order to deal with such non-permutabilities which are *not* completely encoded by the reduction ordering $\propto^\star$ the algorithm must consider *proof relevant positions* and extend $\propto^\star$ by additional $wait_2$-labels. Due to the above results we have to restrict ourselves to C, J and $D, D4, T, S4$ for cumulative and varying domains.

Definition 9. *Let $\ll$ be the tree ordering of $\propto^\star$ and C the set of connections.*

1. *A leaf c of $\ll$ is called* connected *if it occurs in a connection $\{c, c_2\} \in C$. A position x is called* dynamically proof relevant *iff either $solved(x)=\top$ and x itself is connected, or $open(z,x)$ and there is a connected $c \in \{x\} \cup suc^+(x)$.*
2. *A position y is called ν_0*-unclosed *if* Stype $(y)=\nu_0$ *and y is the smallest position (wrt $\ll$) such that (i) either y is open and not marked* reduced *wrt $pred(y)$, or (ii) $pred(y)$ is open and* Stype $(pred(y)) \neq \gamma_0$.

The first part of the definition mirrors the fact that the associated set S of a sequent proof either contains an atom ready for the axiom rule or a formula containing axiom-relevant atoms. The ν_0-*unclosed*–property will be necessary for capturing all possibilities in which a ν-formula, already isolated in 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 γ-reduction. $pred(y)$ will be skipped by further traversing $\propto^\star$ and the corresponding ν-rule will not be applied before reaching y. But in the sequent proof the ν-formula corresponding to $pred(y)$ is already isolated and hence reducible in S when skipping the ν-position. Thus y must be considered *before* skipping $pred(y)$ by calling it ν_0-unclosed.

For defining $wait_2$-labels we have to consider two new aspects. First $wait_2$-labels shall prevent us from deleting relevant positions whose corresponding formulae will not be saved in S^*. Second for T, D, and $D4$ they should keep us from doing *incorrect* reductions. These may occur if the application of a π- or a ν-rule (the latter only for $D, D4$) forces us to do a *macro step*. The reason for this is that in addition to the rule application itself all ν-formulae in the set S become reduced in S^* *without* caring about $wait_1$-labels of the corresponding positions in $\propto^\star$.

For T, D only the ν_0 formulae, for $D4$ all ν_0 *and* ν formulae will be saved in S^*. For traversing $\propto^\star$ we obtain: (i) When reducing a π-position x in $T, D, D4$ or a ν-position z at x in $D, D4$ the $wait_1$-labels caused by x itself (for ν) or

$suc_1(x)$ (for π) have not to be taken into account because they will be deleted in the same step. (ii) In T, D all other $wait_1$-labels blocking an arbitrary ν_0-unclosed position y have to be respected, i.e. $wait_2(x){=}\top$ if there is some ν_0-unclosed y with $wait_1(y){=}\top$ not caused by x or $suc_1(x)$. In $D4$ this test becomes superfluous since all ν-formulae will move to S^* and all $wait_1$-labels are respected in $\propto^\star$. (iii) For all logics: if the π-/ν-rule has been applied each ν_0-unclosed position y becomes *open*, i.e. $solved(pred(y)) := \top$. Additionally, each ν_0-position y receives a new mark *reduced* if $wait_1(y){=}\bot$. This indicates that, when reaching y by further traversing $\propto^\star$, no ν-rule for the ν-position $pred(y)$ has to be applied anymore since after reducing a π-/ν-formula in T, D no generative ν-formula exists in S^*. In $D4$ each ν-formula saved in S^* can be used several times which is encoded in the number of $wait_1$-labels blocking y. Consequently we can set a *reduced*-mark to y after the last "reuse" of its predecessor ν-position.

All these considerations will be taken into account by our definition of $wait_2$-labels which extend the reduction ordering $\propto^\star$. Depending on the traversing 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.

