From ZULU to RERS

Lessons learned in the ZULU challenge

Falk Howar, Bernhard Steffen, Maik Merten

University of Dortmund, Chair of Programming Systems,
Otto-Hahn-Str. 14, 44227 Dortmund, Germany
{falk.howar, steffen, maik.merten}@cs.tu-dortmund.de
Tel. ++49-231-755-7759, Fax. ++49-231-755-5802

Abstract This paper summarizes our experience with the ZULU challenge on active learning without equivalence queries, presents our winning solution, investigates the character of ZULU's rating approach, and discusses how this approach can be taken further to establish a framework for the systematic investigation of domain-specific, scalable learning solutions for practically relevant application. In particular, it discusses the RERS initiative, which provides a community platform together with a learning framework that allows users to interactively compose complex learning solutions on the basis of libraries for various learning components, system connectors, and other auxiliary functionality. This framework will be the backbone for an extended challenge on learning in 2011.

1 Motivation

In the last decade, active automata learning, an originally merely theoretical enterprise, got attention as a method for dealing with black-box or third party systems. Applications ranged from the support of formal verification, e.g. for assume guarantee reasoning [10,22], to usage of learned models as the basis for regression testing. In the meantime there exist a number of approaches exploiting active learning for validation [23,28,14,15,5,1,6]. This success may seem surprising, because automata learning, in practice, is inherently neither correct nor complete, and there are only few examples of learned systems of significant size. Hardly any have more than 1,000 states, and to our knowledge the largest learned realistic system is a router with about 22,000 states [24]. On the other hand, there does not seem to be a good alternative for dealing with black-box systems.

This situation calls for a concerted action to improve the state of the art of practical learning. A first attempt in this direction was the ZULU challenge. ZULU asked for learning solutions that function without the use of so-called equivalence queries, which classically could be used to automatically derive a counterexample distinguishing a learned hypothesis model from the target system until equivalence has been established. The reason for excluding equivalence queries, which guaranteed the correctness and completeness of the theoretical framework, was their lack of realism: in essentially all practical applications, equivalence queries need to be approximated by the so-called membership

queries, which often can be implemented via testing. ZULU therefore took exactly this approach and allowed membership queries only, and, indeed, only a comparatively small number of them. Also this latter limitation was motivated by a practical perspective: typically, the approximations of equivalence queries consume an enormous amount of membership queries, and this was considered unrealistic as well.

This paper summarizes our experience with the ZULU challenge on active learning without equivalence queries, presents our winning solution, and

- investigates the character of ZULU's rating approach, which ranks solutions according to their prediction quality for language containment based on a ZULU-generated test suite, and
- discusses how this approach can be taken further to establish a framework for the systematic investigation of domain-specific, scalable learning solutions for practically relevant application.

In particular, we discuss the RERS initiative [8], which provides a community platform together with a learning framework that allows users to interactively compose complex learning solutions on the basis of libraries for various learning components, system connectors, and other auxiliary functionality.

With RERS we want to establish a community of researchers and practitioners interested in the practical application of automata learning technology. RERS stands for Regular Extrapolation of Reactive Systems. We chose the term Regular Extrapolation to indicate that in practice we will not be able to fully infer the behavior of the target system, which often will not be regular anyway. Rather we are able to construct in some sense optimal regular images/views of these behaviors. We are convinced that a concerted effort for improving these techniques will lead to industrial scale learning solutions in near future.

Outline: After recapitulating briefly the central idea of active learning in Section 2, we will discuss the ZULU challenge, our approach to it, the outcome, and further observations in Section 3. Subsequently, we will give a short introduction to the RERS initiative in Section 4, and conclude in Section 5.

2 Active Learning

Active learning (or *query learning*) attempts to construct a deterministic finite representation, e.g., a deterministic finite automaton (DFA), that matches the behavior (language) of a given target system on the basis of observations of the target system and perhaps some further information on its internal structure. It poses *membership queries* that test whether certain strings/words (potential runs) are contained in the target system's language (its set of runs), and *equivalence queries* that compare intermediately constructed hypothesis automata for equivalence with the target system. As long as the equivalence queries fail, they produce counterexamples revealing some of the remaining differences between

hypothesis and target system. These counterexamples are then taken as the basis for further refinement of the hypothesis. Learning terminates successfully as soon as an equivalence query signals success.

Angluin's algorithm L^*, in its basic form, starts with a hypothesis automaton with only one state and refines this automaton on the basis of query results. In scenarios like for ZULU, which (like most practical settings) do not provide an equivalence oracle for reliably answering equivalence queries (see below), three main steps are iterated:

1. *refining the hypothesis* by means of membership queries,
2. *checking for equivalence* on the basis of membership queries e.g., following ideas from *conformance testing* (e.g., [9,13]), and
3. *analyzing counterexamples* as a means to initiate the next refinement step.

The procedure terminates (for finite state target systems) with a state-minimal deterministic (hypothesis) automaton equivalent to the target system. This is the consequence of the following dual characterizations of states, resembling the idea of Nerode's congruence [21,20]:

from below: by a set, $S \subset \Sigma^*$, of *access sequences*. This characterization of state is too fine, as different words $s_1, s_2 \in S$ may lead to the same state in the target system. L^* constructs such a set S, containing access sequences to all states of the target automaton. In addition, it maintains a second set, SA, which together with S covers all transitions of the hypothesis automaton. During learning, the following invariant is repetitively established: $SA = (S \cdot \Sigma) \setminus S$. L^* initializes S with $\{\varepsilon\}$, the set containing only the access sequence to the initial state, and, accordingly, SA with Σ, covering all transitions leaving in the initial state.

from above: by ordered set(s), $D \subset \Sigma^*$, of distinguishing sequences or *futures*. These sets can be regarded as a projection of Nerode's residual languages. Starting with the singleton set $\{\varepsilon\}$, L^* successively extends D until it suffices to identify all states of the smallest deterministic automaton.

