# Non-clausal Connection Calculi for Non-classical Logics

Jens Otten

*Department of Informatics, University of Oslo*
*PO Box 1080 Blindern, 0316 Oslo, Norway*
jeotten@ifi.uio.no

**Abstract.** The paper introduces non-clausal connection calculi for first-order intuitionistic and several first-order modal logics. The notion of a non-clausal matrix together with the non-clausal connection calculus for classical logic are extended to intuitionistic and modal logics by adding prefixes that encode the Kripke semantics of these logics. Details of the required prefix unification and some optimization techniques are described. Furthermore, compact Prolog implementations of the introduced non-classical calculi are presented. An experimental evaluation shows that non-clausal connection calculi are a solid basis for proof search in these logics, in terms of time complexity and proof size.

## 1 Introduction

Intuitionistic and modal logics are among the most popular *non-classical logics*. *Intuitionistic logic* is used, e.g., within interactive proof assistants, such as NuPRL [4] and Coq [2]. *Modal logics* have applications in, e.g., planning, natural language processing, and program verification. Hence, *(fully) automated reasoning* in these logics is an important task and many applications would benefit from more powerful reasoning tools. Unfortunately, *automated theorem proving (ATP)*, i.e., deciding whether a formula is *valid* in these non-classical logics is even harder than for classical logic. For the propositional fragment, intuitionistic and (most) modal logics are *PSPACE-complete* whereas classical logic is "only" *NP*-complete. Adapting complex ATP systems for classical first-order logic to these non-classical logics is in general not easily possible.

A popular approach for dealing with intuitionistic and modal logics is to encode their *Kripke semantics* with so-called labels or *prefixes* [22, 23]. Two of the most powerful ATP systems for these logics, ileanCoP [11] and MleanCoP [14], use prefixes and are based on *clausal* connection calculi [10]. While the use of a clausal form technically simplifies the proof calculus, the standard translation as well as the definitional translation into clausal form introduce a significant overhead into the proof search [12]. Furthermore, both translations modify the original structure of the formula.

This paper introduces prefixed *non-clausal* connection calculi for first-order intuitionistic and several first-order modal logics. Syntax, semantics, prefixes and the underlying matrix characterizations are described (Section 2). Afterwards, non-clausal calculi are presented together with prefixed non-clausal matrices and the required prefix unifications for intuitionistic logic (Section 3) and modal logic (Section 4). After the description of some optimization techniques (Section 5), compact implementations of these non-clausal calculi are presented (Section 6) and evaluated (Section 7). The paper concludes with a short summary and an outlook on future work (Section 8).

## 2 Preliminaries

The standard notation for first-order formulae is used. Terms (denoted by $t$) are built up from functions (denoted by $f$), constants and (term) variables (denoted by $x$). An atomic formula (denoted by $A$) is built up from predicate symbols and terms. A *(first-order) formula* (denoted by $F, G, H$) is built up from atomic formulae, the connectives $\neg$, $\wedge$, $\vee$, $\Rightarrow$, and the standard first-order quantifiers $\forall$ and $\exists$. A *literal* $L$ has the form $A$ or $\neg A$. Its *complement* $\overline{L}$ is $A$ if $L$ is of the form $\neg A$; otherwise $\overline{L}$ is $\neg L$. A *connection* is a set $\{A, \neg A\}$ of literals with the same predicate symbol but different polarity. A *quantifier* or *term substitution* $\sigma_Q$ is a mapping from the set of term variables to the set of terms. In $\sigma_Q(L)$ all term variables $x$ in $L$ are substituted by their image $\sigma_Q(x)$.

### 2.1 Intuitionistic Logic

*Intuitionistic logic* and classical logic share the same *syntax*, but their *semantics* is different. For example, the formula $A \vee \neg A$ is valid in classical logic but not in intuitionistic logic. The semantics of intuitionistic logic requires a proof for $A$ or for $\neg A$. As this property neither holds for $A$ nor for $\neg A$, the formula is not valid in intuitionistic logic. Formally, the semantics of intuitionistic logic is specified by a Kripke semantics [23].

Hence, the following three rules of the (multi-succedent) *sequent calculus for intuitionistic logic* [6, 23] differ from the ones for classical logic:

$$\frac{\Gamma, G \vdash}{\Gamma \vdash \neg G, \Delta}\ \neg\textit{-right}\ , \qquad \frac{\Gamma, G \vdash H}{\Gamma \vdash G \Rightarrow H, \Delta}\ \Rightarrow\textit{-right}\ , \qquad \frac{\Gamma \vdash G[x\backslash a]}{\Gamma \vdash \forall x G, \Delta}\ \forall\textit{-right}\ .$$

In all three rules the set of formulae $\Delta$ does not occur in the sequent of the premises anymore. As these formulae might be necessary within a bottom-up search in order to complete a proof, the application of these rules need to be controlled. To this end, a prefix is assigned to each subformula $G$ of a given formula $F$.

**Definition 1 (Intuitionistic prefix).** *A* prefix *(denoted by $p, q$) is a string (sequence of characters) over an alphabet $\Phi \cup \Psi$, in which $\Phi$ is a set of* prefix variables *($V_1, \ldots$) and $\Psi$ is a set of* prefix constants *($a_1, \ldots$). For every $\neg / \Rightarrow / \forall$ or atomic formula $A$ preceding a subformula $G$ of a formula $F$, an element of $\Phi$ or $\Psi$ is added to the prefix $p$ of $G$ depending on the "polarity" of $\neg / \Rightarrow / \forall / A$ (see [23] or Section 3 for details).*

Semantically, a prefix encodes a sequence of worlds in a Kripke model. Proof-theoretically, prefix constants and variables represent applications of the rules $\neg$*-right*, $\Rightarrow$*-right*, $\forall$*-right*, and $\neg$*-left*, $\Rightarrow$*-left*, $\forall$*-left* in the sequent calculus, respectively. The prefix $p$ of a subformula $G$, denoted by $pre(G)$ or $G : p$, specifies the sequence of these rules that have to be applied (bottom-up) to obtain $G$ in the sequent. In order to preserve the atomic formulae that form an axiom in the intuitionistic sequent calculus, their prefixes need to unify under an intuitionistic substitution $\sigma_I$. An additional *domain condition* ensures that $\sigma_Q$ and $\sigma_I$ are mutually consistent [23].

**Definition 2 (Intuitionistic substitution; $\sigma$-complementary).** *An* intuitionistic substitution $\sigma_I : \Phi \to (\Phi \cup \Psi)^*$ *maps elements of $\Phi$ to strings over $\Phi \cup \Psi$. In $\sigma_I(p)$ prefix variables are replaced according to $\sigma_I$. A connection $\{L_1 : p_1, L_2 : p_2\}$ is* $\sigma$-complementary *for a combined substitution $\sigma = (\sigma_Q, \sigma_I)$ iff $\sigma_Q(L_1) = \sigma_Q(\overline{L_2})$ and $\sigma_I(p_1) = \sigma_I(p_2)$. $\sigma$ is* admissible *iff the cumulative domain condition holds (see Section 2.2).*

### 2.2 Modal Logics

*Modal logics* extend the *syntax* of classical logic with the unary modal operators $\Box$ and $\Diamond$. They are used to represent the modalities "it is necessarily true that" and "it is possibly true that", respectively. The *Kripke semantics* of the standard modal logics are defined by a set of worlds $W$ and a binary *accessibility relation* $R_i \subseteq W \times W$ between these worlds. In each single world $w \in W$ the classical semantics applies to the standard connectives and quantifiers, whereas the modal operators are interpreted with respect to accessible worlds: $\Box F$ or $\Diamond F$ are true in a world $w$, if $F$ is true in *all* worlds $w'$ or *some* world $w'$ with $(w,w') \in R$, respectively. The properties of the accessibility relation $R$ determine the specific modal logic. In this paper the modal logics D, T, S4, and S5 are considered. Their accessibility relation is serial (D)[1], reflexive (T), reflexive and transitive (S4), or an equivalence relation (S5). The standard semantics is considered with rigid term designation, i.e. every term denotes the same object in every world, and terms are local, i.e. any ground term denotes an existing object in every world.

