Clausal Connection-Based Theorem Proving in Intuitionistic First-Order Logic

Jens Otten

Institut für Informatik, University of Potsdam
August-Bebel-Str. 89, 14482 Potsdam-Babelsberg, Germany
jeotten@cs.uni-potsdam.de

Abstract. We present a clausal connection calculus for first-order intuitionistic logic. It extends the classical connection calculus by adding prefixes that encode the characteristics of intuitionistic logic. Our calculus is based on a clausal matrix characterisation for intuitionistic logic, which we prove correct and complete. The calculus was implemented by extending the classical prover leanCoP. We present some details of the implementation, called ileanCoP, and experimental results.

1 Introduction

Automated reasoning in intuitionistic first-order logic is an important task within the formal approach of constructing verifiable correct software. Interactive proof assistants, like NuPRL [5] and Coq [2], use constructive type theory to formalise the notion of computation and would greatly benefit from a higher degree of automation. Automated theorem proving in intuitionistic logic is considerably more difficult than in classical logic, because additional non-permutabilities in the intuitionistic sequent calculus [8] increase the search space. In classical logic (disjunctive or conjunctive) clausal forms are commonly used to simplify the problem of copying appropriate subformulae (so-called multiplicities). Once a formula is converted into clausal form multiplicities can be restricted to clauses. For intuitionistic logic a validity-preserving clausal form does not exist.

An elegant way of encoding the intuitionistic non-permutabilities was given by Wallen [30] by extending Bibel's (non-clausal) *characterisation* of logical validity [3]. The development of proof *calculi* and implementations based on this characterisation (e.g. [11, 21, 22, 25]) were restricted to non-clausal procedures, making it difficult to use more established clausal methods (e.g. [3, 4, 13, 14]).

In this paper we present a clausal matrix characterisation for intuitionistic logic. Extending the usual Skolemization technique makes it possible to adapt a connection calculus that works on prefixed matrices in clausal form. It simplifies the notation of multiplicities and the use of existing clausal connection-based implementations for classical logic. Only a few changes are required to adapt the classical prover leanCoP to deal with prefixed matrices.

The paper is organised as follows. In Section 2 the standard (non-clausal) matrix characterisation is presented before Section 3 introduces a clausal characterisation using Skolemization and a prefixed connection calculus. The ileanCoP

implementation with experimental results is described in Section 4. We conclude with a summary and a brief outlook on further research in Section 5.

2 Preliminaries

We assume the reader to be familiar with the language of classical first-order logic (see, e.g., [7]). We start by defining some basic concepts before briefly describing the matrix characterisation for classical and intuitionistic logic.

2.1 Formula Trees, Types and Multiplicities

Some basic concepts are formula trees, types and multiplicities (see [4, 7, 30]). Multiplicities encode the contraction rule in the sequent calculus [8].

Definition 1 (Formula Tree). *A* formula tree *is the graphical representation of a formula F as a tree. Each node is labeled with a connective/quantifier or atomic subformula of F and marked with a unique name, its* position*, denoted by $a_0, a_1, \ldots$. The set of all positions is denoted by $\mathcal{P}os$. The* tree ordering *$< \subseteq \mathcal{P}os \times \mathcal{P}os$ is the (partial) ordering on the positions in the formula tree, i.e. $a_i < a_j$ iff position a_i is below position a_j in the formula tree.*

Example 1. Figure 1 shows the formula tree for $F_1 = \forall x Px \Rightarrow Pb \wedge Pc$ with $\mathcal{P}os = \{a_1, \ldots, a_5\}$. It is, e.g., $a_1 < a_2$ and $a_0 < a_4$. Note that each subformula of a given formula corresponds to exactly one position in its formula tree, e.g. $\forall x Px$ corresponds to the position a_1.

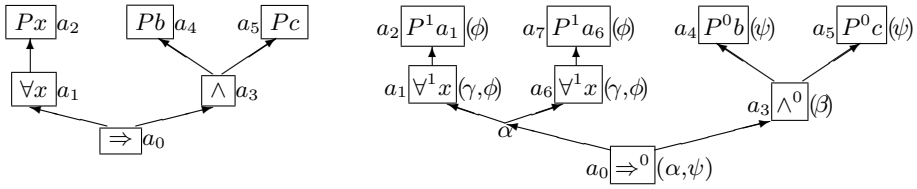

Fig. 1. Formula tree for F_1 and F_1^μ with $\mu(a_1) = 2$

Definition 2 (Types, Polarity). *The* principal *and* intuitionistic type *of a position is determined by its label and polarity according to Table 1 (e.g. a formula/position $A \wedge B$ with polarity 1 has type α). Atomic positions have no principal type and some positions have no intuitionistic type. We denote the sets of positions of type γ, δ, ϕ, and ψ by Γ, Δ, Φ, and Ψ, respectively. The* polarity *(0 or 1) of a position is determined by the label and polarity of its predecessor in the formula tree according to table 1 (e.g. if $A \wedge B$ has polarity 1 then both subformula A and B have polarity 1 as well). The root position has polarity 0.*

Example 2. In the left formula tree for F_1 in Figure 1 (ignore the additional branch for now) each node/position is marked with its types and its polarity. The positions a_0, a_1, a_3 have principal type α, γ and β, respectively; the positions a_1, a_2 and a_0, a_4, a_5 have intuitionistic type ϕ and ψ, respectively.

Table 1. Principal type, intuitionistic type and polarity of positions/subformulae

principal type α	$(A \wedge B)^1$	$(A \vee B)^0$	$(A \Rightarrow B)^0$	$(\neg A)^1$	$(\neg A)^0$
successor polarity	A^1, B^1	A^0, B^0	A^1, B^0	A^0	A^1
principal type β	$(A \wedge B)^0$	$(A \vee B)^1$	$(A \Rightarrow B)^1$		
successor polarity	A^0, B^0	A^1, B^1	A^0, B^1		

principal type γ	$(\forall x A)^1$	$(\exists x A)^0$	*principal type* δ	$(\forall x A)^0$	$(\exists x A)^1$
successor polarity	A^1	A^0	successor polarity	A^0	A^1

intuitionistic type ϕ	$(\neg A)^1$	$(A \Rightarrow B)^1$	$(\forall x A)^1$	P^1 (P is atomic)
intuitionistic type ψ	$(\neg A)^0$	$(A \Rightarrow B)^0$	$(\forall x A)^0$	P^0 (P is atomic)

Definition 3 (Multiplicity). *The* multiplicity $\mu : \Gamma \cup \Phi \to I\!N$ *assigns each position of type* γ *and* ϕ *a natural number. By* F^μ *we denote an* indexed formula*. In the formula tree of* F^μ *multiple instances of subformulae — according to the multiplicity of its positions — have been considered. The branch instances of a subformula have an implicit node/position of type* α *as predecessor.*

Example 3. The formula tree for F_1 with $\mu(a_1)$=2 is shown in Figure 2. Here and in the following we will replace variables in atomic formulae by their quantifier positions. Thus positions of type γ and δ appear in atomic formulae.

