The Higher-Order Prover Leo-III⋆

Alexander Steen[1] and Christoph Benzmüller[2,1]

[1] Freie Universität Berlin, Institute of Computer Science, Berlin, Germany
[2] University of Luxembourg, Computer Science and Communications, Luxembourg
{a.steen,c.benzmueller}@fu-berlin.de

Abstract. The automated theorem prover Leo-III for classical higher-order logic with Henkin semantics and choice is presented. Leo-III is based on extensional higher-order paramodulation and accepts every common TPTP dialect (FOF, TFF, THF), including their recent extensions to rank-1 polymorphism (TF1, TH1). In addition, the prover natively supports almost every normal higher-order modal logic. Leo-III cooperates with first-order reasoning tools using translations to many-sorted first-order logic and produces verifiable proof certificates. The prover is evaluated on heterogeneous benchmark sets.

1 Introduction

Leo-III is an automated theorem prover (ATP) for classical higher-order logic (HOL) with Henkin semantics and choice.[3] It is the successor of the well-known LEO-II prover [1], whose development significantly influenced the build-up of the TPTP THF infrastructure [2]. Leo-III exemplarily utilizes and instantiates the associated LEOPARD system platform [3] for higher-order (HO) deduction systems implemented in Scala.

In the tradition of the cooperative nature of the LEO prover family, Leo-III collaborates with external theorem provers during proof search, in particular, with first-order (FO) ATPs such as E [4]. Unlike LEO-II, which translated proof obligations into untyped FO languages, Leo-III, by default, translates its HO clauses to (monomorphic or polymorphic) many-sorted FO formulas. That way clutter is reduced during translation, resulting in a more effective cooperation.

Leo-III supports all common TPTP [5,2] dialects (CNF, FOF, TFF, THF) as well as the polymorphic variants TF1 and TH1 [6,7]. The prover returns results according to the standardized SZS ontology and additionally produces a TSTP-compatible (refutation) proof certificate, if a proof is found. Furthermore, Leo-III natively supports reasoning for almost every normal HO modal logic, including (but not limited to) logics **K**, **D**, **T**, **S4** and **S5** with constant, cumulative or varying domain quantifiers [8]. These hybrid logic competencies make Leo-III, up to the authors' knowledge, the most widely applicable ATP available to date.

---

[3] Leo-III is freely available (BSD license) at `http://github.com/leoprover/Leo-III`.

The most current release of Leo-III (version 1.2) comes with several novel features, including specialized calculus rules for function synthesis, injective functions and equality-based simplification. An evaluation of Leo-III 1.2 confirms that it is on a par with other current state-of-the-art HO ATP systems.

This paper outlines the base calculus of Leo-III and highlights the features of version 1.2. As a pioneering contribution this also includes Leo-III's native support for reasoning in HO modal logics (which, of course, includes propositional and FO modal logics). Finally, an evaluation of Leo-III is presented for all monomorphic and polymorphic THF problems from the TPTP problem library and for all mono-modal logic problems from the QMLTP library [9].

Related ATP systems. These include TPS, Satallax, cocATP and agsyHOL. Also, some interactive proof assistants such as Isabelle/HOL can be used for automated reasoning in HOL. More weakly related systems include the various recent attempts to lift FO ATPs to the HO domain, e.g. Zipperposition.

Higher-Order Logic. HOL as addressed here has been proposed by Church, and further studied by Henkin, Andrews and others, cf. [10,11] and the references therein. It provides lambda-notation, as an elegant and useful means to denote unnamed functions, predicates and sets (by their characteristic functions). In the remainder a notion of HOL with Henkin semantics and choice is assumed.

2 Higher-Order Paramodulation

Leo-III extends a complete, paramodulation based calculus for HOL with practically motivated, heuristic inference rules, cf. Fig. 1. They are grouped as follows:

Clause normalization. Leo-III employs *definitional clausification* to reduce the number of clauses. Moreover, *miniscoping* is employed prior to clausification. Further normalization rules are straightforward.

Primary inferences. The primary inference rules of Leo-III are *paramodulation* (Para), *equality factoring* (EqFac) and *primitive substitution* (PS) as displayed in Fig. 1. The first two introduce *unification constraints* that are encoded as negative literals. Note that these rules are unordered and produce numerous redundant clauses. Leo-III uses several heuristics to restrict the number of inferences, including a *HO term ordering*. While these restrictions sacrifice completeness in general, recent evaluations confirm practicality of this approach (cf. evaluation in §4); complete search may be retained though. PS instantiates free variables at top-level with approximations of predicate formulas using so-called *general bindings* $\mathcal{GB}_\tau^C$ [1, §2].