The sets S, SA, and D are successively refined during the learning process until a model is reached, which is state-minimal and equivalent (at least up to the distinctive power of the given approximation of the equivalence oracle) to the target system. Algorithmic details and correctness arguments can be found in [2,17,26,20,19,27].

3 The ZULU Competition

ZULU [11] is a competition in active learning from membership queries: contestants had to develop active learning algorithms (following the general approach due to Dana Angluin [2]). Essential to ZULU have been two main ideas:

No equivalence queries: Equivalence queries (in the theoretical formulation of active learning) compare a learned hypothesis model with the target system for language equivalence and, in case of failure, return a counterexample

exposing a difference. Their realization is rather simple in simulation scenarios: if the target system is a model, equivalence can be tested explicitly. In practice, however, the system under test will typically be some kind of black-box and equivalence queries will have to be simulated using membership queries [16]. One goal of the ZULU competition was to intensify research in this practical direction.

Limited membership queries: Equivalence queries can be simulated using model-based testing methods [7] (The close relation between learning and conformance testing is discussed in [4]). If, e.g., an upper bound is known on the number of states the target system can have, the W-method [9] or the Wp-method [13] can be applied. Both methods have an exponential complexity (in the size of the target system and measured in the number of membership queries needed).

By allowing only a (very) limited number of membership queries for every problem, the ZULU competition forced the contestants to change the objective from 'trying to prove equivalence', e.g., by using conformance testing techniques, to 'finding counterexamples fast'.

These two ideas led to the following concrete competition scenario. The contestants had to compete in producing models of finite state acceptors by means of membership queries, which the ZULU platform would answer. For every task, the number of granted membership queries was determined by the ZULU system as the number of membership queries needed by some basic Angluin-style reference implementation to achieve a prediction quality of over 70%. Solutions were ranked according to their quality of prediction relative to a ZULU-generated suite of 1800 test words (the same suite as used to determine the number of allowed membership queries).

During the competition, all contestants had to compete in 12 categories: 9 'regular' categories ranging over finite state acceptors with (a) growing numbers of states and (b) growing alphabets, as well as 3 'extra' categories, in which the number of granted membership queries was reduced further. The overall winner was determined as the contestant with the best average ranking. A detailed description of the technical details can be found on the ZULU web page [12].

The background of our team at the TU Dortmund is the development of realistic reliable systems [24,25]. Accordingly, we are working on techniques to make active learning applicable to industrial size systems. Of course this also requires to (1) saving membership queries and (2) finding counterexamples fast, but for man-made systems. It turned out that dealing with randomly generated systems changes the situation quite a bit. Our solutions therefore arose in a two step fashion:

1. a generically optimized basis, not exploiting any knowledge about the structure or origin of the considered target systems. The main strategy here was to follow an *evolutionary* approach to hypothesis construction which explicitly exploits that the series of hypotheses are successive refinements. This helps organizing the membership and equivalence queries.

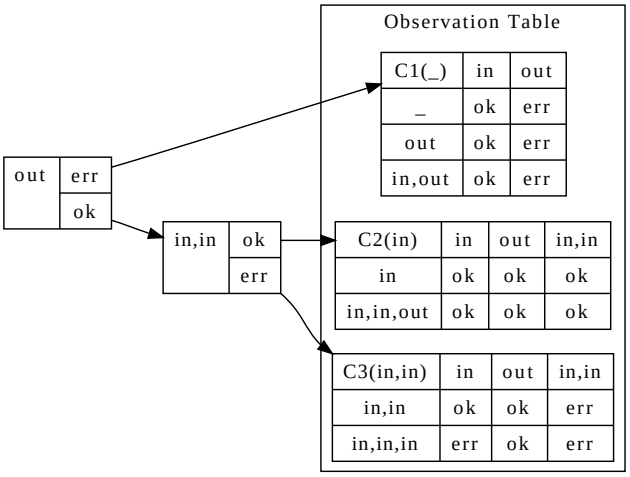

Figure 1. Extended Observation Packs

2. a subsequent customization for exploiting the fact that we are dealing with randomly generated systems. Here we were able to exchange our traditionally breadth-first-oriented approximations of an equivalence oracle, which exploit ideas from conformance testing, by a very fast counterexample finder. Also this customization benefits from the evolutionary character of the hypothesis construction.

The careful experimental investigation of the above options and its variants profited from our learning framework, the LearnLib [25], which we recently considerably extended. In particular, it enabled us to build a highly configurable learning algorithm, which can mimic most of the known algorithms, as well as all our variations in a simple fashion. In addition, the experimentation facilities of LearnLib, comprising, e.g.,remote execution, monitoring, and statistics, saved us a lot of time. We are planning to provide contestants of the RERS challenge [8] with all these facilities in order to allow contestants a jump start.

3.1 A Configurable Inference Framework

The main algorithmic pattern we used for the ZULU competition can best be described as generalized *Observation Pack* [3]: a pack is a set of components (which usually form the observation table). In the original discussion an observation pack is introduced only as a unifying formalism for the algorithms of [2,17]. We used a combination of discrimination trees [17] and reduced observation tables [26] to actually implement an observation pack. Fig. 1 illustrates the resulting data structure for a Mealy machine example (a three element buffer).

