No Choice: Reconstruction of First-order ATP Proofs without Skolem Functions

Michael Färber
michael.faerber@uibk.ac.at

Cezary Kaliszyk
cezary.kaliszyk@uibk.ac.at

Universität Innsbruck
Innsbruck, Austria

Abstract

Proof assistants based on higher-order logic frequently use first-order automated theorem provers as proof search mechanisms. The reconstruction of the proofs generated by common tools, such as MESON and Metis, typically involves the use of the axiom of choice to simulate the Skolemisation steps. In this paper we present a method to reconstruct the proofs without introducing Skolem functions. This enables us to integrate tactics that use first-order automated theorem provers in logics that feature neither the axiom of choice nor the definite description operator.

1 Introduction

Many first-order automated theorem provers (ATPs) operate on formulae with implicitly universally quantified variables. To find proofs with such ATPs for formulae containing existential quantifiers, it is necessary to transform the original problems: Skolemisation replaces existentially quantified variables with fresh function symbols that depend at least on all universally quantified variables in the subformula, i.e. $\exists x.t(x, y_1, \ldots, y_n)$ (t being a term) is replaced by $t(f(y_1, \ldots, y_n), y_1, \ldots, y_n)$, where f is a fresh function symbol. In higher-order foundations it is possible to express that a problem is satisfiable iff its Skolemised version is, by existentially quantifying over the Skolem functions. Proving or even stating equisatisfiability cannot be done in first-order logic.

With the increasing interest in Isabelle [NPW02] object logics based on first-order logic, such as Isabelle/ZF [Pau93] or Kaliszyk's Mizar environment [KPU16], it is natural to provide built-in first-order automated theorem proving methods for these logics. Tools including Metis [Hur03] and MESON [Har96] have provided such methods for Isabelle/HOL. However, as the integration of these tools currently relies on higher-order features, they cannot easily be used in Isabelle/FOL. Proof methods that support also FOL, such as *blast* [Pau99], work very well as a human proof search mechanism, but are often insufficient to reconstruct deeper proofs [BKPU16].

We propose a new method to integrate first-order ATPs in interactive proof systems based on first-order logic. The technique can be used with first-order provers that take a set of implicitly all-quantified formulae as input, and whose proofs can be expressed as a set of formula copies together with the instantiations of the variables, and a natural deduction proof on the formula copies that shows $\bot$. This includes the most common first-order calculi such as resolution, paramodulation (superposition), and tableaux. The method uses the information contained in the Skolem terms to derive an ordering on instantiations, as opposed to the methods existing in Isabelle/HOL and other HOL provers which recreate the Skolem functions directly in the higher-order logic. To show the practical feasibility of the approach, we integrate a first-order tableaux prover in Isabelle/FOL.

In: P. Fontaine, S. Schulz, J. Urban (eds.): Proceedings of the 5th Workshop on Practical Aspects of Automated Reasoning (PAAR 2016), Coimbra, Portugal, 02-07-2016, published at http://ceur-ws.org

Contents: In section 2, we recall several basic concepts such as normal forms and Skolemisation. In section 3, we present our method to construct a proof without Skolem functions from a proof with Skolem functions. In section 4, we describe the implementation of our method as part of a larger proof search tactic for Isabelle. We show in section 5 the limitations of our approach. Section 6 discusses the related work, and in section 7, we conclude.

2 Preliminaries

We distinguish between *quantifier logic* and *quantifier-free logic*, where the former one is a logic with and the latter one without any quantifiers, meaning that all variables are implicitly all-quantified. Typically, in a proof assistant, we are given a problem in quantifier logic, of which we create an equisatisfiable problem for an ATP in quantifier-free logic by Skolemisation.

For both logics we assume the existence of three disjoint sets: variables, functions, and predicates, where every function and predicate has a fixed arity. Constants are functions with arity 0. Terms, atoms, literals, and formulae are defined in the usual way. Implication is right-associative, i.e. $a \implies b \implies c$ is interpreted as $a \implies (b \implies c)$.

Definition 1 (Substitution) *A substitution is a function σ from variables to terms under the condition that the fix point of σ exists, i.e. the substitution is non-circular.*

We naturally extend substitutions to structures containing variables, such as terms and formulae.