Unification. Unification in Leo-III uses a variant of *Huet's pre-unification rules*. Negative equality literals are interpreted as unification constraints and are attempted to be *solved eagerly* by unification. In contrast to LEO-II, Leo-III uses *pattern unification* whenever possible. In order to ensure termination, the pre-unification *search is limited* to a configurable depth.

Extensionality rules. Dedicated *extensionality rules* are used in order to eliminate the need for extensionality axioms in the search space. The rules are similar to those of LEO-II [1].

$$\frac{\mathcal{C} \vee [l \simeq r]^{\mathtt{tt}} \qquad \mathcal{D} \vee [s \simeq t]^{\alpha}}{\mathcal{C} \vee \mathcal{D} \vee [s[r]_\pi \simeq t]^{\alpha} \vee [s|_\pi \simeq l]^{\mathtt{ff}}} \text{ (Para)} \qquad \frac{\mathcal{C} \vee [l \simeq r]^{\alpha} \vee [s \simeq t]^{\alpha}}{\mathcal{C} \vee [l \simeq r]^{\alpha} \vee [l \simeq s]^{\mathtt{ff}} \vee [r \simeq t]^{\mathtt{ff}}} \text{ (EqFac)}$$

$$\frac{\mathcal{C} \vee [X_{\overline{\tau_i} \to o} \; \overline{t^i_{\tau_i}}]^{\alpha} \qquad p \in \mathcal{GB}^{\{\neg, \vee\} \cup \{\Pi^\tau, =^\tau | \tau \in \mathcal{T}\}}_{\overline{\tau_i} \to o}}{\left(\mathcal{C} \vee [X_{\overline{\tau_i} \to o} \; \overline{t^i_{\tau_i}}]^{\alpha}\right)\{X/p\}} \text{ (PS)}$$

(Choice): E is a choice operator or a free variable; X is a fresh variable.
(INJ): sk is a fresh constant symbol of appropriate type.

$$\frac{\mathcal{C} \vee [s[E \; t]]^{\alpha}}{[t \; X]^{\mathtt{ff}} \vee [t \; (\epsilon \; t)]^{\mathtt{tt}}} \text{ (Choice)} \qquad \frac{[f \; X \simeq f \; Y]^{\mathtt{tt}} \vee [X \simeq Y]^{\mathtt{ff}}}{[\text{sk} \; (f \; X) \simeq X]^{\mathtt{tt}}} \text{ (INJ)}$$

$$\frac{\mathcal{C} := \mathcal{C}' \vee [F_{\overline{\tau_j} \to \tau} \; \overline{s^{1,j}}^{1 \leq j \leq n} \simeq t^1_\tau]^{\mathtt{ff}} \vee \cdots \vee [F \; \overline{s^{m,j}}^{1 \leq j \leq n} \simeq t^m_\tau]^{\mathtt{ff}}}{\mathcal{C}\left\{F/\lambda \overline{X^j_{\tau_j}}.\epsilon Z_\tau. \; \bigwedge_{k=1}^{m}\left(\left(\bigwedge_{j=1}^{n} X^j = s^{k,j}\right) \longrightarrow Z = t^k\right)\right\}} \text{ (FS)}$$

Fig. 1. Examples of Leo-III's calculus rules. *Technical preliminaries*: $s \simeq t$ denotes an equation of HOL terms, where $\simeq$ is assumed to be symmetric. A literal ℓ is a signed equation, written $[s \simeq t]^{\alpha}$ where $\alpha \in \{\mathtt{tt}, \mathtt{ff}\}$ is the polarity of ℓ. Literals of form $[s_o]^{\alpha}$ are a shorthand for $[s_o \simeq \top]^{\alpha}$. A clause $\mathcal{C}$ is a multiset of literals, denoting its disjunction. For brevity, if $\mathcal{C}, \mathcal{D}$ are clauses and ℓ is a literal, $\mathcal{C} \vee \ell$ and $\mathcal{C} \vee \mathcal{D}$ denote the multi-union $\mathcal{C} \cup \{\ell\}$ and $\mathcal{C} \cup \mathcal{D}$, respectively. $s|_\pi$ is the subterm of s at position π, and $s[r]_\pi$ denotes the term that is created by replacing the subterm of s at position π by r.

