Efficient translation of sequent calculus proofs into natural deduction proofs

Gabriel Ebner[1⋆] and Matthias Schlaipfer[2⋆⋆]

[1] TU Wien, Vienna, Austria
gebner@gebner.org
[2] TU Wien, Vienna, Austria
mschlaipfer@forsyte.at

Abstract. We present a simple and efficient translation of the classical multi-succedent sequent calculus LK to natural deduction. This translation aims to produce few excluded middle inferences, and to keep the quantifier complexity of the instances of excluded middle low. We evaluate the translation on proofs imported from resolution-based first-order theorem provers, and show that it achieves these goals in an efficient manner.

1 Introduction

Program extraction realizes the computational content of mathematical proofs as executable algorithms. These extractions, such as modified realizability [11] for example, aim to transform proofs of existential statements into an effective procedure that computes a witness for the existential quantifier. Frameworks to extract computational interpretations are often formulated for proofs in natural deduction.

Our particular interest lies in program extraction from classical proofs using a method that interprets natural deduction proofs with excluded middle as programs using exceptions [1]. In this setting, the excluded middle on quantified formulas is implemented by a lambda term encoding an exception mechanism. This point of view imposes a *quantitative* requirement on the desired proofs in natural deduction: the proofs do not need to be constructive, but the use of excluded middle should be limited: on the one hand, the *complexity* of the formula should be low as quantifier-free excluded middle can be interpreted efficiently. On the other hand, the *number* of inferences using excluded middle should be small because exceptions are costly and produce slow programs.

GAPT [8] is an open source system implementing proof transformations in classical first- and higher-order logic with a strong focus on the application of Herbrand's theorem. Proofs are mostly represented in the classical sequent

⋆ Supported by the Vienna Science and Technology Fund (WWTF) project VRG12-004.

⋆⋆ Supported by the Austrian Science Fund (FWF) through grants S11403-N23 (National Research Network RiSE) and W1255-N23 (LogiCS doctoral programme).

calculus LK. To extract programs from these proofs in LK, we want an effective translation to natural deduction in order to apply established program extraction methods.

Ultimately our goal is to extract programs from proofs generated by automated classical first-order theorem provers. GAPT can already reliably import proofs in LK from more than half a dozen external theorem provers. In this paper we describe the missing piece to apply program extraction: a translation to natural deduction.

Translations of LK to natural deduction are as old as the proof systems themselves. However these translations often focus on the preservation of provability, and not on quantitative and qualitative aspects of the output. For example, Gentzen first presented such a translation using a Hilbert-style calculus as an intermediate step [9]. The correspondence of cut-free proofs in the intuitionistic sequent calculus LJ and normal proofs in natural deduction has been studied by Zucker [20]. However, given the focus of the work they only translate single-succedent sequent calculus proofs. The textbook by Troelstra and Schwichtenberg [17, Section 3.3] also only shows the translation for the single-succedent LJ; the proof sketch for the extension to LK gives no information about the concrete proof transformation. In general, these translations do not try to minimize the classical content of the proofs.

Closer to our emphasis on the low amount of classical content is the recent work of Gilbert [10]. He shows how to translate automatically generated proofs in the multi-succedent sequent calculus LK to proofs in intuitionistic LJ, and empirically measures the constructivization success of the translation. He measures whether a complete proof is constructive or not, but does not quantify the amount of classical content.

Our translation uses a focusing approach and preserves the local shape of the proof in a straightforward way. For example, a conjunction-right inference in LK is typically translated to a single conjunction-introduction inference in natural deduction. LK supports classical reasoning by allowing multiple formulas in the succedent of a sequent, and inferences to operate on any one of these formulas. We translate this switching between formulas in the succedent using a simulated exchange rule that is similar in spirit to the μ constructor in $\lambda\mu$-calculus [15].

We define the proof systems LK and natural deduction in Section 2. The translation from LK to natural deduction is then described in Section 3. Finally, Section 4 evaluates an implementation of the translation on real-world proofs generated by an automated theorem prover.

2 Proof systems

We consider first-order formulas with standard connectives $\wedge, \vee, \Rightarrow, \neg, \top, \bot$, quantifiers $\forall, \exists$, and equality $=$. We denote formulas by A, B, C and D, variables by x and y, and constant terms by s, t and u. We write capture-avoiding substitution of x by y in A as $A[y/x]$. We present proofs as trees of inferences on sequents. A sequent $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ is a pair of multisets of formulas interpreted as

$(\bigwedge_{i=1}^{n} A_i) \Rightarrow \left(\bigvee_{j=1}^{m} B_j\right)$. We denote multisets of formulas by Γ, Δ, Σ and Π. We abbreviate union using comma: e.g., $\Gamma, \Sigma = \Gamma \cup \Sigma$ and $\Gamma, A = \Gamma \cup \{A\}$. Negation of a multiset of formulas $\neg\Gamma$ is interpreted as $\{\neg A \,|\, A \in \Gamma\}$. The left-hand side of the sequent symbol is the antecedent, and the right-hand side is the succedent of a sequent. The sequent above the inference rule is the premise and the sequent below the rule is the conclusion. We will also write $\overset{\pi}{\Gamma \vdash \Delta}$ to refer to a proof π with conclusion $\Gamma \vdash \Delta$.

Figure 1 lists the inference rules of the sequent calculus LK[3]. The inference rules of natural deduction (ND) are listed in sequent form in Figure 2. Our version of ND allows classical derivations by providing a rule for the excluded middle. We say the *main formula* of an inference rule in LK is the formula derived by the rule. For instance, for the ($\wedge$:r)-rule the main formula is $A \wedge B$.

Induction rule for natural numbers: In the case of the natural numbers (constructors 0 of arity 0 and s of arity 1) the induction rules for LK and ND reduce to

$$\frac{\overset{\pi_1}{\Gamma_1 \vdash \Delta_1, F[0]} \qquad \overset{\pi_2}{F[x], \Gamma_2 \vdash \Delta_2, F[sx]}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, F[t]}\,(\text{ind}) \qquad \text{and} \qquad \frac{\overset{\pi_1}{\Gamma_1 \vdash F[0]} \qquad \overset{\pi_2}{F[x], \Gamma_2 \vdash F[sx]}}{\Gamma_1, \Gamma_2 \vdash F[t]}\,(\text{ind})$$

respectively.

Symmetry and transitivity of equality: Even though we only include reflexivity and rewriting as inference rules for equality, symmetry and transitivity are derivable. The sequent calculus derivation of symmetry is depicted on the left and the natural deduction derivation on the right.

$$\dfrac{\dfrac{\dfrac{}{\vdash s = s}\,(\text{refl})}{s = t \vdash s = s}\,(\text{w:l})}{s = t \vdash t = s}\,(= \text{:r}) \qquad \dfrac{\dfrac{}{s = t \vdash s = t}\,(\text{ax}) \quad \dfrac{}{\vdash s = s}\,(= \text{:i})}{s = t \vdash t = s}\,(= \text{:e})$$

The derivations of transitivity are as follows—again, sequent calculus on the left and natural deduction on the right.

$$\dfrac{\dfrac{\dfrac{}{s = t \vdash s = t}\,(\text{ax})}{s = t, t = u \vdash s = t}\,(\text{w:l})}{s = t, t = u \vdash s = u}\,(= \text{:r}) \qquad \dfrac{\dfrac{\dfrac{}{s = t \vdash s = t}\,(\text{ax}) \quad \dfrac{}{\vdash s = s}\,(=\text{:i})}{s = t \vdash t = s}\,(=\text{:e}) \quad \dfrac{}{t = u \vdash t = u}\,(\text{ax})}{s = t, t = u \vdash s = u}\,(=\text{:e})$$

Examples: We will illustrate the proof systems using a proof of double-negation elimination in sequent calculus on the left, and using natural deduction on the right.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,(\text{ax})}{\vdash A, \neg A}\,(\neg\text{:r})}{\neg\neg A \vdash A}\,(\neg\text{:l})}{\vdash \neg\neg A \Rightarrow A}\,(\Rightarrow\text{:r}) \qquad \dfrac{\dfrac{}{A \vdash A}\,(\text{ax}) \qquad \dfrac{\dfrac{\dfrac{}{\neg\neg A \vdash \neg\neg A}\,(\text{ax}) \quad \dfrac{}{\neg A \vdash \neg A}\,(\text{ax})}{\neg\neg A, \neg A \vdash \bot}\,(\neg\text{:e})}{\neg\neg A, \neg A \vdash A}\,(\bot\text{:e})}{\dfrac{\neg\neg A \vdash A}{\vdash \neg\neg A \Rightarrow A}\,(\Rightarrow\text{:i})}\,(\text{em})$$

[3] Our implementation furthermore supports definition and theory rules which we omit for brevity.