Definition 2 (Normal forms) *A formula is in negation normal form (NNF) iff negations are only applied to atoms and the formula does not contain any implications. A formula is in prenex normal form (PNF) iff it has the shape $Q^*.P$, where $Q \in \{\exists, \forall\}$ is a quantifier and P is a quantifier-free formula.*

For every first-order formula, we can find equivalent formulae in NNF and PNF.

To replace existential quantifiers in logic formulae, a frequently used method is Skolemisation. We will focus only on outer Skolemisation [NW01].

Definition 3 (Skolemisation) *A single outer Skolemisation step of a formula t is*

$$Sk_1(t) = \begin{cases} \forall x_1, \ldots, x_n.P[y := f(x_1, \ldots, x_n)] & \text{if } t = \forall x_1, \ldots, x_n.\exists y.P \\ t & \text{otherwise,} \end{cases}$$

where f is a fresh function symbol. The final Skolemisation $\mathrm{Sk}(t)$ is the fixpoint of the single-step Skolemisation $\mathrm{Sk}_1(t)$. We call the set of fresh functions Skolem functions, and an application of a Skolem function a Skolem term.

3 Proof deskolemisation

Deskolemisation is the process of creating formulae with existential quantifiers from a formula with Skolem functions. In a similar way, proof deskolemisation is about creating a Skolem-free version of a proof with Skolem functions.

Consider the following scenario: We are given a problem as a set of formulae with quantifiers, which we want to use to produce a proof of $\bot$. To pass the formulae to an ATP, we first convert them to PNF + NNF, yielding a set of formulae F that can be shown without the axiom of choice to be equivalent to the original set of formulae. Then, we Skolemize F and omit the universal quantifier prefixes, thus arriving at a set of formulae without quantifiers. The ATP might return a proof of $\bot$. Without loss of generality, we assume an ATP proof to be a set of formulae F' consisting of arbitrarily many copies of the input formulae with disjoint variables, a substitution σ from variables in F' to terms, and a natural deduction proof that uses $\sigma(F')$ to show $\bot$. We can easily represent many proof types, such as tableaux or resolution, in this format.

As the substitution (as well as the natural deduction proof) may contain references to the Skolem functions, which we do not introduce in quantifier logic, we want to eliminate such references.

The procedure consists of two steps: First, using the substitution σ, we create quantifier-free instances of the formulae F in the quantifier logic. Next, the natural deduction proof from the ATP is converted to a proof in the quantifier logic, which is then performed on the quantifier-free instances of F.

Table 1: Skolemisation of $\exists x \forall y.P(x,y) \implies \forall z \exists w.\neg P(z,w) \implies \bot$.

#	Quantifier formula	Quantifier-free formula
t_1	$\exists x \forall y.P(x,y)$	$P(a,y)$
t_2	$\forall z \exists w.\neg P(z,w)$	$\neg P(z,f(z))$

Consider the problem $\exists x \forall y.P(x,y) \implies \forall z \exists w.\neg P(z,w) \implies \bot$. Its quantifier-free version is shown in Table 1. Assume that the ATP finds a proof of the problem, which contains only one step, namely the resolution of t_1 with t_2, yielding a substitution $\sigma = \{y \mapsto f(z), z \mapsto a\}$[1]. The natural deduction proof of the quantifier-free formulae could look as follows:

1	$\neg P(a,f(a))$	
2	$P(a,f(a))$	
3	$\bot$	$\neg$E, 1, 2

In order to recreate the proof in quantifier logic, we instantiate the quantifier logic formulae. For this, formulae cannot be simply processed in sequence, but the dependencies between the formulae need to be resolved switching between the formulae until all quantifiers have been instantiated. The order is determined by the substitution: In this example, we cannot immediately instantiate t_2, because z depends on a which is a Skolem constant for which we have not created an eigenvariable yet. However, we can eliminate the outermost existential quantifier of t_1, yielding an eigenvariable a for x and the new formula $t_3 = \forall y.P(a,y)$. In the second step, we cannot instantiate the new formula t_3, because y depends on $f(z)$, which is a Skolem term for which we have not obtained an eigenvariable yet. However, t_2 can now be instantiated, because we have previously retrieved the eigenvariable a. This yields $t_4 = \exists w.\neg P(a,w)$. In a similar fashion, we can now obtain a new eigenvariable f_a from t_4, yielding $t_5 = \neg P(a,f_a)$, followed by an instantiation of t_3, giving us the last formula $t_6 = P(a,f_a)$. The first-order resolution step can now be performed on t_5 and t_6, concluding the proof:

1	$\exists x.\forall y.P(x,y)$		
2	$\forall z.\exists w.\neg P(z,w)$		
3	a	$\forall y.P(a,y)$	
4		$\exists w.\neg P(a,w)$	$\forall$E, 2
5		f_a	$\neg P(a,f_a)$
6			$P(a,f_a)$ $\quad \forall$E, 3
7			$\bot$ $\quad \neg$E, 5, 6
8		$\bot$	$\exists$E, 4, 5–7
9	$\bot$		$\exists$E, 1, 3–8

In the next subsections we will present the steps of the process.

3.1 Fixing unsubstituted variables

A variable x with $\sigma(x) = x$ corresponds to a universal quantifier that is instantiated with a fresh variable. For example, given the formulae $\forall x.P(x)$ and $\forall y.\neg P(y)$, a possible substitution obtained from a first-order proof is $\sigma = \{x \mapsto y\}$. To prove $\bot$, we need to instantiate $\forall y.\neg P(y)$ with an eigenvariable, say y', yielding $\neg P(y')$.

This also treats the case where a Skolem term contains a variable x that is not substituted by some $\sigma(x) \neq x$. As an example, take $\forall xy.P(x,y)$ and $\forall z.\exists w.\neg P(z,w)$, respectively its Skolemised version $\forall z.\neg P(z,f(z))$. A proof

[1] Different substitutions are admissible for this example, for example $\{y \mapsto f(a), z \mapsto a\}$.

$$\frac{}{\{\},\ M,\ Path} \quad Axiom$$

$$\frac{C,\ M,\ \{\}}{M} \quad Start \quad \text{where } C \in M, C \text{ is positive}$$

$$\frac{C,\ M,\ Path \cup \{L_2\}}{C \cup \{L_1\},\ M,\ Path \cup \{L_2\}} \quad Reduction \quad \text{where } \sigma(L_1) = \sigma(\overline{L_2})$$

$$\frac{C_2 \setminus \{L_2\},\ M,\ Path \cup \{L_1\} \qquad C,\ M,\ Path}{C \cup \{L_1\},\ M,\ Path} \quad Extension \quad \text{where } \begin{array}{c} \sigma(L_1) = \sigma(\overline{L_2}), \\ \sigma \text{ is rigid}, \\ C_1 \in M, L_2 \in C_2, \\ C_2 \text{ is a copy of } C_1 \\ \text{with variables renamed} \end{array}$$

Figure 1: The clause connection calculus used in leanCoP.

might contain the substitution $\{x \mapsto z, y \mapsto f(z)\}$, where z is an argument of the Skolem term. However, this does not create problems, because $\sigma(z) = z$, so by the previous paragraph, z will be instantiated by a fresh eigenvariable.

3.2 Instantiating quantified formulae

The instantiation algorithm determines a sequence of quantifier eliminations, yielding quantifier-free formulae. Special care is taken to produce different eigenvariables only for different Skolem terms, i.e. Skolem functions applied to arguments that are not convertible with respect to the substitution. This is necessary for correctness of the procedure. Conversely, we reuse existing eigenvariables for existentially quantified variables when all precedent universal quantifiers were instantiated with equivalent terms. To that end, we find common prefixes of quantifier instantiations and instantiate these common prefixes only once. Assuming the substitution is non-circular, it is always possible to find a sequence of quantifier eliminations that respects the substitution, which follows from the non-circularity of the substitution.

3.3 Proof lifting

The natural deduction proof in the quantifier-free logic with Skolem functions is lifted to a proof in the quantifier logic. For this, every used instance of a quantifier-free formula is substituted by the instantiated version in the quantifier logic, and Skolem terms are mapped to appropriate eigenvariables that were obtained in the instantiation phase. In the end, one obtains a proof of $\bot$ in the quantifier logic.

4 Implementation

We implemented the technique presented in section 3 as part of the integration of a first-order tableaux prover in the Isabelle object logic FOL. The developed proof method *IsaCoP*[2] integrates the ML version of the tableaux prover leanCoP. In this section, we will give a short introduction to the prover and present its integration in detail.