Definition 10. *Let x be a position in $\propto^\star$. For modal logics let U_{ν_0} be the set of ν_0-unclosed positions. Let $W_1^k \subseteq U_{\nu_0}$ be the set consisting of those positions y for which $wait_1(y){=}\top$ without observing $(k, y) \in\, \sqsubset$, where either $k{=}x$ being a ν_0-position in D, or $k{=}suc_1(x)$ at a π-position x in T, D. Then $wait_2(x){=}\top$ iff (i) x satisfies the conditions in the upper row of the table below and (ii) there is a proof relevant position $y{\neq}x$ satisfying the conditions in the second row.*

J	$S4$	T	D	$D4$
Stype$(x){=}\psi_0$ *(except atom)*	*Ptype*$(x){=}\pi$		*Ptype*$(x){=}\pi$ or *Stype*$(x){=}\nu_0$ (not *reduced*)	
$pol(y){=}0$	$y \notin U_{\nu_0}$	$y \notin U_{\nu_0}$ or $W_1^k{\neq}\emptyset$		$y \notin U_{\nu_0}$

4.2 Adapting the Transformation Algorithm

Using the above considerations we lift $TOTAL$ to an algorithm $TOTAL^\star$ which converts matrix proofs into sequent proofs for C, J and $S4, T, D, D4$ with cumulative and varying domains. For this purpose we present three extensions of our algorithm in figure 1. First $wait_2$-labels are integrated dynamically into the reduction ordering and analyzed during the transformation. Second for the logics $T, D, D4$ we have to manage the reductions of the ν_0-unclosed positions when a π-rule ($T, D, D4$) or a ν-rule ($D, D4$) has been applied. That means setting $open(z, y)$ for all $y \in U_{\nu_0}$ and distributing *reduced*-marks to some of these ν_0-positions. Furthermore whenever such a ν_0-position is reached we have to check if it is still necessary to reduce its generative predecessor by constructing a ν-rule. Finally splitting at β-positions becomes a more complex operation since in order to guarantee completeness we have to prevent the transformation algorithm from running into deadlocks caused by the new $wait_2$-labels. For details concerning the concept of *β–splits* we refer to [16].

We shall present the modifications of our algorithm which deal with the first two aspects by replacing the boxed areas (1),(2),(3), and (4) as follows:

1. For integrating the *wait*$_2$-labels we assume that the sets U_{ν_0} and $W_1^x, W_1^{suc_1(x)}$ have already been computed. From this we obtain that box (1) has to be replaced by the following (where *blocked*(x)=*wait*$_1(x)$ *or* *wait*$_2(x)$):

```
if $blocked(x)$ then
  if $wait_1(x)$ then
    $(k,r) := free(y, \propto^\star)$, where $(y,x) \in \sqsubset$
    return $SOLVE((k, suc_r(k)), \propto^\star, \mathcal{L})$
  else ($wait_2(x)$)
    $select\ open(z_2, y)$, where $x \neq y$
    return $SOLVE((z_2, y), \propto^\star, \mathcal{L})$
  fi
```

Since there is no heuristic for selecting *open*(z_2, y) we need a *fair strategy* on this selection function to guarantee the termination of this selection process.

2. For checking if reductions at generative ν-positions are necessary $(T, D, D4)$ and managing ν-reductions forming S^* in $D, D4$ we replace box (2) by:

```
case $Ptype(z)$ of
  $\{\gamma\}$ :   $r_1$ := Rule $(\gamma, (z,x), \mathcal{L})$
  $\{\nu\}$ :   if $x$ is not marked $reduced$ then
                $r_1$ := Rule $(\nu, (z,x), \mathcal{L})$
                for all $y \in U_{\nu_0}$ do
                  $solved(pred(y)) := \top$
                  if $\mathcal{L} \in \{D, D4\}$ and $wait_1(y)=\bot$ then set $y$ $reduced$ fi
                od
              else $r_1 := empty$ fi
  $\{else\}$ : $r_1 := empty$
esac
 if $\mathcal{L} \in \{D, D4\}$ and $Stype(x)=\nu_0$ and $Ptype(x)=\pi$
    and $wait_2(x)$ (with the new sets $U_{\nu_0}$ and $W_1^{suc_1(x)}$) then
       $select\ open(z_2, y)$, where $x \neq y$
       return $SOLVE((z_2, y), \propto^\star, \mathcal{L})$
 else
```

Wheras he first statement realizes additional ν-reductions the second takes into account a special case for $D, D4$. Although there is no *wait*$_2$–label at x *before* the ν-reduction at z, a new *wait*$_2$–label can occur afterwards if x is a π-position. Since U_{ν_0}, $W_1^{suc_1(x)}$, and the conditions satisfying *wait*$_2$-labels may have changed we have to repeat the test for *wait*$_2$-labels.