The *sequent calculus for the (cumulative) modal logics D, T, and S4* consists of the axiom and rules of the classical sequent calculus and four additional modal rules:

$$\frac{\Gamma^+, F \vdash \Delta^+}{\Gamma, \Box F \vdash \Delta}\ \Box\text{-}left \qquad \frac{\Gamma^* \vdash F, \Delta^*}{\Gamma \vdash \Box F, \Delta}\ \Box\text{-}right$$

$$\frac{\Gamma^+ \vdash F, \Delta^+}{\Gamma \vdash \Diamond F, \Delta}\ \Diamond\text{-}right \qquad \frac{\Gamma^*, F \vdash \Delta^*}{\Gamma, \Diamond F \vdash \Delta}\ \Diamond\text{-}left$$

| logic | $\Gamma^+$ | $\Delta^+$ | $\Gamma^*$ | $\Delta^*$ |
|---|---|---|---|---|
| D | $\Gamma_{(\Box)}$ | $\Delta_{(\Diamond)}$ | $\Gamma_{(\Box)}$ | $\Delta_{(\Diamond)}$ |
| T | $\Gamma$ | $\Delta$ | $\Gamma_{(\Box)}$ | $\Delta_{(\Diamond)}$ |
| S4 | $\Gamma$ | $\Delta$ | $\Gamma_{\Box}$ | $\Delta_{\Diamond}$ |

with $\Gamma_{\Box} := \{\Box G \,|\, \Box G \in \Gamma\}$, $\Delta_{\Diamond} := \{\Diamond G \,|\, \Diamond G \in \Delta\}$, $\Gamma_{(\Box)} := \{G \,|\, \Box G \in \Gamma\}$, and $\Delta_{(\Diamond)} := \{G \,|\, \Diamond G \in \Delta\}$. To avoid deleting formulae in the sets $\Gamma^*$, $\Delta^*$, $\Gamma^+$ and $\Delta^+$ that are required for a proof, the (bottom-up) application of the modal rules need to be controlled. Again, a prefix is used to name sequences of accessible worlds and assigned to each subformula $G$ of a given formula $F$.

**Definition 3 (Modal prefix).** *A* prefix *(denoted by $p,q$) is a string (sequence of characters) over an alphabet $\mathcal{V} \cup \Pi$, in which $\mathcal{V}$ is a set of* prefix variables $(V_1, \ldots)$ *and $\Pi$ is a set of* prefix constants $(a_1, \ldots)$. *For every $\Box/\Diamond$ preceding a subformula $G$ of a formula $F$, an element of $\mathcal{V}$ or $\Pi$ is added to the prefix $p$ of $G$ depending on the "polarity" of $\Box/\Diamond$ (see [23] or Section 4 for details).*

Semantically, a prefix denotes a sequence of worlds in a model. Proof-theoretically, prefix variables and constants represent applications of the rules $\Box$-*left*/$\Diamond$-*right* and $\Box$-*right*/$\Diamond$-*left*, respectively. A prefix of a formula $F$ captures the modal context of $F$ and specifies the sequence of modal rules that have to be applied (bottom-up) in order to obtain $F$ in the sequent. In order to preserve the atomic formulae that form an axiom in the modal sequent calculus, their prefixes need to unify under a modal substitution $\sigma_M$.

**Definition 4 (Modal substitution).** *A* modal substitution $\sigma_M : \mathcal{V} \to (\mathcal{V} \cup \Pi)^*$ *maps elements of $\mathcal{V}$ to strings over $\mathcal{V} \cup \Pi$. In $\sigma_M(p)$ prefix variables are replaced according to $\sigma_M$. A connection $\{L_1 : p_1, L_2 : p_2\}$ is* $\sigma$-complementary *for a substitution $\sigma = (\sigma_Q, \sigma_M)$ iff $\sigma_Q(L_1) = \sigma_Q(\overline{L_2})$ and $\sigma_M(p_1) = \sigma_M(p_2)$. $\sigma$ is* admissible *iff the accessibility condition and the domain condition hold. For S5 only the last prefix character is considered.*

[1] A relation $R \subseteq W \times W$ is *serial* iff for all $w_1 \in W$ there is some $w_2 \in W$ with $(w_1, w_2) \in R$.

The accessibility and the domain condition ensure that the modal substitution respects the accessibility relation and domain variant of the considered modal logic.

**Definition 5 (Accessibility condition; domain condition).** *For the modal logics D and T the* accessibility condition $|\sigma_M(V)|{=}1$ *or* $|\sigma_M(V)|{\leq}1$*, respectively, has to hold for all* $V \in \mathcal{V}$*. The* domain condition *ensures that all "eigenvariables"* $\bar{x}$ *in a term t assigned to a variable x exist in the same world as x. In case of the varying domain variants, objects may only exists in the world in which they are introduced, hence,* $\sigma_M(\mathrm{pre}(\bar{x})) = \sigma_M(\mathrm{pre}(x))$ *has to hold.*[2] *For the cumulative domain variants,* $\bar{x}$ *has to be introduced before x, hence,* $\sigma_M(\mathrm{pre}(\bar{x})) \preceq \sigma_M(\mathrm{pre}(x))$.[3] *For the constant domain variants there is no restriction on the substitutions as every object exists in every world. Furthermore, the reduction ordering induced by* $\sigma$ *has to be irreflexive (see [23] for details).*

### 2.3 Matrix Characterizations

For the matrix characterization of validity in non-classical logics, the notion of matrices is generalized to arbitrary first-order formulae as already done for classical logic [13].

**Definition 6 (Matrix).** *A (non-clausal) matrix* $M(F)$*, representing a formula F, is a set of clauses, in which a* clause *is a set of literals and matrices (see [13] for details).*

Whereas the definition of paths needs to be generalized to non-clausal matrices[4], all other concepts used for clausal matrices, e.g. the definitions of connections and term substitutions remain unchanged.

**Definition 7 (Path).** *A* path *through a matrix M (or a clause C) is inductively defined as follows. The (only) path through a literal L is* $\{L\}$*. If* $p_1, \ldots, p_n$ *are paths through the clauses* $C_1, \ldots, C_n$*, respectively, then* $p_1 \cup \ldots \cup p_n$ *is a path through the matrix* $M = \{C_1, \ldots, C_n\}$*. If* $p_1, \ldots, p_n$ *are paths through the matrices/literals* $M_1, \ldots, M_n$*, respectively, then* $p_1, \ldots, p_n$ *are also paths through the clause* $C = \{M_1, \ldots, M_n\}$*.*

The notion of *multiplicity* is used to encode the number of clause copies used in a proof. It is a function $\mu : M_C \to \mathbb{N}$ from the set of clauses $M_C$ in $M$ that assigns a natural number to each clause in $M$ specifying how many copies of this clause are considered in a proof. In the *copy of a clause C* every (term and prefix) variable in $C$ is replaced by a unique new variable. $M^\mu$ denotes the matrix that includes these clause copies. Clause copies correspond to applications of the contraction rule in the sequent calculus.

**Theorem 1 (Matrix characterization for non-classical logic [23]).** *A formula F is* valid *in intuitionistic / modal logic iff there is (1) a multiplicity* $\mu$*, (2) an admissible substitution* $\sigma = (\sigma_Q, \sigma_I) / \sigma = (\sigma_Q, \sigma_M)$*, and (3) a set of* $\sigma$*-complementary connections, such that every path through* $M^\mu(F)$ *contains a connection from this set.*

Any proof method that is based on the matrix characterization and operates in a connection-oriented way is called a *connection method* [3, 17]. The specific calculus of a connection method is called a *connection calculus*.

[2] $pre(x)$ for a variable $x$ is the prefix $pre(QxG)$ of the corresponding subformula $QxG$, $Q \in \{\forall, \exists\}$.

[3] $u \preceq w$ holds iff $u$ is an initial substring of $w$ or $u = w$. This condition, as well as the fact that there is no accessibility condition for S5, are slightly corrected conditions of [23].

[4] The original characterization [23] uses a "tableau-like" definition and not non-clausal matrices.

# 3 Intuitionistic Logic

The connection calculus for intuitionistic logic to be introduced now is based on the matrix characterization of logical validity presented in Section 2.3. It uses a *connection-driven* search strategy in order to calculate an appropriate set of connections. The prefixed non-clausal matrix is the main concept used in the intuitionistic connection calculus. Furthermore, the use of prefixes requires an additional prefix unification algorithm.