The observation table is split into independent components (essentially small observation tables in their own), each representing one state of the current hypothesis model, and being defined by one access sequence from the set S (see upper left corner). The second row of each component corresponds to the usual characterizing row for this access sequence in terms of some distinguishing futures taken from D, which label the columns. Additional rows stand for words of SA which happen to possess the same characterization in terms of the considered distinguishing futures, and therefore represent the same state in the hypothesis model. Please note that the description above does not hold for the third component which still needs to be split according to its difference with respect to the distinguishing future *in* (see below).

The left part of Fig. 1 shows the corresponding discrimination tree. Its nodes are labeled with elements of D, and its edges with the corresponding outcome of a membership query (here `ok` and `err` - one could also have chosen accept, non-accept). In the considered situation the discrimination tree simply reflects the fact that the first component can be separated from the other two due to the outcome concerning the distinguishing future `out`, whereas the second and third component can be separated from each other due to the outcome concerning the distinguishing future `in,in`. As indicated already, the third component could be split further due to the outcome concerning the distinguishing future `in`. Such a situation can only arise in the Mealy case, where the set of distinguishing futures is initialized with the full alphabet. In this concrete case, the alphabet symbol `in` is contained in the initial set of distinguishing futures but not yet in the discrimination tree.

In order to enter a new access string s into the discrimination tree, one starts with the root of the discrimination tree and successively poses the membership queries $s \cdot f$, where f is the distinguishing future of the currently considered node, and continues the path according to the corresponding output. If a corresponding continuation is missing, the discrimination tree is extended by establishing a new component for the considered access string. Finally, splitting a component requires the introduction of a new node in the discrimination tree, like e.g. for the distinguishing future `in` to split the third component.

This realization allows us to easily switch between different strategies for handling counterexamples (e.g., [26,19,27]), as well as to use non-uniform observation tables, i.e., observation tables where for different access sequences different distinguishing futures may be considered.

To enable working with non-uniform observation tables, we extended the strategy for analyzing counterexamples from [26]. In its original form, this strategy produces a new distinguishing future d by means of a binary search over the counterexamples. During the search prefixes of the counterexamples are replaced by access sequences to the according states in the hypothesis model, in order to maintain as much of the hypothesis model as possible. Taking this idea a bit further allows us to guarantee that the access sequence s for a new component is always taken from SA. This automatically maintains the structure of the spanning tree, and it guarantees that only the component containing s is refined by

the new distinguishing future d. The result of this refinement is a new node in the discrimination tree and two components (following the approach from [17]).

For ZULU, we configured two versions of our learning algorithm, both using this new strategy for analyzing counterexamples. The registered algorithms differed as follows.

Initial set of distinguishing futures: In one configuration, the initial set of distinguishing futures was initialized as $\{\varepsilon\}$ (as in the literature). In the other configuration, we used $\{\varepsilon\} \cup \Sigma$ in order to simulate the effect of changing from DFA to Mealy models. Please note that considering the empty word also for the Mealy configuration is a technical trick to better deal with the DFA-like systems considered here.

Observation Table: We used uniform and non-uniform observation tables. In a non-uniform table, the sets of distinguishing futures are managed independently for each component (cf. [3]).

Both decisions emphasize the same trade-off. Using the complete alphabet in the initial set and using a uniform observation table can be understood as a heuristic for finding counterexamples (in form of *unclosure*) in the table and thus reducing the number of equivalence queries. Using a non-uniform table and only proven distinguishing futures leads to using less membership queries but more equivalence queries.

3.2 Continuous Equivalence Queries

Essentially, active learning algorithms proceed in rounds triggered by negative outcomes of *equivalence queries* for successively improved hypotheses: returned counterexamples are analyzed and exploited to initialize the next round of refinement. Classically, each of these rounds were considered independent, and in fact, equivalence queries were simply provided with the current hypothesis model, which itself was independently constructed for each round. No knowledge about the algorithmic history was exploited. This is not too surprising for two reasons:

- Classically, equivalence queries were considered as atomic, with no need for a dedicated optimization, a point of view also supported by experimental settings, where the target systems are given as automata, which can efficiently be compared with the hypothesis automata.
- However, there is also a more technical argument. The hypothesis automata, which are refined during the learning process, are defined solely by the state characterization from above (their characterizing set), which is not fully reflected in the hypotheses. Thus the knowledge of the hypotheses alone is insufficient to establish the required notion of refinement.

In the meantime it is widely accepted that membership query-based approximations are key to practical application. The ZULU challenge itself is a very strong indication of this new perspective. Moreover, there are ways to strongly

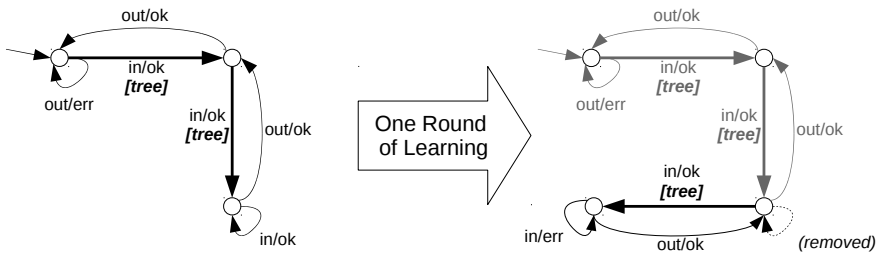

Figure 2. Evolving Hypothesis

link the characterizations from below and above in a way that allow some kind of *incremental hypothesis construction* by means of a global, continuous equivalence querying process. Key to this approach are Rivest's and Schapire's reduced observation tables together with their way of analyzing counterexamples [26]). In their setting, the prefix-closed set S can be understood as a successively produced spanning tree of the target automaton, whose set of nodes is extended by elements from the SA-set. Thus there is always a unique representation of the states of the hypothesis automaton in terms of the monotonically increasing, prefix-closed set S. This invariant is maintained by the specific treatment of counterexamples:

> Each counterexample is minimized by means of binary search to a word/string sa of SA, followed by a distinguishing future d that forces sa to end up in a new state.

