EPiC Series in Computer Science

Volume 36, 2015, Pages 127–136

GCAI 2015. Global Conference on Artificial Intelligence

Metis-based Paramodulation Tactic for HOL Light

Michael Färber and Cezary Kaliszyk

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

Abstract

Metis is an automated theorem prover based on ordered paramodulation. It is widely employed in the interactive theorem provers Isabelle/HOL and HOL4 to automate proofs as well as reconstruct proofs found by automated provers. For both these purposes, the tableaux-based MESON tactic is frequently used in HOL Light. However, paramodulation-based provers such as Metis perform better on many problems involving equality. We created a Metis-based tactic in HOL Light which translates HOL problems to Metis, runs an OCaml version of Metis, and reconstructs proofs in Metis' paramodulation calculus as HOL proofs. We evaluate the performance of Metis as proof reconstruction method in HOL Light.

1 Introduction

1.1 Preliminaries

Interactive theorem provers (ITPs) are programs that are used to formalise mathematical concepts, verifying that properties following from the definitions (theorems) are correctly proven. Assuming that a user trusts the soundness of an ITP, a user can be certain that proofs in an ITP formalisation of a mathematical theory are indeed correct. Traditionally, users of an ITP write mathematical definitions in the ITP's language, followed by an interactive construction of proofs, where the ITP gives the user advice, for example which parts of a conjecture are left to prove, rejecting invalid inference steps and so on. The widespread usage of ITPs in mathematics and related domains, such as software verification, has so far been hindered by the high level of knowledge and experience required to operate ITPs. In particular, users frequently struggle to find facts related to their current problems, and once they have found these facts, how to combine them to solve their problems.

Automated deduction frameworks attempt to ease reasoning with ITPs by automatically proving conjectures, thus reducing the entry barrier for users to employ ITPs. Many frameworks employ automated theorem provers (ATPs), which in contrast to ITPs attempt to solve problems without any user interaction, but whose logic might be different from an ITP's logic. Given a conjecture C, an automated deduction framework might proceed as follows to prove C in an ITP:

1. Identify useful facts to prove conjecture C.
2. Translate C together with found facts from the ITP logic to the ATP logic.

G.Gottlob, G.Sutcliffe and A.Voronkov (eds.), GCAI 2015 (EPiC Series in Computer Science, vol. 36), pp. 127–136

3. Find an ATP proof of C.
4. Reconstruct the ATP proof in ITP.

In some cases, the reconstruction of ATP proofs fails, because the ATP does not provide sufficient information to reconstruct a proof. Still, it is often possible to reconstruct the ATP proof by reproving the conjecture with a different ATP, using in addition the facts the first ATP used in its proof.

In this paper, we describe the integration and performance of an ATP for proof reconstruction in an ITP, which can be used for example in an automated deduction framework.

1.2 Related work

HOL Light [Har96a] and HOL4 [GM93] are LCF-style interactive theorem provers with a small kernel. They feature several *tactics*, which serve the purpose to simplify goals. Examples of tactics are rewriting, decision procedures for arithmetic, and β-reduction. One of the most complex tactics in HOL Light is MESON [Har96b]: It translates the current goal along with an optional list of premises to first-order logic, where it attempts to find a proof with the tableaux method using the premises and then reconstruct the proof in HOL.

HOL(y)Hammer [KU15] is an automated deduction framework for HOL4 and HOL Light. Given a conjecture, it attempts to find suitable premises, then calls external ATPs such as E [Sch13], Vampire [KV13], or Z3 [dMB08], and attempts to reconstruct the proof using the premises used by the ATP. To reconstruct proofs, it uses tactics such as MESON, simplification, and a few other decision procedures, however, these are sometimes not powerful enough to reconstruct proofs found by the external ATPs.

