View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Open Repository and Bibliography - Luxembourg

Agent-Based HOL Reasoning*

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

[1] Freie Universität Berlin, Institute of Computer Science, Germany
{a.steen,m.wisniewski,c.benzmueller}@fu-berlin.de
[2] Stanford University, CSLI, USA

Abstract. In the Leo-III project, a new agent-based deduction system for classical higher-order logic is developed. Leo-III combines its predecessor's concept of cooperating external specialist systems with a novel agent-based proof procedure. Key goals of the system's development involve parallelism on various levels of the proof search, adaptability for different external specialists, and native support for reasoning in expressive non-classical logics.

Keywords: Higher-Order Logic, Automated Theorem Proving, Reasoning, Non-classical logics

1 Introduction

We present the automated theorem prover Leo-III and its associated system platform. In the DFG funded project a novel agent-based deduction system for classical higher-order logic (HOL) is developed which aims at exploiting massive parallelism at various levels in the reasoning process. The system allows ad-hoc inclusion of independent specialist agents that add advanced functionality to the proof search such as consistency checks of the input axiomatization using model finders or augmented deduction processes for non-classical logics. The latter, very powerful, capability is enabled by semantical embedding of the desired goal logic in HOL. Several of such embeddings will be included in Leo-III, yielding an out-of-the-box automation tool for a great number of (quantified) non-classical logics relevant in mathematics (e.g. inclusive/free logic as used in projective geometry), philosophy (e.g. modal logics) and computer science (e.g many-valued logics, paraconsistent logics).

In its current state, Leo-III is based on an ordered paramodulation calculus for typed lambda-terms, augmented with special means of extensionality treatment. The employment of agents allows parallelism on the search level by introducing and-/or-splits of the search space. The scheduling of the agents' actions is realized as optimization procedure using combinatorical auction games.

2 Classical Higher-Order Logic

Simple type theory, also referred to as classical higher-order logic (HOL), is an expressive logic formalism that allows for higher-order quantification [Fre79],

* This work has been supported by the DFG under grant BE 2501/11-1 (Leo-III).

that is quantification over arbitrary set and function variables. It is based on the simply typed λ-calculus and was, in its current formulation, developed by Church [Chu40]. In the following, we briefly introduce the syntax and semantics of HOL. For thorough discussions we refer to the literature[3].

HOL is a typed logic. The set of *simple types* $\mathcal{T}$ is thereby freely generated using the binary function type constructor $\to$ and the set of base types T. We assume that T consists of at least two elements $\{o, \iota\} \subseteq T$, where o and ι denote the type of Booleans and some non-empty domain of individuals, respectively.

The *terms* of HOL are then given by the following grammar ($\tau, \nu \in \mathcal{T}$):

$$s, t ::= c_\tau \mid X_\tau \mid (\lambda X_\tau .\, s_\nu)_{\tau\to\nu} \mid (s_{\tau\to\nu}\; t_\tau)_\nu \tag{1}$$

where c_τ denotes a typed constant from the signature Σ and X_τ is a variable. The remaining two cases are called *abstraction* and *application*. The type of a term is explicitly stated as subscript but may be dropped for legibility reasons if obvious from the context. Terms s_o of type o are *formulas*.

We require Σ to contain a complete logical signature. To that end, we choose Σ to consist at least of the primitive logical connectives for disjunction, negation, and, for each type, equality and universal quantification. Hence, we have $\{\vee_{o\to o\to o}, \neg_{o\to o}, =^\tau_{\tau\to\tau\to o}\ \Pi^\tau_{(\tau\to o)\to o}\} \subseteq \Sigma$ for all $\tau \in \mathcal{T}$. The remaining logical connectives can be defined as usual, e.g. $s \wedge t := \neg(\neg s \vee \neg t)$.

