Abstract
Automation of classical higher-order logic faces various theoretical and practical challenges. On a theoretical level, powerful calculi for effective equality reasoning from first-order theorem proving cannot be lifted to the higher-order domain in a simple manner. Practically, implementations of higher-order reasoning systems have to incorporate procedures that often have high time complexity or are not decidable in general. In my dissertation, both the theoretical and the practical challenges of designing an effective higher-order reasoning system are studied. The resulting system, the automated theorem prover Leo-III, is one of the most effective and versatile systems, in terms of supported logical formalisms, to date.

Keywords
Higher-order logic · Automated theorem proving · Henkin semantics · Quantified modal logics

1 Introduction
Automated theorem proving (ATP) denotes the automation of deduction procedures that, given a set of assumptions and a conjecture as input, try to decide whether the input conjecture is a logical consequence of the assumptions. In the context of ATP systems, this reasoning process is done fully automatically, i.e., without any further user-interaction during proof search.

The term higher-order logic refers to expressive logical formalisms that allow for quantification over predicate and function variables; such a logic was first studied by Frege in the 1870s [11], and later presented by Church as a language based on top of a simply typed Lamda-calculus [10]. The latter formalism was further studied and refined by Henkin [13], Andrews [1, 2] and others [7, 14]. In the remainder, the term HOL is used synonymously to Henkin’s Extensional Type Theory (ExTT); it constitutes the foundation of most contemporary higher-order automated reasoning systems [8].

HOL provides lambda-notation to denote unnamed functions, predicates and sets, and comes with built-in principles of Boolean and functional extensionality as well as type-restricted comprehension, cf. [8] and the references therein.

Automation of HOL has made major progress in the last years and increasingly many higher-order ATP systems exist. Still, HOL ATP systems are not yet as mature as their propositional and first-order counterparts. In particular, effective handling of interpreted equality is challenging as sophisticated methods from first-order automation, such as superposition-based approaches, do not exist. In the author’s dissertation [16], a calculus based on extensional paramodulation is developed that aims at improved equality handling in HOL. It is called extensional as the extensionality principles of HOL are handled at the calculus level, rather than by adding additional axioms to the search space as done in earlier approaches. This drastically reduces the number of non-productive inferences and improves proof search in practice. Its implementation, the novel ATP system Leo-III [17], is one of the most effective HOL ATP systems available to date [9, 16]. Additionally, Leo-III offers effective automation of various quantified non-classical logics including many higher-order modal logics [12].
2 Extensional Paramodulation for HOL

Paramodulation extends resolution by a native treatment of equality at the calculus level. In the context of first-order logic, it was developed in the late 1960s by Robinson and Wos [15] as an attempt to overcome the shortcomings of resolution-based approaches to handling equality; a paramodulation inference can be regarded as a speculative conditional rewriting step. In first-order theorem proving, superposition-based calculi [3] improve the naive paramodulation approach by imposing ordering restrictions on the inference rules such that only a relevant subset of all possible inferences are generated. Most of the state-of-the-art first-order theorem provers implement some variant of superposition techniques to the higher-order setting [16].

In the author’s dissertation, an extensional paramodulation calculus, denoted EP, is presented that addresses the shortcomings of resolution-based approaches for HOL automation. Although it is unordered, and hence subject to a more prolific proof search, it represents a step towards effective equality reasoning in HOL by handling equality as a native concept rather than a defined notion as done in previous work [4, 16].

The calculus rules of EP can be grouped as follows:

1. Clausification: These rules transform the input formulas into an equisatisfiable set of clauses in clause normal form (CNF). The rules are mostly standard, but some advanced normalization techniques can be applied [21].

2. Primary Inferences: The primary inferences of EP are paramodulation (Para), factorization (Fac) and primitive substitution (PS), cf. Fig. 1. Rule (PS) instantiates free variables at top-level with approximation of predicate formulas using so-called general bindings $GB$ [5]. Unification constraints are encoded as negative literals into the result clause.

3. Unification: A variant of Huet’s higher-order unification is employed in the EP calculus, cf. [16] for details.

4. Extensionality Rules: The extensionality rules of EP, cf. Fig. 1, handle aspects of functional extensionality (PFE and NFE) as well as principles of Boolean extensionalitiy (PBE and NBE) on the calculus level. This way, no explicit extensionality axioms are required in the search space.