2.2 Matrix Characterisation for Classical Logic

To resume the characterisation for classical logic [3, 4] we introduce the concepts of matrices, paths and connections. A path corresponds to a sequent and a connection to an axiom in the sequent calculus [8]. For the first-order case we also need the notation of first-order substitution and reduction ordering.

Definition 4 (Matrix,Path,Connection).

1. *In the* matrix(-representation) *of an indexed formula* F^μ *we place the subformulae of a formula of principal type* α *side by side whereas subformulae of a subformula of principal type* β *are placed one upon the other. Furthermore, we omit all connectives and quantifiers.*
2. *A* path *through an indexed formula* F^μ *is a subset of the atomic formulae of its formula tree; it is a horizontal path through the matrix of* F^μ*.*
3. *A* connection *is a pair of atomic formulae with the same predicate symbol but with different polarities.*

Example 4. The matrix for F_1^μ is given in Figure 2. There are two paths through F_1^μ: $\{P^1a_1, P^1a_6, P^0b\}$ and $\{P^1a_1, P^1a_6, P^0c\}$. They contain the connections $\{P^1a_1, P^0b\}$, $\{P^1a_6, P^0b\}$ and $\{P^1a_1, P^0c\}$, $\{P^1a_6, P^0c\}$, respectively. We will present a more formal definition of matrices and paths in Section 3.

$$\begin{bmatrix} & & P^0b \\ P^1a_1 & P^1a_6 & \\ & & P^0c \end{bmatrix}$$

$$\frac{\Gamma, A \vdash}{\Gamma \vdash \neg A, \Delta}\ \neg\text{-right} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B, \Delta}\ \Rightarrow\text{-right} \qquad \frac{\Gamma \vdash A[x\backslash a]}{\Gamma \vdash \forall x A, \Delta}\ \forall\text{-right}$$

Fig. 2. Matrix for F_1^μ and special intuitionistic sequent rules

Definition 5 (First-order Substitution, Reduction Ordering).

1. *A* first-oder substitution $\sigma_Q : \Gamma \to \mathcal{T}$ *assigns to each position of type* γ *a term* $t \in \mathcal{T}$. $\mathcal{T}$ *is the set of terms made up of constants, functions and elements of* Γ *and* Δ*, which are called* term variables *and* term constants*, respectively. A connection* $\{P^0 s, P^1 t\}$ *is said to be* σ_Q-complementary *iff* $\sigma_Q(s)=\sigma_Q(t)$. σ_Q *induces a relation* $\sqsubset_Q \subseteq \Delta \times \Gamma$ *in the following way: for all* $u \in \Gamma$ $v \sqsubset_Q u$ *holds for all* $v \in \Delta$ *occurring in* $\sigma_Q(u)$.
2. *The* reduction ordering $\lhd := (< \cup \sqsubset_Q)^+$ *is the transitive closure of the tree ordering* $<$ *and the relation* $\sqsubset_Q$. *A first order substitution* σ_Q *is said to be* admissible *iff the reduction ordering* $\lhd$ *is irreflexive.*

Theorem 1 (Characterisation for Classical Logic [4]). *A (first-order) formula* F *is classically valid, iff there is a multiplicity* μ*, an admissible first-order substitution* σ_Q *and a set of* σ_Q*-complementary connections such that every path through* F^μ *contains a connection from this set.*

Example 5. Let $\mu(a_1)$=2 and σ_Q=$\{a_1\backslash b, a_6\backslash c\}$, i.e. $\sigma_Q(a_1)$=b, $\sigma_Q(a_6)$=c. σ_Q is admissible, since the induced reduction ordering (= tree ordering) is irreflexive. $\{\{P^1 a_1, P^0 b\}, \{P^1 a_6, P^0 c\}\}$ is a σ_Q-complementary set of connections and every path through F_1^μ contains a connection from this set. Thus F_1 is classically valid.

2.3 Matrix Characterisation for Intuitionistic Logic

A few extensions are necessary to adapt the characterisation of classical validity to intuitionistic logic. These are prefixes and an intuitionistic substitution. Prefixes encode the characteristics of the special rules (see Figure 2) in the intuitionistic sequent calculus (see [30]). Alternatively they can be seen as an encoding of the possible world semantics of intuitionistic logic (see [30]).

Definition 6 (Prefix). *Let* $u_1 < u_2 < \ldots < u_n \leq u$ *be the positions of type* ϕ *or* ψ *that dominate the position/formula* u *in the formula tree. The* prefix *of* u*, denoted* $pre(u)$*, is a string over* $\Phi \cup \Psi$ *and defined as* $pre(u) := u_1 u_2 \ldots u_n$.

Example 6. For the formula F_1^μ we obtain $pre(a_2)$=$a_0 A_1 A_2$, $pre(a_7)$= $a_0 A_6 A_7$, $pre(a_4)$=$a_0 a_4$ and $pre(a_5)$=$a_0 a_5$. Positions of type ϕ are written in capitals.

Definition 7 (Intuitionistic/Combined Substitution, Admissibility).

1. *An* intuitionistic substitution $\sigma_J : \Phi \to (\Phi \cup \Psi)^*$ *assigns to each position of type* ϕ *a string over the alphabet* $(\Phi \cup \Psi)$. *Elements of* Φ *and* Ψ *are called* prefix variables *and* prefix constants*, respectively.* σ_J *induces a relation* $\sqsubset_J \subseteq \Psi \times \Phi$ *in the following way: for all* $u \in \Phi$ $v \sqsubset_J u$ *holds for all* $v \in \Psi$ *occurring in* $\sigma_J(u)$.
2. *A* combined substitution $\sigma := (\sigma_Q, \sigma_J)$ *consists of a first-order and an intuitionistic substitution. A connection* $\{P^0 s, P^1 t\}$ *is* σ-complementary *iff* $\sigma_Q(s)=\sigma_Q(t)$ *and* $\sigma_J(pre(P^0 s))=\sigma_J(pre(P^1 t))$. *The* reduction ordering $\lhd := (< \cup \sqsubset_Q \cup \sqsubset_J)^+$ *is the transitive closure of* $<$, $\sqsubset_Q$ *and* $\sqsubset_J$.
3. *A combined substitution* $\sigma=(\sigma_Q, \sigma_J)$ *is said to be* admissible *iff the reduction ordering* $\lhd$ *is irreflexive and the following condition holds for all* $u \in \Gamma$ *and all* $v \in \Delta$ *occurring in* $\sigma_Q(u)$: $\sigma_J(pre(u))=\sigma_J(pre(v)) \circ q$ *for some* $q \in (\Phi \cup \Psi)^*$.

Theorem 2 (Characterisation for Intuitionistic Logic [30]). *A formula F is intuitionistically valid, iff there is a multiplicity μ, an admissible combined substitution $\sigma = (\sigma_Q, \sigma_J)$, and a set of σ-complementary connections such that every path through F^μ contains a connection from this set.*

Example 7. Let $\mu(a_1)$=2 and $\sigma = (\sigma_Q, \sigma_J)$ with σ_Q=$\{a_1\backslash b, a_6\backslash c\}$ and $\sigma_J = \{A_1\backslash\varepsilon, A_2\backslash a_4, A_6\backslash\varepsilon, A_7\backslash a_5\}$ in which ε denotes the empty string. Then σ is admissible and $\{\{P^1a_1, P^0b\}, \{P^1a_6, P^0c\}\}$ is a σ_Q-complementary set of connections and every path through F_1^μ contains a connection from this set. Therefore F_1 is intuitionistically valid.

