Certification of Nonclausal Connection Tableaux Proofs

Michael Färber[0000−0003−1634−9525] and Cezary Kaliszyk[0000−0002−8273−6059]

Universität Innsbruck, Austria
michael.faerber@gedenkt.at, cezary.kaliszyk@uibk.ac.at

Abstract. Nonclausal connection tableaux calculi enable proof search without performing clausification. We give a translation of nonclausal connection proofs to Gentzen's sequent calculus LK and compare it to an existing translation of clausal connection proofs. Furthermore, we implement the translation in the interactive theorem prover HOL Light, enabling certification of nonclausal connection proofs as well as a new, complementary automation technique in HOL Light.

1 Introduction

Most automated theorem provers (ATPs) output only limited proof traces for performance reasons. This is in contrast to the LCF approach, which hinges on the correctness of a small, trusted kernel [13]. One way to certify the correctness of proofs produced by ATPs is to translate them to interactive theorem provers (ITPs) [15, 17]. Certification of proofs given by ATPs is also important for the integration of ATPs into interactive theorem provers, providing automation in the form of proof tactics [5].

Most ATPs convert their input problems to clausal normal form as preprocessing step [23]. To reconstruct the resulting clausal proofs in an ITP, it is necessary to verify in the ITP the conversion to clausal normal form. The ATP nanoCoP has demonstrated that a connection prover not requiring clausification can be effectively implemented [27]. The reconstruction of nonclausal proofs eliminates the necessity of proving the correctness of the clausification, but on the other hand, translating the proofs is more involved.

In this paper, we describe the translation of clausal and nonclausal connection proofs to Gentzen's LK. To ease the translation, we introduce slightly modified versions of the clausal and nonclausal connection calculus in section 3. Using these calculi, we describe a translation method from clausal and nonclausal connection proofs to LK in section 4. Based on this translation, we develop in section 5 an automatic proof certification of clausal proofs from leanCoP as well as of nonclausal proofs from nanoCoP in the ITP HOL Light. We evaluate the performance of our implementations on HOL Light problem sets in section 6.

This paper generalises work co-authored by the second author of this paper about the certification of clausal connection tableaux proofs [19]. Whereas [19] is concerned more with technical questions of implementing a clausal prover and a corresponding proof translation in a functional language, this paper abstracts more from technical details in order to treat the more involved nonclausal proof

translation. This paper extends section 6.4 of the first author's PhD thesis [11], where a preliminary version of the nonclausal proof translation described in this paper was introduced.

2 Connection Calculi

In this section, we will give a brief overview of the clausal and the nonclausal connection tableaux calculus. For more details and examples, see [26, 27].[1]

Let us start by fixing some notation. The transitive closure of a relation R is denoted by R^+, and the transitive reflexive closure by R^*. A term t is either a variable x, a constant a, or $f(t_1, \ldots, t_n)$, where f is a function symbol of arity n and $t_1, \ldots, t_n$ are terms. An atom A is $P(t_1, \ldots, t_n)$, where P is a predicate of arity n and $t_1, \ldots, t_n$ are terms. A (first-order) formula F is (A), $(F_1 \vee F_2)$, $(F_1 \wedge F_2)$, $(F_1 \implies F_2)$, $(\neg F_1)$, $(\forall x.F_1)$, or $(\exists x.F_1)$, where F_1 and F_2 are formulas, A is an atom, and x is a variable. We write a sequence of quantifiers $\forall x_1 \ldots x_n.F$ as $\forall \boldsymbol{x}.F$. The formula $F[t/x]$ denotes the formula F with all unbound occurrences of x replaced by t. A literal L is either $\neg A$ or A, where A is an atom. The complement $\overline{L}$ of a literal is A if L is of the shape $\neg A$, and $\neg A$ otherwise. A substitution σ is a function from variables to terms.

In the clausal calculus, a clause C is $\forall \boldsymbol{x}.(L_1 \vee \cdots \vee L_n)$ and a matrix M is $C_1 \wedge \cdots \wedge C_n$. In the nonclausal calculus, a clause C is $\forall \boldsymbol{x}.(X_1 \vee \cdots \vee X_n)$, where X is either a literal or a matrix, and a matrix M is $C_1 \wedge \cdots \wedge C_n$.[2] We refer to matrices in the clausal calculus as clausal matrices and to matrices in the nonclausal calculus as nonclausal matrices.

We can write a clause $\forall \boldsymbol{x}.(L_1 \vee \cdots \vee L_n)$ as a set $\{L_1, \ldots, L_n\}$ and we can write a matrix $C_1 \wedge \cdots \wedge C_n$ as a set $\{C_1, \ldots, C_n\}$. Alternatively, we write matrices as row vectors and clauses as column vectors.

For any formula F, there are equisatisfiable closed formulas $M(F)$ and $\bar{M}(F)$, where $M(F)$ is a nonclausal matrix and $\bar{M}(F)$ is a clausal matrix. We can convert any formula to a nonclausal matrix by conversion to negation normal form, Skolemisation (eliminating existential quantifiers), and pushing universal quantifiers inwards via $\forall \boldsymbol{x}.(F_1 \wedge F_2) \equiv (\forall \boldsymbol{x}.F_1) \wedge (\forall \boldsymbol{x}.F_2)$.
Example 1. Consider the following equivalent formulas F and $\bar{F}$.

$$F = Q \wedge P(a) \wedge \forall x.(\neg P(x) \vee (\neg P(s^2 x) \wedge (P(sx) \vee \neg Q)))$$
$$\bar{F} = Q \wedge P(a) \wedge (\forall x.\neg P(x) \vee \neg P(s^2 x)) \wedge (\forall x.\neg P(x) \vee P(sx) \vee \neg Q)$$

[1] We diverge from [26] by using a refutational point of view; that is, instead of proving formulas directly, we refute their negations. This shows up for example when we interpret clauses and matrices: In this paper, a clause (of a negated formula) represents a disjunction, whereas in [26], a clause (of an unnegated formula) represents a conjunction. Our refutational view is historically motivated by other proof certification methods, namely those for MESON [15] and leanCoP [19].

[2] We represent clauses with quantifiers to reduce the size of the translated proofs.

For brevity, we write sx for $s(x)$ and s^2x for $s(s(x))$. The nonclausal matrix M corresponds to F and the clausal matrix $\bar{M}$ to $\bar{F}$:

$$M = \left[[Q]\ [P(a)] \left[\begin{bmatrix} \neg P(x) \\ [\neg P(s^2x)] \end{bmatrix} \begin{bmatrix} P(sx) \\ \neg Q \end{bmatrix} \right] \right]$$

$$\bar{M} = \left[[Q]\ [P(a)] \begin{bmatrix} \neg P(x) \\ \neg P(s^2x) \end{bmatrix} \begin{bmatrix} \neg P(x) \\ P(sx) \\ \neg Q \end{bmatrix} \right]$$

The words of the connection calculi treated in this paper are tuples $\langle C, M, Path \rangle$, where C is a clause, M is a matrix, and $Path$ is a set of literals and matrices called the active path.[3] In the calculus rules, σ is a global (or rigid) term substitution, i.e. it is applied to the whole derivation. We say that a (non)clausal connection proof of M is a derivation of $\langle \emptyset, M, \emptyset \rangle$ in the (non)clausal connection calculus.