A set of sentences is said to have a refutation in EP iff an empty clause can be derived in EP. A clause is called empty, if it only consists of unification constraints with variables on head position on both sides [16].

\[ \frac{C \lor [s \simeq t]_{\alpha}}{D \lor [\nu \simeq v]_{\beta}} \quad \text{(Para)} \]

\[ \frac{C \lor [s \simeq t]_{\alpha} \land \varphi \lor [s \simeq t]_{\beta}}{C \lor [s \simeq t]_{\alpha} \land \varphi \lor [s \simeq t]_{\beta}} \quad \text{(Fac)} \]

\[ \frac{C \lor [s \simeq t]_{\alpha} \lor [u \simeq v]_{\beta}}{C \lor [s \simeq t]_{\alpha} \lor [u \simeq v]_{\beta}} \]

\[ \frac{C \lor [H \simeq G]_{\beta}}{C \lor [H \simeq G]_{\beta}} \quad \text{(PS)} \]

\[ \frac{\exists x \in \Gamma, \alpha \in \{\top, \bot\} \quad \text{if } s_{x} \text{ is of type } \nu \text{ and } \text{fv}(s_{x}) \subseteq \text{fv}(s)}{\}

\[ \frac{C \lor [s \simeq t]_{\alpha}}{D \lor [\nu \simeq v]_{\beta}} \quad \text{(PBE)} \]

\[ \frac{C \lor [s]_{\alpha} \lor [t]_{\beta}}{C \lor [s]_{\alpha} \lor [t]_{\beta}} \quad \text{(NBE)} \]

\[ \frac{C \lor [s]_{\alpha} \lor [t]_{\beta}}{C \lor [s]_{\alpha} \lor [t]_{\beta}} \]

\[ \frac{C \lor [s \simeq t]_{\alpha} \lor [t \simeq s]_{\beta}}{C \lor [s \simeq t]_{\alpha} \lor [t \simeq s]_{\beta}} \quad \text{(PFE)} \]

\[ \frac{C \lor [s \simeq t]_{\alpha} \lor [s \simeq t]_{\beta}}{C \lor [s \simeq t]_{\alpha} \lor [s \simeq t]_{\beta}} \quad \text{(NFE)} \]

Fig. 1 Primary inference rules and extensionality rules of EP. Technical notes: $s \simeq t$ denotes an equation, where $\simeq$ is assumed to be symmetric. A literal $\ell$ is a signed equation, written $[s \simeq t]$ where $s, t \in \Gamma$. Literals of form $[s \simeq t]_{\alpha}$ are a shorthand for $[s \simeq t]_{\alpha}$. A clause $C$ is a multiset of literals, denoting its disjunction. For brevity, if $C, D$ are clauses and $\ell$ is a literal, $C \lor \ell$ and $C \lor \ell \lor D$ denote the multi-union $C \cup \ell$ and $C \cup \ell \lor D$, respectively. $s[x]_{\alpha}$ is the sub-term of $s$ at position $x$, and $s[r]_{\alpha}$ denotes the term that is created by replacing the subterm of $s$ at position $x$ by $r$. The free variables of a term $t$ are given by $\text{fv}(t)$.

Theorem 1 EP is sound and refutationally complete for HOL with Henkin semantics.

Proof See [16, §3].

3 The Leo-III Theorem Prover

Leo-III is an effective ATP system for HOL with Henkin semantics and choice [16, 17]. It is implemented in Scala; its source code, and that of related projects, is publicly available under BSD-3 license on GitHub. The system accepts all common TPTP input syntax formats [19], including untyped clause normal form (CNF), untyped and typed first-order logic (FOF and TFF, respectively) and, as primary input format, monomorphic higher-order logic (THF). Additionally, as one of the first higher-order ATP systems, Leo-III
supports reasoning in rank-1 polymorphic variants of the above logics (TF1 and TH1 syntax) [18]. The prover returns results according to the standardized TPTP SZS ontology [19] and produces a verifiable TPTP-compatible proof certificate, if a proof is found.