This form of counterexample allows maintaining the above invariant by moving the corresponding s from SA to S, add d to the set of futures D, and to continue with the usual procedure establishing closedness. Besides avoiding the construction of hypotheses from scratch, this procedure leads to a sequence of incrementally refined hypotheses, which allows for organizing the equivalence tests from a global perspective, sensitive to all the tests which have been performed in earlier iterations.

Fig. 2 shows how an evolving hypothesis is refined during one round of the learning process. The hypothesis in the left of the figure corresponds to the extended observation packs from Fig. 1. All transitions are labeled by annotations, indicating whether they belong to the spanning-tree or not. This information is vital, as it indicates proven knowledge. Adding a new state to the hypothesis leaves most of the original hypothesis and its annotations unaffected. In the example of Fig. 2 only the former loop at the third state is modified and two new transitions are introduced.

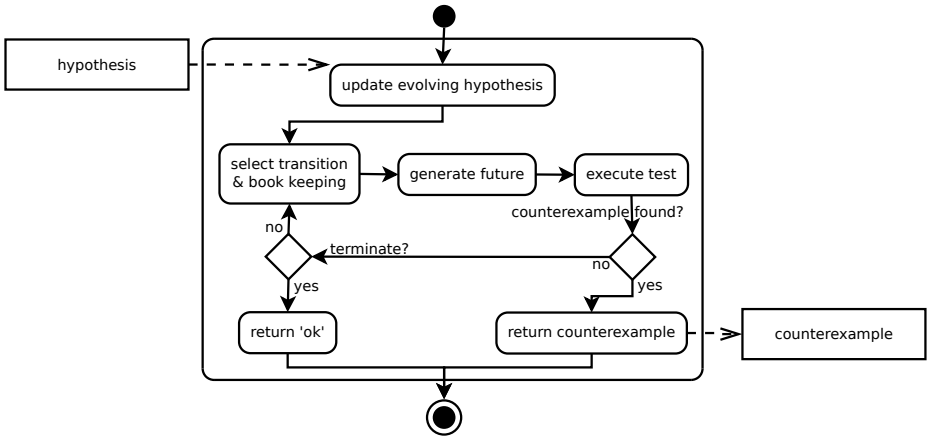

Figure 3. Continuous Equivalence Query

Beyond being a possible target for counterexample reduction (after some oracle provided an arbitrarily structured counterexample), the counterexample pattern

$$c = sa \cdot d, \text{ with } sa \in SA \text{ and } d \in \Sigma^+.$$

turned out to be ideal for realizing membership oracle-based equivalence oracles, or better, for implementing a method for revealing counterexamples fast along the lines shown in Fig. 3. Counterexample candidates sa are are tested for some heuristically chosen futures d until either a counterexample is found or some termination criterion is met. The effectiveness of the search heuristics for selecting counter example candidates and finding distinguishing futures d may strongly depend on side knowledge. For the ZULU challenge, we exploited our knowledge that the languages were randomly generated as follows:

Select transitions & Book keeping: For the *E.H.Blocking* algorithm, transitions from the SA-set were chosen randomly. Once used, a transition was excluded from subsequent tests. When no transitions were left to choose from, all transitions were re-enabled. The *E.H.Weighted* algorithm uses weights on all transitions, which are increased each time a transition is selected, and chooses transitions with a probability inversely proportional to its weight.

Generate futures: The futures were generated randomly with increasing length. The length was initialized as some ratio of the number of states of the hypothesis automaton, and was increased after a certain number of unsuccessful tests. The exact adjustment of the future length was developed in a experimentally to fit the properties of the problems in the ZULU challenge. This strategy of guessing comparatively long futures d turned out to be rather effective. It reduced the number of required membership queries to an average

Table 1. Algorithms: Configuration and Ranking

Algorithm	Dist. Set		Equivalence Query		Training	Rank
	Init.	Uniform	Continuous	Strategy	(Avg.)	
E.H.Blocking			yes	block transitions	89.38	1
E.H.Weighted	{ε}	no	yes	weight transitions	89.26	2
Random			no	random walks	88.93	6
run_random			no	random walks	80.17	14
run_blocking1	{ε}∪Σ	yes	yes	block transitions	79.89	15
run_weighted1			yes	weight transitions	79.65	16

of 2-4, which radically outperformed our initial breath-first trials. Of course, this is very much due to the profile of the ZULU examples, and indeed, this choice of futures was the only truly domain-dependent optimization we developed.

We did not use an explicit termination criterion. A query terminated as soon as the number of queries granted by ZULU was exhausted.

3.3 Results

For the actual competition, we registered six candidate algorithms, split in two groups of three:

- the first group used a non-uniform observation table with a DFA-style initial set of distinguishing futures, and
- the second group used a uniform observation table with a (modified) Mealy-style initial set of distinguishing futures.