3. We replace (3) by adding reduced-marks for $T, D, D4$ during π-reduction:

```
$\{\pi\}$ :  for all $y \in U_{\nu_0}$ do
          $solved(pred(y)) := \top$
          if $\mathcal{L} \in \{T, D, D4\}$ and $wait_1(y)=\bot$ then set $y$ $reduced$ fi
        od
        return $[r_1$, Rule $(\pi, x, \mathcal{L})]$
```

4. (4) must be replaced by $\boxed{\textbf{fi}}$ to close an 'open' **else**-branch in (2).

For instantiating the sequent rules we again use the upper part of table 4. At position y the uniform rule construction from *Ptype*(y) and *sform*(y) remains unchanged. The starting point is now *sform*$(w)=\langle A, 0\rangle$. When transforming modal matrix proofs with varying domains the admissibility of σ_Q and σ_M ensures a correct order of quantifier reductions such that the resulting terms satisfy the conditions on varying domains sequent calculi too. From this we obtain:

Theorem 3. *For the logics C, J and $D, D4, T, S4$ in cumulative and varying domains the algorithm* TOTAL* *using* wait$_2$*-labels,* reduced*-marks, and subrelation–reductions after β-splits is complete for conversion into sequent calculi.*

5 Conclusion

We have presented a procedure for transforming non-classical and classical matrix proofs into sequent proofs. It is based on a unified representation of matrix characterizations of logical validity and of sequent calculi for various logics and 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, efficient, and simple way. It would be easy to extend our algorithm to logics not yet considered simply by extending the tables appropriately. Another advantage of our algorithm is that it converts a given matrix proof into a sequent proof without any additional search. If combined with an efficient proof search procedure for the underlying logics it can therefore be used for efficiently constructing formal proofs in a humanly comprehensible form.

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 also intend to investigate extensions of our procedure to fragments of linear logic and the possibilities for integrating induction techniques by similar techniques.

References

1. E. W. Beth. *The foundations of mathematics.* North–Holland, 1959.
2. W. Bibel, S. Brüning, U. Egly, T. Rath. Komet. In *CADE–12*, LNAI 814, pp. 783–787. Springer, 1994.
3. W. Bibel. On matrices with connections. *J. of the ACM*, 28:633–645, 1981.
4. W. Bibel. *Automated Theorem Proving.* Vieweg Verlag, 1987.
5. B. I. Dahn, J. Gehne, Th. Honigmann, L. Walther, A. Wolf. Integrating Logical Functions with *ILF*; Preprint 94-10, Humboldt University Berlin, 1994.
6. M. C. Fitting. *Intuitionistic logic, model theory and forcing.* Studies in logic and the foundations of mathematics. North–Holland, 1969.
7. M. C. Fitting. *Proof Methods for Modal and Intuitionistic Logic.* D. Reidel, 1983.
8. R. Letz, J. Schumann, S. Bayerl, W. Bibel. Setheo: A high-performance theorem prover. *Journal of Automated Reasoning*, 8:183–212, 1992.
9. C. Lingenfelder. Structuring computer generated proofs. *IJCAI-89*, 1989.
10. C. Lingenfelder. *Transformation and Structuring of Computer Generated Proofs.* PhD thesis, Universität Kaiserslautern, 1990.
11. H. J. Ohlbach. A resolution calculus for modal logics. Ph.D. Thesis, Universität Kaiserslautern, 1988.
12. J. Otten, C. Kreitz. A connection based proof method for intuitionistic logic. *TABLEAUX-95*, LNAI 918, pp. 122–137, Springer, 1995.
13. J. Otten, C. Kreitz. A Uniform Proof Procedure for Classical and Non-Classical Logics Technical Report, TH Darmstadt, 1996.
14. J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. of the ACM*, 12(1):23–41, 1965.
15. S. Schmitt, C. Kreitz. On transforming intuitionistic matrix proofs into standard-sequent proofs. *TABLEAUX–95*, LNAI 918, pp. 106–121, Springer, 1995.
16. S. Schmitt. Ein erweiterter intuitionistischer Sequenzenkalkül und dessen Anwendung im intuitionistischen Konnektionsbeweisen. TH Darmstadt, 1994.
17. L. Wallen. Matrix proof methods for modal logics. *IJCAI-87*, p. 917–923. 1987.
18. L. Wallen. *Automated deduction in nonclassical logic.* MIT Press, 1990.
19. L. Wos et. al. Automated reasoning contributes to mathematics and logic. *CADE–10*, LNCS 449, p. 485–499. Springer 1990.