3 Clausal Connection-Based Theorem Proving

This section presents a clausal version of the matrix characterisation for intuitionistic logic and a prefixed connection calculus based on this characterisation.

3.1 Prefixed Matrix and Skolemization

In order to develop a prefixed connection calculus more formal definitions of the concepts of a matrix and paths (as introduced in Definition 4) are required. Afterwards we will also introduce a Skolemization technique, which extends the Skolemization used for classical logic to intuitionistic logic.

A (non-clausal, i.e. nested) matrix is a set of clauses and contains α-related subformulae. A clause is a set of matrices and contains β-related subformulae. A (non-clausal) matrix is a compact representation of a (classical) formula and can be represented in the usual two dimensional graphical way. Adding prefixes will make it suitable to represent intuitionistic formulae as well. The formal definition of paths through a (matrix of a) formula is adapted accordingly.

Definition 8 (Prefixed Matrix).

1. *A* prefix p *is a string with* $p \in (\Phi \cup \Psi)^*$. *Elements of* Φ *and* Ψ *play the role of variables and constants, respectively. A* prefixed (signed) formula $F^{pol}{:}p$ *is a formula* F *marked with a polarity* $pol \in \{0, 1\}$ *and a prefix* p.
2. *A* prefixed matrix M *is a set of clauses in which a* clause *is a set of prefixed matrices and prefixed atomic formulae. The prefixed matrix of a prefixed formula* $F^{pol}{:}p$ *is inductively defined according to Table 2. In Table 2* A, B *are formulae,* Ps *is an atomic formula,* $p \in (\Phi \cup \Psi)^*$ *is a prefix,* $Z \in \Phi$, $a \in \Psi$ *are new (unused) elements.* A_x *and* A_c *is the formula* A *in which* y *is replaced by new (unused) elements* $x \in \Gamma$ *and* $c \in \Delta$, *respectively. The prefixed matrix of a formula* F, *denoted* $matrix(F)$, *is the prefixed matrix of* $F^0{:}\varepsilon$.

Definition 9 (Path). *A* path *through a (prefixed) matrix* M *or clause* C *is a set of prefixed atomic formulae and defined as follows: If* M=$\{\{P^{pol}s{:}p\}\}$ *is a matrix and* Ps *an atomic formula then* $\{P^{pol}s{:}p\}$ *is the only path through* M. *Otherwise if* M=$\{C_1, .., C_n\}$ *is a matrix and* $path_i$ *is a path through the clause* C_i *then* $\bigcup_{i=1}^{n} path_i$ *is a path through* M. *If* C=$\{M_1, .., M_n\}$ *is a clause and* $path_i$ *is a path through the matrix* M_i *then* $path_i$ *(for* $1 \leq i \leq n$*) is a path through* C.

Table 2. The prefixed matrix of a formula $F^{pol}{:}p$

Formula $F^{pol}{:}p$	Prefixed matrix of $F^{pol}{:}p$	M_A, M_B is matrix of
$(P^1 s){:}p$	$\{\{P^1 s{:}p \circ Z\}\}$	
$(P^0 s){:}p$	$\{\{P^0 s{:}p \circ a\}\}$	
$(A \wedge B)^1{:}p$	$\{\{M_A\},\{M_B\}\}$	$A^1{:}p\ ,\ B^1{:}p$
$(A \vee B)^0{:}p$	$\{\{M_A\},\{M_B\}\}$	$A^0{:}p\ ,\ B^0{:}p$
$(A \Rightarrow B)^0{:}p$	$\{\{M_A\},\{M_B\}\}$	$A^1{:}p\ ,\ B^0{:}p$
$(\forall y A)^1{:}p$	M_A	$A^1_x{:}p \circ Z$
$(\exists y A)^0{:}p$	M_A	$A^0_x{:}p$

Formula $F^{pol}{:}p$	Prefixed matrix of $F^{pol}{:}p$	M_A, M_B is matrix of
$(\neg A)^1{:}p$	M_A	$A^0{:}p \circ Z$
$(\neg A)^0{:}p$	M_A	$A^1{:}p \circ a$
$(A \wedge B)^0{:}p$	$\{\{M_A, M_B\}\}$	$A^0{:}p\ ,\ B^0{:}p$
$(A \vee B)^1{:}p$	$\{\{M_A, M_B\}\}$	$A^1{:}p\ ,\ B^1{:}p$
$(A \Rightarrow B)^1{:}p$	$\{\{M_A, M_B\}\}$	$A^0{:}p\ ,\ B^1{:}p$
$(\forall y A)^0{:}p$	M_A	$A^0_c{:}p \circ a$
$(\exists y A)^1{:}p$	M_A	$A^1_c{:}p$

Example 8. The prefixed matrix of formula F_1 (see examples in Section 2) is $M_1{=}matrix(F_1){=}\ \{\{P^1 x_1{:}a_2 Z_3 Z_4\},\{P^0 b{:}a_2 a_5, P^0 c{:}a_2 a_6\}\}$, in which submatrices of the form $\{\{M_1,..,M_n\}\}$ have been simplified to $M_1,..,M_n$. The paths through M_1 are $\{P^1 x_1{:}a_2 Z_3 Z_4, P^0 b{:}a_2 a_5\}$ and $\{P^1 x_1{:}a_2 Z_3 Z_4, P^0 c{:}a_2 a_6\}$.

An indexed formula F^{μ} is defined in the usual way, i.e. each subformula F' of type γ or ϕ is assigned its multiplicity $\mu(F') \in I\!N$ encoding the number of instances to be considered. Before F^{μ} is translated into a matrix, each F' is replaced with $(F'_1 \wedge \ldots \wedge F'_{\mu(F')})$ in which F'_i is a copy of F'. The notation of σ-complementary is slightly modified, i.e. a connection $\{P^0 s{:}p, P^1 t{:}q\}$ is *σ-complementary* iff $\sigma_Q(s){=}\sigma_Q(t)$ and $\sigma_J(p){=}\sigma_J(q)$.

For a combined substitution σ to be admissible (see Definition 7) the reduction ordering $\lhd$ induced by σ has to be irreflexive. In classical logic this restriction encodes the Eigenvariable condition in the classical sequent calculus [8]. It is usually integrated into the σ-complementary test by using the well-known Skolemization technique together with the occurs-check of term unification. We extend this concept to the intuitionistic substitution σ_J. To this end we introduce a new substitution $\sigma_<$, which is induced by the tree ordering $<$. $\sigma_<$ assigns to each constant (elements of Δ and Ψ) a Skolemterm containing all variables (elements of Γ and Φ) occurring below this constant in the formula tree. It is sufficient to restrict $\sigma_<$ to these elements, since $\sqsubset_J \subseteq \Psi \times \Phi$ and $\sqsubset_Q \subseteq \Delta \times \Gamma$.

