# Active Learning of Extended Finite State Machines

Frits Vaandrager*

Institute for Computing and Information Sciences, Radboud University Nijmegen,
P.O. Box 9010, 6500 GL Nijmegen
the Netherlands

Once they have high-level models of the behavior of software components, engineers can construct better software in less time. A key problem in practice, however, is the construction of models for existing software components, for which no or only limited documentation is available. In this talk, I will present an overview of recent work by my group — done in close collaboration with the Universities of Dortmund and Uppsala — in which we use machine learning to infer state diagram models of embedded controllers and network protocols fully automatically through observation and test, that is, through black box reverse engineering.

Starting from the well-known $L^*$ algorithm of Angluin [6], our aim is to develop algorithms for active learning of richer classes of (extended) finite state machines. Abstraction is the key when learning behavioral models of realistic systems. Hence, in practical applications, researchers manually define abstractions which, depending on the history, map a large set of concrete events to a small set of abstract events that can be handled by automata learning tools. Our work, which builds on earlier results from concurrency theory and the theory of abstraction interpretation, shows how such abstractions can be constructed fully automatically for a restricted class of extended finite state machines in which one can test for equality of data parameters, but no operations on data are allowed [2, 1]. Our approach uses counterexample-guided abstraction refinement (CEGAR): whenever the current abstraction is too coarse and induces nondeterministic behavior, the abstraction is refined automatically. In the talk, I will compare our approach with the related work of Howar et al [8, 9] on register automata.

Using the LearnLib [11, 10] tool from Dortmund in combination with Tomte [1], a prototype implementation of our CEGAR algorithm, we have succeeded to learn models of several realistic software components, such as the SIP protocol [3, 1], the new biometric passport [5], banking cards, and printer controllers.

Once we have learned a model of a software component, we may use model checking technology to analyze this model and model-based testing to automatically infer test suites. This allows us to check, for instance, whether no new faults have been introduced in a modified version of the component (regression testing), whether an alternative implementation by some other vendor agrees

* Supported by STW project 11763 Integrating Testing And Learning of Interface Automata (ITALIA), `http://www.italia.cs.ru.nl/`.

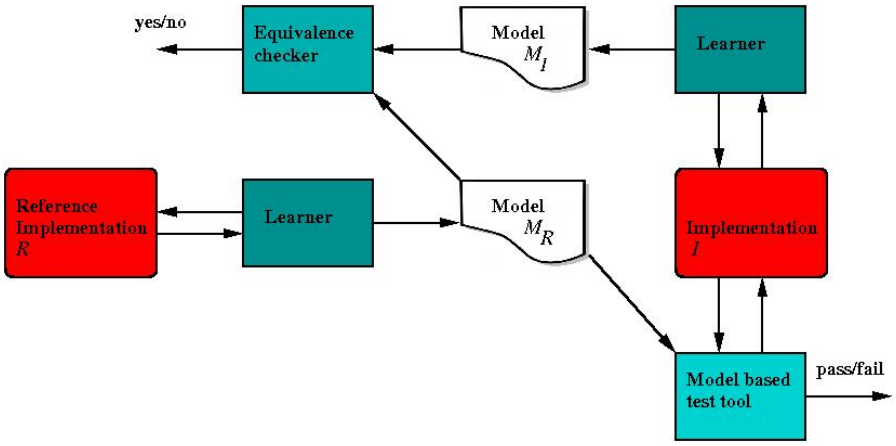


**Fig. 1.** Use of automata learning to establish conformance of implementations.

with a reference implementation, or whether some communication protocol is secure. Using a well-known industrial case study from the verification literature, the bounded retransmission protocol [7], we show how active learning can be used to establish the correctness of protocol implementation $I$ relative to a given reference implementation $R$. Using active learning, we learn a model $M_R$ of reference implementation $R$, which serves as input for a model based testing tool that checks conformance of implementation $I$ to $M_R$. In addition, we also explore an alternative approach in which we learn a model $M_I$ of implementation $I$, which is compared to model $M_R$ using an equivalence checker. Our work uses a unique combination of software tools for model construction (Uppaal), active learning (LearnLib, Tomte), model-based testing (JTorX, TorXakis) and verification (CADP, MRMC). We show how these tools can be used for learning these models, analyzing the obtained results, and improving the learning performance [4].