Axioms

$$\frac{}{A \vdash A}\ (\text{ax}) \qquad \frac{}{\vdash \top}\ (\top) \qquad \frac{}{\vdash t = t}\ (\text{refl}) \qquad \frac{}{\bot \vdash}\ (\bot)$$

Cut

$$\frac{\Gamma \vdash \Delta, A \qquad A, \Sigma \vdash \Pi}{\Gamma, \Sigma \vdash \Delta, \Pi}\ (\text{cut})$$

Structural

$$\frac{\Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}\ (\text{w:l}) \qquad \frac{A, A, \Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}\ (\text{c:l}) \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\ (\text{w:r}) \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\ (\text{c:r})$$

Propositional

$$\frac{A, B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}\ (\wedge\text{:l}) \qquad \frac{\Gamma \vdash \Delta, A \qquad \Sigma \vdash \Pi, B}{\Gamma, \Sigma \vdash \Delta, \Pi, A \wedge B}\ (\wedge\text{:r})$$

$$\frac{A, \Gamma \vdash \Delta \qquad B, \Sigma \vdash \Pi}{A \vee B, \Gamma, \Sigma \vdash \Delta, \Pi}\ (\vee\text{:l}) \qquad \frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B}\ (\vee\text{:r})$$

$$\frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta}\ (\neg\text{:l}) \qquad \frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\ (\neg\text{:r})$$

$$\frac{\Gamma \vdash \Delta, A \qquad B, \Sigma \vdash \Pi}{A \Rightarrow B, \Gamma, \Sigma \vdash \Delta, \Pi}\ (\Rightarrow\text{:l}) \qquad \frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \Rightarrow B}\ (\Rightarrow\text{:r})$$

Quantification

$$\frac{A[t/x], \Gamma \vdash \Delta}{\forall x A, \Gamma \vdash \Delta}\ (\forall\text{:l}) \qquad \frac{A[y/x], \Gamma \vdash \Delta}{\exists x A, \Gamma \vdash \Delta}\ (\exists\text{:l}) \qquad \frac{\Gamma \vdash \Delta, A[y/x]}{\Gamma \vdash \Delta, \forall x A}\ (\forall\text{:r}) \qquad \frac{\Gamma \vdash \Delta, A[t/x]}{\Gamma \vdash \Delta, \exists x A}\ (\exists\text{:r})$$

The variable y must not occur free in Γ, Δ or A.

Equational

$$\frac{s = t, A[T/s], \Gamma \vdash \Delta}{s = t, A[T/t], \Gamma \vdash \Delta}\ (=\text{:l}) \qquad \frac{s = t, \Gamma \vdash \Delta, A[T/s]}{s = t, \Gamma \vdash \Delta, A[T/t]}\ (=\text{:r})$$

Each equation rule replaces an arbitrary number of occurrences of T.

Induction

$$\frac{\dfrac{\pi_1}{\mathcal{S}_1} \quad \dfrac{\pi_2}{\mathcal{S}_2} \quad \ldots \quad \dfrac{\pi_n}{\mathcal{S}_n}}{\Gamma_1, \ldots, \Gamma_n \vdash \Delta_1, \ldots, \Delta_n, F[t]}\ (\text{ind})$$

The induction rule applies to arbitrary algebraic data types. Let $c_1, \ldots, c_n$ be the constructors of a type and let k_i be the arity of c_i. Let $F[x]$ be a formula with x a free variable of the appropriate type. Then we call the sequent $\mathcal{S}_i \stackrel{\text{def}}{=} F[x_1], \ldots, F[x_{k_i}], \Gamma_i \vdash \Delta_i, F[c_i(x_1, \ldots, x_{k_i})]$ the i-th induction step.

Fig. 1. The sequent calculus LK.

Axioms

$$\frac{}{A \vdash A}\ (\text{ax}) \qquad \frac{}{\vdash \top}\ (\top)$$

Structural

$$\frac{\Gamma \vdash B}{A, \Gamma \vdash B}\ (\text{w}) \qquad \frac{A, A, \Gamma \vdash B}{A, \Gamma \vdash B}\ (\text{c})$$

Propositional

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ (\wedge\text{:e1}) \qquad \frac{\Gamma \vdash A \qquad \Pi \vdash B}{\Gamma, \Pi \vdash A \wedge B}\ (\wedge\text{:i})$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ (\wedge\text{:e2}) \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ (\vee\text{:i1})$$

$$\frac{\Gamma \vdash A \vee B \qquad \Pi, A \vdash C \qquad \Delta, B \vdash C}{\Gamma, \Pi, \Delta \vdash C}\ (\vee\text{:e}) \qquad \frac{\Gamma \vdash A}{\Gamma \vdash B \vee A}\ (\vee\text{:i2})$$

$$\frac{\Gamma \vdash \neg A \qquad \Pi \vdash A}{\Gamma, \Pi \vdash \bot}\ (\neg\text{:e}) \qquad \frac{A, \Gamma \vdash \bot}{\Gamma \vdash \neg A}\ (\neg\text{:i})$$

$$\frac{\Gamma \vdash A \Rightarrow B \qquad \Pi \vdash A}{\Gamma, \Pi \vdash B}\ (\Rightarrow\text{:e}) \qquad \frac{A, \Gamma \vdash B}{\Gamma \vdash A \Rightarrow B}\ (\Rightarrow\text{:i})$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ (\bot\text{:e})$$

Quantification

$$\frac{\Gamma \vdash \forall x A}{\Gamma \vdash A[t/x]}\ (\forall\text{:e}) \qquad \frac{\Gamma \vdash A[y/x]}{\Gamma \vdash \forall x A}\ (\forall\text{:i})$$

$$\frac{\Gamma \vdash \exists x A \qquad \Pi, A[y/x] \vdash B}{\Gamma, \Pi \vdash B}\ (\exists\text{:e}) \qquad \frac{\Gamma \vdash A[t/x]}{\Gamma \vdash \exists x A}\ (\exists\text{:i})$$

The variable y must not occur free in Γ, Π or B.