Definition 10 ($\sigma_<$-Skolemization). *A* tree ordering substitution $\sigma_< : (\Delta \cup \Psi) \rightarrow \mathcal{T}$ *assigns a Skolemterm to every element of $\Delta \cup \Psi$. It is induced by the tree ordering $<$ in the following way: $\sigma_< := \{\, c \backslash c(x_1,...x_n) \mid c \in \Delta \cup \Psi$ and for all $x \in \Gamma \cup \Phi$: $(x \in \{x_1,\ldots x_n\}$ iff $x < c)\}$.*

Example 9. The tree ordering of formula F_1 induces the substitution $\sigma_< = \{a_2 \backslash a_2(), a_5 \backslash a_5(), a_6 \backslash a_6()\}$, since no variables occur before any constant.

Note that we follow a purely proof-theoretical view on Skolemization, i.e. as a way to integrate the irreflexivity test of the reduction ordering into the condition of σ-complementary. For classical logic this close relationship was pointed out by Bibel [4]. Like for classical logic the use of Skolemization simplifies proof calculi and implementations. There is no need for an explicit irreflexivity check and subformulae/-matrices can be copied by just renaming all their variables.

Lemma 1 (Admissibility Using $\sigma_<$-Skolemization). *Let F be a formula and $<$ be its tree ordering. A combined substitution $\sigma = (\sigma_Q, \sigma_J)$ is* admissible *iff (1) $\sigma' = \sigma_< \cup \sigma_Q \cup \sigma_J$ is idempotent, i.e. $\sigma'(\sigma') = \sigma'$, and (2) the following holds for all $u \in \Gamma$ and all $v \in \Delta$ occurring in $\sigma_Q(u)$: $\sigma_J(pre(u)) = \sigma_J(pre(v)) \circ q$ for some $q \in (\Phi \cup \Psi)^*$.*

Proof. It is sufficient to show that $\lhd$ is reflexive iff $\sigma_< \cup \sigma_Q \cup \sigma_J$ is not idempotent. "$\Rightarrow$": Let $\lhd$ be reflexive. Then there are positions with $a_1 \lhd \ldots \lhd a_n \lhd a_1$. For each $a_i \lhd a_j$ there is $a_i < a_j$, $a_i \sqsubset_Q a_j$ or $a_i \sqsubset_J a_j$. According to Definition 10, 5 and 7 there is some substitution with $\{a_j \backslash t\}$ in $\sigma_<$, σ_Q or σ_J, respectively, in which a_i occurs in t. Then $\sigma_< \cup \sigma_Q \cup \sigma_J$ is not idempotent. "$\Leftarrow$": Let $\sigma_< \cup \sigma_Q \cup \sigma_J$ be not idempotent. Then there is $\sigma' = \{a_1 \backslash .. a_n .., a_n \backslash .. a_{n-1} .., \ldots, a_2 \backslash .. a_1 ..\} \subseteq \sigma_< \cup \sigma_Q \cup \sigma_J$. Each $\{a_j \backslash .. a_i ..\} \in \sigma'$ is part of $\sigma_<$, σ_Q or σ_J. According to Definition 10, 5 and 7 it is $a_i < a_j$, $a_i \sqsubset_Q a_j$ or $a_i \sqsubset_J a_j$. Therefore $\lhd := (< \cup \sqsubset_Q \cup \sqsubset_J)^+$ is reflexive. □

3.2 A Clausal Matrix Characterisation

We will now define prefixed matrices in clausal form and adapt the notation of multiplicities to clausal matrices before presenting a clausal matrix characterisation for intuitionistic logic. The restriction of multiplicities to clauses makes it possible to use existing clausal calculi for which multiplicities can be increased during the proof search in an easy way. The transformation of a prefixed matrix to clausal form is done like for classical logic. Note that we apply the substitution $\sigma_<$ to the clausal matrix, i.e. to terms and prefixes of atomic formulae.

Definition 11 (Prefixed Clausal Matrix). *Let $<$ be a tree ordering and M be a prefixed matrix. The* (prefixed) clausal matrix *of M, denoted $clausal(M)$, is a set of clauses in which each clause is a set of prefixed atomic formulae. It is inductively defined as follows: If M has the form $\{\{P^{pol} s{:}p\}\}$ then $clausal(M) := M$; otherwise $clausal(M) := \bigcup_{C \in M} \{\{\bigcup_{i=1}^{n} c_i\} \mid c_i \in clausal(M_i)$ and $C = \{M_1, \ldots, M_n\}\}$. The (prefixed) clausal matrix of a formula F (with tree ordering $<$), denoted $M_c = matrix_c(F)$, is $\sigma_<(clausal(matrix(F)))$.*

Definition 12 (Clausal Multiplicity). *The* clausal multiplicity *$\mu_c : M_c \to \mathbb{N}$ assigns each clause in M_c a natural number, specifying the number of clause instances to be considered. $matrix_c^{\mu}(F)$ denotes the clausal matrix of F in which clause instances/copies specified by μ_c have been considered. Term and prefix variables in clause copies are renamed (Skolemfunctions are not renamed).*

Example 10. For F_1 let $\mu_c(\{P^1 x_1{:}a_2()Z_3()Z_4()\}) = 2$. Then $matrix_c^{\mu}(F_1) = \{\{P^1 x_1{:}a_2()Z_3 Z_4\}, \{P^1 x_7{:}a_2()Z_8 Z_9\}, \{P^0 b{:}a_2()a_5(), P^0 c{:}a_2()a_6()\}\}$.

Note that we use the same Skolemfunction (symbol) for instances of the same subformula/clause. This is an optimisation, which has a similar effect like the liberalised δ^+-rule for (classical) semantic tableaux [9]. Since we apply the substitution $\sigma_<$ to the clausal matrix, term and prefix constants can now both contain term and prefix variables. Therefore the first-order and intuitionistic substitutions σ_Q and σ_J are extended so that they assign terms over $\Gamma \cup \Delta \cup \Phi \cup \Psi$ to term and prefix variables (elements of Γ and Φ).

Theorem 3 (Clausal Characterisation for Intuitionistic Logic). *A formula F is intuitionistically valid, iff there is a multiplicity μ_c, an admissible combined substitution $\sigma = (\sigma_Q, \sigma_J)$, and a set of σ-complementary connections such that every path through $matrix^{\mu}_{c}(F)$ contains a connection from this set.*

Proof. Follows directly from the following Lemma 2 and Theorem 2. □

Lemma 2 (Equivalence of Matrix and Clausal Matrix Proofs). *There is a clausal matrix proof for F iff there is a (non-clausal) matrix proof for F.*

