

LEO-II - A Cooperative Automatic Theorem Prover for Classical Higher-Order Logic^{*}

Christoph Benzmüller¹, Lawrence C. Paulson², Frank Theiss¹, and Arnaud Fietzke³

¹Dep. of Computer Science, Saarland University, Saarbrücken, Germany

²Computer Laboratory, The University of Cambridge, UK

³Max Planck Institute for Informatics, Saarbrücken, Germany

Abstract. LEO-II is a standalone, resolution-based higher-order theorem prover designed for effective cooperation with specialist provers for natural fragments of higher-order logic. At present LEO-II can cooperate with the first-order automated theorem provers E, SPASS, and Vampire. The improved performance of LEO-II, especially in comparison to its predecessor LEO, is due to several novel features including the exploitation of term sharing and term indexing techniques, support for primitive equality reasoning, and improved heuristics at the calculus level. LEO-II is implemented in Objective Caml and its problem representation language is the new TPTP THF language.

1 Introduction and Motivation

Automatic theorem provers (ATPs) based on the resolution principle, such as Vampire [18], E [19], and SPASS [23], have reached a high degree of sophistication. They can often find long proofs even for problems having thousands of axioms. However, they are limited to first-order (FO) logic. Higher-order (HO) logic extends FO logic with lambda notation for functions and with function and predicate variables. It supports reasoning in set theory, using the obvious representation of sets by predicates. HO logic is a natural language for expressing mathematics, and it is also ideal for formal verification. Moving from FO to HO logic requires a more complicated proof calculus, but it often allows much simpler problem statements. HO logic's built-in support for functions and sets often leads to shorter proofs. Conversely, elementary identities (such as the distributive law for union and intersection) turn into difficult problems when expressed in FO form.

LEO-II is a standalone, resolution-based HO ATP that is designed for fruitful cooperation with specialist provers for fragments of HO logic. The idea is to combine the strengths of the different systems. On the other hand, LEO-II itself, as an external reasoner, is designed to support HO proof assistants such as Isabelle/HOL [16], HOL [11], and Ω MEGA [20] by efficiently automating subproblems and thereby reducing user effort.

^{*} The LEO-II project was funded by EPSRC under grant EP/D070511/1, "LEO-II: An Effective Higher-Order Theorem Prover."

1.1 Motivation for LEO-II

LEO-II is the successor of LEO [4], which was implemented in Allegro Common Lisp as a part of the Ω MEGA system and which unfortunately was not available as a standalone reasoner. LEO shared the basic data structures such as terms and clauses with Ω MEGA; these data structures were not designed for efficiency and their term indexing support was limited [13]. LEO's performance strongly improved after coupling it with the FO ATP Otter in the agent based Ω ANTS framework [9]. This integration has subsequently been improved and Otter has been replaced by Bliksem and Vampire. This cooperative theorem proving approach [8] outperforms FO ATPs such as Vampire on problems about sets, relations, and functions as given in the TPTP SET domain (see Figure 1). The cooperative approach not only solves more problems in this domain; it also solves them more efficiently. This provides evidence for the following working hypothesis:

It is well known in AI that representation matters. Analogously, problem representation matters in automated theorem proving: many proof problems can be effectively solved when they are initially represented in a suitable way in HO logic and then tackled with a cooperative theorem proving approach in which a HO reasoner subsequently reduces them to a suitable fragment of HO logic in which they can be tackled by an effective specialist reasoner.

So far, we consider only FO logic as a target fragment. The general idea, however, is not limited to this; other fragments will be studied in future work. Examples include decidable fragments like monadic second-order logic and the guarded fragment. Decidability can probably produce useful feedback for improved heuristic control in the HO ATP.