Equational

$$\frac{\Gamma \vdash s = t \qquad \Pi \vdash A[s/x]}{\Gamma, \Pi \vdash A[t/x]}\ (=\text{:e}) \qquad \frac{}{\vdash t = t}\ (=\text{:i})$$

Excluded Middle

$$\frac{\Gamma, A \vdash B \qquad \Pi, \neg A \vdash B}{\Gamma, \Pi \vdash B}\ (\text{em})$$

Induction

$$\frac{\overset{\pi_1}{\mathcal{S}_1} \qquad \overset{\pi_2}{\mathcal{S}_2} \qquad \dots \qquad \overset{\pi_n}{\mathcal{S}_n}}{\Gamma_1, \dots, \Gamma_n \vdash F[t]}\ (\text{ind})$$

The induction rule applies to arbitrary algebraic data types. Let $c_1, \dots, c_n$ be the constructors of a type and let k_i be the arity of c_i. Let $F[x]$ be a formula with x a free variable of the appropriate type. Then we call the sequent $\mathcal{S}_i \overset{\text{def}}{=} F[x_1], \dots, F[x_{k_i}], \Gamma_i \vdash F[c_i(x_1, \dots, x_{k_i})]$ the i-th induction step.

Fig. 2. Natural deduction ND.

3 Translating LK to ND

One of the main differences between our sequent calculus and natural deduction is that natural deduction allows just a single formula in the succedent. In order to translate LK to ND our translation thus focuses on one formula in the succedent of the conclusion of the LK proof. The focused formula forms the succedent of the obtained ND proof, whereas the other formulas are negated and move to the antecedent. If possible, our algorithm directly proves the focused formula in ND. Otherwise, it first *exchanges* the succedent formula of the ND proof with the needed main formula via the macro rule (ex) defined in Figure 3, and then proves it. The translation then proceeds recursively up the LK proof tree.

$$\frac{\Gamma, \neg A \vdash B}{\Gamma, \neg B \vdash A}\,(\mathrm{ex}_{\neg A})$$

(a) Exchange macro rule.

$$\dfrac{\dfrac{}{A \vdash A}\,(\mathrm{ax}) \qquad \dfrac{\dfrac{}{\neg B \vdash \neg B}\,(\mathrm{ax}) \qquad \Gamma, \neg A \vdash B}{\Gamma, \neg B, \neg A \vdash \bot}\,(\neg\mathrm{:e})}{\Gamma, \neg B \vdash A}\,(\bot\mathrm{:e})\,(\mathrm{em})$$

(b) Definition of the exchange rule.

Fig. 3. The exchange rule $(\mathrm{ex}_{\neg A})$ and its definition. If B equals $\bot$ in the premise of the rule, then (ex) simplifies to ($\bot$:e) followed by (em). We drop the subscript if the exchanged formula is clear from the context.

The translation function Tr takes two arguments. The first argument is an LK proof π with conclusion $\Gamma \vdash \Delta$. The second argument is either an arbitrary formula in the succedent Δ, say A, or $-$ if the succedent is empty and the argument is to be ignored. We call A the *focused formula*. Tr then produces an ND proof of the corresponding sequent, where the focused formula is the single formula in the succedent of the ND proof's conclusion:

$$\text{If } \dfrac{\pi}{\Gamma \vdash \Delta} \text{ with } A \in \Delta, \text{ then } \dfrac{\mathrm{Tr}(\pi, A)}{\Gamma, \neg(\Delta - \{A\}) \vdash A}$$

$$\text{and if } \dfrac{\pi}{\Gamma \vdash} \text{, then } \dfrac{\mathrm{Tr}(\pi, -)}{\Gamma \vdash \bot}\,.$$

We define Tr inductively on the structure of the proof. We perform a case distinction on the last rule of the LK proof and on which formula is focused. Before presenting the case analysis, we revisit our example and demonstrate Tr by applying it to the proof of double-negation elimination in Figure 4.

Inductive definition of the translation

Axioms: The axioms form the base cases:

$$\mathrm{Tr}\left(\dfrac{}{A \vdash A}\,(\mathrm{ax}), A\right) \stackrel{\text{def}}{=} \dfrac{}{A \vdash A}\,(\mathrm{ax}), \quad \mathrm{Tr}\left(\dfrac{}{\vdash \top}\,(\top), \top\right) \stackrel{\text{def}}{=} \dfrac{}{\vdash \top}\,(\top)$$

$$\mathrm{Tr}\left(\dfrac{}{\vdash t = t}\,(\mathrm{refl}), t = t\right) \stackrel{\text{def}}{=} \dfrac{}{\vdash t = t}\,(=\mathrm{:i}), \quad \mathrm{Tr}\left(\dfrac{}{\bot \vdash}\,(\bot), -\right) \stackrel{\text{def}}{=} \dfrac{}{\bot \vdash \bot}\,(\mathrm{ax})$$

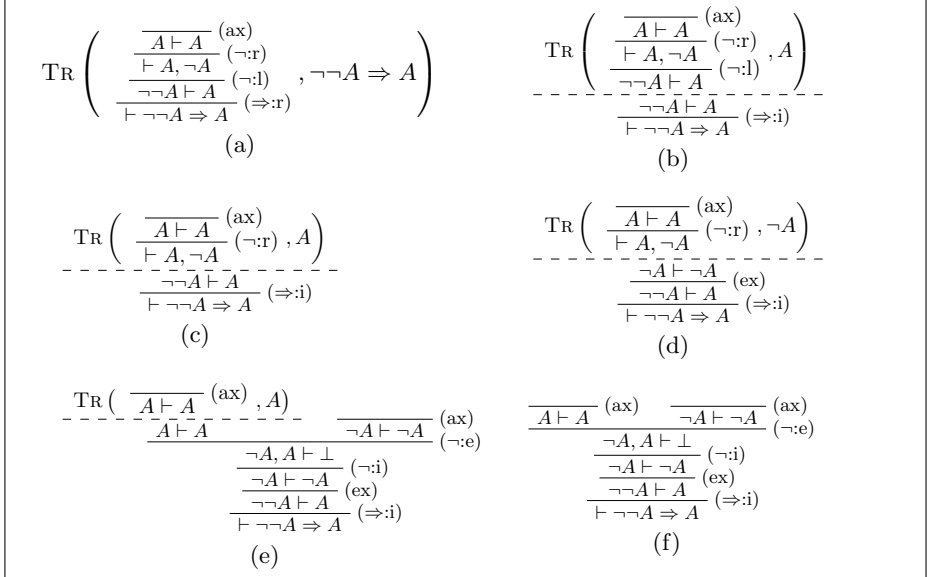

Fig. 4. In order to derive $\neg\neg A \Rightarrow A$, which is the only formula in the succedent of the input proof's conclusion, we set it as the focus (4a). We translate the (⇒:r)-rule using (⇒:i) and call Tr recursively on the proof with the conclusion removed and the new focused formula A (4b). The translation of (¬:l) drops the conclusion (4c). We notice that the focused formula A and the main formula of the last inference rule (¬:r), i.e., $\neg A$, do not match, so we change the focus to $\neg A$ by introducing an exchange inference (4d). Now that the main formula and the focused formula match, we translate (¬:r) using (ax), (¬:e), and (¬:i) (4e). Finally, the translation of the axiom is straightforward and we obtain an ND proof (4f). Notice that the obtained proof is not normalized (unlike the example proof in Section 2).