4.1 Tableau prover

For proof search we use the core reasoning procedure of the first-order tableaux prover leanCoP introduced by Otten and Bibel [OB03, Ott08]. We translated it to Standard ML, based on a previous translation of Kaliszyk to OCaml for HOL Light [KUV15].

leanCoP was chosen due to its very simple calculus (shown in Figure 1) which makes it easy to reconstruct proofs once Skolem functions have been treated.

[2]The source code of IsaCoP is available under http://cl-informatik.uibk.ac.at/users/mfaerber/tactics.html.

4.2 Equality axioms

The core reasoning procedure of leanCoP does not have an inbuilt notion of equality, however the full version of leanCoP supports equality, by inserting equality axioms into the problem. For that reason, before sending the quantifier logic problem to leanCoP, we prove the following formulae and add them to the original problem:

- reflexivity of equality,
- transitivity of equality, and
- congruence axioms for every predicate P and every function f appearing in the problem (excluding Skolem functions and equality itself), such as:

$$x_1 = y_1 \implies \ldots \implies x_n = y_n \implies P(x_1, \ldots, x_n) \implies P(y_1, \ldots, y_n),$$
$$x_1 = y_1 \implies \ldots \implies x_n = y_n \implies f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n).$$

4.3 Translation to clausal form

To use a refutation-based prover on an Isabelle goal

$$A_1 \implies \ldots \implies A_n \implies C,$$

it is necessary to first negate the conjecture. To achieve conjecture directed proof search, leanCoP marks the conjecture-related parts of the goal with a special symbol #:

$$\neg C \vee \neg\# \implies A_1 \implies \ldots \implies A_n \implies \# \implies \bot.$$

Furthermore, it is necessary to translate the goal to a normal form PNF + NNF + CNF. This is achieved using the Isabelle simplifier by rewriting with a fixed set of rules such as

$$\neg(a \wedge b) \leftrightarrow \neg a \vee \neg b,$$
$$(\forall x.P(x)) \vee Q \leftrightarrow \forall x.(P(x) \vee Q).$$

The conversion of the goal to normal form is performed as proof steps inside the Isabelle logic, meaning that every step is logically verified. In contrast, we do not perform Skolemisation inside the logic, because we do not assume the axiom of choice. Instead, we convert the normal form to an equisatisfiable quantifier-free term by introducing Skolem functions outside the logic. From the result, we extract the clauses from the quantifier-free formulae and pass them to leanCoP.

4.4 Proof reconstruction

If leanCoP found a proof (consisting of a substitution and a tableaux proof), IsaCoP extracts all quantifier logic formulae employed in extension steps and instantiates all quantifiers by the proof deskolemisation method shown in section 3. Furthermore, the terms used inside the tableaux proof are converted to terms in the quantifier logic, replacing Skolem terms by appropriate eigenvariables. Finally, the tableaux proof is converted to a natural deduction proof, following the procedure used in MESON proof reconstruction [Har96]. We obtain a proof of $\bot$, showing that the negated conjecture is unsatisfiable, thus proving the conjecture.

5 Limitations

In this section, we shed light on some limitations of our approach, most notably the type of Skolemisation and the usage of equality.

5.1 Optimised Skolemisation

Table 2: Inner Skolemisation example.

#	Quantifier formula	Quantifier-free formula
t_1	$\neg P(a, c)$	$\neg P(a, c)$
t_2	$\forall w.P(w, c)$	$P(w, c)$
t_3	$\forall xy.(P(x, y) \wedge \exists z.\neg P(z, y))$	$P(x, y) \vee \neg P(f(y), y)$

We currently use outer Skolemisation, because optimised Skolemisation methods, such as inner Skolemisation, may require creating different eigenvariables for syntactically equivalent Skolem terms. Consider for example Table 2. The corresponding natural deduction proof for the quantifier-free formula might look as follows:

1	$\neg P(a, c)$	
2	$P(f(c), c)$	
3.1	$P(a, c) \vee \neg P(f(c), c)$	
3.2	$P(f(c), c) \vee \neg P(f(c), c)$	
4	$\quad P(a, c)$	
5	$\quad \bot$	$\neg$E, 4, 1
6	$\quad \neg P(f(c), c)$	
7	$\quad\quad P(f(c), c)$	
8	$\quad\quad \bot$	$\neg$E, 7, 6
9	$\quad\quad \neg P(f(c), c)$	
10	$\quad\quad \bot$	$\neg$E, 2, 9
11	$\quad \bot$	$\vee$E, 3.2, 7–10
12	$\bot$	$\vee$E, 3.1, 4–11