Clause contraction. In addition to standard simplification routines, Leo-III implements are variety of (equational) *simplification procedures*, including *subsumption*, destructive *equality resolution*, heuristic *rewriting* and contextual *unit cutting* (simplify-reflect).

Defined Equalities. Leo-III scans for common definitions of equality predicates and heuristically instantiates (or replaces) them with *primitive equality*.

Choice. Leo-III supports HOL with choice ($\epsilon^\tau_{(\tau \to o) \to \tau}$ being a choice operator for type τ). Rule (Choice) *instantiates choice predicates* for subterms that represent either concrete choice operator applications (if $E \equiv \epsilon$) or potential applications of choice (if E is a free variable of the clause).

Function synthesis. If plain unification fails for a set of unification constraints, Leo-III may try to *synthesise function specifications* by rule (FS) using special choice instances that simulate suitable if-then-else terms. In general, this rule tremendously increases the search space. However, it also enables Leo-III to solve some hard problems (with TPTP rating 1.0). Also, Leo-III supports improved *reasoning with injective functions* by postulating the existence of left-inverses, cf. rule (INJ).

Heuristic instantiation. Prior to clause normalization, Leo-III might *instantiate universally quantified* variables. This include exhaustive instantiation of finite types as well as partial instantiation for otherwise interesting types.

3 Modal Logic Reasoning

Modal logics have many relevant applications in computer science, artificial intelligence, mathematics and computational linguistics. They also play an important role in many areas of philosophy, including ontology, ethics, philosophy of mind and philosophy of science. Many challenging applications, as recently explored in metaphysics, require FO or HO modal logics (HOMLs). The development of ATPs for these logics, however, is still in its infancy.

Leo-III is addressing this gap. In addition to its HOL reasoning capabilities, it is the first ATP that natively supports a very wide range of normal HOMLs. To achieve this, Leo-III internally implements a shallow semantical embeddings approach [12,13]. The key idea in this approach is to provide and exploit faithful mappings for HOML input problems to HOL that encode its Kripke-style semantics. An example is as follows:

A The user inputs a HOML problem in a suitably adapted TPTP syntax, e.g.
```
thf(1,conjecture,( ! [P:$i>$o,F:$i>$i, X:$i]: (? [G:$i>$i]:
      (($dia @ ($box @ (P @ (F @ X)))) => ($box @ (P @ (G @ X))))))).
```
which encodes $\forall P_{\iota \to o} \forall F_{\iota \to \iota} \forall X_{\iota} \exists G_{\iota \to \iota} (\Diamond \Box P(F(X)) \Rightarrow \Box P(G(X)))$, with `$box` and `$dia` representing the (mono-)modal operators. This example formula (an instance of a corollary of Becker's postulate) is valid in **S5**.

B In the header of the input file the user specifies the logic of interest, say **S5** with rigid constants, constant domain quantifiers and a global consequence relation. For this purpose the TPTP language has been suitably extended:[4]
```
thf(simple_s5, logic, ($modal := [
      $constants := $rigid, $quantification := $constant,
      $consequence := $global, $modalities := $modal_system_S5 ])).
```

C When being called with this input file, Leo-III parses and analyses it, automatically selects and unfolds the corresponding definitions of the semantical embedding approach, adds appropriate axioms and then starts reasoning in (meta-logic) HOL. Subsequently, it returns SZS compliant result information and, if successful, also a proof object just as for standard HOL problems. Leo-III's proof for the embedded example is verified by GDV [5] in 356 s.

As of version 1.2, Leo-III supports (but is not limited to) FO and HO extensions of the well known modal logic cube. When taking the different parameter combinations into account (constant/cumulative/varying domain semantics, rigid/non-rigid constants, local/global consequence relation, and further semantical parameters) this amounts to more than 120 supported HOMLs.[5] The exact number of supported logics is in fact much higher, since Leo-III also supports multi-modal logics with independent modal system specification for each modality. Also, user-defined combinations of rigid and non-rigid constants and different quantification semantics per type domain are possible. *Related provers* are in contrast limited to propositional logics or support a small range of FO modal

---

[4] Cf. http://www.cs.miami.edu/~tptp/TPTP/Proposals/LogicSpecification.html

[5] Cf. [13, §2.2]; we refer to the literature [8] for more details on HOML.

logics only [14,15,16]. In the restricted logic settings of the related systems, the embedding approach used by Leo-III is still competitive; cf. evaluation below.