Both groups were equipped with the same three equivalence checking algorithms: (1) E.H.Blocking, (2) E.H.Weighted, and (3) plain random walks. As the random walks algorithm simply tested randomly generated words, the option for continuous equivalence queries did not apply. Table 1 shows the configuration of the algorithms, their average scores during the training phase and the eventual rankings from the competition.
Apparently, group 1 is far superior: about 10 points, and this in a setting where there are only 0.45 points between the first and the sixth place. In order understand these results better, we investigated Problem 49763507 in more detail. In particular we wanted to understand how the ZULU ranking mechanism, which is based on predictions rates for randomly generated test suites, reflects the quality of Angluin-style implementations of automata learning.

3.4 Discussion of the ZULU Rating Approach

In order to better understand the progress of the learning algorithms, let us consider some profile data of the training problem 49763507 in more detail, for

Table 2. Detailed Training Example: Problem 49763507

Algorithm	New Membership Queries			Rounds	States	Score
	Close Obs.	Analyze CEs	Search CEs			
E.H.Blocking	6,744	358	999	259	352	94.11
E.H.Weighted	6,717	349	1,035	262	351	94.61
Random	6,586	519	996	228	332	93.28
run_random	8,080	14	7	5	312	74.89
run_blocking1	8,074	11	16	6	319	73.06
run_weighted1	8,077	9	15	6	319	74.39

which the ZULU environment allowed 8101 membership queries. Table 2 classifies the consumption of these 8101 queries according to their usage during (1) the closure of the Observation Table, (2) the analysis of counterexamples, and (3) the search for counterexamples. Moreover, it shows the number of learning rounds performed, the detected states, and the eventual scores.

These results are quite drastic:

1. The difference between the two groups discussed in the previous section is even more impressive here. It is obvious that the first group profited from the extremely efficient search for counterexamples, which required in average only about 3 membership queries. In fact, the algorithms in the first group executed 50 times as many approximative equivalence queries as the algorithms of the second group.
2. The impact of the continuous equivalence queries, which make only a difference of 1,8% in the ZULU ranking (E.H.Blocking vs Random), make about 6% in the number of detected states, which is a lot, when considering the typical behavior of Angluin-style learning. Just consider the only 3% difference in the number of detected states between the Random options of the first group and the second group. In the ZULU rating they make a difference of 19%.
3. Despite the extremely different distribution of the allowed 8101 membership queries, the average number of membership queries required to obtain a new state seem quite similar. They range between 23 and 26. One should keep in mind, however, that the difficulty to find a new state strongly increases in the course of the algorithm. Thus having detected 8% more states is much.
4. The ZULU ranking seems to increase with the number of states. This interpretation is, however, wrong, as can be seen in Fig. 4 (for ZULU problem 85173129), where in the beginning, the quality of prediction drops from 60 to almost only 40 per cent! For the already discussed example (49763507) this effect is not as strong, but still the non-monotone developing of ratings throughout the learning process can be observed in Fig. 5. As we will discuss later, this effect, which shows here up already for randomly generated systems (in fact, we observed it in almost all Problems we treated as part of the ZULU challenge), gets worse when systems are man-made.

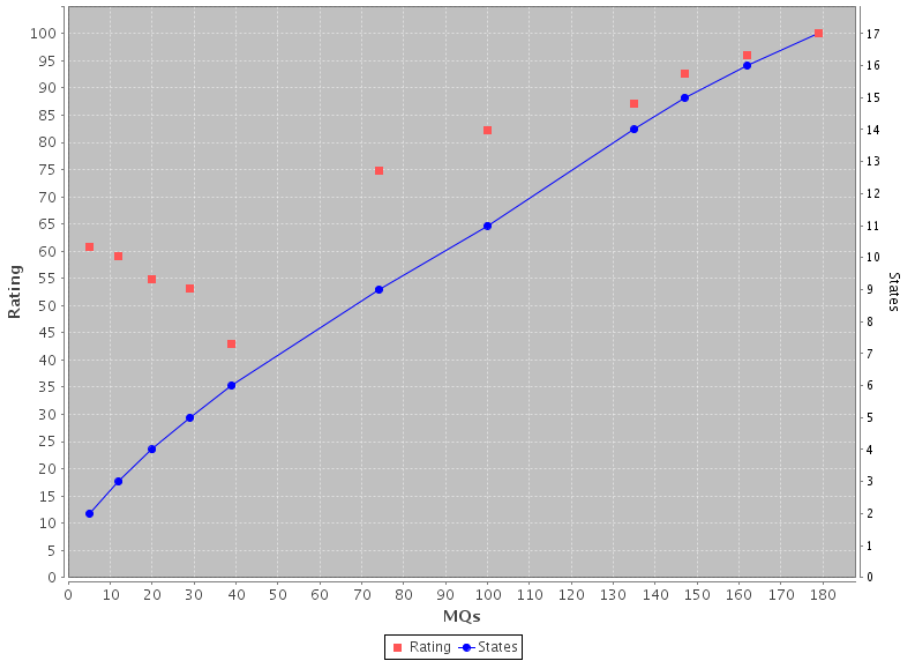

Figure 4. ZULU Rating for all hypotheses in a learning attempt, Problem 85173129

5. There is a part of the learning process, where small improvements make a big difference in the ZULU ranking (e.g. for the two random versions, 13 more states made a difference of 19% in the ZULU score). This is the part after the chaotic behavior at the beginning and before the final phase, where progress gets exponentially harder.