The rules of the clausal connection calculus are shown in Figure 1 [30]. For any closed formula F, we have that F is unsatisfiable iff there is a clausal connection proof of $\bar{M}(F)$ [3]. A clausal connection proof of $\bar{M}$ from Example 1 is given in Figure 2.

Axiom $\dfrac{}{\{\}, M, Path}\,\mathrm{A}$

Start $\dfrac{C_2, M, \{\}}{\varepsilon, M, \varepsilon}\,\mathrm{S}$ where C_2 is copy of $C_1 \in M$

Reduction $\dfrac{C, M, Path \cup \{L'\}}{C \cup \{L\}, M, Path \cup \{L'\}}\,\mathrm{R}$ where $\sigma(L) = \sigma(\overline{L'})$

Extension $\dfrac{C_2 \setminus \{L'\}, M, Path \cup \{L\} \quad C, M, Path}{C \cup \{L\}, M, Path}\,\mathrm{E}$

where C_2 is copy of $C_1 \in M$ and $L' \in C_2$ with $\sigma(L) = \sigma(\overline{L'})$

Fig. 1. Clausal connection calculus rules.

We now proceed to introduce definitions related to the nonclausal connection calculus.

[3] In the original description of the calculi, $Path$ denotes a set of literals. Our generalisation to literals and matrices is motivated by the correctness proof of our translation, in particular Theorem 1. It does, however, not alter the actual proof search with the calculi, as all active paths in a connection proof tree will only contain literals.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{\},\bar{M},\ldots}\,\mathrm{A} \quad \dfrac{}{\{\},\bar{M},\ldots}\,\mathrm{A}}{\{\neg P(\bar{x})\},\bar{M},\{Q,P(sx'),P(s\hat{x})\}}\,\mathrm{E} \quad \dfrac{}{\{\},\bar{M},\{Q,P(sx')\}}\,\mathrm{A}}{\{P(s\hat{x}),\neg Q\},\bar{M},\{Q,P(sx')\}}\,\mathrm{R} \quad \dfrac{}{\{\},\bar{M},\{Q\}}\,\mathrm{A}}{\{P(sx')\},\bar{M},\{Q\}}\,\mathrm{E}}{\{\neg P(x'),P(sx')\},\bar{M},\{Q\}}\,\mathrm{E} \quad \dfrac{}{\{\},\bar{M},\{\}}\,\mathrm{A}}{\{Q\},\bar{M},\{\}}\,\mathrm{E}}{\epsilon,\bar{M},\epsilon}\,\mathrm{S}$$

Fig. 2. Clausal connection proof with $\sigma = \{x' \mapsto a, \hat{x} \mapsto sx', \overline{x} \mapsto x'\}$.

Definition 1 (Clause Predicates). *A clause* C recursively contains *a literal or a matrix* X *iff* $X \in^+ C$.[4] *A clause* $C \in^+ M$ *is* α-related *to* X *iff there is some* $M' \in^* M$ *with* $\{C_X, C_C\} \subseteq M'$ *such that* $C_X \neq C_C$, $X \in^+ C_X$, *and* $C \in^* C_C$. *A variable is* free *in* $C \in^+ M$ *if it occurs only in literals recursively contained in* C *and (possibly) in literals to which* C *is* α*-related. A clause* C' *is a* parent clause *of* C *iff* $M' \in C'$ *and* $C \in M'$ *for some matrix* M'.

Definition 2 (Clause Functions). *A* copy of the clause $C \in^+ M$ *is created by replacing all free variables in* C *with 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.

In a clausal matrix $\bar{M}$, all clauses in $\bar{M}$ can potentially give rise to an extension step. In a nonclausal matrix M, however, we have clauses C for which $C \in^+ M$, but $C \notin M$. It depends on the active path which of these clauses may give rise to an extension step. Those clauses which do are called extension clauses.

Definition 3 (Extension Clause). *The clause* $C \in^+ M$ *is an* extension clause *(*e-clause*) of the matrix* M *with respect to a set* $Path$ *iff either (a)* C *recursively contains an element of* $Path$*, or (b)* C *is* α*-related to all elements of* $Path$ *recursively contained in* M *and if* C *has a parent clause, that parent clause recursively contains an element of* $Path$.

Given an extension clause, its β-clause removes from the clause those parts that are irrelevant to the current subgoal.

Definition 4 (β-clause). *The* β-clause *of* C *with respect to* L *is* C *with* L *and all clauses that are* α*-related to* L *removed.*

Example 2. Consider the nonclausal matrix

$$M = \left[[Q][P(a)]\overbrace{\left[\begin{array}{c}\neg P(x)\\ \left[\underbrace{[\neg P(s^2x)]}_{C_4}\underbrace{\left[\begin{array}{c}P(sx)\\ \neg Q\end{array}\right]}_{C_5}\right]\end{array}\right]}^{C_3}\right]$$

from Example 1. The extension clauses with respect to $\{Q\}$ are all clauses $C \in M$. In particular, the first clause in M, $\{Q\}$, is an extension clause due to condition

[4] We use the term "recursively contains" instead of "contains" as employed in [27] to clearly distinguish it from regular set membership.

(a) of Definition 3, because it contains Q, and the other clauses in M are extension clauses due to condition (b), because they are α-related to Q and do not have parent clauses. Only one of the clauses in M recursively contains $\neg Q$, namely C_3. The β-clause of C_3 with respect to $\neg Q$ is

$$\left[\begin{array}{c}\neg P(x)\\ \Big[\Big[P(sx)\Big]\Big]\end{array}\right]$$

Let us now assume that $\sigma(x) = a$. The extension clauses with respect to $\{Q, P(sx)\} \cup \{P(s^2a)\}$ are all clauses in M, plus C_4 due to condition (b) and C_5 due to condition (a). Two of these extension clauses recursively contain the literal $\neg P(s^2x)$ that can be unified with $\neg P(s^2a)$, namely C_3 and C_4. The β-clause of C_4 with respect to $\neg P(s^2x)$ is $\{\}$, and the β-clause of C_3 with respect to $\neg P(s^2x)$ is

$$\left[\begin{array}{c}\neg P(x)\\ [[]]\end{array}\right]$$

Some β-clauses in this example will be used in a nonclausal proof in Figure 7.