4   Evaluation

In order to quantify the performance of Leo-III, an evaluation based on various benchmarks was conducted. Three benchmark data sets were used:

- *TPTP TH0* (2463 problems) is the set of all monomorphic HOL (TH0) problems from the TPTP library v7.0.0 [5] that are annotated as theorems. The TPTP library is a de-facto standard for the evaluation of ATP systems.
- *TPTP TH1* (442 problems) is the subset of all 666 polymorphic HOL (TH1) problems from TPTP v7.0.0 that are annotated as theorems and do not contain arithmetic. The problems mainly consist of HOL Light core exports and Sledgehammer translations of various Isabelle theories.
- *QMLTP* (580 problems) is the subset of all mono-modal benchmarks from the QMLTP library 1.1 [9]. The QMLTP library only contains propositional and FO modal logic problems. Since each problem may have a different validity status for each semantics of modal logic, all problems (and not only those marked as theorem) are selected. The total number of tested problems thus is 580 (raw problems) $\times$ 5 (logics) $\times$ 3 (domain conditions). QMLTP assumes rigid constant symbols and a local consequence relation.

The evaluation measurements were taken on the StarExec cluster in which each compute node is a 64 bit Red Hat Linux (kernel 3.10.0) machine featuring 2.40 GHz Intel Xeon quad-core processors and a main memory of 128 GB. For each problem, every prover was given a CPU time limit of 240 s. The following theorem provers were employed in one or more of the experiments: Leo-III 1.2 (TH0/TH1/QMLTP) used in conjuction with E, CVC4 and iProver, Isabelle/HOL 2016 (TH0/TH1) [17], Satallax 3.0 (TH0) [18], Satallax 3.2 (TH0), LEO-II 1.7.0 (TH0), Zipperposition 1.1 (TH0) and MleanCoP 1.3 [19] (QMLTP).

The experimental results are discussed next:

TPTP TH0. Table 1 (a) displays each system's performance on the TPTP TH0 data set. For each system the absolute number (Abs.) and relative share (Rel.) of solved problems is displayed. Solved here means that a system is able to establish the SZS status `Theorem` and also emits a proof certificate that substantiates this claim. All results of the system, whether successful or not, are counted and categorized as THM (`Theorem`), CAX (`ContradictoryAxioms`), GUP (`GaveUp`) and TMO (`TimeOut`) for the respective SZS status of the returned result.[6] Additionally, the average and sum of all CPU times and wall clock (WC) times over all solved problems is presented.

Leo-III successfully solves 2053 of 2463 problems (roughly 83.39 %) from the TPTP TH0 data set. This is 735 (35.8 %) more than Zipperposition, 264

---

[6] Remark on CAX: In this special case of THM (theorem) the given axioms are inconsistent, so that anything follows, including the given conjecture. Unlike most other provers, Leo-III checks for this special situation.

Table 1. Detailed result of the benchmark measurements

(a) TPTP TH0 data set (2463 problems)

Systems	Solved		SZS Results				Avg. Time [s]		Σ Time [s]	
	Abs.	Rel.	THM	CAX	GUP	TMO	CPU	WC	CPU	WC
Satallax 3.2	2140	86.89	2140	0	2	321	12.26	12.31	26238	26339
Leo-III	2053	83.39	2045	8	16	394	15.39	5.61	31490	11508
Satallax 3.0	1972	80.06	2028	0	2	433	17.83	17.89	36149	36289
LEO-II	1788	72.63	1789	0	43	631	5.84	5.96	10452	10661
Zipperposition	1318	53.51	1318	0	360	785	2.60	2.73	3421	3592
Isabelle/HOL	0	0.00	2022	0	1	440	46.46	33.44	93933	67610

(b) TPTP TH1 data set (442 problems)

| Systems	Solved		SZS Results				Avg. Time [s]		Σ Time [s]	
	Abs.	Rel.	THM	CAX	GUP	TMO	CPU	WC	CPU	WC
Leo-III	185	41.86	183	2	8	249	49.18	24.93	9099	4613
Isabelle/HOL	0	0.00	237	0	23	182	93.53	81.44	22404	19300