The ZULU organizers also observed that the top teams had quite similar scores. Looking at Table 2 this is not too surprising, as much progress is needed at the end to achieve even marginal improvements for the ZULU rating (20 more in this stage very hard to detect states for 1,8% in the ZULU rating: see E.H.Blocking vs. Random of the first group). It was therefore decided to further reduce the number of allowed membership queries. This certainly diversified the results. However, it also increased the risk of algorithms being punished by the quality drop described under item 4 above.

We consider the ZULU competition as milestone for advancing the state of the art for practical learning. Still, we see a high potential for improvement. The following section describes RERS [8], a new initiative trying to push in this direction.

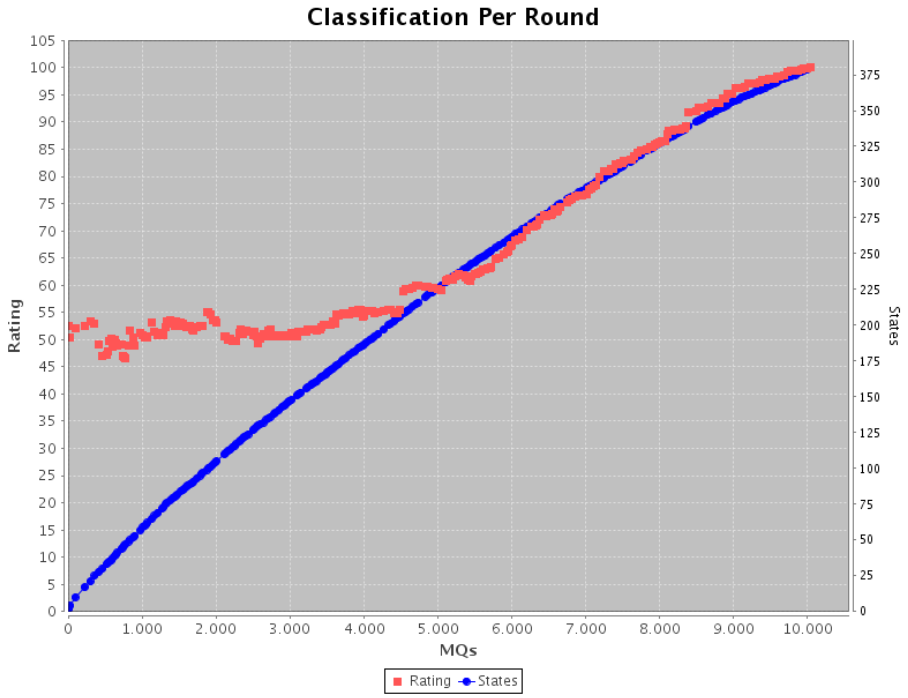

Figure 5. ZULU Rating for all hypotheses in a learning attempt, Problem 49763507

4 RERS - Regular Inference of Reactive Systems

As the discussion of the results from the ZULU competition shows, an adequate quantification of a hypothesis' quality may depend on the domain of application: the test suite-based ZULU rating primarily addresses prediction quality concerning language membership. Perhaps surprisingly, this measure does not necessarily reflect the performance of Angluin-style implementations of learning algorithms. As we have seen, the prediction quality may quite significantly drop in the course of learning. The RERS initiative [8] aims at developing fair evaluation measurements tailored to specific application contexts. The discussion in the previous section already suggests discovered states as a good orientation for algorithmic progress. In fact, in our experience, discovered states are also a good measurement in the context of system testing [14,15]: The more system structure (in terms of states) is revealed, the better are the results of test generation algorithms.

Developing domain-specific notions of quality is, however, only one dimension of RERS, although a very important one, as it sets the standards for other dimensions, like:

– Establishing scenarios for realistic case studies,
– Developing methods making automata learning scalable,
– Providing technology to support the system interaction required for learning,
– Generalizing the notion of learning to capture new kinds of system behavior.

RERS does not only attempt to promote research in this direction, but intends to establish a framework in which progress in these dimensions can be measured, solutions can be compared, and experiences can be exchanged. Organizing challenges with branches capturing the currently most essential features is meant to be one of RERS prominent roles.

For next year we plan a challenge focusing on automata learning as a key technology for dealing with *black-box systems*, i.e., systems that can be observed, but for which no or little knowledge about the internal structure or even their intent is available. By this we intend to reveal the state of the art concerning (at least) the following kinds of scenarios, specifically for both, randomly generated systems (frequently used for evaluation, see e.g., the ZULU challenge), and man-made systems, the predominant case in practice:

1. Black-box systems with equivalence queries are not relevant in practice, but ideal for learning. Usually only approximations of equivalence queries can be generated.
2. For fast black-box systems (e.g. simulated ones) the number of membership queries is not as important as in other scenarios. While seemingly of only little practical use, some use cases, e.g., learning behavior of APIs, can come close.
3. In scenarios with black-box systems with high-cost membership queries (e.g., systems that are slow to respond or generate actual cost per query) it makes much sense to limit the amount of generated queries and to investigate ways how to decrease effective costs.
4. Scenarios where data values have to be considered are common and motivate the introduction of means to abstract from infinite value domains. While manually generated and fixed abstractions may be sufficient in many cases it may be necessary to refine such an abstraction during the learning process, if possible in an automated fashion according to system output.
5. Variants of non-deterministic or probabilistic systems. Such systems typically require more enhanced membership queries, which cannot be realized via normal system testing and therefore pose new implementation requirements (see e.g. [18]).