## 3.1 Prefixed Non-clausal Matrices

An intuitionistic non-clausal matrix is a set of prefixed clauses, which consist of prefixed literals and prefixed (sub)matrices. The *polarity* 0 or 1 is used to represent negation in a matrix, i.e. literals of the form $A$ and $\neg A$ are represented by $A^0$ and $A^1$, respectively,

The *irreflexivity test* of the *reduction ordering* is realized by the *occurs check* during the term and prefix unifications. To this end, the *skolemization* technique, originally used to eliminate *eigenvariables* in classical logic, is extended and also used for prefix constants, a technique that is already used in the intuitionistic clausal calculus [10].

In the following, $\varepsilon$ denotes the empty string, $u \circ w$ (shortly $uw$) denotes the concatenation of the strings $u$ and $w$, and $G[x\backslash t]$ denotes the formula $G$ in which all free occurrences of the variable $x$ are replaced by the term $t$.

**Definition 8 (Intuitionistic non-clausal matrix).** *Let $F$ be a formula, pol be a polarity, and $p$ be a prefix. The* intuitionistic (non-clausal) matrix $M(F^{pol}:p)$ *of a* prefixed formula $F^{pol}:p$ *is a set of prefixed clauses, in which a prefixed clause is a set of prefixed literals and prefixed (non-clausal) matrices, and defined inductively according to Table 1. $x^*$ is a new term variable, $t^*$ is the Skolem term $f^*(x_1,\ldots,x_n)$ in which $f^*$ is a new function symbol and $x_1,\ldots,x_n$ are all free term and prefix variables in $(\forall xG)^0:p$ or $(\exists xG)^1:p$. $V^*$ is a new prefix variable, $a^*$ is a prefix constant of the form $f^*(x_1,\ldots,x_n)$ in which $f^*$ is a new function symbol and $x_1,\ldots,x_n$ are all free term and prefix variables in $A^0:p$, $(\neg G)^0:p$, $(G\Rightarrow H)^0:p$, or $(\forall xG)^0:p$. The* intuitionistic (non-clausal) matrix $M(F)$ *of $F$ is the intuitionistic matrix $M(F^0:\varepsilon)$.*

In the *graphical representation* of a non-clausal matrix, its clauses are arranged horizontally, while the literals and (sub-)matrices of each clause are arranged vertically.

**Table 1.** The definition of the prefixed non-clausal matrix for intuitionistic logic

| type | $F^{pol}:p$ | $M(F^{pol}:p)$ |
|---|---|---|
| atomic | $A^0:p$ | $\{\{A^0:pa^*\}\}$ |
| $\alpha$ | $(G\wedge H)^1:p$ | $\{\{M(G^1:p)\}\},\{\{M(H^1:p)\}\}$ |
| | $(G\vee H)^0:p$ | $\{\{M(G^0:p)\}\},\{\{M(H^0:p)\}\}$ |
| | $(G\Rightarrow H)^0:p$ | $\{\{M(G^1:pa^*)\}\},\{\{M(H^0:pa^*)\}\}$ |
| $\beta$ | $(G\wedge H)^0:p$ | $\{\{M(G^0:p),M(H^0:p)\}\}$ |
| | $(G\vee H)^1:p$ | $\{\{M(G^1:p),M(H^1:p)\}\}$ |
| | $(G\Rightarrow H)^1:p$ | $\{\{M(G^0:pV^*),M(H^1:pV^*)\}\}$ |

| type | $F^{pol}:p$ | $M(F^{pol}:p)$ |
|---|---|---|
| atomic | $A^1:p$ | $\{\{A^1:pV^*\}\}$ |
| $\alpha$ | $(\neg G)^0:p$ | $M(G^1:pa^*)$ |
| | $(\neg G)^1:p$ | $M(G^0:pV^*)$ |
| $\gamma$ | $(\forall xG)^1:p$ | $M(G[x\backslash x^*]^1:pV^*)$ |
| | $(\exists xG)^0:p$ | $M(G[x\backslash x^*]^0:p)$ |
| $\delta$ | $(\forall xG)^0:p$ | $M(G[x\backslash t^*]^0:pa^*)$ |
| | $(\exists xG)^1:p$ | $M(G[x\backslash t^*]^1:p)$ |

For example, the formula $(P(a) \wedge \forall x(P(x) \Rightarrow P(f(x))) \Rightarrow P(f(f(a)))) \wedge (Q \Rightarrow Q)$ has the simplified (redundant brackets are omitted) intuitionistic non-clausal matrix

$$\{\{\ \{\{P(a)^1 : a_1 V_1\}, \{P(x)^0 : a_1 V_2 V_3 a_2(V_2, x, V_3), P(f(x))^1 : a_1 V_2 V_3 V_4\},$$
$$\{P(f(f(a)))^0 : a_1 a_3\}\},\ \{\{Q^1 : a_4 V_5\}, \{Q^0 : a_4 a_5\}\}\ \}\}$$

and the graphical representation (where $a_2(V_2, x, V_3)$ is a (skolemized) prefix constant):

$$\left[\left[\left[\left[P(a)^1 : a_1 V_1\right]\ \left[\begin{array}{c} P(x)^0 : a_1 V_2 V_3 a_2(V_2, x, V_3) \\ P(f(x))^1 : a_1 V_2 V_3 V_4 \end{array}\right]\ \left[P(f(f(a)))^0 : a_1 a_3\right]\right]\right]\right.$$
$$\left.\left[\ \left[Q^1 : a_4 V_5\right]\ \left[Q^0 : a_4 a_5\right]\ \right]\right] .$$

### 3.2 Prefix Unification

The intuitionistic substitution $\sigma_I$ is calculated by a *prefix unification algorithm* [10]. For a given set of prefix equations $\bar{E} = \{p_1 = q_1, \ldots, p_n = q_n\}$, an appropriate substitution $\sigma_I$ is a unifier such that $\sigma_I(p_i) = \sigma_I(q_i)$ for all $1 \le i \le n$. A set of unifiers $\Sigma$ is a set of *most general unifiers* (*mgu*) for $\bar{E}$ if and only if every unifier $\tau$ is an instance of some $\sigma \in \Sigma$ (completeness) and no unifier $\sigma \in \Sigma$ is an instance of another unifier $\tau \in \Sigma$ (minimality). General algorithms for string unification[5] exist, but the number of most general unifiers might not be finite. The following unification algorithm is more efficient, as it takes the *prefix property* into account: for prefixes $p_i = u_1 X w_1$ and $p_j = u_2 X w_2$ (also in case $i = j$) with $X \in \Phi \cup \Psi$ the property $u_1 = u_2$ holds. This reflects the fact that prefixes correspond to sequences of connectives and quantifiers within the same formula.

**Definition 9 (Intuitionistic prefix unification).** *The unification for the prefix equation $\{p = q\}$ is carried out by applying the* rewriting rules *R1 to R10 in Figure 1. It is $V, \bar{V}, V' \in \Phi$ with $V \neq \bar{V}$, $V'$ is a new prefix variable, $a, b \in \Psi$, $X \in \Phi \cup \Psi$, and $u, w, z \in (\Phi \cup \Psi)^*$. For rule 10 the restriction $(*)$ $u = \varepsilon$ or $w \neq \varepsilon$ or $X \in \Psi$ applies. $\sigma_I(V) = u$ is written $\{V \backslash u\}$. The unification starts with the tuple $(\{p = \varepsilon | q\}, \{\})$. The application of a rewriting rule $E \to E', \tau$ replaces the tuple $(E, \sigma_I)$ by the tuple $(E', \tau(\sigma_I))$. $E$ and $E'$ are prefix equations, $\sigma_I$ and $\tau$ are (intuitionistic) substitutions. The unification terminates when the tuple $(\{\}, \sigma_I)$ is derived. In this case, $\sigma_I$ represents a* most general unifier. *Rules can be applied non-deterministically and lead to a set of mgu [10].*

| | | | |
|---|---|---|---|
| R1. | $\{\varepsilon = \varepsilon \mid \varepsilon\} \to \{\}, \{\}$ | R6. | $\{Vu = \varepsilon \mid aw\} \to \{u = \varepsilon \mid aw\}, \{V \backslash \varepsilon\}$ |
| R2. | $\{\varepsilon = \varepsilon \mid Xu\} \to \{Xu = \mid \varepsilon\}, \{\}$ | R7. | $\{Vu = z \mid abw\} \to \{u = \varepsilon \mid bw\}, \{V \backslash za\}$ |
| R3. | $\{Xu = \varepsilon \mid Xw\} \to \{u = \mid w\}, \{\}$ | R8. | $\{Vau = \varepsilon \mid \bar{V}w\} \to \{\bar{V}w = V \mid au\}, \{\}$ |
| R4. | $\{au = \varepsilon \mid Vw\} \to \{Vw = \mid au\}, \{\}$ | R9. | $\{Vau = Xz \mid \bar{V}w\} \to \{\bar{V}w = V' \mid au\}, \{V \backslash XzV'\}$ |
| R5. | $\{Vu = z \mid \varepsilon\} \to \{u = \mid \varepsilon\}, \{V \backslash z\}$ | R10. | $\{Vu = z \mid Xw\} \to \{Vu = zX \mid w\}, \{\}$ $(*)$ |

**Fig. 1.** The prefix unification for intuitionistic logic and modal logic S4

[5] This is also called the *monoid* problem; it is the equation theory in which there is a neutral element $\varepsilon$ and the associativity of the string concatenation operator $\circ$ holds.