Exchange: For a right-inference rule (⊗:r) with $\otimes \in \{\mathrm{c}, \vee, \Rightarrow, \neg, \forall, \exists, =\}$, if the focused formula B is not the main formula A', we first derive A' by changing the focus to A' and add an exchange inference as follows:

$$\mathrm{Tr}\left(\dfrac{\overset{\pi}{\Gamma \vdash \Delta, A, B}}{\Gamma' \vdash \Delta, A', B}\,(\otimes\text{:r})\;, B\right) \overset{\text{def}}{=} \dfrac{\mathrm{Tr}\left(\dfrac{\overset{\pi}{\Gamma \vdash \Delta, A, B}}{\Gamma' \vdash \Delta, A', B}\,(\otimes\text{:r})\;, A'\right)}{\Gamma', \neg\Delta, \neg B \vdash A'}\,(\mathrm{ex})$$

We treat the binary rule (∧:r) similarly. Let's consider the case where an arbitrary formula C, i.e., not the main formula $A \wedge B$, coming from the right subproof

π_r, is focused (symmetric if C comes from π_l):

$$\textsc{Tr}\left(\dfrac{\dfrac{\overline{\Gamma \vdash \Delta, A}^{\pi_l} \quad \overline{\Sigma \vdash \Pi, B, C}^{\pi_r}}{\Gamma, \Sigma \vdash \Delta, \Pi, A \wedge B, C}\,(\wedge\text{:r})\ , C}\right) \stackrel{\text{def}}{=}$$

$$\dfrac{\textsc{Tr}\left(\dfrac{\dfrac{\overline{\Gamma \vdash \Delta, A}^{\pi_l} \quad \overline{\Sigma \vdash \Pi, B, C}^{\pi_r}}{\Gamma, \Sigma \vdash \Delta, \Pi, A \wedge B, C}\,(\wedge\text{:r})\ , A \wedge B}\right)}{\dfrac{\Gamma, \Sigma, \neg\Delta, \neg\Pi, \neg C \vdash A \wedge B}{\Gamma, \Sigma, \neg\Delta, \neg\Pi, \neg(A \wedge B) \vdash C}\,(\text{ex})}$$

We want to point out that the use of (ex) is often not necessary, even if the succedent contains multiple formulas—namely when the main formula is already in focus.

We now present the derivations for left-inferences with arbitrary formulas in focus, and right-inferences with the main formula in focus[4]. Tr maintains the focused formula between recursive calls and does not unnecessarily change the focus. When we have to make a choice, we pick the first formula in the succedent if it is non-empty, otherwise we set the focus to $-$. There is a choice when translating (w:r) (when the weakening formula is in focus) and when translating ($\vee$:r). We omit cases where the succedent is empty and the focus is to be ignored if it does not affect the structure of the translated proof.

Structural rules: The structural rules are translated as follows:

$$\textsc{Tr}\left(\dfrac{\overline{\Gamma \vdash \Delta, B}^{\pi}}{A, \Gamma \vdash \Delta, B}\,(\text{w:l})\ , B\right) \stackrel{\text{def}}{=} \dfrac{\textsc{Tr}(\pi, B)}{A, \Gamma, \neg\Delta \vdash B}\,(\text{w})$$

$$\textsc{Tr}\left(\dfrac{\overline{\Gamma \vdash \Delta, A}^{\pi}}{\Gamma \vdash \Delta, A, B}\,(\text{w:r})\ , B\right) \stackrel{\text{def}}{=} \dfrac{\dfrac{\textsc{Tr}(\pi, A)}{\neg B, \Gamma, \neg\Delta \vdash A}\,(\text{w})}{\neg A, \Gamma, \neg\Delta \vdash B}\,(\text{ex})$$

$$\textsc{Tr}\left(\dfrac{\overline{\Gamma \vdash \Delta, A}^{\pi}}{\Gamma \vdash \Delta, A, B}\,(\text{w:r})\ , A\right) \stackrel{\text{def}}{=} \dfrac{\textsc{Tr}(\pi, A)}{\neg B, \Gamma, \neg\Delta \vdash A}\,(\text{w})$$

$$\textsc{Tr}\left(\dfrac{\overline{A, A, \Gamma \vdash \Delta, B}^{\pi}}{A, \Gamma \vdash \Delta, B}\,(\text{c:l})\ , B\right) \stackrel{\text{def}}{=} \dfrac{\textsc{Tr}(\pi, B)}{A, A, \Gamma, \neg\Delta \vdash B}\,(\text{c})$$

$$\textsc{Tr}\left(\dfrac{\overline{\Gamma \vdash \Delta, B, B}^{\pi}}{\Gamma \vdash \Delta, B}\,(\text{c:r})\ , B\right) \stackrel{\text{def}}{=} \dfrac{\dfrac{}{B \vdash B}\,(\text{ax}) \quad \dfrac{\textsc{Tr}(\pi, B)}{\neg B, \Gamma, \neg\Delta \vdash B}}{\Gamma, \neg\Delta \vdash B}\,(\text{em})$$

Cut: The cut rule is translated as follows. The focus should be on a formula from the right side of the cut. If the focus is instead on a formula $D \in \Delta$ from

[4] The (w:r)-rule is a special case for which we also show the case where an arbitrary formula is in focus.

the left side of the cut, we add an exchange inference ($\mathrm{ex}_{\neg D}$) after ($\Rightarrow$:e).

$$\textsc{Tr}\left(\dfrac{\dfrac{\overline{\Gamma \vdash \Delta, A}^{\pi_l} \qquad \overline{A, \Sigma \vdash \Pi, B}^{\pi_r}}{\Gamma, \Sigma \vdash \Delta, \Pi, B}\,(\mathrm{cut})}{}\,, B\right) \overset{\text{def}}{=} \dfrac{\dfrac{\textsc{Tr}(\pi_l, A) \qquad \dfrac{\textsc{Tr}(\pi_r, B)}{A, \Sigma, \neg\Pi \vdash B}\,(\Rightarrow\text{:i})}{\Gamma, \neg\Delta \vdash A \qquad \Sigma, \neg\Pi \vdash A \Rightarrow B}\,(\Rightarrow\text{:e})}{\Gamma, \Sigma, \neg\Delta, \neg\Pi \vdash B}$$

Propositional rules: Next, we present the translation of the propositional rules, starting with negation. If B is absent in the translation of the ($\neg$:r)-rule (resulting in $A, \Gamma, \neg\Delta \vdash \bot$ being the conclusion of the translation of π) we can omit the (ax) and ($\neg$:e) inferences.

$$\textsc{Tr}\left(\dfrac{\overline{\Gamma \vdash A, \Delta}^{\pi}}{\neg A, \Gamma \vdash \Delta}\,(\neg\text{:l})\,, -\right) \overset{\text{def}}{=} \dfrac{\dfrac{}{\neg A \vdash \neg A}\,(\mathrm{ax}) \qquad \overline{\Gamma \vdash A, \Delta}^{\textsc{Tr}(\pi, A)}}{\neg A, \Gamma \vdash \bot}\,(\neg\text{:e})$$

$$\textsc{Tr}\left(\dfrac{\overline{\Gamma \vdash A, \Delta}^{\pi}}{\neg A, \Gamma \vdash \Delta, B}\,(\neg\text{:l})\,, B\right) \overset{\text{def}}{=} \overline{\neg A, \Gamma, \neg\Delta \vdash B}^{\textsc{Tr}(\pi, B)}$$

$$\textsc{Tr}\left(\dfrac{\overline{A, \Gamma \vdash \Delta, B}^{\pi}}{\Gamma \vdash \Delta, \neg A, B}\,(\neg\text{:r})\,, \neg A\right) \overset{\text{def}}{=} \dfrac{\overline{A, \Gamma, \neg\Delta \vdash B}^{\textsc{Tr}(\pi, B)} \qquad \dfrac{}{\neg B \vdash \neg B}\,(\mathrm{ax})}{\dfrac{\neg B, A, \Gamma, \neg\Delta \vdash \bot}{\neg B, \Gamma, \neg\Delta \vdash \neg A}\,(\neg\text{:i})}\,(\neg\text{:e})$$