The semantics of HOL is now briefly addressed. A frame $\{\mathcal{D}_\tau\}_{\tau\in\mathcal{T}}$ is a collection of non-empty sets $\mathcal{D}_\tau$ such that $\mathcal{D}_o = \{T, F\}$ (for truth and falsehood, respectively) and $\mathcal{D}_{\tau\to\nu} \subseteq {\mathcal{D}_\nu}^{\mathcal{D}_\tau}$ is a collection of functions from $\mathcal{D}_\tau$ to $\mathcal{D}_\nu$. An *interpretation* is a pair $\mathcal{M} = (\{\mathcal{D}_\tau\}_{\tau\in\mathcal{T}}, \mathcal{I})$ where $\{\mathcal{D}_\tau\}_{\tau\in\mathcal{T}}$ is a frame and $\mathcal{I}$ is a function mapping each constant c_τ to some denotation in $\mathcal{D}_\tau$. We assume that the primitive logical connectives are assigned their usual denotation. Given a variable assignment σ we can define a valuation $\|.\|^{\mathcal{M},\sigma}$ by

$$\begin{aligned}
\|c_\tau\|^{\mathcal{M},\sigma} &= \mathcal{I}(c_\tau)\\
\|X_\tau\|^{\mathcal{M},\sigma} &= \sigma(X_\tau)\\
\|s_{\tau\to\nu}\; t_\tau\|^{\mathcal{M},\sigma} &= \|s_{\tau\to\nu}\|^{\mathcal{M},\sigma}\; \|t_\tau\|^{\mathcal{M},\sigma}\\
\|\lambda X_\tau .\, s_\nu\|^{\mathcal{M},\sigma} &= \big(f : z \longmapsto \|s\|^{\mathcal{M},\sigma[z/X_\tau]}\big) \in D_{\tau\to\nu}
\end{aligned} \tag{2}$$

A formula s_o is called valid, iff $\|s_o\|^{\mathcal{M},\sigma} = T$ for every variable assignment σ and every interpretation $\mathcal{M}$. We call $\mathcal{M}$ a *standard model* iff $\mathcal{D}_{\tau\to\nu}$ is the complete set of total functions, i.e. $\mathcal{D}_{\tau\to\nu} = \mathcal{D}_\nu^{\mathcal{D}_\tau}$. As a consequence of Gödel's Incompleteness Theorem [Göd31], HOL with standard semantics is necessarily incomplete. However, if we allow $\mathcal{D}_{\tau\to\nu}$ to be a proper subset of $\mathcal{D}_\nu^{\mathcal{D}_\tau}$ with the constraint that $\|.\|$ remains total, a meaningful notion of completeness can be achieved [Hen50]. We assume this so-called *Henkin semantics* in the following.

[3] Detailed information about typed λ-calculi and formal aspects of HOL can e.g. be found in [BDS13,BM14,Ben15a,BBK04] and references therein).

3 Extensional Paramodulation for HOL

The proof search of Leo-III is guided by a refutation-based calculus which uses the fact that $A_1, \dots, A_n \vdash C$ if and only if $\{A_1, \dots, A_n, \neg C\}$ is inconsistent. To that end, the initial set of formulas is transformed into equational clausal normal form and saturated until the empty clause is found. A popular method for saturating a given set of clauses is resolution, i.e. as employed by LEO-II [BPST15]. In first-order theorem proving, many successful systems use calculi based on ordered *paramodulation* [BG94] (or its even more restricted form, *superposition*), which improves naive resolution not only by an appropriate handling of equality, but also by using ordering constraints to restrict the number of possible inferences. In HOL, however, finding appropriate term orderings is more involved and only few such orderings exist.

We now sketch a (unordered) paramodulation rule for HOL and then briefly discuss, how ordering restrictions can be employed for the paramodulation-based calculus of Leo-III.

An equation is a pair $s \simeq t$ of terms. A literal is a signed equation, written $[s \simeq t]^\alpha$ where $\alpha \in \{\mathfrak{tt}, \mathfrak{ff}\}$ is the polarity of the literal. A clause $\mathcal{C}$ is a multiset of literals, denoting its disjunction. For brevity, if $\mathcal{C}$ and $\mathcal{D}$ are clauses and l is a literal, we write $\mathcal{C} \vee l$ and $\mathcal{C} \vee \mathcal{D}$ for the multi-union of $\mathcal{C} \cup \{l\}$ and $\mathcal{C} \cup \mathcal{D}$, respectively. The paramodulation inference can then be stated as

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

where negative equality literals encode postponed unification tasks, $s|_\pi$ 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. Intuitively, paramodulation is a conditional rewriting step that is justified if the unification tasks can be solved. Further calculus rules include equality factoring, unification handling and clausification.