Proof. Main idea: paths through clausal matrix and (non-clausal) matrix of F are identical and substitutions and multiplicities μ_c/μ can be transfered into each other. Note that in the following proof "variable" refers to term and prefix variables. We assume that $\sigma_<$ is applied to the non-clausal matrix as well.
"$\Rightarrow$": Let $(\mu_c, \sigma, \mathcal{S})$ be a clausal matrix proof of F in which $\mathcal{S}$ is the connection set. We will construct a matrix proof (μ, σ', S') for F. Let M_c=$matrix^{\mu}_{c}(F)$ be the clausal matrix of F (with clause instances according to μ_c). We construct a matrix M=$matrix(F^{\mu})$ in the following way: We start with the original matrix M. For each clause $C_i \in M_c$ we identify the vertical path through M which represents C_i (modulo Skolemfunction names); see matrices on left side of Figure 3. If this vertical path shares variables with already identified clauses in M we copy the smallest subclause of M, so that the new path contains copies $x'_1, .., x'_n$ of all (term and prefix) variables $x_1, .., x_n$ in C_i with $\sigma(x_i) \neq \sigma(x'_i)$. Note that Skolemfunctions in copied subclauses of M are renamed and therefore unique. The constructed matrix M determines μ. Let $\sigma' := \sigma$ and $\mathcal{S}' := \mathcal{S}$. We identify every connection $\{P^0 s{:}p, P^1 t{:}q\} \in \mathcal{S}$ from M_c as $\{P^0 s'{:}p', P^1 t'{:}q'\}$ in M. If s', t', p' or q' contain a Skolemfunction f_i (which is unique in M), we rename Skolemfunctions in σ' so that $\sigma'(s', p') = \sigma(t', q')$. There can be no renaming conflict for the set $\mathcal{S}'$: Let $f(x_1), f(x_2), ..$ be the same Skolemfunctions in M_c, which are represented by different Skolemfunctions $f_1(x_1), f_2(x_2), ..$ in M. If $\sigma(x_i) \neq \sigma(x_j)$ then different Skolemfunctions do not matter, since $f_i(x_i)$ and $f_j(x_j)$ are never assigned to the same variable. The case $\sigma(x_i) = \sigma(x_j)$ does not occur according to the construction of M. If copies of the same variable are substituted by the same term, the branches in the formula tree can be folded up (see right side of Figure 3); otherwise the branches (and Skolemterms) differ. Every path through M contains a path from M_c as a subset. Therefore $(\mu, \sigma', \mathcal{S}')$ is a proof for F.

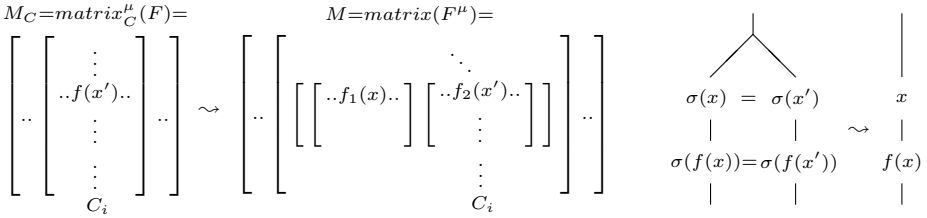

Fig. 3. Clause C_i in (non-)clausal matrix and folding up branches in formula tree

"$\Leftarrow$": Let $(\mu, \sigma', \mathcal{S}')$ be a (non-clausal) matrix proof for F. We will construct a clausal matrix proof $(\mu_c, \sigma, \mathcal{S})$. Let M=$matrix(F^{\mu})$ be the matrix of F^{μ}. We

construct a clausal matrix M_c=$matrix^{\mu}_c(F)$ in the following way: We start with the clausal form of the original formula F, i.e. M_c=$matrix_c(F)$. We extend M_c by using an appropriate μ_c, so that it is the clausal form of M (modulo Skolem-function and variable names). Let σ:=σ' and $\mathcal{S}$:=$\mathcal{S}'$. We extend σ by unifying all variables $x_1, x_2, ..$ in M_c which have the same image in M. Since copied Skolem-function names in M differ, but are identical in M_c we can simply replace them with a unique name in σ. Then all connections in $\mathcal{S}$ are σ-complementary. By induction we can also show that every path through M_c contains a connection from $\mathcal{S}$. Therefore $(\mu_c, \sigma, \mathcal{S})$ is a clausal proof for F. □

3.3 A Prefixed Connection Calculus

The clausal *characterisation* of intuitionistic validity in Theorem 3 serves as a basis for a proof *calculus*. Calculating the first-order substitution σ_Q is done by the well-known algorithms for term unification (see, e.g., [16]). Checking that all paths contain a connection can be done by, e.g., sequent calculi [8] tableau calculi [7] or connection-driven calculi [3, 4, 13, 14]. The clausal connection calculus is successfully used for theorem proving in classical logic (see, e.g., [12]). A basic version of the connection calculus for first-order classical logic is presented in [19]. The calculus uses a connection-driven search strategy, i.e. in each step a connection is identified along an active (sub-)path and only paths not containing the active path and this connection will be investigated afterwards. The clausal multiplicity μ_c is increased dynamically during the path checking process. We adapt this calculus to deal with intuitionistic logic (based on the clausal characterisation) by adding prefixes to the atomic formulae of each clause as specified in Definition 8 and 11.

Definition 13 (Prefixed Connection Calculus). *The axiom and the rules of the* prefixed connection calculus *are given in Figure 3. M is a prefixed clausal matrix, C, C_p, C_1, C'_2 are clauses of M and C_p is a positive clause (i.e. contains no atomic formulae with polarity 1), $C_1 \cup \{\overline{L}\}$ contains no (term or prefix) variables, $C'_2 \cup \{\overline{L'}\}$ contains at least one variable and $C_2/\overline{L}$ are copies of $C'_2/\overline{L'}$ in which all variables have been renamed. $\{L, \overline{L}\}$ and $\{L, \overline{L'}\}$ are σ-complementary connections, and $Path$ is the* active path*, which is a subset of some path trough M. A formula F is valid iff there is an admissible substitution $\sigma = (\sigma_Q, \sigma_J)$ and a derivation for $matrix_c(F)$ in which all leaves are axioms.*

Axiom:	*Start Rule:*	*Reduction Rule:*
$\dfrac{}{(\{\}, M, Path)}$	$\dfrac{(C_p, M \cup \{C_p\}, \{\})}{M \cup \{C_p\}}$	$\dfrac{(C, M, Path \cup \{\overline{L}\})}{(C \cup \{L\}, M, Path \cup \{\overline{L}\})}$

Extension Rule:

$$\frac{(C,M,Path) \quad (C_1, M, Path \cup \{L\})}{(C \cup \{L\}, M \cup \{C_1 \cup \{\overline{L}\}\}, Path)}$$

Extension Rule:*

$$\frac{(C,M,Path) \quad (C_2, M \cup \{C'_2 \cup \{\overline{L'}\}\}, Path \cup \{L\})}{(C \cup \{L\}, M \cup \{C'_2 \cup \{\overline{L'}\}\}, Path)}$$

Fig. 4. The connection calculus for first-order logic

Example 11. Let $matrix_c(F_1)$=$\{\{P^1x_1{:}a_2()Z_3Z_4\},\{P^0b{:}a_2()a_5(), P^0c{:}a_2()a_6()\}\}$ $=M$ be the clausal matrix of F_1. The following is a derivation for M.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(\{\},M,\{\})}Axiom \quad \dfrac{}{(\{\},M,\{P^0c{:}a_2()a_6()\})}Axiom}{(\{P^0c{:}a_2()a_6()\},M,\{\})}Extension^* \quad \dfrac{}{(\{\},M,\{P^0b{:}a_2()a_5()\})}Axiom}{(\{P^0b{:}a_2()a_5(),P^0c{:}a_2()a_6()\},M,\{\})}Extension^*}{\{\{P^1x_1{:}a_2()Z_3Z_4\},\{P^0b{:}a_2()a_5(),P^0c{:}a_2()a_6()\}\}}Start$$

This derivation is a proof for F_1 with the admissible substitution σ=(σ_Q,σ_J) and σ_Q=$\{x_1'\backslash b, x_2''\backslash c\}$, σ_J=$\{Z_3'\backslash\varepsilon, Z_4'\backslash a_2()a_5(), Z_3''\backslash\varepsilon, Z_4''\backslash a_2()a_6()\}$, i.e. F_1 is valid.