The rules of the nonclausal calculus are shown in Figure 3. The difference in the calculus rules to the clausal variant is the addition of a decomposition rule, and the adaptation of the extension rule to the nonclausal setting. For any closed formula F, we have that F is unsatisfiable iff there is a nonclausal connection proof of $M(F)$ [26]. A nonclausal proof of M from Example 1 as well as a shorter clausal proof of M from the same example will be given using slightly modified versions of the calculi in section 3.

Axiom $\dfrac{}{\{\}, M, Path}\,\mathrm{A}$

Start $\dfrac{C_2, M, \{\}}{\varepsilon, M, \varepsilon}\,\mathrm{S}$ where C_2 is copy of $C_1 \in M$

Reduction $\dfrac{C, M, Path \cup \{L'\}}{C \cup \{L\}, M, Path \cup \{L'\}}\,\mathrm{R}$ where $\sigma(L) = \sigma(\overline{L'})$

Extension $\dfrac{C_3, M[C_1 \backslash C_2], Path \cup \{L\} \quad C, M, Path}{C \cup \{L\}, M, Path}\,\mathrm{E}$

where C_3 is the β-clause of C_2 with respect to L', C_2 is copy of C_1, C_1 is e-clause of M with respect to $Path \cup \{L\}$, $L' \in^+ C_2$ with $\sigma(L) = \sigma(\overline{L'})$

Decomposition $\dfrac{C \cup C', M, Path}{C \cup \{M'\}, M, Path}\,\mathrm{D}$ where $C' \in M'$

Fig. 3. Nonclausal connection calculus rules.

3 Compressed Connection Calculi

In Otten's presentation of connection calculi [26], all proof rules have a fixed number of premises. To ease the presentation of proofs in this paper, we present slightly reformulated versions of Otten's calculi. We call these calculi *compressed*, because proofs in these calculi usually consist of fewer proof steps and take up less space. The compressed calculi can be considered a mixture between Otten's and Letz's presentation of connection tableaux [21].

We introduce the following notation for rules with an arbitrary number of premises:

$$\frac{\bigwedge_i P_i}{C} \quad \equiv \quad \frac{P_1 \quad \dots \quad P_n}{C}$$

The compressed connection calculi are shown in Figures 4 and 5. In the original calculi, the words are $\langle C, M, Path\rangle$. In the compressed calculi, the words are $\langle X, M, Path\rangle$, where X denotes an arbitrary clause element, i.e. a matrix or a literal. In the compressed calculi, the axiom rule becomes obsolete.

$$\text{Start} \quad \frac{\bigwedge_i \langle X_i, M, \{\}\rangle}{\varepsilon, M, \varepsilon}\,\mathrm{S} \quad \text{where } \{X_1, \dots, X_n\} \text{ is copy of } C \in M$$

$$\text{Reduction} \quad \frac{}{L, M, Path \cup \{L'\}}\,\mathrm{R} \quad \text{where } \sigma(L) = \sigma(\overline{L'})$$

$$\text{Extension} \quad \frac{\bigwedge_i \langle L_i, M, Path \cup \{L\}\rangle}{L, M, Path}\,\mathrm{E}$$
$$\text{where } \{L_1, \dots, L_n\} \cup \{L'\} \text{ is copy of } C \in M \text{ and } \sigma(L) = \sigma(\overline{L'})$$

Fig. 4. Compressed clausal connection calculus.

We will now show how proofs can be translated between the compressed calculi in this section and the original calculi in section 2.

Lemma 1. *The sequent $\langle \{X_1, \dots, X_n\}, M, Path\rangle$ has a proof in a connection calculus iff all sequents $\langle X_1, M, Path\rangle, \dots, \langle X_n, M, Path\rangle$ have proofs in the corresponding compressed connection calculus.*

Proof. Any connection proof of $\langle \{X_1, \dots, X_n\}, M, Path\rangle$ has the following shape:

$$R_1 \dfrac{P_1 \qquad R_2 \dfrac{P_2 \qquad \vdots \qquad R_n \dfrac{P_n \qquad \dfrac{}{\{\}, M, Path}\,\mathrm{A}}{\{X_2, \dots, X_n\}, M, Path}}{\{X_1, \dots, X_n\}, M, Path}$$

Start $\dfrac{\bigwedge_i \langle X_i, M, \{\}\rangle}{\varepsilon, M, \varepsilon}\,\mathrm{S}$ where $\{X_1, \ldots, X_n\}$ is copy of $C \in M$

Reduction $\dfrac{}{L, M, Path \cup \{L'\}}\,\mathrm{R}$ where $\sigma(L) = \sigma(\overline{L'})$

Extension $\dfrac{\bigwedge_i \langle X_i, M[C_1 \backslash C_2], Path \cup \{L\}\rangle}{L, M, Path}\,\mathrm{E}$

where $\{X_1, \ldots, X_n\}$ is the β-clause of C_2 with respect to L', C_2 is copy of C_1, C_1 is e-clause of M with respect to $Path \cup \{L\}$, $L' \in^+ C_2$ with $\sigma(L) = \sigma(\overline{L'})$

Decomposition $\dfrac{\bigwedge_i \langle X_i, M, Path\rangle}{M', M, Path}\,\mathrm{D}$ where $\{X_1, \ldots, X_n\} \in M'$

Fig. 5. Compressed nonclausal connection calculus.

From such a proof, we can recursively construct proofs of $\langle X_i, M, Path\rangle$ in the corresponding compressed calculus by

$$R_i \, \frac{P_i'}{X_i, M, Path}$$

where P_i' is the translation of the proof P_i to the compressed calculus. Similarly, we can translate proofs from the compressed to the original calculi. □

Example 3. For the matrices M and $\bar{M}$ in Example 1, proofs in the compressed calculi are given in Figures 6 and 7. The extension steps used to prove $\langle Q, M, \{\}\rangle$ and $\langle P(s\hat{x}), \hat{M}, \ldots\rangle$ in the nonclausal proof of M are explained in Example 2.

$$\dfrac{\dfrac{\dfrac{}{\neg P(x'), \bar{M}, \{Q\}}\,\mathrm{E} \qquad \dfrac{\dfrac{\dfrac{}{\neg P(\bar{x}), \bar{M}, \{Q, P(sx'), P(s\hat{x})\}}\,\mathrm{E}}{P(s\hat{x}), \bar{M}, \{Q, P(sx')\}}\,\mathrm{E} \qquad \dfrac{}{\neg Q, \bar{M}, \{Q, P(sx')\}}\,\mathrm{R}}{P(sx'), \bar{M}, \{Q\}}\,\mathrm{E}}{Q, \bar{M}, \{\}}\,\mathrm{E}}{\epsilon, \bar{M}, \epsilon}\,\mathrm{S}$$

Fig. 6. Proof in the compressed clausal calculus with $\sigma = \{x' \mapsto a, \hat{x} \mapsto sx', \overline{x} \mapsto x'\}$.