The above rule (Para) is unordered and will, especially in a higher-order setting, produce a lot of irrelevant (redundant) clauses in the search space. In order to restrict the inference rules such as (Para), we are employing a higher-order term ordering primarily investigated for automated termination proofs, called *computability path ordering* (CPO) [BJR15].

In its current state, we successfully use CPO to orient equations and preselect maximal literals eligible for paramodulation and factorization inferences. However, the employment of full ordered paramodulation constraints, that additionally discard generated clauses that do not match the ordering constraints after unification, is not yet implemented. This is partly due to the complicated nature of higher-order unification where, in general, there exists no most general unifier between two terms. Nevertheless, already at this point, the use of CPO seems promising as a candidate ordering towards a fully ordered paramodulation calculus for HOL.

In addition to the usual paramodulation inference rules above, extensionality aspects need to be considered explicitly as well. This is because equalities in HOL

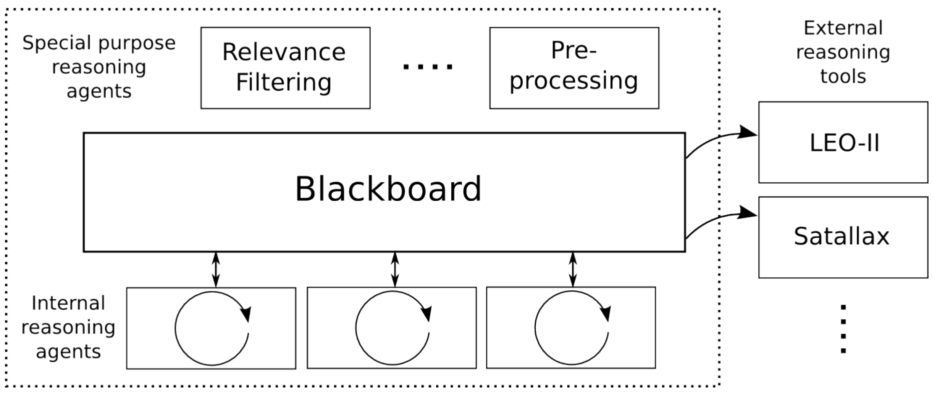

Fig. 1. Leo-III's agent-based proof search cooperation

can occur between terms of any type, in particular between terms of Boolean type or function type. The mere addition of extensionality axioms for each relevant symbol does not suffice, as it leads to a massive explosion of the search space. Hence, we include special means of calculus-level treatment similar to the rules used by LEO-II [BPST15], and combine them with extensionality handling in adequate preprocessing steps [WSKB16].

The overall saturation procedure consisting of the above sketched ingredients is, at the moment, organized as a sequential loop using a variant of the *given-clause* algorithm.

4 Agent-Based Refutation

An *agent* is a software component that can be executed independently of others. Moreover, an agent is given the ability to decide on its own when to execute its functionality. This high amount of autonomy is a key feature of agents [Wei13]. In the Leo-III system, agents are employed as *specialists* for some aspects of the proof search. The underlying architecture of Leo-III is designed as a blackboard which the agents use to collaboratively find a proof. The work of the agents is thereby divided in transactional tasks and organized in auctions, in which it is decided which tasks are performed next in case of interference.[4]

In its current state, Leo-III employs agents in three different scenarios: During the preprocessing phase of the overall proof procedure each agent is responsible (i.e. a specialist) for one sort of normalization to be applied to a formula. Here, the overall goal is to exhaustively apply all normalization procedures [WSKB16] to all clauses. Since normalization is a local problem, an agent can judge solely by observing a particular formula whether it wants to act on it. Due to only little existing interference between the normalization methods, the execution of the different normalization routines can easily be distributed among the agents.

[4] Further information can e.g. be found in [WB16,Wis14].