2 Overview on LEO-II

The results of the LEO-II project (an one year project at Cambridge Univ.) are:

1. LEO-II has been (re-)implemented in Objective Caml 3.09 (approx. 12500 lines of code) as a standalone HO ATP. It can be easily installed, deployed and integrated with other reasoners. The sources are available from the LEO-II web-site at: <http://www.ags.uni-sb.de/~chris/leo/>.
2. Proof search in LEO-II, which is based on the extensional HO resolution calculus [3] has been further improved, e.g., it now supports efficient (recursive) expansion of definitions and primitive equality reasoning (see Section 3).
3. For cooperation with FO ATPs, LEO-II offers two different HO to FO mappings; further mappings can easily be added.
4. LEO-II provides efficient term data structures. It employs perfect term sharing and supports for HO term indexing. LEO-II also provides analysis tools for exploring its proof object, term graph and term index. This includes statistical analysis of the term graph (see Section 4).
5. In addition to a fully automatic mode, LEO-II also provides an interactive mode [6]. This mode supports debugging and inspection of the search space,

Problem	Vampire 9.0	LEO+Vamp.	LEO-II+E	Problem	Vamp. 9.0	LEO+Vamp.	LEO-II+E
SET014+4	114.5	2.60	0.300	SET648+3	98.2	0.12	0.037
SET017+1	1.0	5.05	0.059	SET649+3	117.5	0.25	0.037
SET066+1	-	3.73	0.029	SET651+3	117.5	0.09	0.029
SET067+1	4.6	0.10	0.040	SET657+3	146.6	0.01	0.028
SET076+1	51.3	0.97	0.031	SET669+3	83.1	0.20	0.041
SET086+1	0.1	0.01	0.028	SET670+3	-	0.14	0.067
SET096+1	5.9	7.29	0.033	SET671+3	214.9	0.47	0.038
SET143+3	0.1	0.31	0.034	SET672+3	-	0.23	0.034
SET171+3	68.6	0.38	0.030	SET673+3	217.1	0.47	0.042
SET580+3	0.0	0.23	0.078	SET680+3	146.3	2.38	0.035
SET601+3	1.6	1.18	0.089	SET683+3	0.3	0.27	0.053
SET606+3	0.1	0.27	0.033	SET684+3	-	3.39	0.039
SET607+3	1.2	0.26	0.036	SET716+4	-	0.40	0.033
SET609+3	145.2	0.49	0.039	SET724+4	-	1.91	0.032
SET611+3	0.3	4.00	0.125	SET741+4	-	3.70	0.042
SET612+3	111.9	0.46	0.030	SET747+4	-	1.18	0.028
SET614+3	3.7	0.41	0.060	SET752+4	-	516.00	0.086
SET615+3	103.9	0.47	0.035	SET753+4	-	1.64	0.037
SET623+3	-	2.27	0.282	SET764+4	0.1	0.01	0.032
SET624+3	3.8	3.29	0.047				
SET630+3	0.1	0.05	0.025				
SET640+3	1.1	0.01	0.033				
SET646+3	84.4	0.01	0.032				
SET647+3	98.2	0.12	0.037				

Vamp. 9.0: 2.80GHz, 1GB memory, 600s time limit
LEO+Vamp.: 2.40GHz, 4GB memory, 120s time limit
LEO-II+E: 1.60GHz, 1GB memory, 60s time limit

Fig. 1. The new LEO-II+E cooperation strongly outperforms the old LEO+Vampire cooperation on HO encodings of proof problems as given in the TPTP SET domain (these HO encodings are distributed with LEO-II). The worst approach is to tackle these problems in their original, less elegant FO encoding with Vampire.

but also the tutoring of resolution based HO theorem proving to students. The interactive mode and the automatic mode can be interleaved.

6. LEO-II provides some basic support for prefix-polymorphism; full polymorphism adds many non-trivial choice points to the already challenging search space of LEO-II [6] and is therefore future work.
7. LEO-II employs the new TPTP THF representation language [7].

