Inferring Log-Based Behavioural System Models using Markov Chains

Author: Thomas Werthenbach T.A.K.Werthenbach@student.tudelft.nl
Supervisors: Mitchell Olsthoorn and Dr. Annibale Panichella {M.J.G.Olsthoorn, A.Panichella}@.tudelft.nl

1. Problem

Obtaining a model of system behaviour has several advantages:

- Test case generation [1].
- Analysis of software processes [2].
- Improve software quality [3].

Existing techniques do not scale well, as the problem of inferring a minimalistic finite state machine is NP-Hard [4].

The aim of this research is to evaluate the effectiveness of using Markov chains for inferring a concise yet accurate state model of system behaviour using log analysis.

2. Unique state graph

Log statements correspond to an event type.

Log traces (sequences of log statements) are represented in a graph where all nodes correspond to one or more event types.

- Contains at most one node for every event type.
- Grows with the number of unique event types.
- All log traces start in the same state.
- All log traces end in the same state.
- Every edge has an empirically determined probability for that event to occur, given that the next event type is unknown.

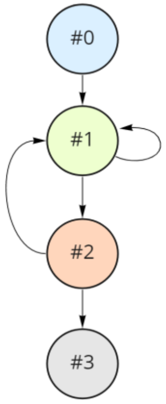

Figure 1. Simple unique state graph.

3. Markov chains

Markov chains are models which describe the probability of a certain state transition based only on the current state.

Adjacency matrices can be compressed by:

- Merging states which correspond to equivalent rows [5].
- Merging states which correspond to equivalent columns [5].
- Merging states with a transition probability of 1.

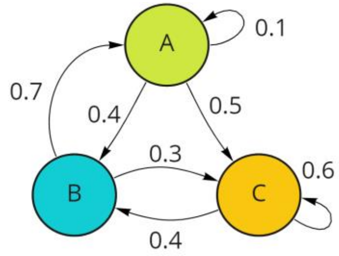

Figure 2. Simple state machine with probabilities indicated per edge.

$$\begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 0.1 & 0.4 & 0.5 \\ 0.7 & 0 & 0.3 \\ 0 & 0.4 & 0.6 \end{pmatrix} \end{array}$$

Figure 3. The corresponding adjacency matrix of the state machine in Figure 2.

4. Empirical study

An empirical study was performed on the logs of the XRP Ledger Consensus Protocol.

The model's accuracy was measured using the following metrics: Specificity, Recall, Precision, and F-measure.
Results were collected by compressing the model to a one-node model, while measuring the metrics on intermediate results.

A run-time experiment was conducted in which a model was trained and compressed for five different dataset sizes.

5. Results

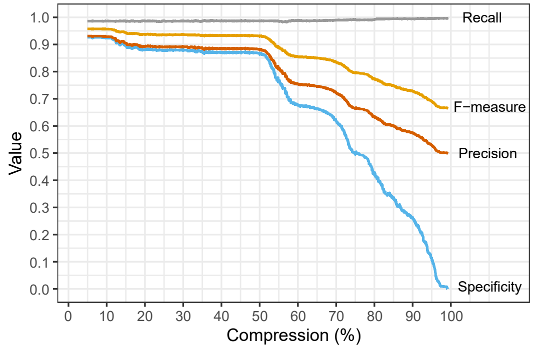

Figure 4. Results of the accuracy experiment.

- All metrics score high before the compression rate of 52%.
- Recall scores high.
- Specificity scores 0 at 100% compression.
- There is a trade-off: accuracy is sacrificed for conciseness.

- Consistent results.
- Run-time scales linearly with the size of the dataset.

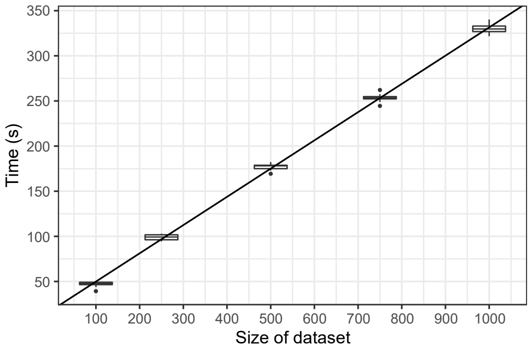

Figure 5. Results of the run-time experiment.

6. Conclusion

- Scales linearly in run-time.
- All metrics score high for compression rates lower than 50%.
- Several clear knee points.

[1] G. Fraser and N. Walkinshaw. 2012. Behaviourally adequate software testing. In 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation. IEEE, 300–309.
[2] R.M. Greenwood. 1992. Using CSP and system dynamics as process engineering tools. In European Workshop on Software Process Technology. Springer, 138–145.
[3] F. Wagner. 2006. Modeling software with finite state machines: a practical approach. auerbach Publications
[4] A. W. Biermann and J. A. Feldman. 1972. On the Synthesis of Finite-State Machines from Samples of Their Behavior. IEEE Trans. Comput. C-21, 6 (1972), 592–597. https://doi.org/10.1109/tc.1972.5009015
[5] W. M. Spears. 1998. A Compression Algorithm for Probability Transition Matrices. SIAM J. Matrix Anal. Appl. 20, 1 (1998), 60–77. https://doi.org/10.1137/s0895479897316916