LearnLib: A Library for Automata Learning and Experimentation

Harald Raffelt and Bernhard Steffen

University of Dortmund,
Chair of Programming Systems,
Baroper Str. 301,
Dortmund 44227, Germany

Abstract. In this tool demonstration we present the LearnLib, a library for automata learning and experimentation. Its modular structure allows users to configure their tailored learning scenarios, which exploit specific properties of the envisioned applications. As has been shown earlier, exploiting application-specific structural features enables optimizations that may lead to performance gains of several orders of magnitude, a necessary precondition to make automata learning applicable to realistic scenarios.

The demonstration of the LearnLib will include the extrapolation of a behavioral model for a realistic (legacy) system, and the statistical analysis of different variants of automata learning algorithms on the basis of random generated models.

1 Motivation

Most systems in use today lack adequate specifications or make use of un(der) specified components. In fact, the much propagated component-based hard- and software design style naturally leads to under specified systems, as most libraries and third party components only provide very partial specifications. To improve this situation automata learning techniques [1] have been proposed. They enable the automatic construction and subsequent update of behavioral models.

2 Automata Learning

Automata learning tries to automatically construct a finite automaton that matches the behavior of a given target automaton on the basis of (systematic) observation [2, 3]. This complements other automatic learning techniques which aim at the construction of invariants [4, 5]. The interested reader may refer to [1, 6, 7] for our (practice-oriented) view of the use of (active) learning.

Active learning assumes an omniscient teacher which is able to answer *membership* and *equivalence* queries. A membership query tests whether a string (a potential run) is contained in the target automatons language (its set of runs), and an equivalence query compares the hypothesis automaton with the target

© Springer-Verlag Berlin Heidelberg 2006

automaton for language equivalence, in order to determine whether the learning procedure was successfully completed.

In any practical attempt of learning legacy systems, equivalence queries can only be treated approximately, but membership queries can be answered by testing the target systems [1, 6]. The LearnLib therefore provides a number of heuristics and techniques for this approximation.

3 The LearnLib

The LearnLib [8] provides a platform for experimenting with different learning algorithms and for statistically analyzing their characteristics in terms of required number and size of e.g., membership queries, run time, and memory consumption. This concerns in particular the analysis of the impact of the various techniques for optimizations.

Besides the fine granular analysis for understanding the individual components of typical learning algorithms, the LearnLib can also be used as a fully automatic tool to systematically build finite state machine models of (specific aspects of) real world systems.

As depicted in Fig. 1 LearnLib consists of three modules:

- the *automata learning* module containing the basic learning algorithms,
- the *filters* module providing a number of strategies to reduce the number of queries, and
- the module for *approximative equivalence queries*, which is based on techniques adopted from the area of conformance testing. suites for the conjectures of the learning algorithms.

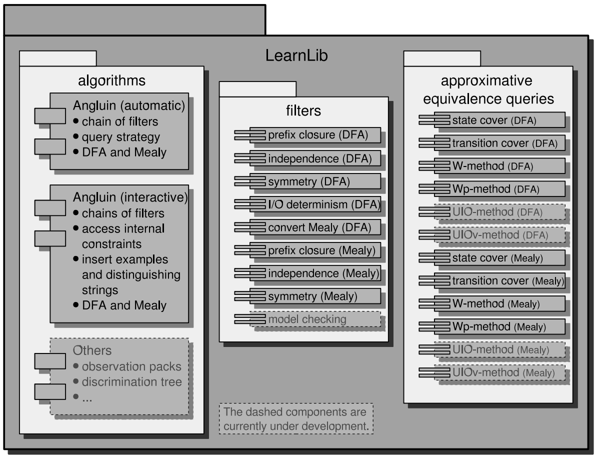

Fig. 1. LearnLib Components

3.1 Automata Learning with the LearnLib

The main module of the LearnLib contains learning algorithms of different flavor. Currently, the LearnLib only supports some variants and extensions of Angluin's algorithm L^*.

Angluin's algorithm [9] works on deterministic finite state machines, which, as usual, consist of a finite set of states Q, a finite (input) alphabet Σ, a transition function $\delta : (Q \times \Sigma) \rightarrow Q$, and a finite set of accepting states $F \subseteq Q$.