4 Connection Proof Translation

In this section, we propose a translation method from connection proofs to Gentzen's sequent calculus LK [12].

$$\dfrac{\dfrac{\dfrac{}{\neg P(x'), M', \{Q\}}\,\mathrm{E} \qquad \dfrac{\dfrac{\dfrac{}{P(s\hat{x}), \hat{M}, \{Q, P(sx')\}}\,\mathrm{E} \quad \dfrac{}{\neg Q, \hat{M}, \{Q, P(sx')\}}\,\mathrm{R}}{\left[[\neg P(s^2\hat{x})]\ \begin{bmatrix} P(s\hat{x}) \\ \neg Q \end{bmatrix}\right], \hat{M}, \{Q, P(sx')\}}\,\mathrm{D}}{P(sx'), M', \{Q\}}\,\mathrm{E}}{Q, M, \{\}}\,\mathrm{E}}{\epsilon, M, \epsilon}\,\mathrm{S}$$

Fig. 7. Proof in the compressed nonclausal calculus with $\sigma = \{x' \mapsto a, \hat{x} \mapsto sx'\}$.

A connection proof for a first-order formula F consists of a connection proof tree and a global substitution σ. Given this information, we want to construct a proof of $F \vdash \bot$, which is written in LK as $F \vdash$. To more concisely present the proof translation, we omit the substitution σ in the LK translation; for example, instead of writing $\sigma(L), \sigma(M), \sigma(Path) \vdash$, we write $L, M, Path \vdash$.

We translate connection proof trees recursively by distinguishing the different rules of the calculus. We denote by $[\Gamma \vdash]$ the LK translation of the connection proof for Γ. We write that C is in M iff $M = C_1 \wedge \cdots \wedge C_n$ with $C = C_i$ for some i with $1 \leq i \leq n$.

We use a rule $\wedge$L to extract a conjunct from a conjunction while keeping the conjunction in the context, as well as a rule $\bot$L to derive $\bot$ from two complementary literals in the context:[5]

$$\dfrac{\Gamma, C_i, C_1 \wedge \cdots \wedge C_n \vdash \Delta}{\Gamma, C_1 \wedge \cdots \wedge C_n \vdash \Delta}\,\wedge\mathrm{L} \qquad \dfrac{}{\Gamma, A, \overline{A} \vdash}\,\bot\mathrm{L}$$

We now describe the translation of connection proofs. Two rules of the connection calculi are translated the same way for clausal and nonclausal proofs, namely the start and the reduction rule. We show the translation of these rules in Figure 8. For the start rule, the translation obtains the formula corresponding to the clause C with the $\wedge$L rule, and instantiates it with the $\forall$L rule. The substitution σ is used to determine the instantiations, where fresh names are invented when a variable is unbound in the substitution. As noted before, we omit σ in the LK translation, writing $X_1 \vee \cdots \vee X_n, M \vdash$ to abbreviate $\sigma(X_1 \vee \cdots \vee X_n), \sigma(M) \vdash$. Then, the sequent is split into several proof trees $[X_i, M, \{\} \vdash]$, which represent the translations of the connection proofs for $\langle X_i, M, \{\}\rangle$.[6]

4.1 Clausal Proof Translation

The translation of the clausal extension rule (shown in Figure 4) is given in Figure 9. First, $L, M, Path \vdash$ is transformed to the equivalent $M, P \vdash$, where

[5] These rules are not part of Gentzen's original LK calculus. However, translating them into Gentzen's LK is straightforward.

[6] In the clausal setting, X_i could be written as L_i, but because the same rule is used in the nonclausal setting, where X_i can represent either a literal or a matrix, we write X_i for the common rules.

Connection Calculus	LK
$\dfrac{\bigwedge_i \langle X_i, M, \{\}\rangle}{\varepsilon, M, \varepsilon}\,\mathrm{S}$ where $\{X_1, \ldots, X_n\}$ is copy of $C \in M$	$\dfrac{\dfrac{\dfrac{[X_1, M, \{\} \vdash] \quad \ldots \quad [X_n, M, \{\} \vdash]}{X_1 \vee \cdots \vee X_n, M \vdash}\vee\mathrm{L}}{\forall \boldsymbol{x}.(X_1 \vee \cdots \vee X_n), M \vdash}\forall\mathrm{L}}{M \vdash}\wedge\mathrm{L}$ where $\forall \boldsymbol{x}.(X_1 \vee \cdots \vee X_n)$ in M
$\dfrac{}{L, M, Path \cup \{L'\}}\,\mathrm{R}$ where $\sigma(L) = \sigma(\overline{L'})$	$\dfrac{}{L, M, Path \cup \{L'\} \vdash}\,\bot\mathrm{L}$ where $L = \overline{L'}$

Fig. 8. LK translation of common connection calculus rules.

$P = Path \cup \{L\}$. The remaining translation resembles that of the start rule, with the exception that it additionally closes a proof branch containing the negated literal $\overline{L}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{[L_1, M, P \vdash] \quad \ldots \quad \overline{\overline{L}, M, P \vdash}\,\bot\mathrm{L} \quad \ldots \quad [L_n, M, P \vdash]}{L_1 \vee \cdots \vee \overline{L} \vee \cdots \vee L_n, M, P \vdash}\vee\mathrm{L}}{\forall \boldsymbol{x}.(L_1 \vee \cdots \vee L_n), M, P \vdash}\forall\mathrm{L}}{M, P \vdash}\wedge\mathrm{L}}{L, M, Path \vdash}$$

where $\forall \boldsymbol{x}.(L_1 \vee \cdots \vee L_n)$ in M and $P = Path \cup \{L\}$

Fig. 9. LK translation of the clausal extension rule.

4.2 Nonclausal Proof Translation

We now proceed with the translation of nonclausal connection proofs, using the calculus introduced in Figure 5. The LK context in the translation of nonclausal proofs now has the shape $X, \boldsymbol{M}, Path$, where $\boldsymbol{M}$ is a set of matrices instead of a single matrix M as in the clausal case. During translation, $\boldsymbol{M}$ is extended such that for each word $\langle L, M, Path\rangle$ in the connection calculus and its corresponding sequent $L, \boldsymbol{M}, Path \vdash$ in LK, the e-clauses of M with respect to $Path \cup \{L\}$ are the clauses C for which C in M' and $M' \in \boldsymbol{M}$. We will see this in detail in the explanation for the extension rule.

The LK translation of nonclausal proofs reuses the translations of the start and the reduction rules given in Figure 8. However, occurrences of M in the LK

translation are replaced by $\boldsymbol{M}$. The start rule uses $\boldsymbol{M} = \{M\}$, i.e. $\boldsymbol{M}$ contains only the initial problem matrix M.

