Technical Communications of the International Conference on Logic Programming, 2010 (Edinburgh), pp. 281–285
http://www.floc-conference.org/ICLP-home.html

HIGHER-ORDER LOGIC LEARNING AND λPROGOL

NIELS PAHLAVI [1]

[1] Department of Computing, Imperial College London
180 Queen's Gate, London, United Kingdom
E-mail address: niels.pahlavi@imperial.ac.uk
URL: http://www.doc.ic.ac.uk/~namdp05/

Abstract. We present our research produced about Higher-order Logic Learning (HOLL), which consists of adapting First-order Logic Learning (FOLL), like Inductive Logic Programming (ILP), within a Higher-order Logic (HOL) context. We describe a first working implementation of λProgol, a HOLL system adapting the ILP system Progol and the HOL formalism λProlog. We compare λProgol and Progol on the learning of recursive theories showing that HOLL can, in these cases, outperform FOLL.

Introduction, Problem Description and Background

Much of logic-based Machine Learning research is based on First-order Logic (FOL) and Prolog, including Inductive Logic Programming (ILP). As such, learning higher-order theories is not possible for such a system, and even some first-order tasks are not handled well, like "learning first-order recursive theories" which "is a difficult learning task" in a normal ILP setting [Mal03]. Yet, [Far08] describes HOL as "a natural extension of first-order logic (FOL) which is simple, elegant, highly expressive, and practical" and recommends its use as an "attractive alternative to first-order logic". HOL, which allows for quantification over predicates and functions, is intrinsically more expressive than FOL, would give sounder logical foundations, and "has generally been under-exploited" [Llo03] in logic-based Machine Learning. According to [Llo03], "the logic programming community needs to make greater use of the power of higher-order features and the related type systems and the use of HOL in Computational Logic is illustrated: functional languages, like Haskell98; Higher-order programming introduced with λProlog [Mil98]; integrated functional logic programming languages like Curry or Escher; or the higher-order logic interactive theorem proving environment "HOL". It is also used in IBAL and for Deep Transfer Learning.

The use of HOL in ILP would allow to consider the learning of higher-order predicates; but it would also make the learning of first-order learning theories sounder, more natural and more intuitive through the use of higher-order predicates in background knowledge. More generally, the expressivity of HOL would make it possible to represent mathematical properties like symmetry, reflexivity or transitivity, which would allow to handle equational reasoning and functions within a logic-based framework. We could also represent such properties in the following fashion (in the case of symmetry) : R@X@Y $\Leftarrow$ [sym@R,R@Y@X], and,

Key words and phrases: Inductive Logic Programming, Progol, Higher-order Logic, Higher-order Logic Learning, λProlog.

Technical Communications of the 26th International Conference on Logic Programming, Edinburgh, July, 2010
Editors: Manuel Hermenegildo, Torsten Schaub
LIPIcs - Leibniz International Proceedings in Informatics. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany
Digital Object Identifier: 10.4230/LIPIcs.ICLP.2010.281

abduce for example that the move of the bishop in chess is symmetric: sym@bishop_move. About the use of probability in a logic-based setting, [Ng08] advocates for probability to be captured directly in the theory itself, which can be done naturally and directly with HOL, as opposed to almost all approaches having a clear separation between the logical statements and the probabilities.

λProlog [Mil98] is a higher-order logic programming language handling scoping over names and procedures, the use of lambda terms as data structures and higher-order programming. It is based on Higher-order Horn Clauses (HOHC) (introduced in [Nad90] where a theorem proving procedure for HOHC based on Huet's unification algorithm in typed λ-calculus [Hue75] is also outlined), which are "a generalization of Horn clauses to a higher-order logic" obtained "by supplanting first-order terms with the terms of a typed λ-calculus and by permitting quantification over function and predicate symbols".

1. Goal of the Research and Overview of the existing Literature

The goal of my PhD research is, therefore, to develop Higher-order Logic Learning (HOLL), which consists of generalizing logic-based Machine Learning, and particularly ILP, from the first-order to the higher-order context. We have already made a first working implementation of λProgol, a HOLL system adapting the ILP system Progol and the HOL formalism λProlog. We decided to choose Higher-order Horn Clauses (HOHC) [Nad90] as a HOL formalism, since it is one of the logical foundations of λProlog. As a ILP system, we chose to adapt Progol [Mug95], which is a popular and efficient implementation.