As a second employment scenario, a relevance filter (cf. [MP09]) is implemented that prunes the search space prior to preprocessing. Relevance filtering can be performed similarly to the preprocessing, except that the problem is not local in the above sense since information about other formulas have to be considered as well. As most of the agents of Leo-III will have this kind of non-local dependencies, a reasonable coordination of those agents is one of the main goals of further development. The last employment of the agents is to parallelize heavy weighted proof procedures. Here, the agent-based approach can be applied on the calculus level, e.g. for single clauses, but also for parallelizing one or more (sequential) proof procedures (so-called multi search). In Leo-III, the calculus sketched in §3 is distributed among multiple agents. Additionally, HOL theorem proving systems such as LEO-II and Satallax [Bro12] are included as external specialists.

In Fig. 1 the connections among the components of Leo-III are visualized. The focus in the current state relies mostly on the last of the three above described cases. We employ sequential proof procedures and external provers to solve the input problem in parallel and wait for the first positive result. These tasks either differ in some parameters of the proof search or in previously applied normalization techniques. For further work, we will experiment with different granularities for the generated tasks and different means of agent coordination. The task sizes can hereby vary from the execution of whole proof procedures to very fine-grained responsibilities (e.g. the application of single inference steps). The coordination is at the lowest level bound to the auction system. On top of that however, additional mechanisms (such as fixed execution priorities, or coalitions and coalition games [CEW11]) can be added.

5 The Leo-III System

In its core, Leo-III is a new higher-order automated theorem prover based on the associated system platform LeoPARD [WSB15]. LeoPARD is a framework for deduction systems (implemented in Scala) providing sophisticated term, search, and indexing data structures for typed λ-terms, as well as an generic agent-based blackboard architecture. Leo-III makes use of these supported data structures and implements the concrete agents as described in §4 on top of the provided blackboard architecture. The internal reasoning agents implement a proof procedure realizing the calculus depicted in §3.

During the development of Leo-III, special care was given to providing maximal compatibility with existing systems and conventions of the application area. As input language, for instance, Leo-III supports every standard dialect of the TPTP syntax [Sut09] (including THF, TFF and FOF). For best possible external utilization, Leo-III can output a proof object used for proof reconstruction pointers (e.g. in Isabelle [NPW02]) or proof verification tools (e.g. IDV [Sut09]).

One major goal of Leo-III is to provide native means of reasoning within (and about) non-classical logics including free logic, (quantified) conditional logic, and (quantified) modal logic[5]. Such logics are of strong interest in many different fields of research, for example in mathematics, artificial intelligence, and philosophy. In its current state, our system is already capable of reasoning in that embedded logics and even – with a few modifications – of parsing the syntax representation of these formalisms. The automated transformation of an input problem stated in such a specialized syntax representation into an equivalent HOL formulation is still in development, but can easily be added to Leo-III as a new preprocessing procedure.

6 Conclusion

In this paper, we have presented a new automated theorem prover for HOL, called Leo-III. A rough sketch of its underlying paramodulation calculus and its extensionality handling has been given. A proof procedure based on that calculus is included in the presented agent-based blackboard architecture. Additionally, external deduction systems can be included as agents. Leo-III is, on the long perspective, intended as a platform for *universal reasoning*, offering not only support for reasoning in classical higher-order logics, but also for reasoning within further expressive, non-classical logics such as modal logics, conditional logics or even many-valued logics. This enables our system to serve as a reasoning tool for a wide spectrum of formal scientific disciplines.

A native out-of-the-box automation of input problems stated in specialized syntax of corresponding non-classical logics is further work. Additionally, we will develop and include further specialized agents to allow a more fine-grained parallelization of the proof search. To that end, experiments with various parameters and heuristics for the guidance of the proof search and the organization of the agents will be conducted.

References

[BBK04] C. Benzmüller, C. Brown, and M. Kohlhase. Higher-Order Semantics and Extensionality. *Journal of Symbolic Logic*, 69(4):1027–1088, 2004.

[BDS13] H. P. Barendregt, W. Dekkers, and R. Statman. *Lambda Calculus with Types*. Perspectives in logic. Cambridge University Press, 2013.