The decomposition rule of the nonclausal calculus can be seen as a generalisation of the start rule. We give its translation to LK in Figure 10.

Connection Calculus	LK
$\dfrac{\bigwedge_i \langle X_i, M, Path\rangle}{M', M, Path}\,\mathrm{D}$ where $\{X_1, \ldots, X_n\} \in M'$	$\dfrac{\dfrac{\dfrac{\left[X_1, \boldsymbol{M}', Path \vdash\right] \quad \ldots \quad \left[X_n, \boldsymbol{M}', Path \vdash\right]}{X_1 \vee \cdots \vee X_n, \boldsymbol{M}', Path \vdash}\vee\mathrm{L}}{\forall \boldsymbol{x}.(X_1 \vee \cdots \vee X_n), \boldsymbol{M}', Path \vdash}\forall\mathrm{L}}{M', \boldsymbol{M}, Path \vdash}\wedge\mathrm{L}$ where $\forall \boldsymbol{x}.(X_1 \vee \cdots \vee X_n)$ in M' and $\boldsymbol{M}' = \{M'\} \cup \boldsymbol{M}$

Fig. 10. LK translation of the decomposition rule.

Let us now consider a nonclausal extension step applied to $\langle L, M, Path\rangle$. Let C_1 denote the e-clause of M with respect to $Path \cup \{L\}$ that was used for the extension step. By construction of $\boldsymbol{M}$ mentioned above, C_1 is some clause in $M_1 \in \boldsymbol{M}$. Furthermore, let β_1 be the β-clause of C_1 with respect to $\overline{L}$. Then we can find some m such that M_1, C_1 and β_1 can be written as in Figure 11.

$$M_i = \begin{cases} \left[\cdots \overbrace{\begin{bmatrix} X_{i,1} \\ \vdots \\ M_{i+1} \\ \vdots \\ X_{i,n_i} \end{bmatrix}}^{C_i} \cdots\right] & \text{if } i \le m \\ \overline{L} & \text{otherwise} \end{cases} \qquad \beta_i = \begin{cases} \begin{bmatrix} X_{i,1} \\ \vdots \\ [\beta_{i+1}] \\ \vdots \\ X_{i,n_i} \end{bmatrix} & \text{if } i \le m \\ [] & \text{otherwise} \end{cases}$$

Fig. 11. Definition of matrix M_i, clause C_i, and β-clause β_i.

The translation of the nonclausal extension rule is shown in Figure 12. We first transform $L, \boldsymbol{M}, Path \vdash$ to $\boldsymbol{M}^0, P \vdash$ which is equivalent due to $\boldsymbol{M}^0 = \boldsymbol{M}$. We then determine $M_1 \in \boldsymbol{M}$ and put it into the context by contraction (CL).

Now we recursively prove the sequent $M_i, \boldsymbol{M}^{i-1}, P \vdash$ as follows: If M_i is the literal $\overline{L}$, we prove the sequent $\overline{L}, \boldsymbol{M}^m, P \vdash$ with the $\bot$L rule. Otherwise, we proceed in the following way: First, we put the appropriate clause C_i of M_i that corresponds to β_i into the context with the $\wedge$L rule. In the same step, we merge M_i with $\boldsymbol{M}^{i-1}$, yielding $\boldsymbol{M}^i$. After the instantiation of C_i with the $\forall$L rule, the

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{[X_{m,1}, \boldsymbol{M}^m, P\vdash]} \quad \cdots \quad \dfrac{}{\overline{L}, \boldsymbol{M}^m, P\vdash}\,\bot\mathrm{L} \quad \cdots \quad [X_{m,n_m}, \boldsymbol{M}^m, P\vdash]}\,\vee\mathrm{L}}{\vdots}}{\big[X_{1,1}, \boldsymbol{M}^1, P\vdash\big] \quad \cdots \quad \dfrac{}{M_2, \boldsymbol{M}^1, P\vdash}\,\wedge\mathrm{L} \quad \cdots \quad \big[X_{1,n_1}, \boldsymbol{M}^1, P\vdash\big]}\,\vee\mathrm{L}}{X_{1,1}\vee\cdots\vee X_{1,n_1}, \boldsymbol{M}^1, P\vdash}\,\forall\mathrm{L}}{\forall\boldsymbol{x}.(X_{1,1}\vee\cdots\vee X_{1,n_1}), \boldsymbol{M}^1, P\vdash}\,\wedge\mathrm{L}}{M_1, \boldsymbol{M}^0, P\vdash}\,\mathrm{CL}}{\boldsymbol{M}^0, P\vdash}}{L, \boldsymbol{M}, Path\vdash}$$

where $\boldsymbol{M}^j = \boldsymbol{M} \cup \{M_i \mid 1 \le i \le j\}$ and $P = Path \cup \{L\}$

Fig. 12. LK translation of the nonclausal extension rule.

clause elements $X_{i,1}$ to X_{i,n_i} give rise to several proof branches where all but one are closed by translation of the proof branches of the connection proof. The one remaining clause element M_{i+1} gives rise to a sequent $M_{i+1}, \boldsymbol{M}^i, P\vdash$, which we translate by recursion. This concludes the translation of the extension rule.

Example 4. Consider the nonclausal proof given in Figure 7. We show its translation to LK in Figure 13, where boxed sequents indicate words of the original proof. We use F from Example 1 to define

$$\begin{aligned}\boldsymbol{M}_0 &= \{F\}\\ \boldsymbol{M}_1 &= \boldsymbol{M}_0 \cup \{\neg P(s^2a) \wedge (P(sa) \vee \neg Q)\}\\ \boldsymbol{M}_2 &= \boldsymbol{M}_1 \cup \{\neg P(s^3a) \wedge (P(s^2a) \vee \neg Q)\}\end{aligned}$$

The question might arise whether the proof translation necessarily needs to keep a set of matrices $\boldsymbol{M}$ containing potential extension clauses. Could one instead reconstruct extension clauses from the initial M and $Path$? The next example shows that extending $\boldsymbol{M}$ with extension clauses is indeed necessary.

Example 5. Consider the extension step that closes $\langle P(s\hat{x}), \hat{M}, \{Q, P(sx')\}\rangle$ in Figure 7. The extension clause used in this extension step is C_4 from Example 2. However, the closest to C_4 we can obtain from M and $\{Q, P(sx')\} \cup \{P(s\hat{x})\}$ is

$$\begin{bmatrix} \neg P(x') \\ \big[\big[\neg P(s^2x')\big]\big] \end{bmatrix}$$

As performed by our translation, extending $\boldsymbol{M}$ in the translation of the extension step for $\langle P(sx'), M', \{Q\}\rangle$ with the α-related clause $[\neg P(s^2x')]$ corresponding to C_4 allows us to translate the extension step for $P(s\hat{x})$ with precisely that clause.