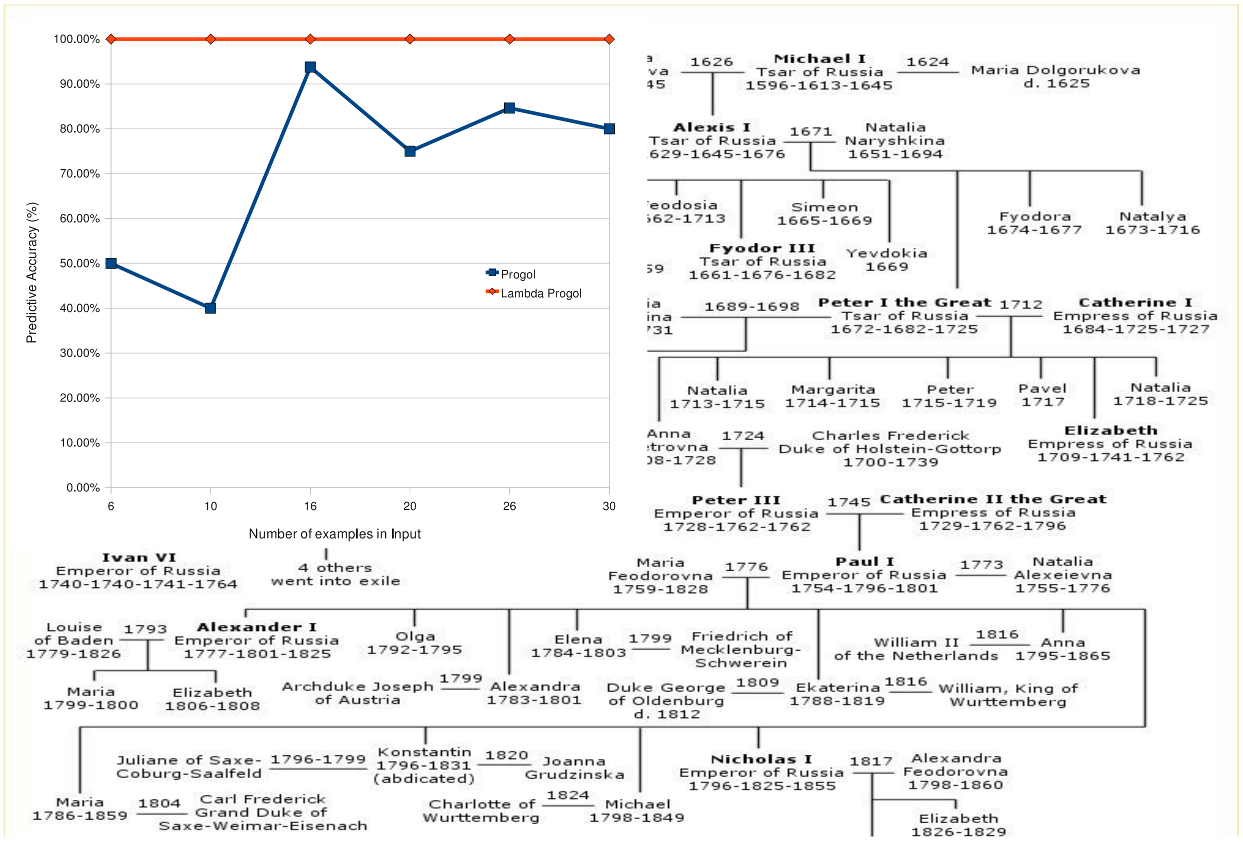
We also want to determine whether HOLL can outperform First-order Logic Learning (FOLL), assessing how the power of expressivity of HOL can be used to improve significantly the learning capacity and efficiency of FOLL, and study the trade-off that there may be between learnability and searching costs (the use of Henkin semantics as in [Wol94], seems to alleviate these and maintain the structure of the search space). ILP seems to be rather intuitively adaptable to a HOL formalism and we aim at developing a theory of HOLL as well.

There have been attempts to use HOL for logic-based Machine Learning such as by Harao starting in [Har90], Feng and Muggleton, and Furukawa and Goebel [Fur96]. They provide different higher-order extensions of least general generalization to handle higher-order terms in a normal ILP setting, whereas we use λProlog, a HOL framework, as a logical foundation to extend first-order ILP to a higher-order context. The main similar work is [Llo03] by Lloyd and Ng, where higher-order machine learning is also developed. It details a learning system, called *ALKEMY*. A main difference is that Lloyd's approach is not based on Logic Programming and therefore on ILP. According to Flach, "it is almost a rational reconstruction of what ILP could have been, had it used Escher-style HOL rather than Prolog"; whereas we intend, through the use of higher-order Horn clauses to keep the Horn clauses foundations of LP and ILP and to extend it.

2. Current Status of the Research and Preliminary Results

λProgol, a λProlog adaptation of the popular and efficient ILP system Progol was introduced in [Pah09a] and [Pah09b] along with its algorithm adapting closely and intuitively Progol and Mode-Directed Inverse Entailment. A first working implementation of λProgol has since been made, tested, is described in [Pah10] and is available at [Pah]. Our first

Figure 1: Left: Comparison between Progol and λProgol on the Ancestor example. Right: Part (around one third) of the Romanov dynasty tree used in the experiments



choice of implementation was based on λProlog but revealed to be too inconvenient and inefficient to use; instead the current implementation is in Prolog, which is more convenient and more efficient; a requirement is the use of a λProlog interpreter, which was implemented using a depth-first approach.

Initial promising results have been obtained so far about learning recursive theories. In order to stress the difference in the learnability of a given problem between HOLL and FOLL, and to ensure fairness and soundness, standard λProgol was compared against standard Progol, whose algorithms are almost the same.