Metis [Hur03] is an ATP based on ordered paramodulation, which due to its small set of inference rules is specially suited for reconstruction. Furthermore, it is a proof search method that is frequently used in HOL4 and Isabelle/HOL [NPW02, PS07]. As shown by the second author [KUV15], Metis can solve many problems that tableaux-based methods included in HOL Light, such as MESON [Har96b] and leanCoP [Ott08], cannot solve. However, up to now, Metis was not available as proof search method in HOL Light.

The challenges that have to be overcome to create a Metis-based tactic in HOL Light are as follows.

- *Goal translation.* To convert a HOL problem to a FOL problem tractable by many ATPs, there exists a multitude of translation methods [Har96b, Hur03, MP08, Bla12], differing in their treatment of types, λ-terms, Hilbert's ϵ operator and many more aspects.
- *Using Metis.* Metis is written in Standard ML, whereas HOL Light is written in OCaml. Calling Metis as an external program increases the burden on users because they need to install and maintain a separate program, whereas an OCaml port of Metis takes a considerable effort due to the size of its code base (around 900kB).
- *Proof reconstruction.* To recreate a proof found by Metis in HOL Light, it is necessary to translate all Metis inferences to equivalent HOL Light inferences, as well as to reconstruct the types of the variables used in the (untyped) Metis proof.

We investigated the effectiveness of a HOL Light tactic using Metis for proof reconstruction in automated deduction systems such as HOL(y)Hammer. To this end, we:

- created a HOL Light tactic that translates HOL problems to first-order logic, reusing parts of HOL Light's MESON infrastructure;
- ported a subset of Metis, namely its given-clause algorithm, to OCaml to allow proof search for first-order problems directly inside HOL Light; and

- created a proof reconstruction routine that translates Metis proofs to HOL Light proofs by using detailed Metis proof objects.

We evaluated the resulting tactic on two different datasets by comparing it with the HOL Light tactics MESON and a single complete strategy of leanCoP. Our new method reconstructs the largest number of problems in the shortest time.

2 Goal Translation

To translate HOL goals to first-order logic, we reuse large parts of the corresponding MESON parts, as done for the HOL Light leanCoP tactic. The biggest difference is that we do not generate equality axioms, because Metis treats equality directly.

Consider the Flyspeck lemma `length_eq_imp_length_tl_eq` as an example of a HOL goal to translate:

$$|s_1| = |s_2| \implies |\operatorname{tl} s_1| = |\operatorname{tl} s_2|,$$

where $|x|$ stands for the length of a list x and $\operatorname{tl} x$ denotes the tail of a list x. An ATP found a proof of the conjecture using the following HOL Light theorems:

$$\forall l.l \neq [] \implies |\operatorname{tl} l| = |l| - 1 \qquad (\texttt{LENGTH\_TL})$$

$$\forall l.|l| = 0 \iff l = [] \qquad (\texttt{LENGTH\_EQ\_NIL})$$

To reconstruct the ATP proof in HOL Light, one first refutes the goal:

$$\neg(|s_1| = |s_2| \implies |\operatorname{tl} s_1| = |\operatorname{tl} s_2|) \implies \bot,$$

then one adds the ATP premises as assumptions via `POLY_ASSUME_TAC`. This tactic attempts to instantiate premises containing polymorphic constants (such as tl) with the types of the objects they are used with:

$$\texttt{LENGTH\_TL} \implies \texttt{LENGTH\_EQ\_NIL} \implies \neg(|s_1| = |s_2| \implies |\operatorname{tl} s_1| = |\operatorname{tl} s_2|) \implies \bot$$

Furthermore, the translation involves the following steps.