In the worst-case, the number of mgu grows exponentially with respect to the length of the prefixes $p$ and $q$. To solve a *set* of prefix equations $\bar{E} = \{p_1 = p_1, \ldots, q_n = t_q\}$, the equations in $\bar{E}$ are solved one after the other and each calculated unifier is applied to the remaining prefix equations in $\bar{E}$.

For example, for the prefix equation $\{a_1V_2V_3 = a_1a_3\}$ there are the two derivations:

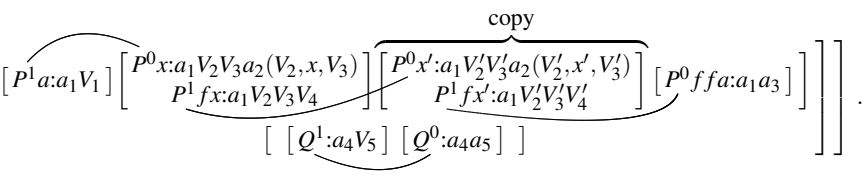
$\{a_1V_2V_3 = \varepsilon | a_1a_3\}, \{\} \xrightarrow{\text{R3}} \{V_2V_3 = \varepsilon | a_3\}, \{\} \xrightarrow{\text{R6}} \{V_3 = \varepsilon | a_3\}, \{V_2 \backslash \varepsilon\} \xrightarrow{\text{R10}} \{V_3 = a_3 | \varepsilon\}, \{V_2 \backslash \varepsilon\}$
$\xrightarrow{\text{R5}} \{\varepsilon = \varepsilon | \varepsilon\}, \{V_2 \backslash \varepsilon, V_3 \backslash a_3\}$ and $\{a_1V_2V_3 = \varepsilon | a_1a_3\}, \{\} \xrightarrow{\text{R3}} \{V_2V_3 = \varepsilon | a_3\}, \{\}$
$\xrightarrow{\text{R10}} \{V_2V_3 = a_3 | \varepsilon\}, \{\} \xrightarrow{\text{R5}} \{V_3 = \varepsilon | \varepsilon\}, \{V_2 \backslash a_3\} \xrightarrow{\text{R5}} \{\varepsilon = \varepsilon | \varepsilon\}, \{V_2 \backslash a_3, V_3 \backslash \varepsilon\},$

which yield the most general unifiers $\sigma_J^1 = \{V_2 \backslash \varepsilon, V_3 \backslash a_3\}$ and $\sigma_J^2 = \{V_2 \backslash a_3, V_3 \backslash \varepsilon\}$.

### 3.3 An Intuitionistic Non-clausal Connection Calculus

The non-clausal connection calculus for intuitionistic logic is an extension of the non-clausal connection calculus for classical logic [13], in which a prefix is added to each literal and an additional prefix unification is used to identify $\sigma$-complementary connections. According to the matrix characterization in Section 2.3, a formula $F$ is valid, iff all paths through its matrix $M^{\mu}(F)$ (with added clause copies) contain a $\sigma$-complementary connection. The calculus uses a *connection-driven* search strategy in order check this property. In each (reduction and extension) step of a derivation in the calculus, a $\sigma$-complementary connection is identified and only paths that do not contain this connection are investigated afterwards. If every path contains a $\sigma$-complementary connection, the proof search succeeds and the given formula is valid. A *non-clausal connection proof* can be illustrated within the graphical matrix representation.

For example, the proof of the matrix from Section 3.1 consists of four connections, marked by an arc in the graphical matrix representation, that are $\sigma$-complementary with $\sigma_Q = \{x \backslash a, x' \backslash fa\}$, $\sigma_J = \{V_1 \backslash a_2(\varepsilon, a, \varepsilon), V_2 \backslash \varepsilon, V_3 \backslash \varepsilon, V_4 \backslash a_2(\varepsilon, fa, \varepsilon), V_2' \backslash \varepsilon, V_3' \backslash \varepsilon, V_4' \backslash a_3\}$:




But, e.g., for the formula $P \vee \neg P$ there is no intuitionistic connection proof of its matrix $\{\{P^0{:}a_1\}, \{P^1{:}a_2A_1\}\}$, as the two prefixes of the only connection cannot be unified.

A few additional concepts are required as follows in order to specify which clauses can be used within the generalized non-clausal extension rule. The term $\alpha$-related is used to express that a clause occurs besides a literal in a matrix. The definitions of free variables and clause copies have to be generalized to cover non-clausal matrices.

**Definition 10 ($\alpha$-related; parent clause; clause copy).** *A clause $C$ is $\alpha$-related to a literal $L$ iff it occurs besides $L$ in the graphical matrix representation; more precisely, $C$ is* $\alpha$-related *to a literal $L$ iff $\{C', C''\} \subseteq M'$ for some matrix $M'$, such that $C'$ contains $L$ and $C''$ contains $C$ (or $C{=}C''$). $C'$ is a* parent clause *of $C$ iff $M' \in C'$ and $C \in M'$ for some $M'$. In the* copy of a clause *$C$ all* free variables *in $C$ are replaced by fresh variables. $M[C_1 \backslash C_2]$ denotes the matrix $M$, in which the clause $C_1$ is replaced by the clause $C_2$.*

**Definition 11 (Extension clause; $\beta$-clause).** *$C$ is an* extension clause (e-clause) of the matrix $M$ with respect to *a set of literals $Path$ iff either (a) $C$ contains a literal of $Path$, or (b) $C$ is $\alpha$-related to all literals of $Path$ occurring in $M$ and if $C$ has a parent clause, it contains a literal of $Path$. In the* $\beta$-clause of $C_2$ with respect to $L_2$, *denoted by* $\beta$-clause$_{L_2}(C_2)$, *$L_2$ and all clauses that are $\alpha$-related to $L_2$ are deleted from $C_2$.*

In the example, the literal $Q^1{:}a_4V_5$ is only $\alpha$-related to the literal $Q^0{:}a_4a_5$. The parent clause of $\{Q^1{:}a_4V_5\}$ is the clause $C'$ in the whole matrix $\{C'\}$ of the example. Furthermore, the clause $\{Q^0{:}a_4a_5\}$ is an extension clause with respect to $\{Q^1{:}a_4V_5\}$.

The non-clausal connection calculus for intuitionistic logic adds prefixes and an intuitionistic substitution $\sigma_J$ to the non-clausal connection calculus for classical logic.

