Learning Finite-State Machines from Inexperienced Teachers

Olga Grinchtein[1]⋆ and Martin Leucker[2]

[1] Department of Computer Systems, Uppsala University, Sweden
[2] Institut für Informatik, TU München, Germany

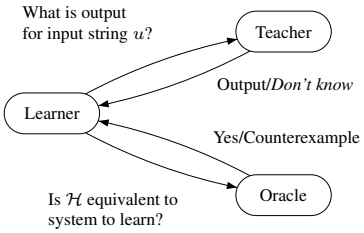
– Extended Abstract –

The general goal of query-based learning algorithms for finite-state machines is to identify a *machine*, usually of *minimum* size, that *agrees* with an *a priori* fixed (class of) machines. For this, *queries* on how the underlying system behaves may be issued.

A popular setup is that of Angluin's L^* algorithm[Ang87], here adapted to the case of finite-state machines, in which a minimal deterministic finite-state machine for a regular language is learned based on so-called *membership* and *equivalence queries*. Using a pictorial language, we have a *learner* whose job is to come up with the automaton to learn, a *teacher* who may answer the output for a given input string as well an *oracle* answering whether the automaton $\mathcal{H}$ currently proposed by the learner is correct or not. This setting is depicted in Figure 1(a) (though assume that the *don't know* is not there).

In Angluin's setting, a teacher will always answer with the correct output symbol. In many application scenarios, however, parts of the machine to learn are not completely specified or not observable. Then, queries may be answered inconclusively, by *don't know*, also denoted by $?$.

In the full version of this paper [GL06], we study a learning algorithm (and variants thereof), called ABSAT, ABSAT$_\mathrm{I}$, and ABSAT$_\mathrm{I}$2, that are designed to work with such an *inexperienced* teacher. The oracle, however, does not change its functionality in the setting discussed here (see Figure 1(a), the *don't know* is new).



(a) L^* versus ABSAT

	bica	ABSAT	ABSAT$_\mathrm{I}$	ABSAT$_\mathrm{I}$2
	time	time	time	time
fo.70	n.a.	39.50	38.57	38.69
th.55	188.87	0.45	0.43	0.43
vmebus.master.m	59.93	1627.37	693.44	93.01
ifsm2	14.08	34.82	34.36	30.78
pe-send-ifc.fc	2.33	6.93	2.75	2.07
pe-send-ifc.fc.m	0.85	6.93	2.75	1.07
rubin1200	10.24	10.28	10.26	10.02
rubin2250	64.78	64.69	64.74	63.29
vbe4a	1.94	4.31	1.86	1.56
ex2	0.78	0.78	0.76	4.67

(b) Running times compared

Fig. 1. The setup for the learning algorithms and their performance

⋆ Part of the work has been done during the author's stay at TU München supported by the C F Liljewalchs fellowship, Uppsala University.

In general, two types of learning algorithms for FSMs can be distinguished, so-called *online* and *offline* algorithms. Online algorithms, such as Angluin's L^* algorithm, query strings to the teacher. Offline algorithms get a fixed set of examples and no further queries are allowed before computing a minimal FSM conforming to the examples. Typical algorithms of this type are based on a characterization in terms of a constraint satisfaction problem (CSP) over the natural numbers due to Biermann [BF72].

Faced with an inexperienced teacher, we cannot rely completely on Angluin's algorithm. We therefore define an algorithm that is a combination of an online algorithm and an offline algorithm and is based on [OS98]. Similar to Angluin's algorithm, we round off the information on the automaton in question by asking queries. As queries can be answered by *?*, we may not be able to complete the information as in Angluin's setting to compute an FSM directly. For this, we use Biermann's approach for obtaining an FSM based on the enriched information. Our combination is conservative in the sense that in case all queries are answered by either *yes* or *no*, we obtain the same efficiency as for Angluin's algorithm. Furthermore, the encoding in terms of CSP is optimized based on the information collected in Angluin's algorithm.

While in [OS01] an efficient implementation for solving the resulting CSP problem is explained, we give an encoding as a SAT problem featuring a simple yet—as the examples show—very efficient inference algorithm by employing powerful SAT solvers.

Actually, our approach is quite similar to the one proposed in [OS98] and [OS01]. The main difference are that we use SAT solvers for solving the corresponding CSP problem (which gives algorithm ABSAT) and that we additionally propose incremental consistency checks (ABSATI) and an incremental construction of the CSP problem (ABSATI2), which both improves the overall efficiency.

To validate our approach in practice, we have employed our techniques to the problem of reducing incompletely specified finite-state machines. We have implemented our extensions within the C++ program called BICA, used in [OS98] and have reexamined BICA as well as our three versions on the same set of examples studied in [OS98] (see Figure 1(b)).

The overall conclusion is that although the behavior of a SAT solver is not completely predictable, our algorithms ABSATI and ABSATI2 are, for many examples, competitive alternatives to BICA, which especially work on examples that are too complex for BICA.

References

[Ang87] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[BF72] A. W. Biermann and J. A. Feldman. On the synthesis of finite-state machines from samples of their behaviour. *IEEE Transactions on Computers*, 21:592–597, 1972.

[GL06] Olga Grinchtein and Martin Leucker. Learning finite-state machines from inexperienced teachers. Technical Report TUM-I0613, TU München, 2006.

[OS98] A. L. Oliveira and J. P. M. Silva. Efficient search techniques for the inference of minimum size finite automata. In *String processing and information retrieval*, 1998.

[OS01] Arlindo L. Oliveira and João P. Marques Silva. Efficient algorithms for the inference of minimum size dfas. *Machine Learning*, 44(1/2):93–119, 2001.