- Elimination of Hilbert's ϵ operator: $\epsilon x.Px$ denotes an object x for which Px holds. As the ϵ operator is not available in Metis' first-order logic, HOL Light's `SELECT_ELIM_TAC` replaces occurrences of the form $\epsilon x.Px$ by a fresh variable v, adding the assumption $\forall x.Px \implies Pv$. As this step does not preserve logical equivalence, it can make a provable goal unprovable.
- Fixing of function arities: If the same function appears multiple times in the goal with a different number of arguments, e.g. fx and fxy, it cannot be directly translated to a Metis FOL function, as they need to always be applied to the same number of arguments. The translation thus replaces applications of such functions with an application operator I, turning fx into Ifx and fxy into $I(Ifx)y$.
- Elimination of λ-abstractions (via λ-lifting), β-conversion
- Conversion to prenex normal form
- Instantiation of universally quantified variables with fresh variables
- Skolemisation
- Conversion to conjunctive normal form (CNF)

- Splitting: Performs case distinction for disjunctions, producing new subgoals. For example, this might break the goal $a \vee b \vee c \implies g$ into subgoals $a \implies g$, $b \implies g$, and $c \implies g$. The splitting limit specifies the maximum number of times case distinction can be performed on a single disjunction.

This procedure deviates only marginally from those used in MESON and the HOL Light version of leanCoP. It produces untyped HOL problems, which correspond to Hurd's *uHOL* problem set [Hur03]. As Blanchette showed in his PhD thesis [Bla12], encoding types as in Isabelle's Sledgehammer [MP08] can increase the number of reconstructible problems. In Isabelle and HOL4, several different translations (untyped and typed) are passed to the ATPs, thus increasing the chance of finding a proof. We expect similar gains for HOL Light when producing different translations, and due to the similarity of FOL translation methods in MESON, leanCoP, and Metis, we believe that all these methods could profit from such an improvement with relatively little adaptation.

For the particular problem at hand, the translation produces the following (untyped) FOL problem, where v_1 and v_2 are the only variables:

$$\begin{aligned}
& v_1 = [] \vee |\operatorname{tl} v_1| = |v_1| - 1 && (\texttt{LENGTH_TL'}) \\
\implies & |v_2| \neq 0 \vee v_2 = [] && (\texttt{LENGTH_EQ_NIL}_{\implies}) \\
\implies & |v_2| = 0 \vee v_2 \neq [] && (\texttt{LENGTH_EQ_NIL}_{\impliedby}) \\
\implies & |s_1| = |s_2| \implies |\operatorname{tl} s_1| \neq |\operatorname{tl} s_2| && \text{(negated conjecture)} \\
\implies & \bot
\end{aligned}$$

The set of assumptions that implies $\bot$ are the clauses that are handed over to Metis.

3 Metis

Metis is an automated theorem prover based on ordered paramodulation. It accepts as input a set of clauses and runs a given clause algorithm that attempts to deduce from the clauses all possible inferences. If one of these inferences shows the empty clause $\square$ (which corresponds to $\bot$ in HOL), Metis returns the proof tree for $\square$. If it is not able to infer any new statements anymore and $\square$ is not among the previous statements, then the original clause set is satisfiable. Because Metis is usually run on translations of negated conjectures ($\neg C$), deducing the empty clause $\square$ corresponds to finding a proof for C.

Metis can be directly used in HOL4 and Isabelle/HOL, because both these projects are written in Standard ML. However, because HOL Light is written in OCaml, we cannot directly use Metis inside it. The first option to use Metis in HOL Light is to call it as external program from HOL Light, as done in HOL Light's Prover9 tactic. The disadvantage of this solution is that it increases the burden on users to install and maintain a separate executable, as well as that it requires us to parse the output of Metis. The second option is to translate Metis to OCaml in a way that it can be directly used inside HOL Light. The disadvantage of this solution is that it requires a lot of work, because the Metis source code weighs about 900kB (For comparison, MESON consists of 10kB of OCaml code, and leanCoP is only 1kB of Prolog.)

We manually ported Metis to OCaml. The port took about one week and resulted in about 400kB of code. We were able to reduce the size by omitting parts of the code (such as the TPTP parser) not related to the given clause algorithm as well as by reusing OCaml data structures that were manually implemented in the SML version of Metis.