The conjunction rules are translated as follows:

$$\textsc{Tr}\left(\dfrac{\overline{A, B, \Gamma \vdash \Delta, C}^{\pi}}{A \wedge B, \Gamma \vdash \Delta, C}\,(\wedge\text{:l})\,, C\right) \overset{\text{def}}{=}$$

$$\dfrac{\dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}\,(\mathrm{ax})}{A \wedge B \vdash B}\,(\wedge\text{:e2}) \qquad \dfrac{\dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}\,(\mathrm{ax})}{A \wedge B \vdash A}\,(\wedge\text{:e1}) \qquad \dfrac{\overline{A, B, \Gamma, \neg\Delta \vdash C}^{\textsc{Tr}(\pi, C)}}{B, \Gamma, \neg\Delta \vdash A \Rightarrow C}\,(\Rightarrow\text{:i})}{A \wedge B, B, \Gamma, \neg\Delta \vdash C}\,(\Rightarrow\text{:e})}{A \wedge B, \Gamma, \neg\Delta \vdash B \Rightarrow C}\,(\Rightarrow\text{:i})}{A \wedge B, A \wedge B, \Gamma, \neg\Delta \vdash C}\,(\Rightarrow\text{:e})}{A \wedge B, \Gamma, \neg\Delta \vdash C}\,(\mathrm{c})$$

$$\textsc{Tr}\left(\dfrac{\dfrac{\overline{\Gamma \vdash \Delta, A}^{\pi_l} \qquad \overline{\Sigma \vdash \Pi, B}^{\pi_r}}{\Gamma, \Sigma \vdash \Delta, \Pi, A \wedge B}\,(\wedge\text{:r})}{}\,, A \wedge B\right) \overset{\text{def}}{=} \dfrac{\dfrac{\textsc{Tr}(\pi_l, A) \qquad \textsc{Tr}(\pi_r, B)}{\Gamma, \neg\Delta \vdash A \qquad \Sigma, \neg\Pi \vdash B}\,(\wedge\text{:i})}{\Gamma, \Sigma, \neg\Delta, \neg\Pi \vdash A \wedge B}$$

The disjunction rules are handled next. If D is absent in the translation of the ($\vee$:l)-rule (resulting in $B, \Sigma, \neg\Pi \vdash \bot$ being the conclusion of the translation of π_r) we can omit the (ax) and ($\neg$:e) inferences in the right branch of the derivation. The case where the focus is on D instead of C is symmetric.

$$\textsc{Tr}\left(\dfrac{\dfrac{\overline{A, \Gamma \vdash \Delta, C}^{\pi_l} \qquad \overline{B, \Sigma \vdash \Pi, D}^{\pi_r}}{A \vee B, \Gamma, \Sigma \vdash \Delta, \Pi, C, D}\,(\vee\text{:l})}{}\,, C\right) \overset{\text{def}}{=}$$

$$\dfrac{\dfrac{}{A \vee B \vdash A \vee B}\,(\mathrm{ax}) \qquad \dfrac{\overline{A, \Gamma, \neg\Delta \vdash C}^{\textsc{Tr}(\pi_l, C)} \qquad \dfrac{\dfrac{\overline{B, \Sigma, \neg\Pi \vdash D}^{\textsc{Tr}(\pi_r, D)} \qquad \dfrac{}{\neg D \vdash \neg D}\,(\mathrm{ax})}{B, \neg D, \Sigma, \neg\Pi \vdash \bot}\,(\neg\text{:e})}{B, \neg D, \Sigma, \neg\Pi \vdash C}\,(\bot\text{:e})}{A \vee B, \neg D, \Gamma, \Sigma, \neg\Delta, \neg\Pi \vdash C}\,(\vee\text{:e})$$

$$\mathrm{Tr}\left(\dfrac{\overset{\pi}{\Gamma \vdash \Delta, A, B}}{\Gamma \vdash \Delta, A \vee B}\,(\vee{:}\mathrm{r})\,, A \vee B\right) \overset{\mathrm{def}}{=} \dfrac{\dfrac{\overline{A \vdash A}\,(\mathrm{ax})}{A \vdash A \vee B}\,(\vee{:}\mathrm{i1}) \qquad \dfrac{\overset{\mathrm{Tr}(\pi, B)}{\neg A, \Gamma, \neg\Delta \vdash B}}{\neg A, \Gamma, \neg\Delta \vdash A \vee B}\,(\vee{:}\mathrm{i2})}{\Gamma, \neg\Delta \vdash A \vee B}\,(\mathrm{em})$$

The implication rules are translated as follows. The translation of ($\Rightarrow$:l) works similar to the translation of the cut rule: the focus should be on a formula from the right subproof. If the focus is on a formula $D \in \Delta$ instead, we add an exchange inference ($\mathrm{ex}_{\neg D}$) after the bottom-most ($\Rightarrow$:e).

$$\mathrm{Tr}\left(\dfrac{\overset{\pi_l}{\Gamma \vdash \Delta, A} \qquad \overset{\pi_r}{B, \Sigma \vdash \Pi, C}}{A \Rightarrow B, \Gamma, \Sigma \vdash \Delta, \Pi, C}\,(\Rightarrow{:}\mathrm{l})\,, C\right) \overset{\mathrm{def}}{=}$$

$$\dfrac{\dfrac{\overline{A \Rightarrow B \vdash A \Rightarrow B}\,(\mathrm{ax}) \qquad \overset{\mathrm{Tr}(\pi_l, A)}{\Gamma, \neg\Delta \vdash A}}{A \Rightarrow B, \Gamma, \neg\Delta \vdash B}\,(\Rightarrow{:}\mathrm{e}) \qquad \dfrac{\overset{\mathrm{Tr}(\pi_r, C)}{B, \Sigma, \neg\Pi \vdash C}}{\Sigma, \neg\Pi \vdash B \Rightarrow C}\,(\Rightarrow{:}\mathrm{i})}{A \Rightarrow B, \Gamma, \Sigma, \neg\Delta, \neg\Pi \vdash C}\,(\Rightarrow{:}\mathrm{e})$$

$$\mathrm{Tr}\left(\dfrac{\overset{\pi}{A, \Gamma \vdash \Delta, B}}{\Gamma \vdash \Delta, A \Rightarrow B}\,(\Rightarrow{:}\mathrm{r})\,, A \Rightarrow B\right) \overset{\mathrm{def}}{=} \dfrac{\overset{\mathrm{Tr}(\pi, B)}{A, \Gamma, \neg\Delta \vdash B}}{\Gamma, \neg\Delta \vdash A \Rightarrow B}\,(\Rightarrow{:}\mathrm{i})$$

Quantifier rules: The quantifier rules are handled as follows:

$$\mathrm{Tr}\left(\dfrac{\overset{\pi}{A[t/x], \Gamma \vdash \Delta, B}}{\forall x A, \Gamma \vdash \Delta, B}\,(\forall{:}\mathrm{l})\,, B\right) \overset{\mathrm{def}}{=} \dfrac{\dfrac{\overset{\mathrm{Tr}(\pi, B)}{A[t/x], \Gamma, \neg\Delta \vdash B}}{\Gamma, \neg\Delta \vdash A[t/x] \Rightarrow B}\,(\Rightarrow{:}\mathrm{i}) \qquad \dfrac{\overline{\forall x A \vdash \forall x A}\,(\mathrm{ax})}{\forall x A \vdash A[t/x]}\,(\forall{:}\mathrm{e})}{\forall x A, \Gamma, \neg\Delta \vdash B}\,(\Rightarrow{:}\mathrm{e})$$

$$\mathrm{Tr}\left(\dfrac{\overset{\pi}{\Gamma \vdash \Delta, A[y/x]}}{\Gamma \vdash \Delta, \forall x A}\,(\forall{:}\mathrm{r})\,, \forall x A\right) \overset{\mathrm{def}}{=} \dfrac{\overset{\mathrm{Tr}(\pi, A[y/x])}{\Gamma, \neg\Delta \vdash A[y/x]}}{\Gamma, \neg\Delta \vdash \forall x A}\,(\forall{:}\mathrm{i})$$

