Refining active learning to increase behavioral coverage

Kousar Aslam[1], Yaping Luo[1,2], Ramon Schiffelers[1,3], Mark van den Brand[1]
1. Eindhoven university of Technology, The Netherlands 2. ESI (TNO) 3. ASML
{k.aslam,y.luo2,r.r.h.schiffelers,m.g.j.v.d.brand}@tue.nl

ABSTRACT

Modern high-tech industry is dealing with the maintenance of complex software systems today which consist of a large number of interconnected software components. Many of these components become legacy over years due to lack of documentation and unavailability of original developers. Several techniques are available in literature to retrieve the behavioral models from the existing software. Among those, the dynamic analysis techniques analyze the actual execution of the software, either via execution traces (passive learning), or by interaction with the software components (active learning). These techniques cannot guarantee alone to learn the complete and correct software behavior due to the limitations of each technique. We present an approach to aid active learning technique with software logs (execution traces) and passive learning result to increase the behavioral coverage of learned models.

1 INTRODUCTION

The dynamic analysis techniques analyze the software in execution mode. In active learning [1], the system under learning (SUL) is isolated from its environment. The learning algorithm interacts with the SUL and observes the responses from the component. A hypothesis is developed based on these observations and sent to an equivalence oracle. The equivalence oracle is a conformance testing tool which tests the hypothesis. If a discrepancy is found between the SUL and the hypothesis, learning needs to be refined. The hypothesis is accepted otherwise as the inferred model.

In practice, the equivalence oracle cannot guarantee the completeness of hypothesis due to the inherent limitation (only finite test cases generated) of the testing methods used. For passive learning techniques, the learning algorithms (particularly those accepting only positive examples, like [2]), usually approximate the behavior from the logs. Due to this reason, the learned models do not necessarily contain the complete behavior of SUL. An approach is proposed in [3] to improve the active learning result with logs and passive learning result. The author suggests to learn a software component with both active and passive learning techniques. Logs are then replayed over the active learning result to make sure that the result contains all of the observed behavior. Logs that are not able to be replayed are used as counterexamples to refine the result. The valid approximation from the passive learning result is then later added to the active learning result.

2 OUR APPROACH

In this paper, we propose an efficient version of the approach discussed in [3]. We suggest to intervene the active learning framework by enhancing the equivalence oracle with logs and passive learning result. Instead of learning the whole component first with the active learning and refining it later, we construct a hypothesis with active learning algorithm and then use a sequential equivalence oracle which already makes sure that the logs and valid parts of passive learning result are included in the active learning result, as shown in Fig. 1. We enhance the equivalence oracle as described below.

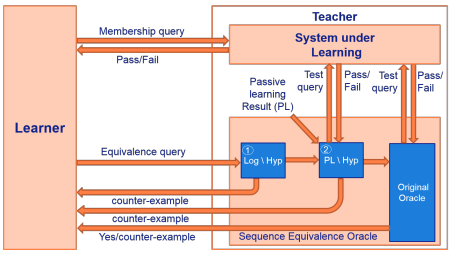

Figure 1: Enhanced equivalence oracle.

1. Use logs as free source of counterexamples: Logs represent how a software component uses, and is used by, its surrounding components. The hypothesis constructed by active learning algorithm should be able to replay all the traces from the logs because logs record real software behavior. To ensure that the hypothesis constructed by active learning algorithm contains the logs, we feed logs to the equivalence oracle and compute the difference between logs and the hypothesis. The divergent traces found are used as counterexamples which refine the learning. In this way, we generate counterexamples without making any test queries to the SUL.

2. Use valid approximation from passive learning result:

When the behavior of a software component is learned with passive learning technique from logs, the resulting model may contain an over-approximation or under-approximation introduced by the passive learning algorithm. The approximation can be valid or invalid. Once we differentiate between valid and invalid approximation, the valid behavior inferred by passive learning algorithm can be used to refine active learning. In this step, we compute the difference between passive learning result and the hypothesis from first oracle. Traces are generated from this difference and posted as test queries to the SUL. If accepted, the trace is valid and should be included in the behavioral model. So the trace is sent to the learner to refine the learning. The rejected traces are discarded.

The hypothesis is then sent to original oracle which either accepts it or sends a counterexample to the learner. Following this approach we reduce the uncertainty caused by equivalence oracle and increase the confidence in active learning result. Currently we are working on validating our approach over a large set of software components to estimate the efficiency of this approach.

REFERENCES

[1] Dana Angluin. 1987. *Learning regular sets from queries and counterexamples. Information and computation* 75, 2 (1987), 87–106.
[2] Wil MP Van der Aalst. 2016. *Process mining: data science in action.* Springer.
[3] Nan Yang. 2018. *Combining model learning results for interface protocol inference.* Master's thesis. Eindhoven University of Technology.