The intuitionistic substitution σ_J can be calculated using the prefix unification algorithm in [20]. It calculates a minimal set of most general unifiers for a given set of prefixes $\{s_1{=}s_1,..,t_n{=}t_n\}$ by using a small set of rewriting rules (similar to the algorithm in [16] for term unification). The following definition briefly describes this algorithm (see [20] for a detailed introduction).

Definition 14 (Prefix Unification). *The* prefix unification *of two prefixes s and t is done by applying the rewriting rules defined in Table 3. The rewriting rules replace a tuple (E,σ_J) by a modified tuple (E',σ_J') in which E and E' represent a prefix equation and σ_J,σ_J' are intuitionistic substitutions. Rules are applied non-deterministically. We start with the tuple $(\{s=\varepsilon|t\},\{\})$, for technical reasons we divide the right part of t, and stop when the tuple $(\{\},\sigma_J)$ is derived. In this case σ_J represents a most general unifier. There can be more than one most general unifier (see [20]). The algorithm can also be used to calculate a unifier for a set of prefix equations $\{s_1{=}t_1,..,s_n{=}t_n\}$ in a stepwise manner.*

We use these symbols: $s,t,z\in(\Phi\cup\Psi)^*$, $s^+,t^+,z^+\in(\Phi\cup\Psi)^+$, $X\in(\Phi\cup\Psi)$, V,V_1 $V'\in\Phi$ and $C,C_1,C_2\in\Psi$. V' is a new variable which does not refer to a position in the formula tree and does not occur in the substitution σ_J computed so far.

Table 3. Rewriting rules for prefix unification

R1.	$\{\varepsilon=\varepsilon\|\varepsilon\},\sigma_J$	$\rightarrow$	$\{\},\sigma_J$	
R2.	$\{\varepsilon=\varepsilon\|t^+\},\sigma_J$	$\rightarrow$	$\{t^+=\varepsilon\|\varepsilon\},\sigma_J$	
R3.	$\{Xs=\varepsilon\|Xt\},\sigma_J$	$\rightarrow$	$\{s=\varepsilon\|t\},\sigma_J$	
R4.	$\{Cs=\varepsilon\|Vt\},\sigma_J$	$\rightarrow$	$\{Vt=\varepsilon\|Cs\},\sigma_J$	
R5.	$\{Vs=z\|\varepsilon\},\sigma_J$	$\rightarrow$	$\{s=\varepsilon\|\varepsilon\},\{V\backslash z\}\cup\sigma_J$	
R6.	$\{Vs=\varepsilon\|C_1t\},\sigma_J$	$\rightarrow$	$\{s=\varepsilon\|C_1t\},\{V\backslash\varepsilon\}\cup\sigma_J$	
R7.	$\{Vs=z\|C_1C_2t\},\sigma_J$	$\rightarrow$	$\{s=\varepsilon\|C_2t\},\{V\backslash zC_1\}\cup\sigma_J$	
R8.	$\{Vs^+=\varepsilon\|V_1t\},\sigma_J$	$\rightarrow$	$\{V_1t=V\|s^+\},\sigma_J$	$(V{\neq}V_1)$
R9.	$\{Vs^+=z^+\|V_1t\},\sigma_J$	$\rightarrow$	$\{V_1t=V'\|s^+\},\{V\backslash z^+V'\}\cup\sigma_J$	$(V'\in\Phi)$
R10.	$\{Vs=z\|Xt\},\sigma_J$	$\rightarrow$	$\{Vs=zX\|t\},\sigma_J$	($s{=}\varepsilon$ or $t{\neq}\varepsilon$ or $X\in\Psi$)

Example 12. In the first step/connection of the proof for F_1 we need to unify the prefixes $s=a_2Z_3'Z_4'$ and $t=a_2a_5$. This is done as follows: $(\{a_2Z_3'Z_4'{=}\varepsilon|a_2a_5\},\{\})$ $\xrightarrow{\text{R3}}$ $(\{Z_3'Z_4'{=}\varepsilon|a_5\},\{\})$ $\xrightarrow{\text{R6}}$ $(\{Z_4'{=}\varepsilon|a_5\},\{Z_3'\backslash\varepsilon\})$ $\xrightarrow{\text{R10}}$ $(\{Z_4'{=}a_5|\varepsilon\},\{Z_3'\backslash\varepsilon\})$ $\xrightarrow{\text{R5}}$ $(\{\varepsilon{=}\varepsilon|\varepsilon\},\{Z_3'\backslash\varepsilon, Z_4'\backslash a_5\})$ $\xrightarrow{\text{R1}}$ $(\{\},\{Z_3'\backslash\varepsilon, Z_4'\backslash a_5\})$. Note that the second (most general) unifier can be calculated by using rule R10 instead of R6.

4 An Implementation: ileanCoP

The calculus presented in Section 3 was implemented in Prolog. We will present details of the implementation and performance results. The source code (and more information) can be obtained at `http://www.leancop.de/ileancop/`.

4.1 The Code

leanCoP is an implementation of the clausal connection calculus presented in Section 3 for *classical* first-order logic [19]. The reduction rule is applied before extension rules are applied and open branches are selected in a depth-first way. We adapt leanCoP to intuitionistic logic by adding

a. prefixes to atomic formulae and a prefix unification procedure,
b. to each clause the set of term variables contained in it and an admissibility check in which these sets are used to check condition (2) of Lemma 1.