Table 1: Reconstruction of the Metis proof rules in HOL Light. Typewriter font (as in `C`) indicates HOL versions of corresponding Metis objects: In particular, C and D stands for clauses, L for a literal, t for a term, and p for a path (denoting the position of a subterm). The term of a literal L at position p is denoted by $L[p]$, and the replacement of a subterm of L at position p by a term t is denoted by $L[p \mapsto t]$.

Metis (i)	HOL Light ($\mathcal{I}(i, m)$)
$\overline{C}$ axiom	$\overline{\texttt{C}}$ `ASSUME`
$\overline{\{L, \neg L\}}$ assume L	$\dfrac{\overline{\forall x.x \vee \neg x}\ \texttt{TAUT}}{\texttt{L} \vee \neg \texttt{L}}\ \texttt{SPEC L}$
$\dfrac{C}{\sigma C}$ subst σ	$\dfrac{\mathcal{I}(C, m\{x \in \operatorname{dom}\sigma \mid x \mapsto \operatorname{Type}(\sigma(x))\})}{\sigma C}$ `SUBST` σ
$\dfrac{\{L\} \cup C \quad \{\neg L\} \cup D}{C \cup D}$ resolve L	$\dfrac{\dfrac{\mathcal{I}(\{L\} \cup C, m)}{\texttt{L} \vee \texttt{C}}\ \texttt{FRONT L} \quad \dfrac{\mathcal{I}(\{\neg L\} \cup D, m)}{\neg \texttt{L} \vee \texttt{D}}\ \texttt{FRONT } \neg\texttt{L}}{\texttt{C} \vee \texttt{D}}\ \texttt{RESOLVE}$
$\overline{\{t = t\}}$ refl t	$\overline{\texttt{t} = \texttt{t}}$ `REFL t`
$\overline{\{\neg(L[p] = t), \neg L, L[p \mapsto t]\}}$ eq $L\ p\ t$	$\dfrac{\dfrac{\overline{\texttt{L}}\ \texttt{ASSUME} \quad \overline{\texttt{L[p]} = \texttt{t}}\ \texttt{ASSUME}}{\texttt{L[p} \mapsto \texttt{t]}}\ \texttt{PATH\_CONV p}}{\neg(\texttt{L[p]} = \texttt{t}) \vee \neg\texttt{L} \vee \texttt{L[p} \mapsto \texttt{t]}}\ \texttt{DISCH\_DISJS}$

In the next section, we show how to convert Metis proofs of $\square$ back to HOL Light proofs of $\bot$.

4 Reconstruction

While there exist stronger ATPs than Metis such as Vampire [KV13] and Z3 [dMB08], a reason to choose Metis for proof reconstruction is its small set of only six different proof rules. In a Metis proof, every inference is represented by the inference type (e.g. resolve), a list of premises, and a conclusion. However, for proof reconstruction, it is convenient to have more information, such as the resolvent used in a resolution step. Conveniently, Metis provides a function to yield more detailed proof objects, similarly to the `prooftrans` tool, which translates Prover9 [McC10] proofs to a format that can be verified by the IVY proof checker [MS00] and which is used in HOL Light's Prover9 tactic. We show the HOL Light inferences for each detailed Metis proof rule in Table 1.

The reconstruction of a Metis proof is done recursively top-down, starting at the proof of the empty clause ($\square$). The HOL translation of a Metis inference i is denoted as $\mathcal{I}(i, m)$, where m is a map from variables to types, and the HOL translation of a whole Metis proof is $\mathcal{I}(i_\square, \{\})$,

$$\dfrac{\dfrac{}{g(a) \neq f(b)}\ \text{axiom} \qquad \dfrac{\dfrac{}{f(x) = g(y)}\ \text{axiom} \qquad \dfrac{\dfrac{\dfrac{}{x_0 = x_0}\ \text{refl} \qquad \dfrac{}{x_0 \neq x_0 \vee x_0 \neq y_0 \vee y_0 = x_0}\ \text{eq}}{x_0 \neq y_0 \vee y_0 = x_0}\ \text{resolve}}{f(x) \neq g(y) \vee g(y) = f(x)}\ \text{subst}}{g(y) = f(x)}\ \text{resolve}}{g(a) = f(b)}\ \text{subst}}{\square}\ \text{resolve}$$