Typical reactive systems do not terminate and are thereforer not modelled appropriately with accepting states. It turned out that Mealy machines, which adequately model input/output behavior provide a much better modelling, which is not only more concise, but can also be learned much faster [7]. The LearnLib therefore provides learning algorithms for both, finite state machines and an adaptation to Mealy machines. These versions can be used in two modes: an automatic mode, which automatically proceeds in a predefined fashion, and an interactive mode, which gives the user a handle to steer the learning process in more detail.

The learning algorithms can be freely combined with a number of filters, optimizing the number of required membership queries (see [10]), and with an adequate approximate solver for equivalence queries. Whereas these approximate solvers are necessary, whenever one attempts to learn a legacy system, they can can be replaced by a perfect equivalence checker for the statistical analysis of learning scenarios using randomly generated target models. The LearnLib provides a bisimulation checker for this purpose.

3.2 Simulation and Analysis of Learning Algorithms

Different learning algorithms have very different characteristics. Perhaps most importantly, they may drastically differ in the number of membership- and equivalence queries, but they also differ in the size of their queries. Our simulator analyzed these differences, and helps to gain knowledge about how learning algorithms essentially work, and where their bottlenecks lie. The interested reader may refer to [11], where the interdependency between our optimizing filters is experimentally evaluated.

Such studies are specifically supported be the LearnLib, which, besides the configuration of individual learning scenarios, also allows the configuration of whole scenarios for the statistical analysis of learning scenarios. In fact, as will be illustrated during the demonstration, it is possible to configure a set of comparative learning scenarios and to automatically generate the corresponding comparative statistical charts.

4 Conclusion

In this paper we have presented the LearnLib, a modular library for automata learning, which is explicitly designed for experimentation. As will be illustrated during the tool demo, its modular structure allows users to systematically analyze and then construct learning algorithms tailored for their specific application scenario.

References

1. Hagerer, A., Hungar, H., Niese, O., Steffen, B.: Model generation by moderated regular extrapolation. In R. Kutsche, H.W., ed.: FASE'02: Proc. of the 5th International Conference on Fundamental Approaches to Software Engineering. Volume 2306 of LNCS., Heidelberg, Germany, Springer-Verlag (2002) 80–95
2. Cook, J.E., Wolf, A.L.: Discovering models of software processes from event-based data. ACM Transactions on Software Engineering and Methodology (TOSEM) **7** (1998) 215–249
3. Peled, D., Vardi, M.Y., Yannakakis, M.: Black box checking. In Wu, J., Chanson, S.T., Gao, Q., eds.: FORTE/PSTV '99: Proc. of the Joint Int. Conference on Formal Description Techniques for Distributed System and Communication/Protocols and Protocol Specification, Testing and Verification, Kluwer Academic Publishers (1999) 225–240
4. Brun, Y., Ernst, M.D.: Finding latent code errors via machine learning over program executions. In: ICSE'04: Proc. of the 26th International Conference on Software Engineering, Edinburgh, Scotland (2004) 480–490
5. Nimmer, J.W., Ernst, M.D.: Automatic generation of program specifications. In: ISSTA'02: Proceedings of the 2002 International Symposium on Software Testing and Analysis, Rome, Italy (2002) 232–242
6. Steffen, B., Hungar, H.: Behavior-based model construction. In Mukhopadhyay, S., Zuck, L., eds.: VMCAI'03: Proc. of the 4th International Conference on Verification, Model Checking, and Abstract Interpretation. Volume 2575 of LNCS., Springer-Verlag (2003) 5–19
7. Steffen, B., Margaria, T., Raffelt, H., Niese, O.: Efficient test-based model generation of legacy systems. In: HLDVT'04: Proc. of the 9th IEEE Int. Workshop on High Level Design Validation and Test, Sonoma (CA), USA, IEEE Computer Society Press (2004) 95–100
8. Raffelt, H., Steffen, B., Berg, T.: Learnlib: A library for automata learning and experimentation. In Halbwachs, N., Zuck, L.D., eds.: FMICS'05: Proc. of the 10th International Workshop on Formal Methods for Industrial Critical Systems. Volume 3440 of LNCS., New York, NY, USA, ACM Press (2005) 557–562
9. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation **2** (1987) 87–106
10. Hungar, H., Niese, O., Steffen, B.: Domain-specific optimization in automata learning. In: Proc. of the 15th Computer Aided Verification Conference. Volume 2725 of LNCS., Springer Verlag (2003) 315–327
11. Margaria, T., Raffelt, H., Steffen, B.: Analyzing second-order effects between optimizations for system-level test-based model generation. In: ITC'05: Proc. of IEEE International Test Conference. (2005)