Line 1 to 3.2 contain the instantiated versions of the quantifier-free formulae with Skolem functions. Note that the second disjuncts of line 3.1 and 3.2 contain two occurrences of $f(c)$, which in the quantifier logic proof will be mapped to two different eigenvariables, namely some e_1 when x is substituted to a, and some e_2 when x is substituted to $f(c)$. This matters because line 10 depends on the correct instantiation done in line 2, where it is not immediately clear without an analysis of the refutation whether e_1 or e_2 should be used in lieu of $f(c)$ to instantiate $\forall w.P(w, c)$.

5.2 Equality

To use our reconstruction method, proofs in calculi with equality are not allowed to rewrite subterms of Skolem terms. We respect this restriction in our implementation by not generating congruence axioms for Skolem functions. To see what can go wrong when Skolem subterms are rewritten, consider the problem

$$\forall x.\exists y.P(x, y) \wedge \neg P(x, y) \wedge x = c.$$

The Skolemised version is

$$\forall x.P(x, f(x)) \wedge \neg P(x, f(x)) \wedge x = c,$$

and instantiating it with a and b yields

$$P(a, f(a)) \wedge \neg P(a, f(a)) \wedge a = c,$$
$$P(b, f(b)) \wedge \neg P(b, f(b)) \wedge b = c.$$

Assume that the proof shows $\bot$ using $P(a, f(a))$ and $\neg P(b, f(b))$, rewriting b to a. Reproducing this proof in quantifier logic is difficult, because the Skolem terms do not exist in quantifier logic, so rewriting underneath a Skolem term cannot be simply translated to quantifier logic.

6 Related work

The axiom of choice can be used to represent Skolem functions: Given a formula $\forall x.\exists y.P(x, y)$, it can be Skolemised to $\forall x.P(x, f(x))$, and the Skolem function is then defined to be $f = \lambda x.\epsilon y.P(x, y)$ [HB39]. While

such reasoning can be in principle expressed in FOL-based logics, it creates an ugly dependency on the axiom of choice, which is not assumed in Isabelle/FOL, and certain derived logics such as the Mizar environment do not assume it in general.

Various approaches for reducing the reliance on the axiom of choice are discussed by Blanchette in the section "Skolemization without Choice" of his Ph.D. thesis [Bla12]. In particular, he shows how to simulate Skolem functions as (higher-order) schematic functions in Isabelle to reconstruct proofs of the automated theorem prover Metis. Isabelle's higher-order unifier can then frequently find the function that implements the Skolem function.

Another approach by de Nivelle [dN05] is about the translation of resolution proofs into first-order proofs: He introduces Skolem relations to simulate the effect of Skolem functions. This approach relies on the less controversial definite description operator. Such an operator is however still not provided for any first-order Mizar type [KPU16].

Avigad [Avi03] adds new functions by building finite approximations thereof via a forcing argument. All these approaches aim in some or the other way at introducing the Skolem functions in the logic where the proof is carried out. Our method reconstructs proofs without introducing Skolem functions in the proof assistant.

A more general approach to reconstruct proofs in natural deduction calculi can be achieved by using *expansion trees* [Mil83, Pfe87] which implicitly encode quantifier instantiations and also allow reconstruction of proofs that do not rely on prenex normal form. We plan to adopt expansion trees in the future.

7 Conclusion

We showed a method to create instances of logic formulae without occurrences of Skolem functions from proofs with Skolem functions. This allows reconstruction of ATP proofs with Skolem functions in first-order logic. Furthermore, we implemented our approach as a proof tactic for Isabelle/FOL, using it to reconstruct proofs from a tableaux prover. Further work can be done to improve the tactic, making it a contestant for existing tactics: Treatment of lambda abstractions (which occur in a few places because FOL is encoded as an Isabelle object logic), e.g. by translating them to combinators, and extension to logics beyond FOL. Furthermore, our method might be usable to reconstruct also proofs from more powerful ATPs, such as E [Sch13] and Vampire [KV13], provided one can obtain suitable substitutions from their proofs and one restricts the usage of equality, see subsection 5.2. In the future, we would like to use expansion trees and to adapt a non-clausal prover [Ott11] for proof search.