The LK translation uses $Path$ only for reduction steps and $\boldsymbol{M}$ for extension steps, whereas the original calculus uses $Path$ for both. Future work might explore whether a calculus closer to the translation yields more efficient proof search.

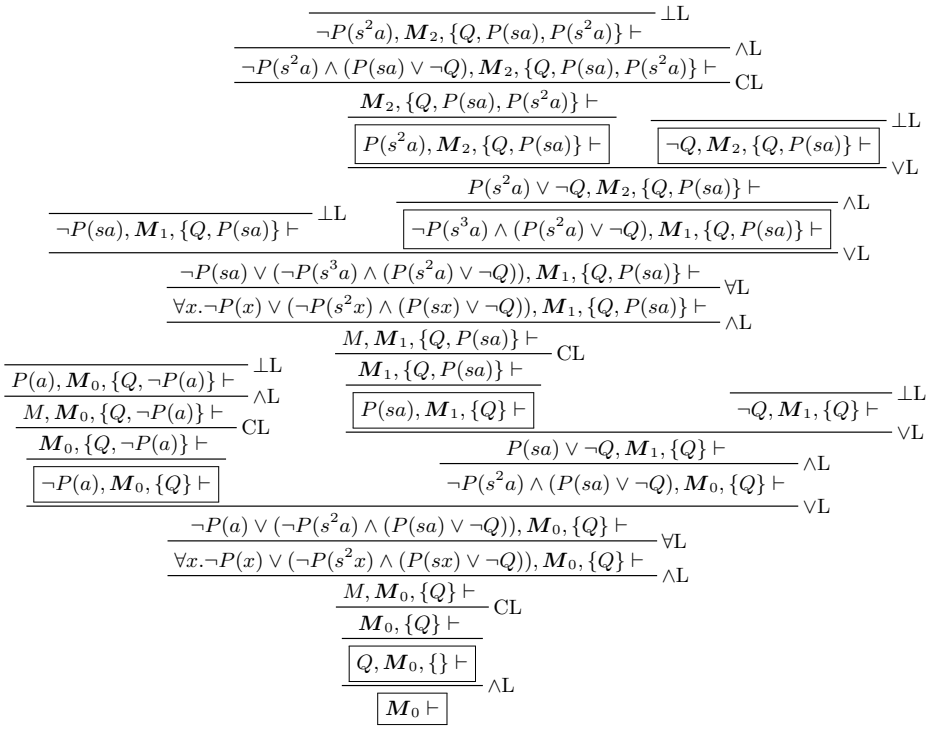

Fig. 13. Translation of the nonclausal proof in Figure 7 to LK.

Theorem 1. *Let $\langle X, M, Path\rangle$ be a word in the nonclausal connection proof Γ and let $\boldsymbol{M}$ contain the extension clauses of M with respect to $Path \cup \{X\}$. For every premise $\langle X', M', Path'\rangle$ of the proof step in Γ with the conclusion $\langle X, M, Path\rangle$, the translation $[X, \boldsymbol{M}, Path \vdash]$ has a sub-proof tree $[X', \boldsymbol{M'}, Path' \vdash]$ such that $\boldsymbol{M'}$ contains the extension clauses of M' with respect to $Path' \cup \{X'\}$.*

Proof. We distinguish the calculus rule to close $\langle X, M, Path\rangle$. The reduction rule is trivial because it has no premises.

Let us first consider the start rule in Figure 8. The translation of the start rule yields several proof trees of the shape $[X_i, \boldsymbol{M}, \{\} \vdash]$, where $\boldsymbol{M} = \{M\}$. For every i, the extension clauses of M with respect to X_i are all the clauses in M, as was illustrated in Example 2. Because all clauses in M are also contained in $\boldsymbol{M}$, the start rule satisfies the property.

Now for the decomposition rule shown in Figure 10. By hypothesis, $\boldsymbol{M}$ contains the extension clauses of M with respect to $Path \cup \{M'\}$. This implies that $\boldsymbol{M}$ contains all clauses that recursively contain M'. For every i, X_i is contained in M', therefore $\boldsymbol{M'}$ contains all clauses that recursively contain X_i, satisfying condition (a) of Definition 3. Furthermore, those clauses α-related to X_i that are required by condition (b) and that are not contained in $\boldsymbol{M}$ are $M' \setminus \{\forall \boldsymbol{x}.(X_1 \vee \cdots \vee X_n)\}$ and thus in $\boldsymbol{M'}$.

Finally we treat the extension rule shown in Figure 12. By hypothesis, $\boldsymbol{M}$ contains the extension clauses of M with respect to $Path \cup \{L\}$. We have to show that for each i and j, the extension clauses of M with respect to $P \cup \{X_{i,j}\}$ correspond to the clauses in $\boldsymbol{M}^i$. For every i and j, we have that $\boldsymbol{M}^i$ contains all clauses that recursively contain $X_{i,j}$, which in addition to some clauses in $\boldsymbol{M}$ are the clauses C_k (see Figure 11) with $k \leq i$. This covers condition (a) of Definition 3. Furthermore, those clauses α-related to $X_{i,j}$ that are required by condition (b) and that are not contained in $\boldsymbol{M}$ are the clauses $M_k \setminus \{C_k\}$ with $k \leq i$, which are contained in $\boldsymbol{M}^i$. □

Corollary 1. *For every formula F, if Γ is a nonclausal connection proof of $M(F)$, then the translation $[\Gamma \vdash]$ is an LK proof of $M(F) \vdash$.*

Proof. By induction on Γ and Theorem 1.

In four large test sets of nonclausal and clausal connection proofs, all translated proofs yielded by our implementations of the proof translations in this section are successfully verified by an interactive theorem prover, see section 6.

5 Implementation

HOL Light is an interactive theorem prover developed by Harrison in OCaml [16]. leanCoP and nanoCoP are clausal and nonclausal connection provers developed by Otten in Prolog [30, 27]. We developed proof search tactics for HOL Light based on leanCoP/nanoCoP and the proof translation shown in section 4.[7] To ease integration with HOL Light, all parts of the tactics are written in

[7] The source code can be retrieved at http://cl-informatik.uibk.ac.at/users/mfaerber/tactics.html.

OCaml, including functional implementations of leanCoP and nanoCoP using the compressed calculi in section 3.

The structure of the proof search tactics is shown in Figure 14: First, we convert given proof goals from higher-order logic to first-order logic. For this, we reuse a large part of the MESON [15] infrastructure, such as instantiation of higher-order axioms. This leaves us with first-order problems of the shape $(A_1 \wedge \cdots \wedge A_n) \implies C$, on which we run leanCoP and nanoCoP in the same interpreter as HOL Light [11]. Finally, we translate the resulting connection proofs to HOL Light proofs: We implemented the proof translation shown in section 4 such that it directly yields HOL Light instead of LK proofs.