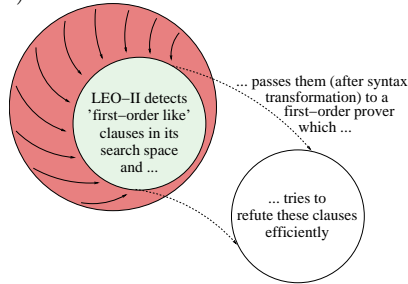
At present, LEO-II cooperates with FO ATPs only in a sequential mode and not via the agent based architecture Ω ANTS. Figure 1 shows that LEO-II's performance has nevertheless strongly improved.

In the remainder we sketch LEO-II's cooperative proof search (Section 3) and its term sharing and term indexing (Section 4).

3 Cooperative Proof Search

LEO-II's clause set generally consists of HO clauses (dark-colored area), which are processed with LEO-II's calculus rules. Some of the clauses in LEO-II's search space additionally attain a special status (the ones in the light-colored area): they are in the domain of a transformation function to a particular fragment of HO logic, here FO logic. Light-colored clauses are available to both LEO-II's proof search and to a specialist prover for the target fragment via the transformation function.

Roughly speaking, currently all clauses which do not contain any λ -term and embedded predicate or proposition are light-colored and our default FO transformation function employs Kerber's ideas [12]: it recursively translates every



application $(p_{\alpha \rightarrow \beta} q_{\alpha})$ into a term $@^{((\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta)}(p, q)$, where $@^{((\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta)}$ is a new function or predicate constant. LEO-II's extensional HO resolution calculus is well suited to subsequently generate more and more light-colored clauses from dark colored ones. In some proof problems (such as Examples 1 and 2 below) the light-colored area quickly collects enough information for constructing a refutation; however, LEO-II is often too weak to find this refutation on its own. In other proof problems (see Example 3 below) the refutation can only be found in the dark-colored area. This observation motivates a distributed architecture in which LEO-II dynamically cooperates with incremental specialist reasoners; future work includes the adaptation of the Ω ANTS framework to LEO-II. At present, LEO-II sequentially launches a fresh call of the cooperating specialist prover every n iterations (currently $n = 10$).

Next, we discuss three proof problems from the domain of normal multimodal logics. For this domain, our cooperative approach yields elegant problem encodings and efficient solutions [5].

The encoding of multimodal logic in simple type theory is intuitive. Choose a base type — we choose ι — to denote the set of all possible worlds. Certain formulas of type $\iota \rightarrow o$ then correspond to multimodal logic expressions. The modal operators \neg , \vee , and \Box_R become λ -terms of types $(\iota \rightarrow o) \rightarrow (\iota \rightarrow o)$, $(\iota \rightarrow o) \rightarrow (\iota \rightarrow o) \rightarrow (\iota \rightarrow o)$, and $(\iota \rightarrow \iota \rightarrow o) \rightarrow (\iota \rightarrow o) \rightarrow (\iota \rightarrow o)$ respectively. Note that \neg forms the complement of a set of worlds, while \vee forms the union of two such sets. \Box_R explicitly abstracts over the accessibility relation R :

$$\begin{aligned}\neg_{(\iota \rightarrow o) \rightarrow (\iota \rightarrow o)} &= \lambda A_{\iota \rightarrow o} \bullet \lambda X_{\iota} \bullet \neg A X \\ \vee_{(\iota \rightarrow o) \rightarrow (\iota \rightarrow o) \rightarrow (\iota \rightarrow o)} &= \lambda A_{\iota \rightarrow o} \bullet \lambda B_{\iota \rightarrow o} \bullet \lambda X_{\iota} \bullet A X \vee B X \\ \Box_R_{(\iota \rightarrow \iota \rightarrow o) \rightarrow (\iota \rightarrow o) \rightarrow (\iota \rightarrow o)} &= \lambda R_{\iota \rightarrow \iota \rightarrow o} \bullet \lambda A_{\iota \rightarrow o} \bullet \lambda X_{\iota} \bullet \forall X_{\iota} \bullet R X Y \Rightarrow A X\end{aligned}$$

A multimodal logic formula $A_{\iota \rightarrow o}$ is valid iff for all possible worlds W_{ι} we have $W \in A$, that is, iff AW holds: **valid** = $\lambda A_{\iota \rightarrow o} \bullet \forall W_{\iota} \bullet A W$. Reflexivity and transitivity are defined as (we omit types) **refl** = $\lambda R \bullet \forall X \bullet R X X$ and **trans** = $\lambda R \bullet \forall X \bullet \forall Y \bullet \forall Z \bullet R X Y \wedge R Y Z \Rightarrow R X Z$.