$$\mathrm{Tr}\left(\dfrac{\overset{\pi}{A[y/x], \Gamma \vdash \Delta, B}}{\exists x A, \Gamma \vdash \Delta, B}\,(\exists{:}\mathrm{l})\,, B\right) \overset{\mathrm{def}}{=} \dfrac{\overline{\exists x A \vdash \exists x A}\,(\mathrm{ax}) \qquad \overset{\mathrm{Tr}(\pi, B)}{A[y/x], \Gamma, \neg\Delta \vdash B}}{\exists x A, \Gamma, \neg\Delta \vdash B}\,(\exists{:}\mathrm{e})$$

$$\mathrm{Tr}\left(\dfrac{\overset{\pi}{\Gamma \vdash \Delta, A[t/x]}}{\Gamma \vdash \Delta, \exists x A}\,(\exists{:}\mathrm{r})\,, \exists x A\right) \overset{\mathrm{def}}{=} \dfrac{\overset{\mathrm{Tr}(\pi, A[t/x])}{\Gamma, \neg\Delta \vdash A[t/x]}}{\Gamma, \neg\Delta \vdash \exists x A}\,(\exists{:}\mathrm{i})$$

Equational rules: The equational rules are translated as follows:

$$\mathrm{Tr}\left(\dfrac{\overset{\pi}{s = t, A[T/s], \Sigma \vdash \Pi, B}}{s = t, A[T/t], \Sigma \vdash \Pi, B}\,(=:\mathrm{l})\,, B\right) \overset{\mathrm{def}}{=}$$

$$\dfrac{\dfrac{\dfrac{\overline{A[T/t] \vdash A[T/t]}\,(\mathrm{ax}) \qquad \dfrac{\overline{s = t \vdash s = t}\,(\mathrm{ax}) \qquad \overline{\vdash s = s}\,(=:\mathrm{i})}{s = t \vdash t = s}\,(=:\mathrm{e})}{s = t, A[T/t] \vdash A[T/s]}\,(=:\mathrm{e}) \qquad \dfrac{\overset{\mathrm{Tr}(\pi, B)}{s = t, A[T/s], \Sigma, \neg\Pi \vdash B}}{s = t, \Sigma, \neg\Pi \vdash A[T/s] \Rightarrow B}\,(\Rightarrow{:}\mathrm{i})}{s = t, s = t, A[T/t], \Sigma, \neg\Pi \vdash B}\,(\Rightarrow{:}\mathrm{e})}{s = t, A[T/t], \Sigma, \neg\Pi \vdash B}\,(\mathrm{c})$$

$$\textsc{Tr}\left(\ \dfrac{\dfrac{\overset{\pi}{\overline{s=t,\Sigma\vdash\Pi,A[T/s]}}}{s=t,\Sigma\vdash\Pi,A[T/t]}\,(=\!:\mathrm{r})}\ ,A[T/t]\right)\overset{\text{def}}{=}$$

$$\dfrac{\dfrac{}{s=t\vdash s=t}\,(\mathrm{ax})\qquad\overset{\textsc{Tr}(\pi,A[T/s])}{\overline{s=t,\Sigma,\neg\Pi\vdash A[T/s]}}}{\dfrac{s=t,s=t,\Sigma,\neg\Pi\vdash A[T/t]}{s=t,\Sigma,\neg\Pi\vdash A[T/t]}\,(\mathrm{c})}\,(=\!:\mathrm{e})$$

Induction rule: The induction rule is translated as

$$\textsc{Tr}\left(\ \dfrac{\overset{\pi_1}{\overline{\mathcal{S}_1}}\qquad\overset{\pi_2}{\overline{\mathcal{S}_2}}\qquad\dots\qquad\overset{\pi_n}{\overline{\mathcal{S}_n}}}{\Gamma_1,\dots,\Gamma_n\vdash\Delta_1,\dots,\Delta_n,F[t]}\,(\mathrm{ind})\ ,F[t]\right)\overset{\text{def}}{=}$$

$$\dfrac{\overset{\textsc{Tr}(\pi_1,\mathcal{F}_1)}{\overline{\mathcal{S}'_1}}\qquad\overset{\textsc{Tr}(\pi_2,\mathcal{F}_2)}{\overline{\mathcal{S}'_2}}\qquad\dots\qquad\overset{\textsc{Tr}(\pi_n,\mathcal{F}_n)}{\overline{\mathcal{S}'_n}}}{\Gamma_1,\dots,\Gamma_n,\neg\Delta_1,\dots,\neg\Delta_n\vdash F[t]}\,(\mathrm{ind})$$

with

$$\mathcal{S}_i\overset{\text{def}}{=}F[x_1],\dots F[x_{k_i}],\Gamma_i,\vdash\Delta_i,F[c_i(x_1,\dots,x_{k_i})],$$
$$\mathcal{S}'_i\overset{\text{def}}{=}F[x_1],\dots F[x_{k_i}],\Gamma_i,\neg\Delta_i\vdash F[c_i(x_1,\dots,x_{k_i})],\text{and}$$
$$\mathcal{F}_i\overset{\text{def}}{=}F[c_i(x_1,\dots,x_{k_i})].$$

Lemma 1. Tr *terminates and is linear in the size of the* LK *proof.*

Proof. Tr either recurses and operates on a subproof with one less inference, or it recurses and changes the focused formula once and then operates on a smaller subproof. Hence Tr terminates. Each translation step is constant, hence Tr is linear in the size of the LK proof.

Optimizations: The translation of ($\vee$:r) seems inherently classical because the premise of the rule contains multiple formulas in the succedent. However, note that we can obtain an intuitionistic translation if the rule is preceded by a (w:r) inference, where the weakening adds one of the disjuncts. We make use of the following optimization, if the weakening immediately precedes the disjunction rule (symmetric if weakening with A instead of B):

$$\textsc{Tr}\left(\ \dfrac{\dfrac{\overset{\pi}{\overline{\Gamma\vdash\Delta,A}}}{\Gamma\vdash\Delta,A,B}\,(\mathrm{w})}{\Gamma\vdash\Delta,A\vee B}\,(\vee\!:\mathrm{r})\ ,A\vee B\right)\overset{\text{def}}{=}\ \dfrac{\overset{\textsc{Tr}(\pi,A)}{\overline{\Gamma,\neg\Delta\vdash A}}}{\Gamma,\neg\Delta\vdash A\vee B}\,(\vee\!:\mathrm{i1})$$

Let us revisit the example in Figure 4: Notice that in the intermediate step in Figure 4d, the proof could be completed by inferring $\neg A\vdash\neg A$ via (ax). The naïve translation, as we presented it, instead continues and introduces an indirection into the ND proof. We make use of an optimization that catches such cases and stops the translation early by closing the proof with (ax): We check if the conclusion of the input proof is a sequent of the form $\vdash A,\neg A$ with focus

$\neg A$. If this is the case we simply return $\frac{}{\neg A \vdash \neg A}\,(\text{ax})$. The example proof when using the optimization is then translated as

$$\dfrac{\dfrac{\dfrac{}{\neg A \vdash \neg A}\,(\text{ax})}{\neg\neg A \vdash A}\,(\text{ex})}{\vdash \neg\neg A \Rightarrow A}\,(\Rightarrow\text{:i})$$

which, after inlining the (ex) macro rule, is equivalent to the one presented in Section 2.