3.1 Architecture and Implementation

Leo-III is a refutational reasoning system. This means that the initial, possibly empty, set of axioms and the negated conjecture are transformed into an equisatisfiable set of formulas in clause normal form, which is then iteratively saturated until the empty clause is found. Leo-III extends the EP calculus with practically motivated, partly heuristic inference rules. These rules include, among others, equational simplification, function synthesis, and choice reasoning routines [17]. Additionally, Leo-III uses several heuristics to restrict the number of inferences, including others, equational simplification, function synthesis, and choice reasoning routines [17].

An overview of Leo-III’s top-level architecture is displayed at Fig. 2. After parsing the problem statement, a symbol-based relevance filter is used for premise selection. This may reduce the number of considered axioms in large problems by magnitudes. The proof search is organized as a sequential procedure that iteratively saturates the set of input clauses w.r.t. the extended EP calculus. The saturation process is thereby controlled by a dedicated Control module that may restrict or guide the application of calculus rules. Indexing data structures are employed for speeding up frequently used procedures. Leo-III collaborates with external reasoning systems, in particular, with first-order ATPs such as E, iProver and Vampire as well as SMT solvers, e.g. with CVC4. Cooperation is not restricted to first-order systems, and further specialized systems such as higher-order model finders may be utilized by Leo-III.

Apart from handling equality, the implementation of an effective theorem proving system for HOL poses various challenges: Due to the nature of HOL, tedious and computationally heavy procedures are required in order to produce a theoretically complete reasoning system. Also, unification, which is one of the most central procedures in theorem proving—is neither decidable nor unitary for the case of HOL. Finally, extensionality aspects need to be addressed. A more exhaustive discussion of challenges to HOL automation are discussed in the authors thesis [16].

3.2 Reasoning in Non-Classical Logics

A long term goal of the Leo-III project is to provide means for reasoning in expressive non-classical logics (NCL) [20]. This is enabled by utilizing shallow semantic embedding (SSE) approaches that encode various logics into HOL [6], thus making HOL ATP systems applicable to non-classical logics. One particular class of NCLs that has been shallowly embedded into HOL are higher-order modal logics (HOML). Modal logics have many relevant applications in computer science, artificial intelligence, mathematics, computational linguistics and philosophy. Many challenging applications, as recently explored in metaphysics, require first-order or even higher-order quantified modal logics. The development of ATPs for these logics, however, is still in its infancy.

Leo-III is addressing this gap. In addition to its classical 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 SSE procedure [12] that is fully integrated into its automated proof search. When taking the different semantical parameter combinations into account, Leo-III supports more than 120 different HOMLs. This work has pioneered an ongoing extension of the TPTP library and framework towards a standard representation for quantified modal logic problems and their semantics.²

3.3 Evaluation

Multiple evaluation studies underline the practical contribution of the Leo-III prover to the field: in the author’s thesis, an extensive evaluation on different benchmarks sets, including monomorphic and polymorphic HOL problems as well as modal logic problems, display its effectiveness in different application areas [16, §6]. A large independent evaluation study, called GRUNGE, suggests that Leo-III is the most effective reasoning system and, in terms of supported logical formalisms, also the most versatile to date [9]. Additionally, the effectiveness of Leo-III was confirmed in the most recent

² See http://tptp.org/TPTP/Proposals/LogicSpecification.html for more information.
ATP system competition (CASC), where Leo-III won the Large Theory Batch division that is of particular relevance for practical applications of ATP systems.3

4 Conclusion

In the author’s dissertation, the EP calculus for automated HOL reasoning is introduced. The practical part of the thesis presents the state-of-the-art higher-order reasoning system Leo-III, which is based on an extension of EP. Due to its wide range of supported classical and non-classical logical formalisms, including polymorphic higher-order and first-order logic, and numerous quantified modal logics, the system has many topical applications in computer science, AI, maths and philosophy. Several evaluation studies on heterogeneous benchmark sets shows that Leo-III is one of the most effective HOL ATP systems to date [9, 16].

Further work includes the integration of more SSEs into Leo-III and a formal study of a polymorphic variant of the EP calculus.