Central will be here the investigation of the practical limitations and the potential of *active* automata learning, which is characterized by its specific means of observation, i.e., its proactive way of posing membership queries and equivalence queries. ZULU has addressed the problem that equivalence queries are typically not realistic and need to be approximated via membership queries. However, also membership queries do not come for free but must be realized e.g. via testing in practice. The following subsections discuss important practical challenges according to the various characteristics of application scenarios, and illustrate that 'black does not equal black' in real-life black-box scenarios:

Interacting with real systems The interaction with a realistic target system comes with two problems: A merely technical problem of establishing an adequate interface that allows one to apply test cases for realizing membership queries, and a conceptual problem of bridging the gap between the abstract learned model and the concrete runtime scenario.
The first problem is rather simple for systems designed for connectivity (e.g., Web-services or code libraries) which have a native concept of being invoked from the outside and come with documentation on how to accomplish this. Establishing connectivity may be arbitrarily complicated, however, for, e.g., some embedded systems which work within well-concealed environments and are only accessible via some proprietary GUI.
The second problem is conceptually more challenging. It concerns establishing an adequate abstraction level in terms of a communication alphabet, which on one hand leads to a useful model structure, but on the other hand also allows for an automatic back and forth translation between the abstract model level and the concrete target system level.

Membership queries Whereas small learning experiments typically require only a few hundred membership queries, learning realistic systems may easily require several orders of magnitude more. This directly shows that the speed of the target system when processing membership queries, or as in most practical settings the corresponding test cases, is of the utmost importance. In contrast to simulation environments, which typically process several thousand of queries per second, real systems may well need many seconds or sometime even minutes per test case. In such a case, rather than parallelization, minimizing the number of required test cases is the key to success.

Reset Active learning requires membership queries to be independent. Whereas this is no problem for simulated system, it may be quite problematic in practice. Solutions range here from reset mechanisms via homing sequences [26] or snapshots of the system state to the generation of independent fresh system scenarios. Indeed, in certain situations, executing each membership query with a separate independent user scenario may be the best one can do. Besides the overhead of establishing these scenarios, this also requires an adequate aggregation of the query results. E.g., the different user password combinations of the various used scenarios must be abstractly identified.

Parameters and value domains Active learning classically is based on abstract communication alphabets. Parameters and interpreted values are only treated to an extend expressible within the abstract alphabet. In practice, this typically is not sufficient, not even for systems as simple as communication protocols, where, e.g., increasing sequence numbers must be handled, or where authentication requires matching user/password combinations. Due to the complexity of this problem, we do not expect any comprehensive solutions here. We rather think that domain- and problem-specific approaches must be developed in order to produce dedicated solutions.

Within RERS we want to provide the means for contestants to comparatively easily enter this exciting field of research, and to contribute to next year's RERS

challenge. Accordingly, the LearnLib (comprising libraries of algorithms, system connectors, algorithmic patterns, abstraction function etc.) will be made publicly available. This will allow contestants to start from an existing basis and extend it with new functionality.

5 Conclusion

We have summarized our experience with the ZULU challenge on active learning, presented our winning solution, investigated the character of ZULU's rating approach, and discussed how this approach can be taken further to establish a framework for the systematic investigation of domain-specific, scalable learning solutions for practically relevant application. In particular, we have discussed the RERS initiative, which provides a community platform together with a learning framework that allows users to interactively compose complex learning solutions on the basis of libraries for various learning components, system connectors, and other auxiliary functionality. This framework will be the backbone for an extended challenge on learning in 2011.

There are multiple dimensions to RERS, like the design of problems specific model structures, the development of flexible abstraction technologies, and the adequate treatment of parameters and values treatment. These need to be related to learning-specific technologies for e.g. treating counterexamples, realizing equivalence queries, or exploiting domain-specific knowledge about the target systems, and, a potential show stopper in practice, to adequate testing technology enabling the required kind of querying.

We therefore envisage to establish targeted sub-communities specialized and interested in specific scenarios. One such scenario could be the 'classical' learning scenario as addressed e.g. by the ZULU challenge, but there are many more, each of them with a particular learning profile. Examples range from the learning of I/O automata or Mealy machines for complex reactive systems or protocols, over the construction of (abstract) behavioral models of software components e.g. as a means to automate assume guarantee reasoning, to the detection of business processes by real life observation.

We hope that RERS will develop as a platform for synergetic cooperation, that the organized challenges will help to identify the strength and weaknesses of the various solutions and their mutual interplay, and that the possibility for easy experimentation, e.g. using the LearnLib, will make automata learning a convenient tool in many application contexts.

References

1. Fides Aarts, Julien Schmaltz, and Fritz Vaandrager. Inference and abstraction of the biometric passport. In *4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation 18-20 October 2010 - Amirandes, Heraclion, Crete*, LNCS, 2010.