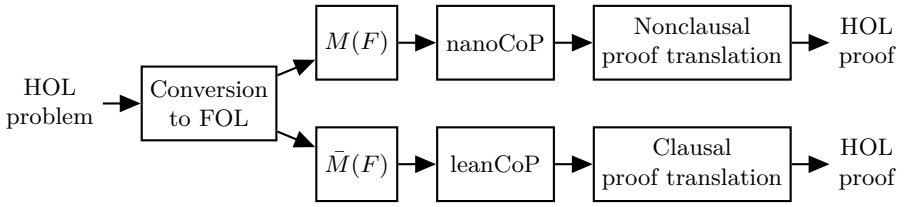

Fig. 14. Structure of the proof search tactics in HOL Light.

6 Evaluation

We compare the performance of our proof search tactics based on leanCoP 2.1 and nanoCoP 1.0 with the Metis [10] and MESON [15] tactics. Similarly to [19], we disable splitting for MESON. We evaluate the tactics on two kinds of problems derived from HOL Light: toplevel and MESON problems.

A toplevel problem results from any HOL Light theorem that is given a name on the OCaml toplevel. It consists of the conclusion of the theorem and the premises used to prove it. A MESON problem results from any call to the MESON tactic. It consists of the statement proven by MESON as well as the premises given to the MESON tactic. Note that toplevel problems are not necessarily solvable by first-order tactics, whereas MESON problems are, because the (first-order) tactic MESON is able to prove them.

We evaluate both toplevel and MESON problems with some tactic by letting the tactic find a proof of the problem conclusion using the problem premises. The problem counts as proven if the tactic finds a proof within a given time limit. We consider toplevel ("top") and MESON ("msn") problems from core HOL Light ("HL") and the Flyspeck project ("FS"), which finished in 2014 a formal proof of the Kepler conjecture [14]. We use the Git version 08f4461 of HOL Light from March 2017, running every tactic with a timeout of 10 seconds on each problem. We use a 48-core server with AMD Opteron 6174 2.2GHz CPUs, 320 GB RAM,

Listing 1.1. Flyspeck problem WLOG_LINEAR_INJECTIVE_IMAGE_ALT.

```
!P. (!f s. P s /\ linear f ==> P (IMAGE f s))
    ==> (!f. linear f /\ (!x y. f x = f y ==> x = y)
             ==> (!s. P (IMAGE f s) <=> P s))
==>
!P f s. (!g t. P t /\ linear g ==> P (IMAGE g t)) /\
        linear f /\ (!x y. f x = f y ==> x = y)
        ==> (P (IMAGE f s) <=> P s)
```

and 0.5 MB L2 cache per CPU. Each problem is always assigned one CPU. We run all provers with a timeout of 10 seconds per problem.

The results are shown in Table 1: Metis solves the largest number of problems among all considered datasets. The comparatively low performance of leanCoP/nanoCoP inside HOL Light is due to their heavy use of array operations for unification: Array access is more than 30 times faster in native OCaml programs compared to programs compiled in OCaml's toplevel (as used in HOL Light). When compiled as native OCaml programs, we have shown that leanCoP/nanoCoP solve more problems than Metis on four out of six datasets that we evaluated [11]. Running leanCoP/nanoCoP outside HOL Light and translating the resulting proofs inside HOL Light would thus very likely increase the performance of the corresponding tactics.

Table 1. Number of problems solved by various HOL Light tactics.

Prover	HL-top	HL-msn	FS-top	FS-msn
Problems in dataset	2499	1119	27112	44468
Metis	807	1029	4626	42829
MESON	736	900	4221	39227
leanCoP+cut	724	948	3714	39922
leanCoP−cut	717	844	3800	38528
nanoCoP+cut	538	802	2743	34213
nanoCoP−cut	550	811	2351	34769

Example 6. Listing 1.1 shows a Flyspeck toplevel problem which among the evaluated tactics, only nanoCoP can solve in the given time limit of 10 seconds. It is proven by nanoCoP in 2.27 seconds.

7 Related Work

Certification of ATP found proofs has been especially important for the integration of ATPs into interactive proof assistants. Such components provide automation in

the form of proof tactics for smaller steps. HOL Light includes the certified proof producing model elimination prover MESON [15]. The paramodulation-based prover Metis [17] was designed with a small certified proof core to simplify its integration with interactive theorem provers [10]. There exists a proof-certifying version of the intuitionistic first-order automated theorem prover JProver for Coq and Nuprl [33, 20] as well as a proof certifying version of an ordered paramodulation prover for Matita [1]. Proofs from several SAT/SMT solvers can be certified in Coq [9] and Isabelle [4]. The logical framework Dedukti allows for the import of superposition proofs from iProver [6] as well as of tableaux proofs from Zenon [7]. The GAPT framework provides translations for a multitude of calculi and automated theorem provers, such as Vampire, E, Prover9, and leanCoP [8, 31].

Among all provers whose proof certification is described in the cited work above, the only nonclausal one is JProver. However, its performance is far behind nanoCoP and the intuitionistic version of nanoCoP, nanoCoP-i, with nanoCoP and nanoCoP-i solving about three times as many problems as JProver on the TPTP and the ILTP benchmarks, respectively [27, 28]. On the other hand, unlike for nanoCoP-i, there already exists a proof certification method for JProver in an intuitionistic proof assistant, namely in Coq. This leaves as future work the extension of the proof certification in this paper to an intuitionistic setting, in order to enable stronger automated proof search via nanoCoP-i in proof assistants like Coq.

8 Conclusion

We proposed a translation from clausal and nonclausal connection proofs to LK, yielding a sound proof certification and a proof search tactic for HOL Light. The tactic certifies every nanoCoP and leanCoP proof output in our evaluation.

Future work includes the improvement of the proof search tactics, for example by calling external instances of nanoCoP/leanCoP, but also by improved preprocessing of the tactics, for example by reordering the clauses in the ITP before proof search [29]. The proof search tactic could also be integrated into other ITPs, such as Isabelle [34] and Coq [2]. The latter being an intuitionistic system motivates the translation of nonclassical connection proofs, such as given by ileanCoP and nanoCoP-i [25, 28]. Finally, we hope that the present article helps to prepare the ground for ITP-checked proofs of soundness/completeness of connection calculi as well as of their implementations.

Acknowledgements We thank the reviewers of JAR and TABLEAUX for their valuable comments as well as Jens Otten for clarifying details of his work on nonclausal connection proving. This work has been supported by a doctoral scholarship of the University of Innsbruck and the European Research Council (ERC) grant no. 714034 *SMART*.