Figure 1: Metis proof of $f(x) = g(y) \implies g(a) \neq f(b) \implies \bot$.

where $i_\square$ is the Metis inference that proves $\square$. The type map m is necessary because when reconstructing Metis inferences that contain terms where the type of variables cannot be inferred, such as $x = x$, it is necessary to know the type of the variable x to correctly reconstruct the inference in HOL.

The substitution rule is the key to successfully inferring variable types: When a substitution σ maps the variable x to a constant c, we can conclude that x will have the type of c in the remaining proof reconstruction branch. For example, consider the Metis proof in Figure 1: When reconstructing the HOL proof of $g(y) = f(x)$, it is necessary to know the types of the variables y and x. The reconstruction can infer the types because the substitution rule was used in a lower part of the proof, instantiating y with a and x with b. As the types of a and b are known, the types of y and x can be inferred.

Metis stores clauses as sets of literals, whereas in HOL Light, clauses are represented by disjunctions of literals, where the order of the literals matters. We only have to deal with this difference in Metis' axiom rule; for if we naively build the disjunction of all Metis literals to create a HOL clause C, C will not necessarily match any assumption in the HOL goal because of the different order of literals. To account for this, we always convert all HOL assumptions and conclusions of the axiom rule to canonical form, which allows us to find the corresponding HOL assumption for a Metis axiom conclusion by simple comparison.

The paths used by Metis in the equality rule are not directly usable in HOL Light because of their different format; see Figure 2 for an example. To convert a path p in Metis format to HOL Light format, we require also the term t to which the path refers:

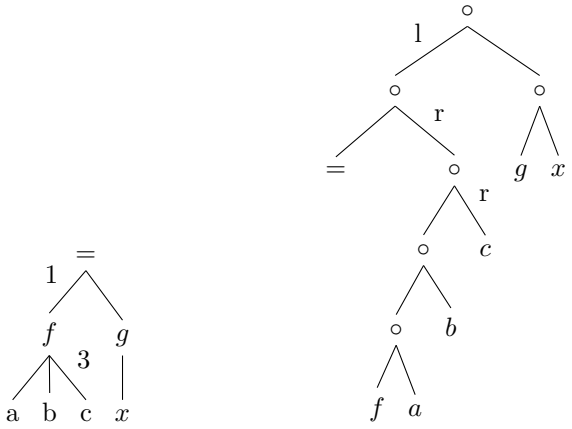
$$\text{Path}(p, t) = \begin{cases} l^{|\vec{x}|-i-1} \cdot r \cdot \text{Path}(x_i, p') & \text{if } t = f\vec{x} \wedge p = i : p' \\ [] & \text{otherwise.} \end{cases}$$

Once the path is converted to HOL Light, we use it via `PATH_CONV` to reconstruct equality inferences. The reconstruction also uses our `DISCH_DISJS` rule which behaves like HOL Light's `DISCH` rule, but creates a disjunction instead of an implication.

5 Evaluation

We use HOL Light (SVN revision: 193) to evaluate the reconstruction performance of three different methods.

- *MESON*: We use the version with a splitting limit of 8, as used by default in HOL Light.
- *leanCoP*$^-$: We use a single strategy of the OCaml version of leanCoP integrated in HOL Light[KUV15] without cut and without splitting. If cut is enabled, leanCoP may discard search branches. This leads to the loss of completeness, but was found to increase the



(a) Metis (b) HOL Light

Figure 2: Representations of the term $f(a, b, c) = g(x)$ and the path to its subterm c ("13" in Metis, "lrr" in HOL Light).

number of proven problems in fixed time limits. As MESON and Metis use complete strategies, we use a complete strategy here as well.