2. D. Angluin. Learning Regular Sets from Queries and Counterexamples. *Information and Computation*, 75(2):87–106, 1987.
3. José L. Balcázar, Josep Díaz, and Ricard Gavaldà. Algorithms for Learning Finite Automata from Queries: A Unified View. In *Advances in Algorithms, Languages, and Complexity*, pages 53–72, 1997.
4. Therese Berg, Olga Grinchtein, Bengt Jonsson, Martin Leucker, Harald Raffelt, and Bernhard Steffen. On the Correspondence Between Conformance Testing and Regular Inference. In Maura Cerioli, editor, *Proc. FASE '05, 8^{th} Int. Conf. on Fundamental Approaches to Software Engineering*, volume 3442 of *Lecture Notes in Computer Science*, pages 175–189. Springer Verlag, April 4-8 2005.
5. Therese Bohlin, Bengt Jonsson, and Sivash Soleimanifard. Inferring compact models of communication protocol entities. In *4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation 18-20 October 2010 - Amirandes, Heraclion, Crete*, LNCS, 2010.
6. Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, and Martin Leucker. Smyle: A Tool for Synthesizing Distributed Models from Scenarios by Learning. In Franck van Breugel and Marsha Chechik, editors, *Proc. of 19^{th} Int. Conf. on Concurrency Theory (CONCUR '08), Toronto, Canada, August 19-22, 2008*, volume 5201 of *Lecture Notes in Computer Science*, pages 162–166. Springer, 2008.
7. Manfred Broy, Bengt Jonsson, Joost-Pieter Katoen, Martin Leucker, and Alexander Pretschner. *Model-Based Testing of Reactive Systems:*, volume 3472 of *Lecture Notes in Computer Science*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
8. TU Dortmund Chair of Programming Systems, Department of Computer Science. RERS - A Challenge In Active Learning. `http://leo.cs.tu-dortmund.de:8100/`. Version from 20.06.2010.
9. Tsun S. Chow. Testing Software Design Modeled by Finite-State Machines. *IEEE Trans. on Software Engineering*, 4(3):178–187, May 1978.
10. Jamieson M. Cobleigh, Dimitra Giannakopoulou, and Corina S. Pasareanu. Learning assumptions for compositional verification. In *Proc. TACAS '03, 9^{th} Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 2619 of *Lecture Notes in Computer Science*, pages 331–346. Springer Verlag, 2003.
11. David Combe, Colin de la Higuera, and Jean-Christophe Janodet. Zulu: an Interactive Learning Competition. In *Proceedings of FSMNLP 2009*, 2010. to appear.
12. David Combe, Colin de la Higuera, Jean-Christophe Janodet, and Myrtille Ponge. Zulu - Active learning from queries competition. `http://labh-curien.univ-st-etienne.fr/zulu/index.php`. Version from 01.08.2010.
13. Susumu Fujiwara, Gregor von Bochmann, Ferhat Khendek, Mokhtar Amalou, and Abderrazak Ghedamsi. Test Selection Based on Finite State Models. *IEEE Trans. on Software Engineering*, 17(6):591–603, 1991.
14. Andreas Hagerer, Hardi Hungar, Oliver Niese, and Bernhard Steffen. Model generation by moderated regular extrapolation. *Lecture Notes in Computer Science*, pages 80–95, 2002.
15. Andreas Hagerer, Tiziana Margaria, Oliver Niese, Bernhard Steffen, Georg Brune, and Hans-Dieter Ide. Efficient regression testing of CTI-systems: Testing a complex call-center solution. *Annual review of communication, Int.Engineering Consortium (IEC)*, 55:1033–1040, 2001.
16. Hardi Hungar, Oliver Niese, and Bernhard Steffen. Domain-Specific Optimization in Automata Learning. In Warren A. Hunt Jr. and Fabio Somenzi, editors, *Proc. 15^{th} Int. Conf. on Computer Aided Verification*, volume 2725 of *Lecture Notes in Computer Science*, pages 315–327. Springer Verlag, July 2003.

17. Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, USA, 1994.
18. Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. Assume-Guarantee Verification for Probabilistic Systems. In *TACAS*, pages 23–37, 2010.
19. Oded Maler and Amir Pnueli. On the Learnability of Infinitary Regular Sets. *Information and Computation*, 118(2):316–326, 1995.
20. Tiziana Margaria, Oliver Niese, Harald Raffelt, and Bernhard Steffen. Efficient test-based model generation for legacy reactive systems. In *HLDVT '04: Proceedings of the High-Level Design Validation and Test Workshop, 2004. Ninth IEEE International*, pages 95–100, Washington, DC, USA, 2004. IEEE Computer Society.
21. A. Nerode. Linear Automaton Transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958.
22. Corina S. Pasareanu, Dimitra Giannakopoulou, Mihaela Gheorghiu Bobaru, Jamieson M. Cobleigh, and Howard Barringer. Learning to divide and conquer: applying the L* algorithm to automate assume-guarantee reasoning. *Formal Methods in System Design*, 32(3):175–205, 2008.
23. Doron Peled, Moshe Y. Vardi, and Mihalis Yannakakis. Black Box Checking. In Jianping Wu, Samuel T. Chanson, and Qiang Gao, editors, *Proc. FORTE '99*, pages 225–240. Kluwer Academic, 1999.
24. Harald Raffelt, Maik Merten, Bernhard Steffen, and Tiziana Margaria. Dynamic testing via automata learning. *Int. J. Softw. Tools Technol. Transf.*, 11(4):307–324, 2009.
25. Harald Raffelt, Bernhard Steffen, Therese Berg, and Tiziana Margaria. LearnLib: a framework for extrapolating behavioral models. *Int. J. Softw. Tools Technol. Transf.*, 11(5):393–407, 2009.
26. Ronald L. Rivest and Robert E. Schapire. Inference of finite automata using homing sequences. *Inf. Comput.*, 103(2):299–347, 1993.
27. Muzammil Shahbaz and Roland Groz. Inferring Mealy Machines. In *FM '09: Proceedings of the 2nd World Congress on Formal Methods*, pages 207–222, Berlin, Heidelberg, 2009. Springer Verlag.
28. Muzammil Shahbaz, Keqin Li, and Roland Groz. Learning Parameterized State Machine Model for Integration Testing. In *Proc. 31st Annual Int. Computer Software and Applications Conf.*, volume 2, pages 755–760, Washington, DC, USA, 2007. IEEE Computer Society.