LM: A Miner for Scenario-Based Specifications

Tuan-Anh Doan[1], David Lo[2], Shahar Maoz[3], Siau-Cheng Khoo[1]

[1]School of Computing, National University of Singapore
[2]School of Information Systems, Singapore Management University
[3]Department of Computer Science 3 (Software Engineering), RWTH-Aachen, Germany

doanta@comp.nus.edu.sg, davidlo@smu.edu.sg,
maoz@se.rwth-aachen.de, khoosc@comp.nus.edu.sg

ABSTRACT

We present LM, a tool for mining scenario-based specifications in the form of Live Sequence Charts, a visual language that extends sequence diagrams with modalities. LM comes with a project management component, a wizard-like interface to the mining algorithm, a set of pre- and post-processing extensions, and a visualization module.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*Restructuring, reverse engineering, and reengineering*

General Terms

Algorithms, Design, Experimentation

Keywords

Specification Mining, Live Sequence Charts, Tool

1. INTRODUCTION

Specification mining [3] aims at extracting behavioral models of a program under investigation from a set of its execution traces. The mined models can assist in program comprehension, documentation, and debugging tasks, in the absence of complete and up-to-date specifications. Specifically, in this paper we present LM, a tool for *scenario-based specification mining* [10], where a data mining approach is used to process a set of execution traces into scenario-based specifications in the form of Live Sequence Charts (LSC).

LSC [4, 5] is an intuitive yet expressive specification language used to represent interactions between system components. Roughly, LSC extends industry standard Message Sequence Charts (MSC) with existential and universal modalities. A (universal) LSC consists of two parts, a pre-chart and a main-chart, and carries the following semantics: Whenever the events in the pre-chart occur in the specified order, eventually the events in the main-chart must occur in the specified order. The LSC does not restrict events not appearing in it to occur or not to occur during a run.

The popularity and intuitive nature of sequence diagrams as a specification language in general, together with the additional unique features of LSC, motivate our choice for the target formalism of our work.

Based on the LSC semantics, we extract positive and negative witnesses of an LSC in an input trace. Given an input trace, our goal is to extract *significant* LSCs: those with enough number of positive witnesses and relatively few negative witnesses. We refer to the former notion as the *support* of the LSC and the latter as the *confidence* of the LSC over the given trace.

The core mining algorithm is statistically sound and complete: given input traces and thresholds of minimum support and confidence, we guarantee that all mined LSCs are significant, and all significant LSCs are mined (see [10]).

LSC mining extends sequential pattern mining [2, 6]. Different from typical sequential pattern mining that mines from sequences of atomic symbols, LSC mining mines from an event-trace where each event is a triple consisting of caller, callee, and the signature of the method being invoked. Our output is also not simply a sequence of symbols, but rather a modal sequence chart obeying the semantics of LSC.

In order to make the result more useful with the help of input from the engineer, LM provides, in addition to its core mining, a set of pre- and post-processing features – user-defined filters and abstractions – optionally used before and after the mining is performed. Some pre- and post-processing functions are embedded into the mining process to accelerate it and to remove uninteresting search spaces. Others aim at allowing the engineer to customize the mining results to the needs of the specific task at hand. For example to let the engineer define events that the miner should ignore although they do appear in the raw traces captured from the system under analysis or group the mined LSCs into sets representing corresponding class-level LSCs (see [8]).

It is important to note that scenario-based approach to modeling, in general, does not aim at providing complete systems' specifications. Rather, the strength of the scenario-based approach to modeling is that it allows the specifier to break up a specification into pieces of behavior or scenarios, each of which cuts across multiple objects.

The success of any software engineering technique in general and of specification mining in particular depends not only on its solid theoretical foundations but also on its ac-

ICSE '10, May 2-8 2010, Cape Town, South Africa

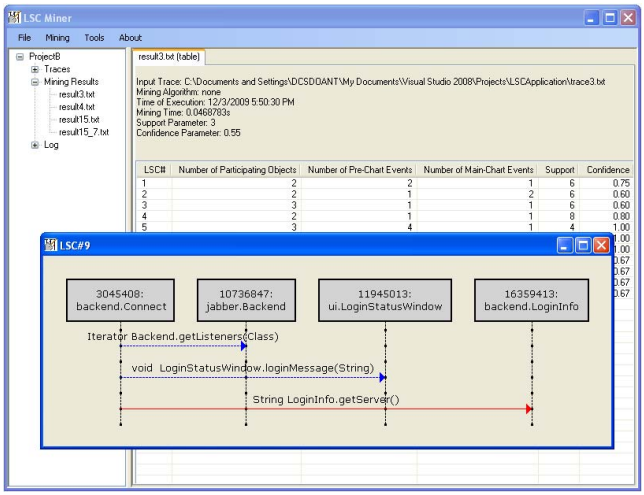

Figure 1: A screen dump from LM

cessibility to engineers. Specifically, in scenario-based specification mining, a scalable implementation with acceptable performance and an intuitively usable interface to define the mining goals and explore the mining results are critical to the future adaptation of our approach in industrial settings. LM indeed tries to address these challenges.