The main part of the ileanCoP code is depicted in Figure 5. The underlined text was added to leanCoP; no other changes were done. A prefix `Pre` is added to atomic formulae `q:Pre` in which `Pre` is a list of prefix variables and constants. The set of term variables `Var` is added to each clause `Var:Cla` in which `Var` is a list containing pairs `[V,Pre]` in which `V` is a term variable and `Pre` its prefix.

```
 (1)   prove(Mat,PathLim) :-
 (2)        append(MatA,[FV:Cla|MatB],Mat), \+ member(-(_):_,Cla),
 (3)        append(MatA,MatB,Mat1),
 (4)        prove([!:[]],[FV:[-(!):(-[])|Cla]|Mat1],[],PathLim,[PreSet,FreeV]),
 (5)        check_addco(FreeV), prefix_unify(PreSet).
 (6)   prove(Mat,PathLim) :-
 (7)        \+ ground(Mat), PathLim1 is PathLim+1, prove(Mat,PathLim1).

 (8)   prove([],_,_,_,[[],[]]).
 (9)   prove([Lit:Pre|Cla],Mat,Path,PathLim,[PreSet,FreeV]) :-
(10)        (-NegLit=Lit;-Lit=NegLit) ->
(11)           ( member(NegL:PreN,Path), unify_with_occurs_check(NegL,NegLit),
(12)             \+ \+ prefix_unify([Pre=PreN]), PreSet1=[], FreeV3=[]
(13)             ;
(14)             append(MatA,[Cla1|MatB],Mat), copy_term(Cla1,FV:Cla2),
(15)             append(ClaA,[NegL:PreN|ClaB],Cla2),
(16)             unify_with_occurs_check(NegL,NegLit),
(17)             \+ \+ prefix_unify([Pre=PreN]),
(18)             append(ClaA,ClaB,Cla3),
(19)             ( Cla1==FV:Cla2 ->
(20)                  append(MatB,MatA,Mat1)
(21)                  ;
(22)                  length(Path,K), K<PathLim,
(23)                  append(MatB,[Cla1|MatA],Mat1)
(24)             ),
(25)             prove(Cla3,Mat1,[Lit:Pre|Path],PathLim,[PreSet1,FreeV1]),
(26)             append(FreeV1,FV,FreeV3)
(27)           ),
(28)           prove(Cla,Mat,Path,PathLim,[PreSet2,FreeV2]),
(29)           append([Pre=PreN|PreSet1],PreSet2,PreSet),
(30)           append(FreeV2,FreeV3,FreeV).
```

Fig. 5. Main part of the ileanCoP source code

Condition (2) of the admissibility check, `check_addco`, and the unification of prefixes, `prefix_unify`, are done after a classical proof, i.e. a set of connections, has been found. Therefore the set of prefix equations `PreSet` and the clause-variable sets `FreeV` are collected in an additional argument of `prove`. `check_addco` and `prefix_unify` require 5 and 26 more lines of code, respectively (see [20, 22]). Condition (1) of the admissibility check of Lemma 1 is realized by the occurs-check of Prolog during term unification. Two prefix constants are considered equal if they can be unified by term unification. Note that we perform a *weak prefix unification* (line 12 and 17) for each connection already during the path checking process (double negation prevents any variable bindings).

To prove the formula F_1 from Section 2 and 3 we call `prove(M,1).` in which `M= [ [[X,[1^[],Z]]]:[-(p(X)): -([1^[],Z,Y])], []:[p(b):\[1^[],2^[]],p(c): [1^[],3^[]]] ]` is the prefixed matrix of F_1 (with clause-variable sets added).

4.2 Refuting some First-Order Formulae

In order to obtain completeness ileanCoP performs iterative deepening on the proof depth, i.e. the size of the active path. The limit for this size, `PathLim`, is increased after the proof search for a given path limit has failed (line 7 in Figure 5). If the matrix contains no variables, the given matrix is refuted.

We will integrate a more restrictive method in which the path limit will only be increased if the current path limit was actually reached. In this case the predicate `pathlim` is written into Prolog's database indicating the need to increase the path limit if the proof search fails. It can be realized by modifying the following lines of the code in Figure 5:

```
(7)     retract(pathlim), PathLim1 is PathLim+1, prove(Mat,PathLim1).
(22)            length(Path,K), ( K<PathLim ->
(23a)           append(MatB,[Cla1|MatA],Mat1) ;
(23b)           \+ pathlim -> assert(pathlim), fail )
```

The resulting implementation is able to refute a large set of first-order formulae (but it does not result in a decision procedure for propositional formulae).

4.3 Performance Results

We have tested ileanCoP on all 1337 non-clausal (so-called FOF) problems in the TPTP library [27] of version 2.7.0 that are classically valid or are not known to be valid/invalid. The tests were performed on a 3 GHz Xeon system running Linux and ECLiPSe Prolog version 5.7 ("nodbgcomp." was used to generate code without debug information). The time limit for all proof attempts was 300 s.

The results of ileanCoP are listed in the last column of Table 4. We have also included the results of the following five theorem provers for intuitionistic first-order logic: JProver [25] (implemented in ML using a non-clausal connection calculus and prefixes), the Prolog and C versions of ft [24] (using an intuitionistic tableau calculus with many additional optimisation techniques and a contraction-free calculus [6] for propositional formulae), ileanSeP (using an intuitionistic tableau calculus; see `http://www.leancop.de/ileansep`) and ileanTAP

[22] (using a tableau calculus and prefixes). The timings of the classical provers Otter [17], leanTAP [1] and leanCoP [19] are provided as well.

Table 4. Performance results of ileanCoP and other provers on the TPTP library

	Otter	leanTAP	leanCoP	ileanSeP	JProver	ft_{Prolog}	ft_{C}	ileanTAP	ileanCoP
solved	509	220	335	90	96	112	125	116	**234**
proved	509	220	335	87	94	99	112	113	**188**
refuted	0	0	0	3	2	13	13	3	**46**
0 to <1s	431	202	280	76	80	106	122	110	**192**
to <10s	54	9	23	8	9	5	1	2	**16**
to <100s	22	8	29	4	7	1	2	2	**17**
to <300s	2	1	3	2	0	0	0	2	**9**
intui. 0.0	-	-	-	60	76	74	76	76	**76**
to ≤0.7	-	-	-	21	20	33	40	34	**39**
to ≤1.0	-	-	-	9	0	5	9	6	**119**
class. 0.0	256	163	227	75	85	92	100	88	**175**
to ≤1.0	253	57	108	15	11	20	25	28	**59**
time out	360	1012	944	1173	1237	1171	667	1165	**1041**
error	468	105	58	74	4	54	545	56	**62**

Of the 234 problems solved by ileanCoP 112 problems could not be solved by any other intuitionistic prover. The rating (see rows eight to twelve) expresses the relative difficulty of the problems from 0.0 (easy) to 1.0 (very difficult). The intuitionistic rating is taken from the ILTP library [23].

4.4 Analysing Test Runs

The proof search process of ileanCoP can be divided into the following four sections: transformation into clausal form, search for a "classical" proof (using weak prefix unification), admissibility check and prefix unification. We mark the transition to these sections with the letters a, b and c as follows:

	a		b		c	
clausal transformation		"classical"-proof		admissibility check		prefix unification

Table 5 contains information about how often each section is entered when running ileanCoP on the 1337 TPTP problems. The numbers assigned to the letters express the number of times this section is entered. For example the column a=b=c=1 means that each section is entered just once. The rows var(ø,max) contain the average and maximum number of variables collected when entering section b, whereas the rows pre(ø,max) contain the average and maximum number of prefix equations collected when entering section c.

Most of the problems (178 formulae) have been proved without backtracking in sections a/b (1st column) or section a (2nd column). This means that the corresponding classical sequent proofs can be converted into intuitionistic sequent proofs by simply reordering the rules. Most of the refuted problems (40 formulae) failed because of weak prefix unification in section a (5th column).