**Definition 12 (Intuitionistic non-clausal connection calculus).** *The axiom and the rules of the* intuitionistic (non-clausal) connection calculus *are given in Figure 2. It works on tuples "$C, M, Path$", where $M$ is a prefixed non-clausal matrix, $C$ is a prefixed (subgoal) clause or $\varepsilon$ and (the active) $Path$ is a set of prefixed literals or $\varepsilon$. $\sigma = (\sigma_Q, \sigma_J)$ is a combined term and intuitionistic substitution. An* intuitionistic (non-clausal) connection proof *of a prefixed matrix $M$ is an intuitionistic connection proof of $\varepsilon, M, \varepsilon$.*

| | | |
|---|---|---|
| *Axiom (A)* | $\dfrac{}{\{\}, M, Path}$ | *Start (S)* $\dfrac{C_2, M, \{\}}{\varepsilon, M, \varepsilon}$ and $C_2$ is copy of $C_1 \in M$ |
| *Reduction (R)* | $\dfrac{C, M, Path \cup \{L_2 : p_2\}}{C \cup \{L_1 : p_1\}, M, Path \cup \{L_2 : p_2\}}$ | and $\{L_1{:}p_1, L_2{:}p_2\}$ is $\sigma$-complementary |
| *Extension (E)* | $\dfrac{C_3, M[C_1 \setminus C_2], Path \cup \{L_1 : p_1\} \quad C, M, Path}{C \cup \{L_1 : p_1\}, M, Path}$ | and $C_3 := \beta\text{-}clause_{L_2}(C_2)$, $C_2$ is copy of $C_1$, $C_1$ is e-clause of $M$ wrt. $Path \cup \{L_1 : p_1\}$, $C_2$ contains $L_2 : p_2$, $\{L_1{:}p_1, L_2{:}p_2\}$ is $\sigma$-complementary |
| *Decomposition (D)* | $\dfrac{C \cup C_1, M, Path}{C \cup \{M_1\}, M, Path}$ | and $C_1 \in M_1$ |

**Fig. 2.** The non-clausal connection calculus for intuitionistic and modal logic

*Proof search* in the non-clausal connection calculus is carried out by applying the rules of the calculus in an *analytic* way (i.e. bottom-up) starting with $\varepsilon, M, \varepsilon$, in which $M$ is the matrix of the given formula. At first, a start clause is selected. Afterwards, connections are successively identified by applying reduction and extension rules. This process is guided by the *active path*, a subset of a path through $M$. During the proof search, backtracking might be required, i.e. alternative rules or rule instances have to be considered if the chosen rule or rule instance does not lead to a proof. This might happen when choosing the clause $C_1$ in the start and extension rules, the literal $L_2$ in the reduction and extension rules or the clause $C_1$ in the decomposition rule. The multiplicity $\mu$ is increased *dynamically* whenever an extension rule is applied.

The substitutions $\sigma_Q$ and $\sigma_J$ are *rigid*, i.e. applied to the whole derivation, and calculated whenever a reduction or extension rule is applied. The term substitution $\sigma_Q$ is calculated by one of the well-known algorithms for term unification. The intuitionistic substitution is calculated by a prefix unification algorithm (Section 3.2).

**Theorem 2 (Correctness and completeness).** *A first-order formula $F$ is valid in intuitionistic logic iff there is a proof in the non-clausal connection calculus for $M(F)$.*

The proof is based on the matrix characterization for modal logic (Theorem 1), the correctness and completeness of the non-clausal connection calculus for classical logic [13] and the correctness of the prefix unification [7]. It is crucial to use a "general" non-clausal approach [13] without optimizations that work only for classical logic.

## 4 Modal Logic

The non-clausal connection calculus for modal logic is similar to the one for intuitionistic logic; only the prefixed non-clausal matrix and the prefix unification is adapted.

### 4.1 Prefixed Non-clausal Matrices

In the modal non-clausal matrix, prefixes are determined by the modal operators. All other concepts, including the extended skolemization technique, are used in the same way as for the non-clausal matrix for intuitionistic logic (see Section 3.1). See [14] and the references therein for a motivation and examples for the usage of modal prefixes.

**Definition 13 (Modal non-clausal matrix).** *Let $F$ be a formula, pol be a polarity, and $p$ be a prefix. The* modal (non-clausal) matrix $M(F^{pol}{:}p)$ *of a* prefixed formula $F^{pol}{:}p$ *is defined inductively according to Table 2. $x^*$ is a new term variable, $t^*$ is the Skolem term $f^*(x_1,\ldots,x_n)$ in which $f^*$ is a new function symbol and $x_1,\ldots,x_n$ are all free term and prefix variables in $(\forall xG)^0 : p$ or $(\exists xG)^1 : p$. $V^*$ is a new prefix variable, $a^*$ is a prefix constant of the form $f^*(x_1,\ldots,x_n)$ in which $f^*$ is a new function symbol and $x_1,\ldots,x_n$ are all free term and prefix variables in $(\Box G)^0 : p$ or $(\Diamond G)^1 : p$. The* modal (non-clausal) matrix $M(F)$ *of $F$ is the modal matrix $M(F^0 : \varepsilon)$.*

**Table 2.** The definition of the prefixed non-clausal matrix for modal logic

| type | $F^{pol} : p$ | $M(F^{pol} : p)$ |
|---|---|---|
| atomic | $A^0 : p$ | $\{\{A^0 : p\}\}$ |
| $\alpha$ | $(G \wedge H)^1 : p$ | $\{\{M(G^1 : p)\}\}, \{\{M(H^1 : p)\}\}$ |
| | $(G \vee H)^0 : p$ | $\{\{M(G^0 : p)\}\}, \{\{M(H^0 : p)\}\}$ |
| | $(G \Rightarrow H)^0 : p$ | $\{\{M(G^1 : p)\}\}, \{\{M(H^0 : p)\}\}$ |
| $\beta$ | $(G \wedge H)^0 : p$ | $\{\{M(G^0 : p), M(H^0 : p)\}\}$ |
| | $(G \vee H)^1 : p$ | $\{\{M(G^1 : p), M(H^1 : p)\}\}$ |
| | $(G \Rightarrow H)^1 : p$ | $\{\{M(G^0 : p), M(H^1 : p)\}\}$ |
| $\nu$ | $(\Box G)^1 : p$ | $M(G^1 : pV^*)$ |
| | $(\Diamond G)^0 : p$ | $M(G^0 : pV^*)$ |

| type | $F^{pol} : p$ | $M(F^{pol} : p)$ |
|---|---|---|
| atomic | $A^1 : p$ | $\{\{A^1 : p\}\}$ |
| $\alpha$ | $(\neg G)^0 : p$ | $M(G^1 : p)$ |
| | $(\neg G)^1 : p$ | $M(G^0 : p)$ |
| $\gamma$ | $(\forall xG)^1 : p$ | $M(G[x\backslash x^*]^1 : p)$ |
| | $(\exists xG)^0 : p$ | $M(G[x\backslash x^*]^0 : p)$ |
| $\delta$ | $(\forall xG)^0 : p$ | $M(G[x\backslash t^*]^0 : p)$ |
| | $(\exists xG)^1 : p$ | $M(G[x\backslash t^*]^1 : p)$ |
| $\pi$ | $(\Box G)^0 : p$ | $M(G^0 : pa^*)$ |
| | $(\Diamond G)^1 : p$ | $M(G^1 : pa^*)$ |

### 4.2 Prefix Unification

Again, a prefix unification algorithm is used to calculate the modal substitution $\sigma_M$. Depending on the modal logic, the accessibility condition (see Section 2.2) has to be respected when calculating this substitution, i.e. for all $V \in \mathcal{V}$: $|\sigma_M(V)| = 1$ for the modal logic D and $|\sigma_M(V)| \leq 1$ for the modal logic T; there is no restriction for the modal logics S4 and S5. The prefix unification for D is a simple pattern matching, i.e. the standard term unification can be used. For S4 the (general) prefix unification for intuitionistic logic can be used (see Section 3.2). For S5 only the last character of each prefix (or $\varepsilon$ if the prefix is $\varepsilon$) has to be unified. By structural induction it can be shown that the following procedure computes a set of mgu for the modal logic T (that contains fewer mgu than the unification procedure presented in [7]).