Example 1 (A simple equation between modal logic formulas).

$$\forall R \bullet \forall A \bullet \forall B \bullet (\Box_R (A \vee B)) = (\Box_R (B \vee A))$$

Example 2 (In modal logic \mathbf{K} , the axioms T and 4 are equivalent to reflexivity and transitivity of the accessibility relation R).

$$\forall R \bullet (\forall A \bullet \mathbf{valid}(\Box_R A \Rightarrow A) \wedge \mathbf{valid}(\Box_R A \Rightarrow \Box_R \Box_R A)) \Leftrightarrow (\mathbf{refl}(R) \wedge \mathbf{trans}(R))$$

Example 3 (Axiom T does not hold in modal logic \mathbf{K}).

$$\neg \forall R \bullet \forall A \bullet (\mathbf{valid}(\Box_R A \Rightarrow A))$$

After negating the statement in Example 1, recursive expansion of the definitions, and exhaustive clause normalization (with integrated functional and Boolean extensionality treatment), LEO-II generates ten light-colored clauses. Amongst them are $(bV) \vee (aV) \vee \neg((rw)V) \vee \neg((rw)U) \vee (bU) \vee (aU)$ and $\neg(az) \vee \neg(bv)$ and $((rw)z) \vee ((rw)v)$ where upper case and lower case symbols

denote variables and Skolem constants, respectively. This light-colored clause set is immediately refutable by E, so that LEO-II does not even start its main proof loop. (The total proving time is 0.071s on a Linux notebook with 1.60GHz, 1GB memory.) Definition expansion and normalization does not always produce refutable light-colored clauses: When replacing the primitive equality = in Example 1 by Leibniz equality $\lambda X, Y. \forall P. (P X) \Rightarrow (P Y)$, we obtain the dark-colored clauses $(p(\lambda X_l. \forall Y_l. \neg((r X) Y) \vee (a Y) \vee (b Y)))$ and $\neg(p(\lambda X_l. \forall Y_l. \neg((r X) Y) \vee (b Y) \vee (a Y)))$. LEO-II starts its proof search and applies resolution and extensional unification to them [3] which subsequently returns a refutable set of light-colored clauses as above. (The total proving time is 0.166s.)

Example 2 appears more challenging than Example 1. Proof search in LEO-II, however, is analogous with one crucial difference: Definition expansion and normalization generates 70 clauses and E generates more than 20 000 clauses from them before finding the refutation. (The total proving time is 2.48s.) This example illustrates the benefit of the cooperation, since LEO-II alone is not able to find this refutation in its search space.

In Example 3, the refutation is found only in the dark-colored area. (The total proving time is 9.02s.) Definition expansion and normalization generates the clauses $((R W) s^{A,W,R}) \vee (A W)$ and $\neg(A s^{A,W,R}) \vee (A W)$, where $s^{A,W,R} = ((s A) W) R$ is a new Skolem term. The refutation employs only the former clause. LEO-II applies its primitive substitution rule [3, 6] to guess the instantiations $R \leftarrow \lambda X, Y. ((M X) Y) \neq ((N X) Y)$ and $A \leftarrow \lambda X. (O X) \neq (P X)$ where M, N, O, P are new free variables. (Hence, LEO-II proposes to consider inequality for the the accessibility relation, which is not reflexive and does not satisfy axiom T .) Applying this instantiation leads to two unification constraints, i.e., negated equation literals in LEO-II, which both have flexible term heads. Proof search terminates, since such flex-flex unification constraints are always solvable.