We have used LM to perform a number of case studies, the results of which can be found in [1]. Below we give an overview of the tool.

2. OVERVIEW OF LM

LM is implemented in C# and is currently a stand-alone application. A screen dump from the user interface of LM is shown in Fig. 1. Typical usage follows four phases:

1. Trace upload
2. Basic mining (with or without pre-processing options)
3. Post-processing
4. Results exploration and visualization

Mining tasks are organized into projects, each of which may include a number of execution traces. The result of various mining and post-processing operations are stored inside the project, which can be saved and loaded later for further work.

2.1 Input Traces

LM is agnostic to the language the program under analysis is written in. Thus, engineers may use existing instrumentation tools to collect traces, e.g., AspectJ for Java, AspectC for C, Valgrind for binary C, JRAT for Java byte code etc. After a program is instrumented, it could be executed and a trace or a set of traces is collected. The engineer uploads the traces into a project in LM.

2.2 Mining & Post-Processing

Once a trace set is uploaded into a project, it can be mined. Setting the parameters for the mining process is done using a wizard-like interface. The basic settings that the engineer must provide consist of the minimum thresholds of support and confidence. Various advanced settings for pre-processing options can be set too.

The results of the mining are saved within the project, and can be further analyzed using the various post-processing extensions. The results of separate mining sessions over the same trace or set of traces are stored under the same project.

2.3 Exploring the Mining Results

The mining results consist of a set of LSCs, annotated with their support and confidence metric values over the input traces. As typically many scenarios are mined, a simple textual representation of the results is not very useful. We thus follow a *details-on-demand* approach and allow the engineer to explore the mining results at two levels.

First, we show a table, listing the mined LSCs and their metric values. In addition to the support and confidence metric values, this high-level view of the results shows how many objects participated in the interaction described by each of the mined LSCs.

Second, the engineer may drill-down for more details on a selected scenario, by opening up a visual representation of the selected chart. This takes advantage of the intuitive visual syntax of LSCs to show the participating objects, the methods involved, and their orders.

Further exploration of the mining results is available on the trace itself. For example, in order to look for evidence of the calculated support and confidence metric values for a selected mined LSC, the engineer may choose to view the textual trace with colored highlights showing the positive and negative witnesses.

3. CONCLUSION

We presented LM, a tool for mining scenario-based specifications in the form of LSC. Further details about LM and the results of several case studies are available in [1]. Future work includes the extension of LM with additional scenario-based mining algorithms from [7, 9].

4. REFERENCES

[1] http://www.comp.nus.edu.sg/~dlo/lscminer/.
[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE*, 1995.
[3] G. Ammons, R. Bodik, and J. R. Larus. Mining Specification. In *POPL*, 2002.
[4] W. Damm and D. Harel. LSCs: Breathing Life into Message Sequence Charts. *J. on Formal Methods in System Design*, 19(1):45–80, 2001.
[5] D. Harel and S. Maoz. Assert and Negate Revisited: Modal Semantics for UML Sequence Diagrams. *Software and Systems Modeling*, 7(2):237–252, 2008.
[6] D. Lo, S.-C. Khoo, and C. Liu. Efficient mining of iterative patterns for software specification discovery. *SIGKDD*, 2007.
[7] D. Lo and S. Maoz. Mining Scenario-Based Triggers and Effects. In *ASE*, pages 109–118. IEEE, 2008.
[8] D. Lo and S. Maoz. Specification mining of symbolic scenario-based models. In *PASTE*, pages 29–35. ACM, 2008.
[9] D. Lo and S. Maoz. Mining hierarchical scenario-based specifications. In *ASE*, 2009.
[10] D. Lo, S. Maoz, and S.-C. Khoo. Mining Modal Scenario-Based Specifications from Execution Traces of Reactive Systems. In *ASE*, 2007.