- *Metis*: We evaluate our newly created Metis tactic[1] using a splitting limit of 8 and the default Metis parameters.

A reconstruction problem consists of a conjecture and a set of dependencies which an ATP found to prove the conjecture. The ATP dependencies are obtained as in [KU14]. We evaluate a problem by feeding the conjecture and the dependencies to a proof reconstruction method in HOL Light, such as:

```
let goal = "m <= n ==> m..n = m INSERT (SUC m..n)" in
let dependencies = [NUMSEG_LREC; ADD1] (* found by an ATP *) in
MESON dependencies (parse_term goal);;
```

If the method can reconstruct the proof within a given time limit, the problem counts as proven.

As datasets, we use toplevel goals from Flyspeck (SVN revision: 3511) [HAB+15] as well as from HOL Light's standard library. We evaluate every problem in parallel on a 48-core server with AMD Opteron 6174 2.2 GHz CPUs, 320 GB RAM, and 0.5 MB L2 cache per CPU.

The results are given in Table 2 and Table 3: Our Metis tactic performs better than leanCoP$^-$ and MESON, both in terms of speed as well as number of proven problems. The speed gains might be due to the fact that the FOL representations of some problems are satisfiable and therefore not provable, in which case Metis stops as soon as it finds a satisfiable assignment, whereas MESON and leanCoP continue trying to solve the problem.

We evaluated the Metis tactic without splitting (splitting limit = 0), which we found to perform slightly worse than with splitting: For example, on the Flyspeck data with timeout 5s, disabling splitting decreased the number of proven problems from 6924 to 6889, which still places Metis above the other methods.

[1] Source for the Metis tactic and the evaluation is available at: http://cl-informatik.uibk.ac.at/users/mfaerber/tactics.html

Table 2: Results for Flyspeck data (18956 problems, avg. 39.1 dependencies per problem).

(a) Timeout: 5s

Prover	Proven	Unique	Time [s]
MESON	6484	225	1601
leanCoP$^-$	5352	113	1732
Metis	6924	889	1533
Σ	7552		4866

(b) Timeout: 30s

Prover	Proven	Unique	Time [s]
MESON	6665	191	8315
leanCoP$^-$	5609	102	8921
Metis	7398	1037	7193
Σ	7918		24429

Table 3: Results for HOL Light data (1773 problems, avg. 3.6 dependencies per problem).

(a) Timeout: 5s

Prover	Proven	Unique	Time [s]
MESON	1208	37	88
leanCoP$^-$	957	8	114
Metis	1266	132	76
Σ	1357		278

(b) Timeout: 30s

Prover	Proven	Unique	Time [s]
MESON	1236	33	390
leanCoP$^-$	1019	9	545
Metis	1318	132	252
Σ	1392		1187

In the evaluation we focused on a single strategy of the leanCoP tactic running inside HOL Light. Using the single cut strategy without completeness has a negative effect on the performance (contrary to the standalone results [KUV15]). Enabling cut would *decrease* the number of proven problems, for example on the Flyspeck data with timeout 5s from 5352 to 3984 proven problems. The results of leanCoP with multiple strategies in [KUV15] and our evaluation are also different for other reasons: the stand-alone OCaml version of leanCoP, whereas we the HOL Light version reuses the slower HOL Light structures. Even if its performance would match the stand-alone version, the results from [KUV15] show that Metis is still very interesting as it complements leanCoP and MESON.

6 Conclusion

We developed a HOL Light tactic using the paramodulation-based ATP Metis. This tactic can be used by HOL(y)Hammer to reconstruct proofs found by stronger ATPs, or it can be used directly by users as a proof search method. For the Flyspeck dataset at timeout 30s, our tactic was able to reconstruct over 7% more proofs than the union of MESON and the complete strategy of the HOL Light version of leanCoP, while being the fastest proof reconstruction method of the three. Further experiments could be conducted with different translations from higher-order to first-order logic, as done for Isabelle [Bla12].