References

1. Andrews PB (1972) General models and extensionality. J Symb Log 37(2):395–397. https://doi.org/10.2307/2272982
2. Andrews PB (1972) General models, descriptions, and choice in type theory. J Symb Log 37(2):385–394. https://doi.org/10.2307/2272981
3. Bachmair L, Ganzinger H (1994) Rewrite-based equational theorem proving with selection and simplification. J Log Comput 4(3):217–247. https://doi.org/10.1093/logcom/4.3.217
4. Benzmüller C (1999) Extensional higher-order paramodulation and RUE-resolution. In: CADE-16, LNCS, 1632:399–413. Springer
5. Benzmüller C (2015) Higher-order automated theorem provers. In: Delahaye D, Woltzenlogel Paleo B (eds) All about proofs, proof for all, mathematical logic and foundations. All about proofs, proof for all, mathematical logic and foundations. London, pp 171–214
6. Benzmüller C (2019) Universal (meta-)logical reasoning: recent successes. Sci Comput Progr 172:48–62. https://doi.org/10.1016/j.scico.2018.10.008
7. Benzmüller C, Brown CE, Kohlhase M (2004) Higher-order semantics and extensionality. J Symb Log 69(4):1027–1088. https://doi.org/10.2178/jsl/1102022211
8. Benzmüller C, Miller D (2014) Automation of higher-order logic. In: Siefmann JH (ed) Computational logic, handbook of the history of logic, vol 9. Elsevier, Amsterdam, pp 215–254. https://doi.org/10.1016/B978-0-444-51624-4.50005-8
9. Brown C, et al. (2019) GRUNGE: A Grand Unified ATP Challenge. In: P. Fontaine (ed.) Automated deduction – CADE-27, LNCS, vol. 11716, pp. 123–141. Springer. https://doi.org/10.1007/978-3-030-29436-6_8
10. Church A (1940) A formulation of the simple theory of types. J Symb Log 5(2):56–68. https://doi.org/10.2307/2266170
11. Frege G (1879) Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens. Verlag von Louis Nebert, Halle
12. Gleißner T, Steen A, Benzmüller C (2017) Theorem provers for every normal modal logic. In: T. Eiter, D. Sands (eds.) LPAR-21, EPICS Series in Computing, vol. 46, pp. 14–30. EasyChair. https://doi.org/10.29007/jsb9
13. Henkin L (1950) Completeness in the theory of types. J Symb Log 15(2):81–91. https://doi.org/10.2307/2266967
14. Muskens R (2007) Intensional models for the theory of types. J Symb Log 72(1):98–118. https://doi.org/10.2178/jsl/1174668386
15. Robinson G, Wos L (1969) Paramodulation and theorem-proving in first-order theories with equality. Mach Intell 4:135–150
16. Steen A (2018) Extensional Paramodulation for Higher-Order Logic and its Effective Implementation Leo-III, DISKI – Dissertations in Artificial Intelligence, vol. 345. Akademische Verlagsgesellschaft AKA GmbH, Berlin, Dissertation. Freie Universität Berlin, Germany
17. Steen A, Benzmüller C (2018) The higher-order prover Leo-III. In: D. Galmiche, et al. (eds.) IJCAR 2018, LNCS, vol. 10900, pp. 108–116. Springer. https://doi.org/10.1007/978-3-319-94205-6_8
18. Steen A, et al (2017) Going polymorphic—TH1 reasoning for Leo-III. In: Eiter T et al (eds) IJWIL@LPAR 2017, Kalpa Publications in Computing, vol. 1. EasyChair. https://doi.org/10.29007/gkww
19. Sutcliffe G (2017) The TPTP problem library and associated infrastructure—from CNF to th0. TPTP v6.4.0. J Autom Reason 59(4):483–502
20. Wisniewski M, Steen A, Benzmüller C (2016) TPTP and beyond: Representation of quantified non-classical logics. In: C. Benzmüller, J. Otten (eds.) ARQNL 2016, CEUR Workshop Proceedings, vol. 1770, pp. 51–65. CEUR-WS.org
21. Wisniewski M, Steen A, Kern K, Benzmüller C (2016) Effective normalization techniques for HOL. In: Olivetti N, Tiwari A (eds) IJCAR 2016, LNCS, vol. 9706, pp. 362–370. Springer. https://doi.org/10.1007/978-3-319-40229-1_25

3 See tptp.org/CASC/27 for details on the competition and its results.