**Definition 14 (Modal T prefix unification).** *The unification for the prefix equation $\{p = q\}$ is carried out by applying the rewriting rules in Figure 3. It is $V, \bar{V} \in \mathcal{V}$ with $V \neq \bar{V}$, $a \in \Pi$, $X \in \mathcal{V} \cup \Pi$, $u, w \in (\mathcal{V} \cup \Pi)^*$. The rules are applied in the same way as those for intuitionistic logic (see Section 3.2), but the tuple has the form $(E, \sigma_M)$ and terminates with the tuple $(\{\}, \sigma_M)$, in which case $\sigma_M$ represents a most general unifier.*

| | | | |
|---|---|---|---|
| R1. | $\{\varepsilon = \varepsilon \mid \varepsilon\} \rightarrow \{\}\{\}$ | R6. | $\{au = \varepsilon \mid Vw\} \rightarrow \{Vw = a \mid u\}, \{\}$ |
| R2. | $\{\varepsilon = \varepsilon \mid Xw\} \rightarrow \{Xw = \varepsilon \mid \varepsilon\}, \{\}$ | R7. | $\{Vu = \varepsilon \mid \bar{V}w\} \rightarrow \{w = V \mid u\}, \{\bar{V} \backslash \varepsilon\}$ |
| R3. | $\{Vu = \varepsilon \mid \varepsilon\} \rightarrow \{u = \varepsilon \mid \varepsilon\}, \{V \backslash \varepsilon\}$ | R8. | $\{Vu = X \mid w\} \rightarrow \{u = X \mid w\}, \{V \backslash \varepsilon\}$ |
| R4. | $\{Xu = \varepsilon \mid Xw\} \rightarrow \{u = \varepsilon \mid w\}, \{\}$ | R9. | $\{Vu = X \mid w\} \rightarrow \{u = \varepsilon \mid w\}, \{V \backslash X\}$ |
| R5. | $\{\bar{V}u = \varepsilon \mid Xw\} \rightarrow \{\bar{V}u = X \mid w\}, \{\}$ | R10. | $\{au = V \mid w\} \rightarrow \{u = \varepsilon \mid w\}, \{V \backslash a\}$ |

**Fig. 3.** The prefix unification for the modal logic T

### 4.3 A Modal Non-clausal Connection Calculus

The non-clausal connection calculus for modal logic uses the same concepts as the one for intuitionistic logic. The intuitionistic substitution $\sigma_I$ is replaced by the one for modal logic $\sigma_M$, and the definition of $\sigma$-complementary is adapted (see Section 2.2). Proof search is carried out in the same way as for intuitionistic logic (see Section 3.3).

**Definition 15 (Modal non-clausal connection calculus).** *The axiom and the rules of the* modal (non-clausal) connection calculus *are given in Figure 2. It works on tuples "$C, M, Path$", where $M$ is a prefixed non-clausal matrix, $C$ is a prefixed (subgoal) clause or $\varepsilon$ and (the active) Path is a set of prefixed literals or $\varepsilon$. $\sigma = (\sigma_Q, \sigma_M)$ is a combined (rigid) term and modal substitution. A* modal (non-clausal) connection proof *of a prefixed matrix $M$ is a modal connection proof of $\varepsilon, M, \varepsilon$.*

**Theorem 3 (Correctness and completeness).** *A modal first-order formula $F$ is valid in modal logic iff there is a proof in the non-clausal connection calculus for $M(F)$.*

The proof is based on the matrix characterization for modal logic (Theorem 1), the correctness and completeness of the non-clausal connection calculus for classical logic [13] and the correctness of the prefix unifications. Again, the "general" non-clausal calculus [13] without any optimizations for classical logic has to be used.

## 5 Optimizations

*Optimization techniques*, such as positive start clauses, regularity, lemmata and restricted backtracking, can be employed in a similar way as in the non-clausal connection calculus for classical logic [13] if the prefixes are additionally taken into account.

**Positive Start Clause.** Like for the clausal connection calculus, the start clause of the non-clausal connection calculus can be restricted to positive clauses. A clause is positive iff all of its elements (matrices and literals) are positive; a matrix is positive iff it contains at least one positive clause; a literal is positive iff its polarity is 0. If there is no positive clause in a matrix $M$ of $F$, then there exists a path through $M$ that contains no positive literal, hence, according to the matrix characterization $F$ cannot be valid. The *positive clause* $C_1$ *of a clause* $C$, consists only of the clauses of $C$ that are positive.

**Regularity.** Regularity is an effective technique for pruning the search space in clausal connection calculi [8]. The *regularity condition* ensures that no literal occurs more than once in the active path. It is integrated into the non-classical non-clausal connection calculus in Figure 2 by adding the following restriction to the reduction and the extension rule: $\forall\ L'{:}\,p' \in C \cup \{L_1{:}\,p_1\} :\ \sigma(L'{:}\,p') \notin \sigma(Path)$, in which the combined substitution $\sigma$ is applied to term/prefix variables. Additional backtracking can be avoided if the combined substitution $\sigma$ is not modified in order to satisfy the regularity condition.

**Lemmata.** The idea of *lemmata* (or *factorization*) is to reuse subproofs during the proof search [8]. To this end an additional set of lemmata (i.e. literals) and a lemma rule [12] is added to the non-clausal connection calculus. Again, a lemma literal $L{:}\,p$ has to unify under a *combined* substitution $\sigma$, in order to apply the lemma rule.

**Restricted Backtracking.** Proof search in the non-clausal connection calculus is *not* confluent. In order to achieve completeness, *backtracking* (see remarks in Section 3.3) is necessary. The idea of *restricted backtracking* is to cut off any alternative connections once a literal from the subgoal clause has been solved [12]. A literal $L$ is called *solved* if it is the literal $L_1$ of a reduction or extension rule application (see Figure 2) and in the case of the extension rule, there is also a proof for the left premise. Restricted backtracking is correct (as the search space is only pruned), but incomplete [12]. It can be applied in the intuitionistic or modal non-clausal connection calculus as well.

## 6 Implementation

The implementations of the intuitionistic and modal non-clausal connection calculi of Figure 2 follow the *lean methodology* [1], which is already used for the clausal connection provers leanCoP [16, 11], ileanCoP [11] and MleanCoP [14]. It uses very compact Prolog code to implement the basic calculus and adds a few essential optimization techniques in order to prune the search space. The resulting *na*tural *no*nclausal *co*nnection *p*rovers for *i*ntuitionistic logic nanoCoP-i and *m*odal logics nanoCoP-M are available at `http://www.leancop.de/nanocop-i/` and `http://www.leancop.de/nanocop-m/`.

**Modal and Intuitionistic Non-clausal Matrices.** In the first step the input formula $F$ is translated into its intuitionistic/modal non-clausal matrix $M{:=}M(F)$ according to Table 1 or 2; redundant brackets of the form "$\{\{\ldots\}\}$" are removed. Additionally, every (sub-)clause $(I,V,FV):C$ and (sub-)matrix $J:M$ are marked with unique *indices* $I$ and $J$; clauses $C$ are also marked with a set $V$ of (free) term and prefix variables and a set $FV$ of (free) term variables of the form $x{:}pre(x)$ that are newly introduced in $C$. Atomic formulae and term/prefix constants are represented by Prolog atoms, term/prefix variables by Prolog variables; literals with polarity 1 are marked with "$-$". Sets, e.g. clauses and matrices, are represented by Prolog lists (representing multisets); prefixes are represented by Prolog lists and marked with the polarity of the corresponding literal. For example, the non-clausal matrix from Section 3.1 is represented by the Prolog term

```
[(2^K)^[]^[]: [16^K:[(17^K)^[]^[]: [-(q): -([15^[]])], (19^K)^[]^[]:[q:[15^[]]]],
      5^K: [(6^K)^[]^[]: [-(p(a)): -([3^[]])],
            (8^K)^[W,X,V]^[X:[3^[],V]]: [p(X):[3^[],V,W], -(p(f(X))): -([3^[],V,W])],
            (14^K)^[]^[]: [p(f(f(a))):[3^[]]]] ] ]
```

in which the Prolog variable `K` is instantiated later on in order to enumerate clause copies (as an optimization for intuitionistic logic, prefix characters introduced by atomic formulae are *only* considered during the unification). In a second step, the matrix $M$ is written into Prolog's database. For every literal `Lit` in $M$ the fact `lit(Lit,ClaB,ClaC,Grnd)` is asserted into the database where `ClaC` $\in M$ is the (largest) clause in which `Lit` occurs and `ClaB` is the $\beta$-clause of `ClaC` with respect to `Lit`. `Grnd` is set either to `g` or `n` depending if the smallest clause in which `Lit` occurs is ground or not. No other modifications of the original formula (structure) are done during these two preprocessing steps.

**Non-classical Non-clausal Proof Search.** The nanoCoP-i/M source code is shown in Figure 4. The underlined text was added to the nanoCoP code for classical logic [15]: (1) prefixes are added to all literals, (2) the sets `PreS` and `VarS` are added, which contain prefix equations and free (prefixed) term variables, respectively, and (3) a prefix unification is added. First, nanoCoP-i/M performs a classical proof search, in which the prefixes of each connection are stored in `PreS`. If the search succeeds, the domain condition is checked and the prefixes in `PreS` are unified (line 4), using the predicates `domain_cond` and `prefix_unify` (which need 18 and between 7 to 22 lines of code).