One of the results (used in [Mal03]) consists of learning the predicate *ancestor* given a genealogical tree defined by facts for the predicates *parent* and *married* as background knowledge and positive and negative examples of the predicate *ancestor*; for λProgol, the higher-order predicate *trans*, which "given a predicate of two arguments, constructs its transitive closure" is added to the background knowledge. The genealogical tree used for this experiment is described in Fig.1 and contains 119 members over 11 relations of the Romanov Russian dynasty.

To compare the two systems, we created files containing positive and negative examples of *ancestor*. These files contain an equal number of positive and negative examples generated randomly. We then compared the respective predicative accuracy of Progol and

λProgol on these examples by doing a leave-one-out cross-validation. The results of this experiment are shown in Fig.1.

For Progol, which has to learn the definition recursively, the larger the input and the smaller the number of examples, the smaller the probability to learn the definition correctly. Hence the difficulty to learn and the observation that the accuracy seems to decrease with the number of examples. On the other hand, λProgol learns the correct definition in all the cases, which is ancestor@X@Y $\Leftarrow$ [trans@parent@X@Y]. This definition is non recursive and can be learned from any given positive example. On this example consisting of learning a recursive definition from large data with few examples, we have showed that HOLL can outperform FOLL. This result along with similar others are available at [Pah].

3. Expected Achievements and Open Issues

We intend to continue the tests and comparisons of λProgol against already existing ILP systems to determine how HOLL may outperform FOLL as it was shown above. We aim to present theoretical results for HOLL. ILP theory seems to be rather intuitively adaptable within a HOL framework. For λProgol, we will have to prove that higher-order inverse entailment is possible and to generalize correctness and complexity results for the Progol Bottom Clause and Search algorithms. In [Wol94], a model-theoretic semantics for HOHC is provided. We also want to investigate tasks and discoveries not learnable by first-order ILP. It could be of interest to look at HOL theorem provers, or integrated functional logic programming languages and Mathematical Discovery. Further objectives may be to investigate abduction within λProgol, active learning, introduce Probability and adapt Probabilistic Logic Learning (that I have studied during my MSc, [Mug06b] and [Mug06a]) within HOL, look at applications such as Bioinformatics, where ILP has been successfully applied, and consider other logics within λProlog.

Acknowledgement

I mostly wish to thank my supervisor Stephen Muggleton and Imperial College London for allowing me to pursue my PhD in great conditions. I am also grateful to NICTA for having given me the opportunity to visit their Canberra Research Laboratory for a month and thoroughly interact and discuss my work with Kee Siong Ng, John Lloyd and Scott Sanner.

References

[Far08] W. Farmer. The seven virtues of simple type theory. *J. Applied Logic*, 2008.
[Fur96] K. Furukawa, M. Imai, and R. Goebel. Hyper least general generalization and its application to higher-order concept learning. Tech. report, Keio University, 1996.
[Har90] M. Harao. Analogical reasoning based on higher-order unification. In *ALT*. 1990.
[Hue75] G. Huet. A unification algorithm for typed λcalculus. *Theor. Comp. Sci.*, 1975.
[Llo03] J.W. Lloyd. *Logic for Learning*. Springer, Berlin, 2003.
[Mal03] D. Malerba. Learning recursive theories in the normal ILP setting. *Fundam. Inform.*, 57(1):39–77, 2003.
[Mil98] Dale Miller. *λProlog: An Introduction to the Language and its Logic*. 1998.
[Mug95] S.H. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995. URL `http://www.doc.ic.ac.uk/\~shm/Papers/InvEnt.pdf`

[Mug06a] S.H. Muggleton and N. Pahlavi. The Complexity of Translating BLPs to RMMs. In *Proceedings of the 16th International Conference on Inductive Logic Programming.* Springer-Verlag, 2006.

[Mug06b] S.H. Muggleton and N. Pahlavi. Stochastic Logic Programs: A Tutorial. In L. Getoor and B. Taskar (eds.), *Statistical Relational Learning.* MIT Press, 2006.

[Nad90] G. Nadathur and D. Miller. Higher-order Horn Clauses. *Journal of the ACM*, 1990.

[Ng08] K. S. Ng, J. W. Lloyd, and W. T. B. Uther. Probabilistic modelling, inference and learning using logical theories. *Ann. Math. Artif. Intell.*, 54(1-3):159–205, 2008.

[Pah] N. Pahlavi. λProgol Homepage. http://www.doc.ic.ac.uk/~namdp05/.

[Pah09a] N. Pahlavi and S. Muggleton. Higher-order Logic Learning. 19th International Conference on Inductive Logic Programming. 2009. (Poster).

[Pah09b] N. Pahlavi and S. Muggleton. Higher-order Logic Learning and λProgol. IJCAI09 Workshop on Abductive and Inductive Knowledge Development. 2009.

[Pah10] N. Pahlavi and S. Muggleton. Can HOLL outperform FOLL? In *Proceedings of the 20th International Conference on Inductive Logic Programming.* 2010. To Appear.

[Wol94] D. A. Wolfram. A semantics for λProlog. *Theor. Comp. Sci.*, pp. 277–289, 1994.