4 Term Sharing and Term Indexing

Term indexing techniques are widely used in major FO ATPs [18, 19, 23]. The indexing data structures store large numbers of terms and, for a given query term t , support the fast retrieval of terms from the index that satisfy a certain relation with t , such as matching, unifiability or syntactic equality [15]. Performance can be further enhanced by representation of terms in efficient data structures, such as shared terms, used for instance in E [19].

HO term indexing techniques are rarely addressed in the literature — an exception is Pientka [17] — which hampers the progress of systems in this field.

LEO-II's implementation at term level is based on a perfectly shared term graph, i.e., syntactically equal terms are represented by a single instance. Ideas from FO term sharing are adapted to HO logic by (i) keeping indexed terms in $\beta\eta$ normal form (i.e., η short and β normal) and (ii) using de Bruijn indices [10] to allow λ -abstracted terms to be shared.

The resulting data structure represents terms in a directed acyclic graph (DAG). LEO-II supports the visualization of such term graphs.

```

----- The Termset Analysis -----
[...]

Sharing rate: 17 nodes with 18 bindings
Average sharing rate (bindings per node): 1.05
Average term size: 6.58
Average number of supernodes: 5.47
Average number of supernodes (symbols): 5.80
Average number of supernodes (nonprimitive terms): 4.50
Rate of term occurrences PST size / term size: 0.36
Rate of symbol occurrences PST size / term size: 0.39
Rate of bound occurrences PST size / term size: 0.57
----- End Termset Analysis -----

```

```

----- The Termset Analysis -----
[...]

Sharing rate: 2094 nodes with 3415 bindings
Average sharing rate (bindings per node): 1.63
Average term size: 13.95
Average number of supernodes: 9.47
Average number of supernodes (symbols): 28.83
Average number of supernodes (nonprimitive terms): 5.73
Rate of term occurrences PST size / term size: 0.24
Rate of symbol occurrences PST size / term size: 0.30
Rate of bound occurrences PST size / term size: 0.52
----- End Termset Analysis -----

```

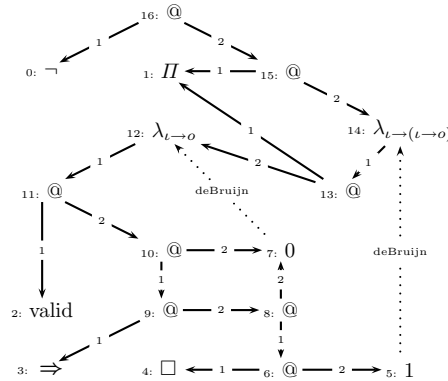
Fig. 2. Statistical analysis of term sharing aspects in LEO-II after initialisation of Example 3 (left) and after the proof has been found (right).

The graph to the right shows the (not heavily shared) term graph after initialization of Example 3.¹

LEO-II also supports the statistical analysis of its term graphs (Figure 2). Future work will investigate whether such information can be exploited for improving heuristic control.

Representation of terms in a shared graph naturally advances the performance of a number of operations, e.g., it allows fast lookup

of all occurrences of syntactically equal terms or subterms, and it improves the performance of rewrite operations, such as global unfolding of definitions. Additionally, LEO-II employs a term indexing data structure, which is based on structural indexing methods from the FO domain [14, 21], as well as road sign techniques. Road signs are features of the data structure which guide operations based on graph traversal. They help to cut branches of the subgraph to be processed early and they are employed, e.g., in the construction of partial syntax trees [22] in which all branches with no occurrences of a given symbol or subterm are cut. This enables LEO-II to avoid potentially costly operations, such as occurs checks, and to speed up basic operations on terms, such as substitution.



5 Conclusion

LEO-II, which replaces the previous LEO system, realizes a cooperative HO-FO theorem proving approach that shows some promising results in selected problem domains. It thus provides an interesting alternative to reasoning solely in FO logic and it also differs from other HO ATPs such as TPS [1] and OTTER- λ [2].