Translating constructive proofs: We want to point out that constructive LK proofs are translated into constructive ND proofs—where constructive proofs are proofs in intuitionistic logic. The constructive subset of LK that we consider is essentially Gentzen's calculus LJ. The only difference is the $(\vee\text{:}r)$-rule, which in our calculus always has two auxiliary formulas in the succedent, and hence requires special treatment. Other constructive subsets of LK, for example multi-succedent calculi such as Maehara's L'J [12], are not always translated into constructive ND proofs.

Definition 1. *An LK proof is constructive if every sequent in the proof contains only a single formula in the succedent. The* $(\vee\text{:r})$*-rule is an exception, which we consider to be constructive if it is immediately preceded by a weakening with one of the disjuncts.*

Definition 2. *An ND proof is constructive if it does not contain an* (em) *inference.*

Lemma 2. Tr *translates constructive sequent calculus proofs into constructive natural deduction proofs.*

Proof. We analyze the cases which lead to introduction of an (ex) or (em) inference and show that they cannot occur: (i) The (c:r)-rule does not occur in a constructive proof. (ii) The (w:r) and $(\vee\text{:r})$ rules only occur together in constructive proofs and the optimization we described takes care of them. (iii) In $(\Rightarrow\text{:l})$ and (cut), Δ must be empty as we are dealing with single-succedent proofs, therefore the case where $D \in \Delta$ is the focus cannot occur. (iv) Finally, the focus never needs to be changed because we are dealing with single-succedent proofs.

4 Experimental evaluation

4.1 Implementation and proof import

The translation described in Section 3 is implemented in our open source[5] library for proof transformations, GAPT [8] version 2.10. Since our ultimate goal is to extract programs from proofs generated by automated theorem provers, we evaluated the translation on such automatically generated proofs. GAPT can

[5] available at `https://logic.at/gapt`

interface with more than half a dozen classical first-order provers, including state-of-the art tools such as Vampire, E, and SPASS.

The resulting proofs are imported in a multi-step process. Proof output from the external provers is parsed and reconstructed using proof replay to obtain resolution proofs. As the next step, we convert the resolution proofs to expansion proofs [13], which store only the quantifier instances of the proof (as used in the ($\exists$:r), ($\forall$:l), ($\exists$:i), and ($\forall$:e) inferences) and ignore the propositional parts of the proof. This conversion [6] also eliminates inferences used for splitting as used by Avatar in Vampire [18] or in SPASS [19] and subformula definitions introduced by our clausification.

We use expansion proofs as an intermediate step for two reasons: first, they "clean up" the proofs considerably. Proofs produced by first-order provers primarily record the proof search, and are not intended to be short, beautiful, or even constructive. Proof replay then adds additional complexity, in particular the reconstruction of a single equational inference in the prover output often produces several inferences in GAPT. All of this superfluous complexity is completely ignored when passing to expansion proofs.

More importantly however, classical first-order provers typically use Skolemization as a preprocessing step. The freshly introduced Skolem functions are then treated like any other function, and also equational inferences can rewrite the arguments of the Skolem functions. In intuitionistic logic, Skolemization does not always produce equivalid formulas: for example, the intuitionistic non-theorem $(\neg\forall x\ P(x)) \to \exists x\ \neg P(x)$ is turned into the theorem $(\neg P(c)) \to \exists x\ \neg P(x)$ via Skolemization. Given that we want to minimize the amount of non-constructive reasoning, it is therefore of paramount importance to forgo Skolemization in this sense. (It would of course also be possible to impose restrictions on the use of Skolem terms so that they behave essentially like eigenvariables. However these restrictions are typically not respected by the proofs produced by classical first-order provers. Hence either approach requires a transformation ensuring that these restrictions are respected.) Expansion proofs allow us to straightforwardly eliminate Skolem functions from first-order proofs using a replacement operation on the instance terms [2]. This deskolemization procedure can theoretically fail on proofs that use equational inferences to rewrite the arguments of Skolem functions. As we will show in this section, this happens only rarely in practice. We are currently investigating a preprocessing step to eliminate congruences for Skolem functions that will extend this proof deskolemization to all proofs with equational inferences.

The expansion proofs are then converted to proofs in the sequent calculus LK via a simple tableaux prover that reproduces the propositional parts of the proof missing from the expansion proof. This proof import method described up to now is not specific to the translation to natural deduction described in this paper, but is in parts independently used by many other applications implemented and evaluated in GAPT, such as cut-introduction [7], inductive theorem proving [5], cut-elimination by resolution [3], and others.

For our translation to natural deduction, we modify the conversion from expansion proofs to proofs in LK slightly using a simple heuristic: usually we want to apply the (¬:l)-rule as soon as possible (that is, close to the bottom) since it is an invertible unary inference rule that simplifies the sequent. However, it is not invertible intuitionistically—the premise may be unprovable even though the conclusion would be an intuitionistic theorem. We hence try the (¬:l)-rule last.

4.2 Large-scale tests

For the empirical evaluation of our translation, we took the 662 problems in the first-order `FEQ`, `FNE`, `FNN`, and `FNQ` divisions of the CASC-26 competition whose size was less than one megabyte after including the separate axiom files. On these 662 problems, we used the translation on proofs imported from the E theorem prover[6] [16] version 2.1 (as submitted to CASC).

We set a total time limit of 5 minutes for each of these problems, and 2 minutes for the first-order prover. Of the 662 problems, GAPT fails to clausify 15 of the problems due to excessive runtime. (These problems—e.g. `HWV053+1`—have blocks of more than a thousand quantifiers.) On the remaining 647 problems, E successfully returns 291 proofs. Using proof replay, GAPT reconstructs 285 resolution proofs, from there we get 283 expansion proofs, and then 261 proofs in LK. The translation to natural deduction finally results in 261 proofs. (These proofs do not contain induction inferences since E is a first-order theorem prover.)

From a performance point of view, the runtime of the translation in Section 3 is negligible compared to the rest of the proof import. Figure 5 shows the relative runtime of the different proof import phases described in Section 4.1. The translation to natural deduction only makes up 0.65% of the total proof import time, making it practically feasible to obtain natural deduction proofs whenever necessary.

Parser	CNF	E	replay	→ exp.	deskolem.	→ LK	→ ND
2.95%	3.58%	50.49%	12.50%	4.28%	7.89%	17.66%	0.65%

Fig. 5. Relative runtime of phases during proof import to natural deduction.

One of the main goals of the translation is to minimize the amount of non-constructive reasoning in the produced proofs, that is, the number of the inferences for excluded middle as well as the complexity of their auxiliary formulas. Of the 261 natural deduction proofs produced, 154 (59%) do not contain any use of the excluded middle, and 224 (85%) do not contain *quantified* excluded middle, i.e., excluded middle on a formula that contains quantifiers. For applications in program extraction, excluded middle on quantifier-free formulas is

[6] We picked the highest-ranking prover in the first-order theorem category of the CASC-26 competition whose license allows competitive evaluation.

typically not a problem since quantifier-free formulas are often decidable. Hence, it is important to consider the number of quantified excluded middle inferences. Figure 6 shows the average number of inferences for excluded middle grouped by the TPTP category of the problem. The problems in most categories require little use of the excluded middle (em), except for SYO, which contains syntactic problems that have no obvious semantic interpretation.