## References

1. F. Aarts, F. Heidarian, H. Kuppens, P. Olsen, and F.W. Vaandrager. Automata learning through counterexample-guided abstraction refinement. In D. Giannakopoulou and D. Méry, editors, *18th International Symposium on Formal Methods (FM 2012), Paris, France, August 27-31, 2012. Proceedings*, volume 7436 of *Lecture Notes in Computer Science*, pages 10–27. Springer, August 2012.
2. F. Aarts, F. Heidarian, and F.W. Vaandrager. A theory of abstractions for learning interface automata. In M. Koutny and I. Ulidowski, editors, *23rd International Conference on Concurrency Theory (CONCUR), Newcastle upon Tyne, UK, September 3-8, 2012. Proceedings*, volume 7454 of *Lecture Notes in Computer Science*, pages 240–255. Springer, September 2012.
3. F. Aarts, B. Jonsson, and J. Uijen. Generating models of infinite-state communication protocols using regular inference with abstraction. In A. Petrenko, J.C.

Maldonado, and A. Simao, editors, *22nd IFIP International Conference on Testing Software and Systems, Natal, Brazil, November 8-10, Proceedings*, volume 6435 of *Lecture Notes in Computer Science*, pages 188–204. Springer, 2010.

4. F. Aarts, H. Kuppens, G.J. Tretmans, F.W. Vaandrager, and S. Verwer. Learning and testing the bounded retransmission protocol. In J. Heinz, C. de la Higuera, and T. Oates, editors, *Proceedings 11th International Conference on Grammatical Inference (ICGI 2012), September 5-8, 2012. University of Maryland, College Park, USA*, volume 21 of *JMLR Workshop and Conference Proceedings*, pages 4–18, 2012.
5. F. Aarts, J. Schmaltz, and F.W. Vaandrager. Inference and abstraction of the biometric passport. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification, and Validation - 4th International Symposium on Leveraging Applications, ISoLA 2010, Heraklion, Crete, Greece, October 18-21, 2010, Proceedings, Part I*, volume 6415 of *Lecture Notes in Computer Science*, pages 673–686. Springer, 2010.
6. D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.
7. L. Helmink, M.P.A. Sellink, and F.W. Vaandrager. Proof-checking a data link protocol. In H. Barendregt and T. Nipkow, editors, *Proceedings International Workshop TYPES'93,* Nijmegen, The Netherlands, May 1993, volume 806 of *Lecture Notes in Computer Science*, pages 127–165. Springer-Verlag, 1994.
8. F. Howar, B. Steffen, B. Jonsson, and S. Cassel. Inferring canonical register automata. In V. Kuncak and A. Rybalchenko, editors, *Verification, Model Checking, and Abstract Interpretation - 13th International Conference, VMCAI 2012, Philadelphia, PA, USA, January 22-24, 2012. Proceedings*, volume 7148 of *Lecture Notes in Computer Science*, pages 251–266. Springer, 2012.
9. M. Merten, F. Howar, B. Steffen, S. Cassel, and B. Jonsson. Demonstrating learning of register automata. In C. Flanagan and B. König, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 18th International Conference, TACAS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings*, volume 7214 of *Lecture Notes in Computer Science*, pages 466–471. Springer, 2012.
10. M. Merten, B. Steffen, F. Howar, and T. Margaria. Next generation LearnLib. In P.A. Abdulla and K.R.M. Leino, editors, *TACAS*, volume 6605 of *Lecture Notes in Computer Science*, pages 220–223. Springer, 2011.
11. H. Raffelt, B. Steffen, T. Berg, and T. Margaria. LearnLib: a framework for extrapolating behavioral models. *STTT*, 11(5):393–407, 2009.