An interesting future direction is the integration of machine learning techniques techniques similar to those used in MaLeCoP [UVŠ11] to reduce the search space. Providing multiple strategies, including incomplete ones, would allow directly comparing Metis to leanCoP.

Acknowledgements

This work has been supported by the Austrian Science Fund (FWF) grant P26201.

References

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

[dMB08] Leonardo M. de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Hungary, March 29-April 6, 2008. Proceedings*, pages 337–340, 2008.

[GM93] Michael J.C. Gordon and Thomas F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, 1993.

[HAB+15] Thomas C. Hales, Mark Adams, Gertrud Bauer, Dat Tat Dang, John Harrison, Truong Le Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Tat Nguyen, Truong Quang Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, An Hoai Thi Ta, Trung Nam Tran, Diep Thi Trieu, Josef Urban, Ky Khac Vu, and Roland Zumkeller. A formal proof of the Kepler conjecture. *CoRR*, abs/1501.02155, 2015.

[Har96a] John Harrison. HOL Light: A tutorial introduction. In *Formal Methods in Computer-Aided Design, First International Conference, FMCAD '96, Palo Alto, California, USA, November 6-8, 1996, Proceedings*, pages 265–269, 1996.

[Har96b] John Harrison. Optimizing proof search in model elimination. In *Automated Deduction - CADE-13, 13th International Conference on Automated Deduction, New Brunswick, NJ, USA, July 30 - August 3, 1996, Proceedings*, pages 313–327, 1996.

[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.

[KU14] Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with Flyspeck. *J. Autom. Reasoning*, 53(2):173–213, 2014.

[KU15] Cezary Kaliszyk and Josef Urban. HOL(y)Hammer: Online ATP service for HOL Light. *Mathematics in Computer Science*, 9(1):5–22, 2015.

[KUV15] Cezary Kaliszyk, Josef Urban, and Jiří Vyskočil. Certified connection tableaux proofs for HOL Light and TPTP. In Xavier Leroy and Alwen Tiu, editors, *Proceedings of the 4th ACM-SIGPLAN Conference on Certified Programs and Proofs*, pages 59–66, 2015.

[KV13] Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, pages 1–35, 2013.

[McC10] William McCune. Prover9 and Mace4. 2005–2010.

[MP08] Jia Meng and Lawrence C. Paulson. Translating higher-order clauses to first-order clauses. *J. Autom. Reasoning*, 40(1):35–60, 2008.

[MS00] William McCune and Olga Shumsky. Ivy: A preprocessor and proof checker for first-order logic. In M. Kaufmann, P. Manolios, and J Moore, editors, *Computer-Aided Reasoning: ACL2 Case Studies*. Kluwer Academic Publishers, 2000.

[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.

[Ott08] Jens Otten. leanCoP 2.0 and ileanCoP 1.2: High performance lean theorem proving in classical and intuitionistic logic (system descriptions). In *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, pages 283–291, 2008.

[PS07] Lawrence C. Paulson and Kong Woei Susanto. Source-level proof reconstruction for interactive theorem proving. In *Theorem Proving in Higher Order Logics, 20th International Conference, TPHOLs 2007, Kaiserslautern, Germany, September 10-13, 2007, Proceedings*, pages 232–245, 2007.

[Sch13] Stephan Schulz. System description: E 1.8. In *Logic for Programming, Artificial Intelligence, and Reasoning - 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings*, pages 735–743, 2013.

[UVŠ11] Josef Urban, Jiří Vyskočil, and Petr Štěpánek. MaLeCoP machine learning connection prover. In *Automated Reasoning with Analytic Tableaux and Related Methods - 20th International Conference, TABLEAUX 2011, Bern, Switzerland, July 4-8, 2011. Proceedings*, pages 263–277, 2011.