Bibliography

[1] A. Asperti and E. Tassi. Higher order proof reconstruction from paramodulation-based refutations: The unit equality case. In M. Kauers, M. Kerber, R. Miner, and W. Windsteiger, editors, *MKM*, volume 4573 of *LNCS*, pages 146–160. Springer, 2007.

[2] Y. Bertot. A short presentation of Coq. In Mohamed et al. [22], pages 12–16.

[3] W. Bibel. *Automated theorem proving*. Artificial intelligence. Vieweg, second edition, 1987.

[4] J. C. Blanchette, S. Böhme, and L. C. Paulson. Extending Sledgehammer with SMT solvers. In N. Bjørner and V. Sofronie-Stokkermans, editors, *CADE-23*, volume 6803 of *LNCS*, pages 116–130. Springer, 2011.

[5] J. C. Blanchette, C. Kaliszyk, L. C. Paulson, and J. Urban. Hammering towards QED. *J. Formalized Reasoning*, 9(1):101–148, 2016.

[6] G. Burel. A shallow embedding of resolution and superposition proofs into the $\lambda\Pi$-calculus modulo. In J. C. Blanchette and J. Urban, editors, *PxTP*, volume 14 of *EPiC Series in Computing*, pages 43–57. EasyChair, 2013.

[7] R. Cauderlier and P. Halmagrand. Checking Zenon Modulo proofs in Dedukti. In Kaliszyk and Paskevich [18], pages 57–73.

[8] G. Ebner, S. Hetzl, G. Reis, M. Riener, S. Wolfsteiner, and S. Zivota. System description: GAPT 2.0. In Olivetti and Tiwari [24], pages 293–301.

[9] B. Ekici, A. Mebsout, C. Tinelli, C. Keller, G. Katz, A. Reynolds, and C. W. Barrett. SMTCoq: A plug-in for integrating SMT solvers into Coq. In R. Majumdar and V. Kuncak, editors, *CAV*, volume 10427 of *LNCS*, pages 126–133. Springer, 2017.

[10] M. Färber and C. Kaliszyk. Metis-based paramodulation tactic for HOL Light. In G. Gottlob, G. Sutcliffe, and A. Voronkov, editors, *GCAI*, volume 36 of *EPiC Series in Computing*, pages 127–136. EasyChair, 2015.

[11] M. Färber. *Learning Proof Search in Proof Assistants*. PhD thesis, Universität Innsbruck, 2018.

[12] G. Gentzen. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, 39(1):176–210, 1935.

[13] M. Gordon. From LCF to HOL: a short history. In G. D. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, pages 169–186. The MIT Press, 2000.

[14] T. C. Hales, M. Adams, G. Bauer, D. T. Dang, J. Harrison, T. L. Hoang, C. Kaliszyk, V. Magron, S. McLaughlin, T. T. Nguyen, T. Q. Nguyen, T. Nipkow, S. Obua, J. Pleso, J. Rute, A. Solovyev, A. H. T. Ta, T. N. Tran, D. T. Trieu, J. Urban, K. K. Vu, and R. Zumkeller. A formal proof of the Kepler conjecture. *Forum of Mathematics, Pi*, 5, 2017.

[15] J. Harrison. Optimizing proof search in model elimination. In M. A. McRobbie and J. K. Slaney, editors, *CADE-13*, volume 1104 of *LNCS*, pages 313–327. Springer, 1996.

[16] J. Harrison. HOL Light: An overview. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *TPHOLs*, volume 5674 of *LNCS*, pages 60–66. Springer, 2009.
[17] J. Hurd. First-order proof tactics in higher-order logic theorem provers. In M. Archer, B. D. Vito, and C. Muñoz, editors, *Design and Application of Strategies/Tactics in Higher Order Logics (STRATA)*, number NASA/CP-2003-212448 in NASA Technical Reports, pages 56–68, Sept. 2003.
[18] C. Kaliszyk and A. Paskevich, editors. *PxTP*, volume 186 of *EPTCS*, 2015.
[19] C. Kaliszyk, J. Urban, and J. Vyskočil. Certified connection tableaux proofs for HOL Light and TPTP. In X. Leroy and A. Tiu, editors, *CPP*, pages 59–66. ACM, 2015.
[20] C. Kreitz and S. Schmitt. A uniform procedure for converting matrix proofs into sequent-style systems. *Inf. Comput.*, 162(1-2):226–254, 2000.
[21] R. Letz and G. Stenz. Model elimination and connection tableau procedures. In Robinson and Voronkov [32], pages 2015–2114.
[22] O. A. Mohamed, C. A. Muñoz, and S. Tahar, editors. *TPHOLs*, volume 5170 of *LNCS*. Springer, 2008.
[23] A. Nonnengart and C. Weidenbach. Computing small clause normal forms. In Robinson and Voronkov [32], pages 335–367.
[24] N. Olivetti and A. Tiwari, editors. *IJCAR*, volume 9706 of *LNCS*. Springer, 2016.
[25] J. Otten. Clausal connection-based theorem proving in intuitionistic first-order logic. In B. Beckert, editor, *TABLEAUX*, volume 3702 of *LNCS*, pages 245–261. Springer, 2005.
[26] J. Otten. A non-clausal connection calculus. In K. Brünnler and G. Metcalfe, editors, *TABLEAUX*, volume 6793 of *LNCS*, pages 226–241. Springer, 2011.
[27] J. Otten. nanoCoP: A non-clausal connection prover. In Olivetti and Tiwari [24], pages 302–312.
[28] J. Otten. Non-clausal connection calculi for non-classical logics. In R. A. Schmidt and C. Nalon, editors, *TABLEAUX*, volume 10501 of *LNCS*, pages 209–227. Springer, 2017.
[29] J. Otten. Proof search optimizations for non-clausal connection calculi. In B. Konev, J. Urban, and P. Rümmer, editors, *PAAR*, volume 2162 of *CEUR Workshop Proceedings*, pages 49–57. CEUR-WS.org, 2018.
[30] J. Otten and W. Bibel. leanCoP: lean connection-based theorem proving. *J. Symb. Comput.*, 36(1-2):139–161, 2003.
[31] G. Reis. Importing SMT and connection proofs as expansion trees. In Kaliszyk and Paskevich [18], pages 3–10.
[32] J. A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning (in 2 volumes)*. Elsevier and MIT Press, 2001.
[33] S. Schmitt, L. Lorigo, C. Kreitz, and A. Nogin. JProver: Integrating connection-based theorem proving into interactive proof assistants. In R. Goré, A. Leitsch, and T. Nipkow, editors, *IJCAR*, volume 2083 of *LNCS*, pages 421–426. Springer, 2001.
[34] M. Wenzel, L. C. Paulson, and T. Nipkow. The Isabelle framework. In Mohamed et al. [22], pages 33–38.