Table 5. Analysis of the TPTP test runs

	$a{=}b{=}c{=}1$	$a{=}b{=}1; c{>}1$	$a{=}1; b,c{>}1$	$a{=}1; b{\geq}1; c{=}0$	$a{=}1; b{=}c{=}0$	$a{=}b{=}c{=}0$
proved	159	19	10	-	-	-
var(ø,max)	(13,216)	(21,62)	(11,23)	-	-	-
pre(ø,max)	(29,1191)	(22,119)	(13,26)	-	-	-
refuted	1	-	3	2	40	-
var(ø,max)	(2,2)	-	(3,5)	(5,5)	-	-
pre(ø,max)	(3,3)	-	(4,4)	-	-	-
time out	3	31	17	2	987	62
var(ø,max)	(26,34)	(25,106)	(41,146)	(47,70)	-	-
pre(ø,max)	(15,19)	(21,99)	(40,112)	-	-	-

For 987 problems no "classical" proof could be found (5th column). For 62 problems (6th column) the size of the clausal form is too big and the transformation results in a stack overflow ("error" in Table 4). For 31 problems sections b and c did not finish after a "classical" proof has been found (2nd column).

5 Conclusion

We have presented a clausal matrix characterisation for intuitionistic logic based on the standard characterisation [30] (see also [29]). Encoding the tree-ordering by an additional substitution replaces the reflexivity test of the reduction ordering by checking if the (extended) combined substitution is idempotent. Due to this Skolemization technique and the transformation into clausal form, multiplicities can be restricted to clauses and clause instances can be generated by just renaming the term and prefix variables of the original clauses. This allows the use of existing clausal calculi and implementations for classical logic (in contrast to specialised calculi [28]), without the redundancy caused by (relational) translations [18] into classical first-order logic. Instead an additional prefix unification is added, which captures the specific restrictions of intuitionistic logic, i.e.

first-order logic	=	propositional logic	+ term unification ,
intuitionistic logic	=	classical logic	+ prefix unification .

We have adapted a clausal connection calculus and transformed the classical prover leanCoP into the intuitionistic prover ileanCoP with only minor additions. Experimental results on the TPTP library showed that the performance of ileanCoP is significantly better than any other (published) automated theorem prover for intuitionistic first-order logic available today. The correctness proof provides a way to convert the clausal matrix proofs back into non-clausal matrix proofs, which can then be converted into sequent-style proofs [26].

While a clausal form technically simplifies the proof search procedure, it has a negative effect on the size of the resulting clausal form and the search space. A non-clausal connection-based proof search, which increases multiplicities in a demand-driven way, would not suffer from this weakness. Whereas the used prefix unification algorithm is very straightforward to implement, it can be improved by solving the set of prefix equations in a more simultaneous way. The clausal characterisation, the clausal connection calculus and the implementation can be adapted to all modal logics considered within the matrix characterisation

framework [30, 21, 11] by changing the unification rules [20]. By adding tests for the linear and relevant constraints within the complementary condition it can even be adapted to fragments of linear logic [10, 11, 15].

Acknowledgements

The author would like to thank Thomas Raths for providing the performance results for the listed classical and intuitionistic theorem proving systems.

References

1. B. Beckert, J. Posegga. leanTAP: lean, tableau-based theorem proving. *12th CADE*, LNAI 814, pp. 793–797, Springer, 1994.
2. Y. Bertot, P. Castéran. *Interactive Theorem Proving and Program Development*. Texts in Theoretical Computer Science, Springer, 2004.
3. W. Bibel. Matings in matrices. *Communications of the ACM*, 26:844–852, 1983.
4. W. Bibel. *Automated Theorem Proving*. Vieweg, second edition, 1987.
5. R. L. Constable et. al. *Implementing Mathematics with the NuPRL proof development system*. Prentice Hall, 1986.
6. R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *Journal of Symbolic Logic*, 57:795–807, 1992.
7. M. C. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 1990.
8. G. Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.
9. R. Hähnle, P. Schmitt The Liberalized δ-rule in free variable semantic tableaux. *Journal of Automated Reasoning*, 13:211–221, 1994.
10. C. Kreitz, H. Mantel, J. Otten, S. Schmitt Connection-based proof construction in linear logic. *14th CADE*, LNAI 1249, pp. 207–221, Springer, 1997.
11. C. Kreitz, J. Otten. Connection-based theorem proving in classical and non-classical logics. *Journal of Universal Computer Science*, 5:88–112, Springer, 1999.
12. R. Letz, J. Schumann, S. Bayerl, W. Bibel. Setheo: A high-performance theorem prover. *Journal of Automated Reasoning*, 8:183–212, 1992.
13. R. Letz, G. Stenz Model elimination and connection tableau procedures. *Handbook of Automated Reasoning*, pp. 2015–2114, Elsevier, 2001.
14. D. Loveland. Mechanical theorem proving by model elimination. *JACM*, 15:236–251, 1968.
15. H. Mantel, J. Otten. linTAP: A tableau prover for linear logic. *8th TABLEAUX Conference*, LNAI 1617, pp. 217–231, Springer, 1999.
16. A. Martelli, U. Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4:258–282, 1982.
17. W. McCune. Otter 3.0 reference manual and guide. Technical Report ANL-94/6, Argonne National Laboratory, 1994.
18. H. J. Ohlbach et al. Encoding two-valued nonclassical logics in classical logic. *Handbook of Automated Reasoning*, pp. 1403–1486, Elsevier, 2001.
19. J. Otten, W. Bibel. leanCoP: lean connection-based theorem proving. *Journal of Symbolic Computation*, 36:139–161, 2003.
20. J. Otten, C. Kreitz. T-string-unification: unifying prefixes in non-classical proof methods. *5th TABLEAUX Workshop*, LNAI 1071, pp. 244–260, Springer, 1996.
21. J. Otten, C. Kreitz. A uniform proof procedure for classical and non-classical logics. *KI-96: Advances in Artificial Intelligence*, LNAI 1137, pp. 307–319, Springer, 1996.
22. J. Otten. ileanTAP: An intuitionistic theorem prover. *6th TABLEAUX Conference*, LNAI 1227, pp. 307–312, Springer, 1997.
23. T. Raths, J. Otten, C. Kreitz. The ILTP library: benchmarking automated theorem provers for intuitionistic logic. *TABLEAUX 2005*, this volume, 2005. (`http://www.iltp.de`)
24. D. Sahlin, T. Franzen, S. Haridi. An intuitionistic predicate logic theorem prover. *Journal of Logic and Computation*, 2:619–656, 1992.
25. S. Schmitt et. al. JProver: Integrating connection-based theorem proving into interactive proof assistants. *IJCAR-2001*, LNAI 2083, pp. 421–426, Springer, 2001.
26. S. Schmitt, C. Kreitz Converting non-classical matrix proofs into sequent-style systems. *13th CADE*, LNAI 1104, pp. 2–16, Springer, 1996.
27. G. Sutcliffe, C. Suttner. The TPTP problem library - CNF release v1.2.1. *Journal of Automated Reasoning*, 21: 177–203, 1998. (`http://www.cs.miami.edu/~tptp`)
28. T. Tammet. A resolution theorem prover for intuitionistic logic. *13th CADE*, LNAI 1104, pp. 2–16, Springer, 1996.
29. A. Waaler Connections in nonclassical logics. *Handbook of Automated Reasoning*, pp. 1487–1578, Elsevier, 2001.
30. L. Wallen. *Automated deduction in nonclassical logic*. MIT Press, 1990.