Assignment - Model Learning

Testing Techniques 2024 - 2025

In this assignment you will learn models of two systems: one with a user interface, such that it is easy to see what the system does, and one without user interface. The end-result should be a report with answers to questions Q1 - Q10. Before starting, download `learning-project.zip` from Brightspace, extract the zip, and follow the included LearnLib tutorial `learnlib-tutorial.pdf` to get started learning with LearnLib. The learning project used in this tutorial is also used for this assignment.

1 Chocolate bar machine

You will learn a model of an SUT (the chocolate bar machine website), by providing counterexamples to hypotheses yourself. Furthermore, you will learn the same SUT by configuring appropriate testing methods to find counterexamples for the learned models automatically.

Learning methods & counterexamples

- Open in your browser the chocolate bar machine website, which is located at `sul/chocolate_bar_machine/website.html`, and click around to see how it works.
- Learn for each of the learning methods L*, RivestShapire and TTT a model by providing counterexamples yourself. You do not need to learn the complete model, just learn enough to answer the following question.

Q1: What differences do you notice between the learning methods? Think about performance and learning results. Performance of a learning method is usually expressed in the number of input symbols required.

- Now, try to learn the complete model with a learning method of your choice.

Q2: Describe the learning process: Which learning method did you choose? What counterexamples did you provide? Which intermediate hypotheses did you get?

Q3: Why do you think you have learned the complete model? Give an argument.

- Using the RivestShapire learning method start learning from scratch again, but now you provide the following counterexample: `10ct 10ct 5ct 10ct 5ct snickers twix 5ct mars`. Provide this counterexample multiple times until the learner say it isn't a counterexample anymore.

Q4: Explain, based on the theory from the lecture (i.e. with L* and explained counterexample handling), why it is possible that an input trace is accepted multiple times as a counterexample.

Learn automatically using an automated testing method

- Experiment with learning using automated testing by changing the TestingMethod from UserQueries to one of the automated testing methods.
- Choose one learning method to be used for the rest of the assignment. Using this fixed learning method we are going to study the different testing methods in LearnLib.

Q5: Which learning method did you choose and why?

- Learn a model with the testing methods RandomWalk, WMethod, and WpMethod. Use the default settings as given in the code.

Q6: For each of the testing methods, answer the following questions:

  - After how much time does the learner stop?
  - Is the found model the same model as you learned by supplying counterexamples yourself? Why (not)?

Q7: Explain which testing method worked the best according to you.

2 Bounded Retransmission Protocol

The Bounded Retransmission Protocol [1] has been developed by Philips to support infrared communication between a remote control and a television. There is a reference implementation of the sender of the protocol. Your task is to find out which out of six implementations built by other manufacturers conform to the reference implementation.

- Every implementation is a `.jar`-file in the `sut/brp/` folder. Start one of them by entering `java -jar filename.jar` in a command line terminal.
- We connect a learner to it via a socket-interface. Create a learning experiment, by replacing in `ExampleExperiment.java` the `ExampleSUL()` by `SocketSul(ip,port, true, ‘‘reset’’)`:
  - As IP-address, use `InetAddress.getLoopbackAddress()`.
  - As port-number, use the port printed by the implementation.
  - As command to reset the SUT, use the string “reset”.
  - Let the learner send a newline after every input.

The input symbols are: IACK, IREQ_0_0_0, IREQ_0_0_1, IREQ_0_1_0, IREQ_0_1_1, IREQ_1_0_0, IREQ_1_0_1, IREQ_1_1_0, IREQ_1_1_1, ISENDFRAME, ITIMEOUT.

- Learn a model of all implementations.
- The output symbols OCONF_0, OCONF_1, and OCONF_2 should all appear in the final model. For these SUTs, RandomWalk finds counterexample the quickest, when choosing appropriate values for `BasicLearner.randomWalk_chanceOfResetting` and `BasicLearner.randomWalk_numberOfSymbols`.

Q8: What values did you choose for these settings?

Q9: Did you manage to learn models with all outputs for all the implementations of the manufacturers? If not, why do you think you did not succeed?

- To find out which of the implementations of the manufacturers is equal to the reference implementation, call the `main` method of `BRPCompare.java` in the `brpcompare` package in the learner project to compare two `.dot`-models. It uses LearnLib's `Automata.findSeparatingWord` method to find a separating sequence between two models. Compare the model of the reference implementation with every implementation of a manufacturer.

Q10: For each of the implementations of the manufacturers, write down whether it was equal to the model of the reference implementation. Provide the found counterexample if the models are unequal. Do not forget to provide the expected and found output of the counterexample.

References

[1] Helmink, Leen, Martin Paul Alexander Sellink, and Frits W. Vaandrager. "Proof-checking a data link protocol." *International Workshop on Types for Proofs and Programs.* Springer Berlin Heidelberg, 1993.