The predicate `prove(Mat,PathLim,Set,Proof)` implements the start rule (lines 1–4) and iterative deepening on the size of the active path (lines 5–9). `Mat` is the matrix generated in the preprocessing step, `PathLim` is the size limit for `Path`, and `Proof` contains the returned (non-clausal) connection proof. `Set` $\subseteq\{$`cut`,`comp(`$I$`)`$\}$, for $I\in\mathbb{N}$, is used to control restricted backtracking [12]. The predicate `positiveC(Cla,Cla1)` returns the positive clause `Cla1` of `Cla` (needs 7 additional lines of code). The predicate `prove(Cla,Mat,Path,PathI,PathLim,Lem,PreS,VarS,Set,Proof)` implements the axiom (line 10), the decomposition rule (lines 11–16), the reduction rule (lines 17–20, 24–26, 37–38), and the extension rule (lines 17–20, 28–49) of the calculus in Figure 2. `Cla`, `Mat`, and `Path` represent the subgoal clause $C$, the prefixed matrix $M$ and the (active) *Path*. The *indexed path* `PathI` contains the indices of all clauses and matrices that contain literals of `Path`; it is used for calculating extension clauses. The substitution $\sigma$ is stored implicitly by Prolog. The predicate `prove_ec(ClaB,Cla1,Mat,ClaB1,Mat1)` calculates extension clauses (lines 39–49). Additional optimization techniques (see Section 5) are regularity (line 19), lemmata (line 21), and restricted backtracking (line 36).

```
        % start rule
(1)     prove(Mat,PathLim,Set,[(I^0)^V:Cla1|Proof]) :-
(2)         member((I^0)^V^VS:Cla,Mat), positiveC(Cla,Cla1),
(3)         prove(Cla1,Mat,[],[I^0],PathLim,[],PreS,VarS,Set,Proof),
(4)         append(VarS,VS,VarS1), domain_cond(VarS1), prefix_unify(PreS).

(5)     prove(Mat,PathLim,Set,Proof) :-
(6)         retract(pathlim) ->
(7)         ( member(comp(PathLim),Set) -> prove(Mat,1,[],Proof) ;
(8)           PathLim1 is PathLim+1, prove(Mat,PathLim1,Set,Proof) ) ;
(9)         member(comp(_),Set) -> prove(Mat,1,[],Proof).

        % axiom
(10)    prove([],_,_,_,_,_,[],[],_,[]).

        % decomposition rule
(11)    prove([J^K:Mat1|Cla],MI,Path,PI,PathLim,Lem,PreS,VarS,Set,Proof) :- !,
(12)        member(I^_^FV:Cla1,Mat1),
(13)        prove(Cla1,MI,Path,[I,J^K|PI],PathLim,Lem,PreS1,VarS1,Set,Proof1),
(14)        prove(Cla,MI,Path,PI,PathLim,Lem,PreS2,VarS2,Set,Proof2),
(15)        append(PreS2,PreS1,PreS), append(FV,VarS1,VarS3),
(16)        append(VarS2,VarS3,VarS), append(Proof1,Proof2,Proof).

        % reduction and extension rules
(17)    prove([Lit:Pre|Cla],MI,Path,PI,PathLim,Lem,PreS,VarS,Set,Proof) :-
(18)        Proof=[[I^V:[NegLit|ClaB1]|Proof1]|Proof2],
(19)        \+ (member(LitC,[Lit:Pre|Cla]), member(LitP,Path), LitC==LitP),
(20)        (-NegLit=Lit;-Lit=NegLit) ->
(21)           ( member(LitL,Lem), Lit:Pre==LitL, ClaB1=[], Proof1=[],
(22)             PreS3=[], VarS3=[]
(23)             ;
(24)             member(NegL:PreN,Path), unify_with_occurs_check(NegL,NegLit),
(25)             ClaB1=[], Proof1=[],
(26)             \+ \+ prefix_unify([Pre=PreN]), PreS3=[Pre=PreN], VarS3=[]
(27)             ;
(28)             lit(NegLit:PreN,ClaB,Cla1,Grnd1),
(29)             ( Grnd1=g -> true ; length(Path,K), K<PathLim -> true ;
(30)               \+ pathlim -> assert(pathlim), fail ),
(31)             \+ \+ prefix_unify([Pre=PreN]),
(32)             prove_ec(ClaB,Cla1,MI,PI,I^V^FV:ClaB1,MI1),
(33)             prove(ClaB1,MI1,[Lit:Pre|Path],[I|PI],PathLim,Lem,PreS1,VarS1,
(34)                   Set,Proof1), PreS3=[Pre=PreN|PreS1], append(VarS1,FV,VarS3)
(35)           ),
(36)           ( member(cut,Set) -> ! ; true ),
(37)           prove(Cla,MI,Path,PI,PathLim,[Lit:Pre|Lem],PreS2,VarS2,Set,Proof2),
(38)           append(PreS3,PreS2,PreS), append(VarS2,VarS3,VarS).

        % extension clause (e-clause)
(39)    prove_ec((I^K)^V:ClaB,IV:Cla,MI,PI,ClaB1,MI1) :-
(40)        append(MIA,[(I^K1)^V1:Cla1|MIB],MI), length(PI,K),
(41)        ( ClaB=[J^K:[ClaB2]|_], member(J^K1,PI),
(42)          unify_with_occurs_check(V,V1), Cla=[_:[Cla2|_]|_],
(43)          append(ClaD,[J^K1:MI2|ClaE],Cla1),
(44)          prove_ec(ClaB2,Cla2,MI2,PI,ClaB1,MI3),
(45)          append(ClaD,[J^K1:MI3|ClaE],Cla3),
(46)          append(MIA,[(I^K1)^V1:Cla3|MIB],MI1)
(47)          ;
(48)          (\+member(I^K1,PI);V\==V1) ->
(49)          ClaB1=(I^K)^V:ClaB, append(MIA,[IV:Cla|MIB],MI1) ).
```

**Fig. 4.** Source code of the nanoCoP-i and nanoCoP-M core provers

## 7 Evaluation

The following tests were conducted on a Xeon system with 4 GB of RAM running Linux and ECLiPSe Prolog 5.10. The CPU time limit was set to 100 seconds.

**ILTP Library.** Table 3 shows the number of solutions on all 2550 first-order problems of the ILTP library v1.1.2 [19] for the intuitionistic theorem provers JProver, ileanTAP, ft, ileanCoP, and nanoCoP-i. JProver [21] is based on a simple prefixed non-clausal connection calculus [7]; ileanTAP [9] uses a prefixed free-variable tableau calculus; ft [20] is a C implementation of an analytic tableau calculus; ileanCoP [10, 11] implements a prefixed *clausal* connection calculus. In order to make the results comparable, the *core prover* of ileanCoP was used with the standard ("[nodef]") and the definitional ("[def]") translation into clausal form. nanoCoP-i was tested without and with restricted backtracking, i.e. `Set=[]` and `Set=[cut,comp(6)]`, respectively. The "full" version of ileanCoP 1.2, which additionally uses a fixed strategy scheduling, proves 787 problems. Compared to ileanCoP "[nodef]", 39% of the proofs of nanoCoP-i "[]" are on average 36% shorter (in terms of number of connections); 60% have the same size. Compared to ileanCoP "[def]", 51% of the proofs are on average 38% shorter; 48% have the same size. There is a significant performance improvement of nanoCoP-i compared to ileanCoP, even though most of the tested problems have a "clausal-like" structure, which also explains that about half of the proofs have the same size.