(12.86 %) more than LEO-II and 81 (3.95 %) more than Satallax 3.0. The only ATP system that solves more problems is the most recent version of Satallax (3.2) that successfully solves 2140 problems, which is approximately 4.24 % more than Leo-III. Isabelle currently does not emit proof certificates (hence zero solutions). Even if results without explicit proofs are counted, Leo-III would still have a slightly higher number of problems solved than Satallax 3.0 and Isabelle/HOL with 25 (1.22 %) and 31 (1.51 %) additional solutions, respectively. Leo-III, Satallax (3.2), Zipperposition and LEO-II produce 18, 17, 15 and 3 unique solutions, respectively. Evidently, Leo-III currently produces more unique solutions than any other ATP system in this setting. Leo-III solves twelve problems that are currently not solved by any other system indexed by TPTP.[7]

Satallax, LEO-II and Zipperposition show only small differences between their individual CPU and WC time on average and sum. A more precise measure for a system's utilization of multiple cores is the so-called *core usage*. It is given by the average of the ratios of used CPU time to used wall clock time over all solved problems. The core usage of Leo-III for the TPTP TH0 data set is roughly 2.52. This means that, on average, two to three CPU cores are used during proof search by Leo-III. Satallax (3.2), LEO-II and Zipperposition show a quite opposite behavior with core usages of 0.64, 0.56 and 0.47, respectively.

TPTP TH1. Currently, there exist only few ATP systems that are capable of reasoning within polymorphic HOL as specified by TPTP TH1. The only exceptions are HOL(y)Hammer and Isabelle/HOL that schedule proof tactics within HOL Light and Isabelle/HOL, respectively. Unfortunately, only Isabelle/HOL was available for instrumentation in a reasonably recent and stable version. Ta-

---

[7] This information is extracted from the TPTP problem rating information that is attached to each problem. The unsolved problems are `NLP004^7`, `SET013^7`, `SEU558^1`, `SEU683^1`, `SEV143^5`, `SYO037^1`, `SYO062^4.004`, `SYO065^4.001`, `SYO066^4.004`, `MSC007^1.003.004`, `SEU938^5` and `SEV106^5`.

Fig. 2. Comparison of Leo-III and MleanCoP on the QMLTP data set (580 problems)
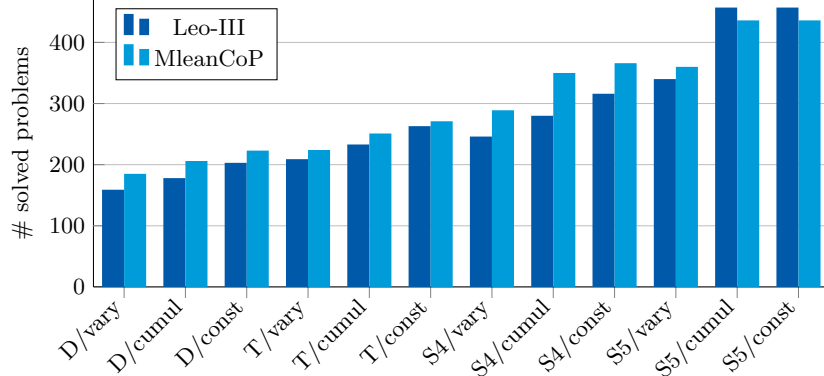
ble 1 (b) displays the measurement results for the TPTP TH1 data set. When disregarding proof certificates, Isabelle/HOL finds 237 theorems (53.62 %) which is roughly 28.1 % more than the number of solutions founds by Leo-III. Leo-III and Isabelle/HOL produce 35 and 69 unique solutions, respectively.

QMLTP. For each semantical setting supported by MleanCoP, which is the strongest FO modal logic prover available to date [16], the number of theorems found by both Leo-III and MleanCoP in the QMLTP data set is presented in Fig. 2. Leo-III is fairly competitive with MleanCoP (weaker by maximal 14.05 %, minimal 2.95 % and 8.90 % on average) for all **D** and **T** variants. For all **S4** variants, the gap between both systems increases (weaker by maximal 20.00 %, minimal 13.66 % and 16.18 % on average). For **S5** variants, Leo-III is very effective (stronger by 1.36 % on average) and it is ahead of MleanCoP for **S5**/const and **S5**/cumul (which coincide). This is due to the encoding of the **S5** accessibility relation in Leo-III 1.2 as the universal relation between possible worlds as opposed to its prior encoding (cf. [16,13]) as an equivalence relation. Leo-III contributes 199 solutions to previously unsolved problems.

For HOML there exist no competitor systems with which Leo-III could be compared. However, Leo-III can e.g. prove all problems from [20].