¹ To further visualize the evolution of the term graph during proof search, we modified LEO-II to output a snapshot of its state after each processing step. This data was used to create animations of dynamically changing term graphs during proof search. The video clips can be obtained at <http://www.ags.uni-sb.de/~chris/leo/art/>.

References

1. P. B. Andrews and C. E. Brown. TPS: A hybrid automatic-interactive system for developing proofs. *J. Applied Logic*, 4(4):367–395, 2006.
2. M. Beeson. Mathematical induction in Otter-Lambda. *J. Autom. Reasoning*, 36(4):311–344, 2006.
3. C. Benzmüller. *Equality and Extensionality in Automated Higher-Order Theorem Proving*. PhD thesis, Saarlandes University, 1999.
4. C. Benzmüller and M. Kohlhase. LEO – a higher-order theorem prover. In *Proc. of CADE-15*, number 1421 in LNAI. Springer, 1998.
5. C. Benzmüller and L. Paulson. *Festschrift in Honour of Peter B. Andrews on his 70th Birthday*, chapter Exploring Properties of Normal Multimodal Logics in Simple Type Theory with LEO-II. IFCoLog, 2007. To appear.
6. C. Benzmüller, L. Paulson, F. Theiss, and A. Fietzke. Progress report on LEO-II – an automatic theorem prover for higher-order logic. In *Emerging Trends at TPHOLs-20*. University Kaiserslautern, Germany, 2007.
7. C. Benzmüller, F. Rabe, and G. Sutcliffe. THF0 — the core TPTP language for classical higher-order logic. Submitted to IJCAR 2008.
8. C. Benzmüller, V. Sorge, M. Jamnik, and M. Kerber. Combined reasoning by automated cooperation. *J. of Applied Logic*, 2008.
9. C. Benzmüller et al. Experiments with an Agent-Oriented Reasoning System. In *In Proc. of KI 2001*, volume 2174 of LNAI. Springer, 2001.
10. N.G. de Bruijn. Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indag. Math.*, 34(5):381–392, 1972.
11. M.J. Gordon and T.F. Melham. *Introduction to HOL: A Theorem-Proving Environment for Higher-Order Logic*. Cambridge University Press, 1993.
12. M. Kerber. *On the Representation of Mathematical Concepts and their Translation into First-Order Logic*. PhD thesis, Universität Kaiserslautern, Germany, 1992.
13. L. Klein. Indexing für Terme höherer Stufe. Master’s thesis, Saarland Univ., 1997.
14. W. McCune. Experiments with discrimination-tree indexing and path indexing for term retrieval. *JAR*, 9(2):147–167, 1992.
15. R. Nieuwenhuis et al. On the evaluation of indexing techniques for theorem proving. In *In Proc. of IJCAR-01*, volume 2083 of LNAI. Springer, 2001.
16. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of LNCS. Springer, 2002.
17. B. Pientka. Higher-order substitution tree indexing. In *Proc. of ICLP*, volume 2916 of LNCS, 2003.
18. A. Riazanov and A. Voronkov. The design and implementation of Vampire. *AICOM*, 15(2-3):91–110, jun 2002.
19. S. Schulz. E – A Brainiac Theorem Prover. *Journal of AI Communications*, 15(2/3):111–126, 2002.
20. J. Siekmann, C. Benzmüller, and S. Autexier. Computer supported mathematics with OMEGA. *Journal of Applied Logic*, 4(4):533–559, 2006.
21. M. Stickel. The path-indexing method for indexing terms. Technical Report 473, Artificial Intelligence Center, SRI International, USA, 1989.
22. F. Theiss and C. Benzmüller. Term indexing for the LEO-II prover. In *Proc. of IWIL-6 WS at LPAR 2006*, Pnom Penh, Cambodia, 2006.
23. C. Weidenbach et al. Spass version 2.0. In *Proc. of CADE-18*, volume 2392 of LNAI, pages 275–279. Springer, 2002.