**Table 3.** Results on the first-order problems of the ILTP library

| | JProver | ileanTAP | ft (C) | — ileanCoP 1.2 — | | — nanoCoP-i 1.0 — | |
|---|---|---|---|---|---|---|---|
| | 11-2005 | 1.17 | 1.23 | [nodef] | [def] | [] | [cut,comp(6)] |
| proved | 258 | 308 | 334 | 601 | 640 | **704** | 764 |
| refuted | 4 | 4 | 30 | 82 | 78 | 89 | 89 |

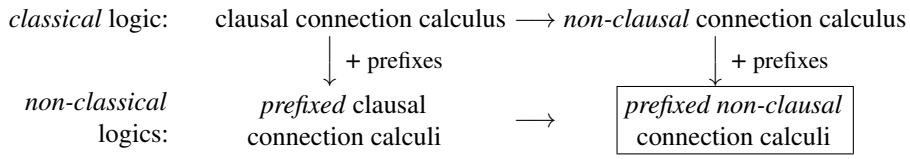
**QMLTP Library.** Table 4 shows the number of solutions on all 580 unimodal problems of the QMLTP library v1.1 [18] (for the varying, cumulative, and constant domain variants) for the modal theorem provers MleanTAP, MleanCoP, and nanoCoP-M. MleanTAP implements a prefixed tableau calculus; MleanCoP [14] uses a prefixed *clausal* connection calculus. Again, the *core prover* of MleanCoP was tested with the (better performing) definitional translation ("[def]") into clausal form. nanoCoP-M was tested without and with restricted backtracking, i.e. `Set=[]` and `Set=[cut,comp(6)]`, respectively; both versions refute the same number of (invalid) formulae. Compared to MleanCoP "[def]", between 33% and 48% of the proofs of nanoCoP-M "[]" are on average between 38% and 41% shorter (in terms of number of connections) depending on the specific modal logic; at most 3% of the proofs are larger (due to a different proof search order of MleanCoP and nanoCoP-M). There is a significant performance improvement; nanoCoP-M proves or refutes up to 10% more problems than MleanCoP. Again, most of the tested problems have a "clausal-like" structure, hence, for about half of the proved problems, the proofs of MleanCoP and nanoCoP-M have the same size.

**Table 4.** Results on the unimodal problems (varying/cumul./constant) of the QMLTP library

| | MleanTAP | MleanCoP 1.3 | | nanoCoP-M 1.0 | | |
|---|---|---|---|---|---|---|
| | 1.3 | [def] | [def] | [] | [] | [cut,comp(6)] |
| Logic | (proved) | (proved) | (refuted) | (proved) | (refuted) | (proved) |
| D | 100/120/135 | 152/170/187 | 246/226/209 | **158/177/194** | 266/246/230 | 167/187/204 |
| T | 138/162/175 | 188/212/229 | 148/126/112 | **211/231/248** | 153/133/119 | 222/244/263 |
| S4 | 169/205/220 | 236/282/296 | 121/95/82 | **261/306/320** | 124/98/85 | 271/321/336 |
| S5 | 219/272/272 | 313/372/372 | 90/41/41 | **329/392/392** | 92/44/44 | 343/414/414 |

## 8 Conclusion

This paper introduced non-clausal connection calculi for some popular non-classical logics. Combining the notion of prefixes with an efficient non-clausal calculus provides the foundation for an efficient proof search in these logics. The resulting *prefixed non-clausal* connection calculi can either be seen as *non-clausal* versions of prefixed *clausal* connection calculi for non-classical logics [10, 14], or as extensions of the non-clausal connection calculus for classical logic [13], to which prefixes have been added.




Using prefixed non-clausal matrices, the proof search works directly on the original structure of the input formula; no translation steps to any clausal or normal form are required. Hence, the presented calculi for intuitionistic and several modal logics combine the advantages of more *natural* non-clausal tableau or sequent calculi with the goal-oriented *efficiency* of a connection-based proof search.

An experimental evaluation of two compact implementations of the introduced calculi shows that the non-clausal approach does not only speed up the proof search, but that the resulting non-clausal proofs are also significantly shorter. nanoCoP-i also has a significantly higher performance than JProver [21], which uses a non-clausal connection calculus [7] that does not add clause copies dynamically during the proof search.

Future work includes the adaption of the non-clausal calculus to other non-classical logics, e.g. to those modal logics for which a (prefixed) matrix characterization exists [23] and to description logics [5], as well as the integration of additional proof search techniques, e.g. strategy scheduling and learning.

**Acknowledgements.** The author would like to thank Arild Waaler for his support through the Sirius Center at the University of Oslo funded by the Research Council of Norway. Furthermore, he would like to thank Wolfgang Bibel for his comments.

## References


1. Beckert, B., Posegga, J.: leanTAP: Lean tableau-based deduction. Journal of Automated Reasoning 15(3), 339–358 (1995)
2. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development Coq'Art: The Calculus of Inductive Constructions. EATCS Series, Springer, Heidelberg (2004)
3. Bibel, W.: Automated theorem proving. Artificial intelligence, F. Vieweg und Sohn, Wiesbaden, 2nd edn. (1987)
4. Constable, R.L., et al.: Implementing Mathematics with the NuPRL proof development system. Prentice–Hall, Englewood Cliffs, NJ (1986)
5. Freitas, F., Otten, J.: A connection calculus for the description logic ALC. In: Khoury, R., Drummond, C. (eds.) Canadian AI 2016. LNCS, vol. 9673, pp. 243–256. Springer (2016)
6. Gentzen, G.: Untersuchungen über das Logische Schließen. Mathematische Zeitschrift 39, 176–210, 405–431 (1935)
7. Kreitz, C., Otten, J.: Connection-based theorem proving in classical and non-classical logics. Journal of Universal Computer Science 5(3), 88–112 (1999)
8. Letz, R., Stenz, G.: Model elimination and connection tableau procedures. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, pp. 2015–2112. Elsevier Science, Amsterdam (2001)
9. Otten, J.: ileanTAP: An intuitionistic theorem prover. In: Galmiche, D. (ed.) TABLEAUX 1997. LNAI, vol. 1227, pp. 307–312. Springer, Heidelberg (1997)
10. Otten, J.: Clausal connection-based theorem proving in intuitionistic first-order logic. In: TABLEAUX 2005. LNAI, vol. 3702, pp. 245–261. Springer, Heidelberg (2005)
11. Otten, J.: leanCoP 2.0 and ileanCoP 1.2: High performance lean theorem proving in classical and intuitionistic logic. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNAI, vol. 5195, pp. 283–291. Springer, Heidelberg (2008)
12. Otten, J.: Restricting backtracking in connection calculi. AI Commun. 23(2–3), 159–182 (2010)
13. Otten, J.: A non-clausal connection calculus. In: Brünnler, K., Metcalfe, G. (eds.) TABLEAUX 2011. LNAI, vol. 6793, pp. 226–241. Springer, Heidelberg (2011)
14. Otten, J.: MleanCoP: A connection prover for first-order modal logic. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNAI, vol. 8562, pp. 269–276. Springer, Heidelberg (2014)
15. Otten, J.: nanoCoP: A non-clausal connection prover. In: Olivetti, N., Tiwari, A. (eds.) IJCAR 2016. LNAI, vol. 9706, pp. 302–312. Springer, Heidelberg (2016)
16. Otten, J., Bibel, W.: leanCoP: lean connection-based theorem proving. Journal of Symbolic Computation 36(1–2), 139–161 (2003)
17. Otten, J., Bibel, W.: Advances in connection-based automated theorem proving. In: Bowen, J., Hinchey, M., Olderog, E.R. (eds.) Provably Correct Systems. Springer, Heidelberg (2017)
18. Raths, T., Otten, J.: The QMLTP problem library for first-order modal logics. In: Gramlich, B., et al. (eds.) IJCAR 2012. LNAI, vol. 7364, pp. 454–461. Springer, Heidelberg (2012)
19. Raths, T., Otten, J., Kreitz, C.: The ILTP problem library for intuitionistic logic. Journal of Automated Reasoning 38, 261–271 (2007)
20. Sahlin, D., Franzen, T., Haridi, S.: An intuitionistic predicate logic theorem prover. Journal of Logic and Computation 2(5), 619–656 (1992)
21. Schmitt, S., Lorigo, L., Kreitz, C., Nogin, A.: JProver: Integrating connection-based theorem proving into interactive proof assistants. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) IJCAR 2001. LNAI, vol. 2083, pp. 421–426. Springer, Heidelberg (2001)
22. Waaler, A.: Connections in nonclassical logics. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, pp. 1487–1578. Elsevier Science, Amsterdam (2001)
23. Wallen, L.A.: Automated Deduction in Non-Classical Logics. MIT Press, Cambridge (1990)