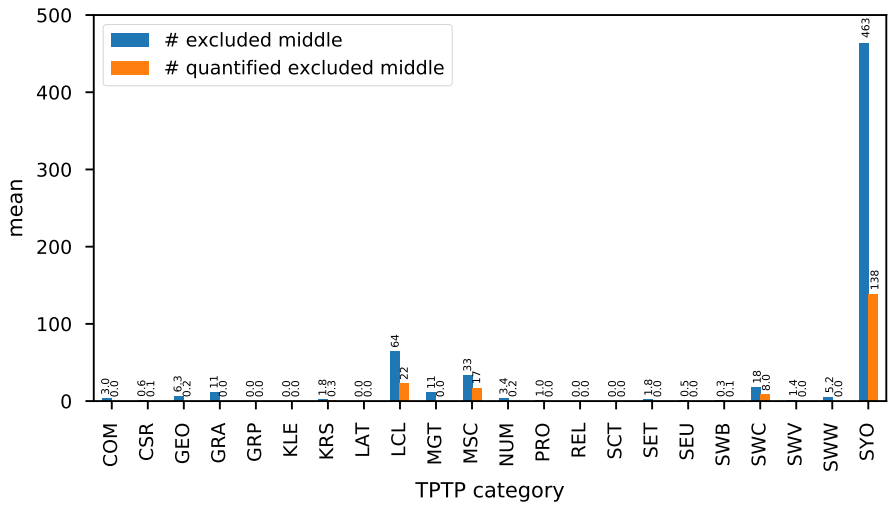

Fig. 6. Mean number of excluded middle inferences, grouped by TPTP category.

We can also interpret the quantitative results on excluded middle inferences from the point of view of proof constructivization. If there are no uses of excluded middle in the produced proof, then we have successfully converted an a priori classical proof into an intuitionistic proof. Automated conversion of proofs produced by classical first-order theorem provers into intuitionistic logic has been studied before using the Dedukti system and the Zenon prover: using a rewriting system on natural deduction proofs [4], the authors obtain a constructivization rate of 61.8% on proofs produced by Zenon. Another approach converts sequent calculus proofs in LK to the intuitionistic sequent calculus LJ [10] using a focusing strategy similar to ours. They report a constructivization rate of 85%, again on proofs produced by Zenon. We believe that we observe a lower constructivization here due to the choice of the classical prover and the resulting different problem selection. The produced proofs are likely to differ significantly, since Zenon is a tableaux prover, and E is a superposition prover. Additionally, E can prove many more problems and find more complicated proofs, which may be harder to constructivize.

A similar observation on the constructivity of classical proofs has been made with the intuitionistic first-order theorem prover ileanCoP [14]. This prover first searches for a classical connection proof and then checks whether this proof is intuitionistic, backtracking otherwise to guarantee completeness. In the evaluation on TPTP problems ileanCoP found 188 proofs, of which 178 (94.6%) have been proved without backtracking, which means that the original classical proof was already intuitionistic modulo reordering of inferences.

Our proof deskolemization approach can fail if there are equational inferences that rewrite the arguments of a Skolem function. This only happens three times in the 283 expansion proofs considered here. The corresponding TPTP problems are `GEO084+1`, `NUM855+2`, and `PRO004+1`. It is not easy to pinpoint the failure to a single responsible inference in the proof output, since the input of the deskolemization algorithm is far away from the resolution proof and we only know that the deskolemized deep formula is not a tautology modulo equality.

In Section 3, we introduced a special case for ($\vee$:r) inferences that are directly following a (w:r) inference. Among the 802 ($\vee$:r) inferences in total, 349 (43.5%) directly follow a (w:r) inference, making this a worthwhile optimization. On average, the produced proofs in natural deduction contain 2.9 times as many inferences as the proofs in LK.

5 Conclusion

We have presented a simple translation of the sequent calculus LK to natural deduction. It is efficient and its cost is negligible compared to other processing steps. The produced proofs have few excluded middle inferences, and a large part of them are quantifier-free. A disadvantage of this translation is that it does not produce normal proofs in natural deduction. In particular the left-inferences in LK introduce redexes. We plan to address this issue by either normalizing the proofs in natural deduction, or simplifying the extracted programs.

As the next step we want to use the proofs generated by this translation in program extraction, and program synthesis using proofs generated by automated theorem provers.

References

1. Federico Aschieri and Margherita Zorzi. On natural deduction in classical first-order logic: Curry-Howard correspondence, strong normalization and Herbrand's theorem. *Theoretical Computer Science*, 625:125–146, 2016.
2. Matthias Baaz, Stefan Hetzl, and Daniel Weller. On the complexity of proof deskolemization. *Journal of Symbolic Logic*, 77(2):669–686, 2012.
3. Matthias Baaz and Alexander Leitsch. Cut-elimination and redundancy-elimination by resolution. *Journal of Symbolic Computation*, 29(2):149–177, 2000.
4. Raphaël Cauderlier. A rewrite system for proof constructivization. In Gilles Dowek, Daniel R. Licata, and Sandra Alves, editors, *Eleventh Workshop on Logical Frameworks and Meta-Languages: Theory and Practice, LFMTP*, pages 2:1–2:7. ACM, 2016.

5. Sebastian Eberhard and Stefan Hetzl. Inductive theorem proving based on tree grammars. *Annals of Pure and Applied Logic*, 166(6):665–700, 2015.
6. Gabriel Ebner. Extracting expansion trees from resolution proofs with splitting and definitions. 2018. Preprint available at `https://gebner.org/pdfs/2018-01-29_etimport.pdf`.
7. Gabriel Ebner, Stefan Hetzl, Alexander Leitsch, Giselle Reis, and Daniel Weller. On the generation of quantified lemmas. *Journal of Automated Reasoning*, pages 1–32, 2018.
8. Gabriel Ebner, Stefan Hetzl, Giselle Reis, Martin Riener, Simon Wolfsteiner, and Sebastian Zivota. System description: GAPT 2.0. In Nicola Olivetti and Ashish Tiwari, editors, *International Joint Conference on Automated Reasoning (IJCAR)*, volume 9706 of *Lecture Notes in Computer Science*, pages 293–301. Springer, 2016.
9. Gerhard Gentzen. Untersuchungen über das logische Schließen. II. *Mathematische Zeitschrift*, 39(1):405–431, 1935.
10. Frédéric Gilbert. Automated constructivization of proofs. In Javier Esparza and Andrzej S. Murawski, editors, *Foundations of Software Science and Computation Structures, FOSSACS*, volume 10203 of *Lecture Notes in Computer Science*, pages 480–495, 2017.
11. Georg Kreisel. Interpretation of analysis by means of constructive functionals of finite types. In Arend Heyting, editor, *Constructivity in Mathematics*, pages 101–128. Amsterdam: North-Holland Publishing Company, 1959.
12. Shôji Maehara. Eine Darstellung der intuitionistischen Logik in der Klassischen. *Nagoya Mathematical Journal*, 7:45–64, 1954.
13. Dale A. Miller. A compact representation of proofs. *Studia Logica*, 46(4):347–370, 1987.
14. Jens Otten. Clausal connection-based theorem proving in intuitionistic first-order logic. In Bernhard Beckert, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX*, volume 3702 of *Lecture Notes in Computer Science*, pages 245–261. Springer, 2005.
15. Michel Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In Andrei Voronkov, editor, *Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer, 1992.
16. Stephan Schulz. System Description: E 1.8. In Ken McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *19th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 8312 of *LNCS*. Springer, 2013.
17. Anne S. Troelstra and Helmut Schwichtenberg. *Basic Proof Theory*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2000.
18. Andrei Voronkov. AVATAR: the architecture for first-order theorem provers. In Armin Biere and Roderick Bloem, editors, *26^{th} International Conference on Computer Aided Verification, CAV 2014*, volume 8559 of *Lecture Notes in Computer Science*, pages 696–710. Springer, 2014.
19. Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischnewski. SPASS version 3.5. In Renate A. Schmidt, editor, *22nd International Conference on Automated Deduction (CADE)*, volume 5663 of *Lecture Notes in Computer Science*, pages 140–145. Springer, 2009.
20. J. Zucker. The correspondence between cut-elimination and normalization. *Annals of Mathematical Logic*, 7(1):1–112, 1974.