[Ben15a] C. Benzmüller. Higher-Order Automated Theorem Provers. In D. Delahaye and B. Woltzenlogel Paleo, editors, *All about Proofs, Proof for All*, Mathematical Logic and Foundations, pages 171–214. College Publications, London, UK, 2015.

[Ben15b] C. Benzmüller. Invited talk: On a (quite) universal theorem proving approach and its application in metaphysics. In De Nivelle. H., editor, *TABLEAUX 2015*, volume 9323 of *LNAI*, pages 209–216, Wroclaw, Poland, 2015. Springer. (Invited paper, mildly reviewed).

[5] The reasoning in such non-classical logics is enabled by a semantical embedding of the target logic into HOL. Detailed information about this approach can be found, e.g. in [Ben15b] and the references therein.

[BG94] L. Bachmair and H. Ganzinger. Rewrite-Based Equational Theorem Proving with Selection and Simplification. *J. Log. Comput.*, 4(3):217–247, 1994.

[BJR15] F. Blanqui, J.-P. Jouannaud, and A. Rubio. The computability path ordering. *CoRR*, abs/1506.03943, 2015.

[BM14] C. Benzmüller and D. Miller. Automation of Higher-Order Logic. In D. M. Gabbay, J. H. Siekmann, and J. Woods, editors, *Handbook of the History of Logic, Volume 9 — Computational Logic*, pages 215–254. North Holland, Elsevier, 2014.

[BPST15] C. Benzmüller, L. C. Paulson, N. Sultana, and F. Theiß. The Higher-Order Prover LEO-II. *Journal of Automated Reasoning*, 55(4):389–404, 2015.

[Bro12] C. E. Brown. Satallax: An Automatic Higher-Order Prover. In B. Gramlich, D. Miller, and U. Sattler, editors, *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, volume 7364 of *LNCS*, pages 111–117. Springer, 2012.

[CEW11] G. Chalkiadakis, E. Elkind, and M. Wooldridge. *Computational Aspects of Cooperative Game Theory*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011.

[Chu40] A. Church. A formulation of the simple theory of types. *J. Symb. Log.*, 5(2):56–68, 1940.

[Fre79] G. Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Verlag von Louis Nebert, Halle, 1879.

[Göd31] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. *Monatshefte für Mathematik und Physik*, 38(1):173–198, 1931.

[Hen50] L. Henkin. Completeness in the theory of types. *J. Symb. Log.*, 15(2):81–91, 1950.

[MP09] J. Meng and L. C. Paulson. Lightweight relevance filtering for machine-generated resolution problems. *Journal of Applied Logic*, 7(1):41 – 57, 2009.

[NPW02] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[Sut09] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. *J. Autom. Reasoning*, 43(4):337–362, 2009.

[WB16] M. Wisniewski and C. Benzmüller. Is it Reasonable to Employ Agents in Theorem Proving? In J. van den Heerik and J. Filipe, editors, *Proc. of the 8th International Conference on Agents and Artificial Intelligence (ICAART)*, volume 1, pages 281–286, Rome, Italy, 2016. SCITEPRESS – Science and Technology Publications, Lda.

[Wei13] G. Weiss, editor. *Multiagent Systems*. MIT Press, 2013.

[Wis14] M. Wisniewski. Agent-based Blackboard Architecture for a Higher-Order Theorem Prover. Master's thesis, Freie Universität Berlin, 2014.

[WSB15] M. Wisniewski, A. Steen, and C. Benzmüller. LeoPARD - A Generic Platform for the Implementation of Higher-Order Reasoners. In M. Kerber, J. Carette, C. Kaliszyk, F. Rabe, and V. Sorge, editors, *Intelligent Computer Mathematics - International Conference, CICM, Proceedings*, volume 9150 of *LNCS*, pages 325–330. Springer, 2015.

[WSKB16] M. Wisniewski, A. Steen, K. Kern, and C. Benzmüller. Effective Normalization Techniques for HOL. In N. Olivetti and A. Tiwari, editors, *Automated Reasoning, Eight International Joint Conference, Proceedings*, LNCS. Springer, 2016. Accepted for publication; to appear in 2016.