5 Summary

Leo-III is a state-of-the-art higher-order reasoning system offering many relevant features and capabilities. Due to its wide range of natively supported classical and non-classical logics, which include polymorphic HO logic and numerous FO and HO modal logics, the system has many topical applications in computer science, AI, maths and philosophy. Additionally, an evaluation on heterogeneous benchmark sets shows that Leo-III is also one of the most effective HO ATP systems to date. Leo-III complies with existing TPTP/TSTP standards, gives detailed proof certificates and plays a pivotal role in the ongoing extension of the TPTP library and infrastructure to support modal logic reasoning.

References

1. Benzmüller, C., Sultana, N., Paulson, L.C., Theiß, F.: The higher-order prover LEO-II. Journal of Automated Reasoning **55**(4) (2015) 389–404
2. Sutcliffe, G., Benzmüller, C.: Automated reasoning in higher-order logic using the TPTP THF infrastructure. Journal of Formalized Reasoning **3**(1) (2010) 1–27
3. Wisniewski, M., Steen, A., Benzmüller, C.: LeoPARD - A generic platform for the implementation of higher-order reasoners. In Kerber, M., et al., eds.: Intelligent Computer Mathematics. Volume 9150 of LNCS., Springer (2015) 325–330
4. Schulz, S.: E - a brainiac theorem prover. AI Commun. **15**(2-3) (2002) 111–126
5. Sutcliffe, G.: The TPTP problem library and associated infrastructure - from CNF to TH0, TPTP v6.4.0. J. Autom. Reasoning **59**(4) (2017) 483–502
6. Blanchette, J.C., Paskevich, A.: TFF1: the TPTP typed first-order form with rank-1 polymorphism. In Bonacina, M.P., ed.: Automated Deduction. Volume 7898 of LNCS., Springer (2013) 414–420
7. Kaliszyk, C., Sutcliffe, G., Rabe, F.: TH1: the TPTP typed higher-order form with rank-1 polymorphism. In Fontaine, P., et al., eds.: 5th PAAR Workshop. Volume 1635 of CEUR Workshop Proceedings., CEUR-WS.org (2016) 41–55
8. Blackburn, P., van Benthem, J.F., Wolter, F.: Handbook of modal logic. Volume 3. Elsevier (2006)
9. Raths, T., Otten, J.: The QMLTP Problem Library for First-Order Modal Logics. In: IJCAR 2012. Volume 7364 of LNCS., Springer (2012) 454–461
10. Andrews, P.: Church's type theory. In Zalta, E.N., ed.: The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University (2014)
11. Benzmüller, C., Miller, D.: Automation of higher-order logic. In Gabbay, D.M., Siekmann, J.H., Woods, J., eds.: Handbook of the History of Logic, Volume 9 — Computational Logic. North Holland, Elsevier (2014) 215–254
12. Benzmüller, C., Paulson, L.: Multimodal and intuitionistic logics in simple type theory. The Logic Journal of the IGPL **18**(6) (2010) 881–892
13. Gleißner, T., Steen, A., Benzmüller, C.: Theorem provers for every normal modal logic. In Eiter, T., Sands, D., eds.: LPAR-21. Volume 46 of EPiC Series in Computing., Maun, Botswana, EasyChair (2017) 14–30
14. Hustadt, U., Schmidt, R.A.: Mspass: Modal reasoning by translation and first-order resolution. In: TABLEAUX. Volume 1847 of LNCS., Springer (2000) 67–71
15. Tishkovsky, D., Schmidt, R.A., Khodadadi, M.: The tableau prover generator MetTeL2. In: European Workshop on Logics in AI, Springer (2012) 492–495
16. Benzmüller, C., Otten, J., Raths, T.: Implementing and evaluating provers for first-order modal logics. In Raedt, L.D., et al., eds.: ECAI 2012. Volume 242 of Frontiers in AI and Applications., Montpellier, France, IOS Press (2012) 163–168
17. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic. Lecture Notes in Computer Science. Springer (2002)
18. Brown, C.E.: Satallax: An automatic higher-order prover. In: Automated Reasoning. Volume 7364 of LNCS. Springer Berlin Heidelberg (2012) 111–117
19. Otten, J.: Mleancop: A connection prover for first-order modal logic. In: Automated Reasoning. Volume 8562 of LNCS., Springer (2014) 269–276
20. Benzmüller, C., Woltzenlogel Paleo, B.: The inconsistency in Gödel's ontological argument: A success story for AI in metaphysics. In: IJCAI 2016. Volume 1-3., AAAI Press (2016) 936–942