Input-Output Conformance Testing of Software Product Lines

Harsh Beohar and Mohammad Reza Mousavi

Center for Research on Embedded Systems, Halmstad University, Sweden
harsh.beohar@hh.se, m.r.mousavi@hh.se

1 Introduction

Software Product Lines (SPLs) have become common practice in software development and have been proven effective in mass production and customization of software. There have been several attempts to provide a structured discipline for testing SPLs. Furthermore, composing test suites in a structured way is also studied in different areas of software engineering. For instance, in [10], the authors present a model-based approach to test aspect-oriented programs. However, it appears from recent surveys [3, 4, 7, 6] that several fundamental approaches to model-based testing (based on finite state machines and labeled transition systems) are not yet fully adapted to and adopted in this domain.

In this abstract, we propose to adopt Input-Output Featured Transition Systems (IOFTSs) as fundamental and expressive models for model-based testing of SPLs. To this end, we adapt the traditional Input-Output Conformance (IOCO) theory [11] to allow for using IOFTSs as test models.

We define a notion of the test suite and the set of test cases generated from an IOFTS, which can be used for checking conformance. We define two notions of refinement, one at the level of IOFTSs and another one at the level of test suites, that allow for focusing on particular sets of features and eventually on a particular product. We show that these two refinements interact nicely, in that they lead to the same set of test cases.

This abstract is organized as follows. In Section 2, the notions of IOFTS and product derivation are explained informally. Section 3 provides a brief overview of our main results. In Section 4, some open issues for future research are outlined. Due to the space restriction, we only present a brief overview of our approach. We refer to [1] for a precise and detailed treatment of the approach.

2 Background

Feature diagrams [5, 9] have been used to model variability constraints in SPLs using a graphical notation. A feature diagram represents all valid products of an SPL in terms of features that are arranged hierarchically. Usually, feature diagrams are represented by a directed acyclic graph, of which each node is a feature. There are different kinds of edges between a parent node (feature) and its children (sub-features), namely, the ones representing the *mandatory* sub-features, and the others representing the *optional* sub-features. In addition, a feature diagram can specify extra constraints on features; namely, the *alternative* relationship, the *exclude* relationship, and the *require* relationship.

A feature diagram only specifies the structural aspects of variability in an SPL; however, to formally analyze the behavior of an SPL, we follow the approach of [2] in annotating the transitions of a labeled transition system with logical constraints on the presence or absence of features; the features used in such logical constraints are assumed to be already specified in a feature diagram. We slightly differ from [2] by distinguishing the alphabets of the labeled transition systems into two disjoint sets of inputs and outputs. This is a necessary ingredient for extending the theories of testing, and particularly IOCO, to this setting. In short, a *input-output feature transition system* (IOFTS) consists of a *input-output labeled transition system*, a *feature diagram*, a *feature annotation* function, and a set of *product configurations* representing the set of valid products induced by a feature diagram.

Subsequently, we define a family of product derivation operators (parameterized by feature constraints), which project the behavior of an IOFTS into another IOFTS representing a selection of products (a product sub-line). Using such representations as test models different products of an SPL can be analyzed simultaneously.

3 Results

Given a specification modeled as an IOFTS and assuming that an implementation under test can be expressed as an (unkown) IOFTS (similar to the testing assumption of [11]), it is possible to define an Input-Output Conformance (IOCO) relation between the two. Intuitively, the defined IOCO relation asserts that the experiments derived from a specification and executed on the implementation under test, results in outputs that are always allowed by the specification. This corresponds exactly to the extensional definition of IOCO on labelled transition systems [11].

To complement the intensional definition, we give an operational definition of test suites and test cases, which can also be expressed as IOFTSs derived from a given specification. Moreover, we define a notion of refinement that projects a test-suite into the part that satisfies a certain feature constraint. This allows us to generate a test suite for a product line and refine it into test suites for more specific sub-lines (and eventually generating test cases for a specific product). Furthermore, we show that the two notions of refinement (one at the specification level and the other at the test suite level) are consistent. In particular, we showed that by refining a test suite of a specification, we obtain a test suite that is isomorphic to the test suite generated from the refined specification (assuming that both refinements use the same feature constraint).

4 Open issues

In future, we would like to research the following open issues.

1. *Factoring test suites.* The main goal of this research line is to provide a theory of SPL testing that allows for testing common features among different products once and for all. As a first step to this end, we intend to define an operator that given two models (or two test suites) with different feature constraints, returns a test suite, which represents the common features of the two models and two test suits that cover the specific features of each of the two models.

2. *Incremental testing.* Our refinement operators (both at the level of specification and test suite) are top-down in nature, i.e., these operators refine the behavior of an abstract specification (test suite) by strengthening the associated feature constraint. Conversely, it is also possible to perform testing in a bottom-up manner, where the behavior of concrete products are combined to validate an SPL and the test suite of an SPL is generated compositionally.

3. *Empirical research.* Lastly, we would like to implement our theoretical framework and perform empirical research on its effectiveness and efficiency.

References

[1] H. Beohar and M.R. Mousavi. Input-output conformance testing based on featured transition systems. Submitted for publication, August 2013. Available from: `http://ceres.hh.se/mediawiki/images/0/0c/Fioco_2013.pdf`.

[2] A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay, and J.-F. Raskin. Featured transition systems: Foundations for verifying variability-intensive systems and their application to LTL model checking (to appear). *IEEE Trans Software Eng (TSE)*, 2012.

[3] P.A. da Mota Silveira Neto, I. do Carmo Machado, J.D. McGregor, E.S. de Almeida, and S.R. de Lemos Meira. A systematic mapping study of software product lines testing. *Inf. Softw. Technol.*, 53(5):407–423, 2011.

[4] E. Engström and P. Runeson. Software product line testing - a systematic mapping study. *Information & Software Technology*, 53(1):2–13, 2011.

[5] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, 1990.

[6] B.P. Lamancha, M.P. Usaola, and M.P. Velthius. Systematic review on software product line testing. In *Software and Data Technologies*, volume 170 of *Comm. in Computer and Information Science*, pages 58–71. Springer, 2013.

[7] S. Oster, A. Wübbeke, G. Engels, and A. Schürr. Model-based software product lines testing survey. In *Model-based Testing for Embedded Systems*, pages 339–381. CRC Press, 2011.

[8] G.D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.

[9] P.-Y. Schobbens, P. Heymans, and J.-C. Trigaux. Feature diagrams: A survey and a formal semantics. In *Proc. of the 14th IEEE International Conference on Requirements Engineering*, pages 136–145. IEEE, 2006.

[10] W. Xu and D. Xu. A Model-Based Approach to Test Generation for Aspect-Oriented Programs. In *Proc. of the 1st Workshop on Testing Aspect-Oriented Programs*, ACM. Chicago, 2005.

[11] J. Tretmans. Model based testing with labelled transition systems. In *Formal Methods and Testing*, volume 4949 of *LNCS*, pages 1–38. Springer, 2008.