Acknowledgements

We would like to thank Chad Brown for the examples in section 5 and Thomas Powell for the discussions on the role of Skolem functions. This work has been supported by the Austrian Science Fund (FWF) grant P26201.

References

[Avi03] Jeremy Avigad. Eliminating definitions and Skolem functions in first-order logic. *ACM Trans. Comput. Log.*, 4(3):402–415, 2003.

[BKPU16] Jasmin Christian Blanchette, Cezary Kaliszyk, Lawrence C. Paulson, and Josef Urban. Hammering towards QED. *J. Formalized Reasoning*, 9(1):101–148, 2016.

[Bla12] Jasmin C. Blanchette. *Automatic Proofs and Refutations for Higher-Order Logic.* PhD thesis, Universität München, 2012.

[dN05] Hans de Nivelle. Translation of resolution proofs into short first-order proofs without choice axioms. *Inf. Comput.*, 199(1-2):24–54, 2005.

[Har96] John Harrison. Optimizing proof search in model elimination. In Michael A. McRobbie and John K. Slaney, editors, *Automated Deduction - CADE-13, 13th International Conference on Automated Deduction, New Brunswick, NJ, USA, July 30 - August 3, 1996, Proceedings*, volume 1104 of *Lecture Notes in Computer Science*, pages 313–327. Springer, 1996.

[HB39] David Hilbert and Paul Bernays. *Grundlagen der Mathematik II.* Springer-Verlag, 1939.

[Hur03] Joe Hurd. First-order proof tactics in higher-order logic theorem provers. In *Design and Application of Strategies/Tactics in Higher Order Logics, number NASA/CP-2003-212448 in NASA Technical Reports*, pages 56–68, 2003.

[KPU16] Cezary Kaliszyk, Karol Pąk, and Josef Urban. Towards a Mizar environment for Isabelle: Foundations and language. In Jeremy Avigad and Adam Chlipala, editors, *Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs, Saint Petersburg, FL, USA, January 20-22, 2016*, pages 58–65. ACM, 2016.

[KUV15] Cezary Kaliszyk, Josef Urban, and Jirí Vyskocil. Certified connection tableaux proofs for HOL Light and TPTP. In Xavier Leroy and Alwen Tiu, editors, *Proceedings of the 2015 Conference on Certified Programs and Proofs, CPP 2015, Mumbai, India, January 15-17, 2015*, pages 59–66. ACM, 2015.

[KV13] Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 1–35. Springer, 2013.

[Mil83] Dale A. Miller. *Proofs in Higher-Order Logic*. PhD thesis, Carnegie Mellon University, 1983.

[NPW02] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.

[NW01] Andreas Nonnengart and Christoph Weidenbach. Computing small clause normal forms. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 335–367. Elsevier and MIT Press, 2001.

[OB03] Jens Otten and Wolfgang Bibel. leanCoP: lean connection-based theorem proving. *J. Symb. Comput.*, 36(1-2):139–161, 2003.

[Ott08] Jens Otten. leanCoP 2.0 and ileanCoP 1.2: High performance lean theorem proving in classical and intuitionistic logic (system descriptions). In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, volume 5195 of *Lecture Notes in Computer Science*, pages 283–291. Springer, 2008.

[Ott11] Jens Otten. A non-clausal connection calculus. In Kai Brünnler and George Metcalfe, editors, *Automated Reasoning with Analytic Tableaux and Related Methods - 20th International Conference, TABLEAUX 2011, Bern, Switzerland, July 4-8, 2011. Proceedings*, volume 6793 of *Lecture Notes in Computer Science*, pages 226–241. Springer, 2011.

[Pau93] Lawrence C. Paulson. Set theory for verification: I. From foundations to functions. *J. Autom. Reasoning*, 11(3):353–389, 1993.

[Pau99] Lawrence C. Paulson. A generic tableau prover and its integration with Isabelle. *J. UCS*, 5(3):73–87, 1999.

[Pfe87] Frank Pfenning. *Proof Transformations in Higher-Order Logic*. PhD thesis, Carnegie Mellon University, 1987.

[Sch13] Stephan Schulz. System description: E 1.8. In Kenneth L. McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings*, volume 8312 of *Lecture Notes in Computer Science*, pages 735–743. Springer, 2013.