On the Expressiveness of MTS with Variability Constraints

Journal First

Maurice H. ter Beek[1], Ferruccio Damiani[2], Stefania Gnesi[1], Franco Mazzanti[1], and Luca Paolini[2]

[1] ISTI–CNR, Pisa, Italy
{maurice.terbeek, stefania.gnesi, franco.mazzanti}@isti.cnr.it
[2] University of Turin, Turin, Italy
{ferruccio.damiani, luca.paolini}@unito.it

Abstract. Modal transition systems and featured transition systems are widely recognised as fundamental behavioural models for software product lines. Modal transition systems with variability constraints are equally expressive as featured transition systems. This is proved by providing transformation of the latter into the former, and a transformation of the former into the latter which are both sound and complete. First, our results contribute to the expressiveness hierarchy of such basic models studied in many papers. Second, it provides an automatic algorithm from FTS to MTS that preserves the original (compact) branching structure, thus paving the way for using the model checking of FTSs with the variability model checker VMC.

Keywords: Behavioural model · Formal specification · Featured transition system · Modal transition system.

1 Background

Software systems are more and more often developed and managed as software product lines (SPLs) to tackle the variability inherent to a collection of individual customization [27]. The variability among the instances of highly-configurable, variant-rich systems is expressed in terms of features, which conceptualise pieces of functionality or aspects of a system that are relevant to the stakeholders [1]. Formal models for the specification and verification of SPL behaviour have been the subject of extensive research throughout the last decade [21, 23, 19, 22, 24, 14, 2, 18, 13, 28, 25, 26, 4, 7].

Behavioral models for SPL are based on the superimposition of multiple Labeled Transition Systems (LTS), each of which represents a different variant (a product model), in a single LTS enriched with feature-based variability (a family model). A family's products (ordinary LTS) can be derived from the enriched LTS by resolving this variability. This boils down to deciding which 'variable' behavior to include in a specific product and which not, based on the combination of features defining the product.

In [11] three of the most fundamental behavioural models for SPLs were compared with respect to their expressive power: MTSs, FTSs and so-called PL-LTSs. @ALL: introdurre FTSs and MTSs They are the two models studied in [9]. While PL-LTSs form the semantic model of the product line process algebra PL-CCS introduced in [22]. PL-CCS extends Milner's calculus of communicating systems with a variants operator XOR enabling the modelling of alternative behaviour. The expressiveness results in [11] state that MTSs are less expressive than PL-LTSs, which in turn are less expressive than FTSs.

In a very recent corrigendum to [11], contained in [29], the authors of [11] reported that their definition of PL-LTSs is more restrictive than the one originally introduced in [22], upon which they have proved that adopting the original and more liberal definition, PL-LTSs are equally expressive as FTSs.

It is important to note that the results in [11, 29] are based on LTS-based SPL models with a possibly infinite number of states. Moreover, in [11] products derived from an FTS do not need to preserve the FTS' branching structure (viz. a product may contain more states than the FTS it is derived from) and they may be infinite in number.

2 Contributions of [9]

In [3], we informally presented an automatic technique to transform an FTS into an MTSυ, but we merely sketched a proof of the soundness of this model transformation. Subsequently, in [9], we contributed to the expressiveness hierarchy of fundamental behavioral models for SPL studied in [11], by proving finite-state MTSυ to be equally expressive as finite-state FTS. Formally:

- We prove that MTSυ are at least as expressive as FTS by defining an algorithm that transforms any FTS into an MTSυ and by proving its soundness and completeness (i.e. an MTSυ results with the same set of variant LTS as the original FTS).
- We prove that MTSυ are even equally expressive as FTS by defining an algorithm that transforms any MTSυ into an FTS and by proving its soundness and completeness (i.e. an FTS results with the same set of variant LTS as the original MTSυ).

Our paper complements the expressiveness hierarchy given in [11] with an expressiveness result for *finite-state* behavioural SPL models.

Since the transformation algorithm from FTS to MTSυ preserves the original (compact) branching structure, we thus pave the way for using an (optimized) algorithm to achieve family-based SPL model checking of FTS with the Variability Model Checker VMC [10, 5]. VMC is a tool for modeling and analyzing behavioral SPL models, which currently accepts only MTSυ defined as MTS (specified in a high-level modal process algebra) together with a set of variability constraints (specified as propositional logic formulae).

3 Conclusion and future works

In [9], we proved that finite-state MTSυs are equally expressive as finite-state FTSs. This result complements the expressiveness results that were reported in [11, 29] for behavioural SPL formalisms with possibly infinite states, viz. MTSs are less expressive than FTSs (with a generalised product-derivation relation), which are equally expressive as PL-LTSs.

In the future, we plan to implement such an optimized model transformation as a front-end of VMC, which would allow VMC to offer SPL model checking of temporal logic properties over either FTS or MTSυ. VMC is the most recent member of the KandISTI product line of model checkers developed at ISTI–CNR over the past decades, including UMC [8] and CMC [20]. KandISTI's model checkers offer explicit-state on-the-fly model checking of functional properties expressed in specific action- and state-based branching-time temporal logics derived from ACTL [16], the action-based version of CTL [12]. Their common model-checking engine has been highly optimised, due to which millions of states can now be verified in minutes.

Currently, efficient SPL model checking over FTSs can be done by using dedicated family-based model checkers like ProVeLines [15] or, alternatively, by using highly optimized off-the-shelf model checkers like SPIN or mCRL2, which have recently been made amenable to family-based SPL model checking over FTS [17, 6].

References

1. Apel, S., Batory, D.S., Kästner, C., Saake, G.: Feature-Oriented Software Product Lines: Concepts and Implementation. Springer (2013). https://doi.org/10.1007/978-3-642-37521-7
2. Asirelli, P., ter Beek, M.H., Fantechi, A., Gnesi, S.: Formal Description of Variability in Product Families. In: Proceedings of the 15th International Software Product Lines Conference (SPLC'11). pp. 130–139. IEEE (2011)
3. ter Beek, M., Damiani, F., Gnesi, S., Mazzanti, F., Paolini, L.: From Featured Transition Systems to Modal Transition Systems with Variability Constraints. In: SEFM. LNCS, vol. 9276, pp. 344–359. Springer (2015)
4. ter Beek, M., Fantechi, A., Gnesi, S., Mazzanti, F.: Modelling and analysing variability in product families: Model checking of modal transition systems with variability constraints. J. Log. Algebr. Meth. Program. **85**(2), 287–315 (2016)
5. ter Beek, M., Mazzanti, F.: VMC: Recent Advances and Challenges Ahead. In: SPLC. pp. 70–77. ACM (2014)
6. ter Beek, M., de Vink, E., Willemse, T.: Family-Based Model Checking with mCRL2. In: FASE. LNCS, vol. 10202, pp. 387–405. Springer (2017)
7. ter Beek, M.H., Legay, A., Lluch Lafuente, A., Vandin, A.: A framework for quantitative modeling and analysis of highly (re)configurable systems. IEEE Transactions on Software Engineering (2018)
8. ter Beek, M.H., Fantechi, A., Gnesi, S., Mazzanti, F.: A state/event-based model-checking approach for the analysis of abstract system properties. Sci. Comput. Program. **76**(2), 119–135 (2011)

9. ter Beek, M., Damiani, F., Gnesi, S., Mazzanti, F., Paolini, L.: On the expressiveness of modal transition systems with variability constraints. Sci. Comput. Program. **169**, 1–17 (2019). https://doi.org/10.1016/j.scico.2018.09.006
10. ter Beek, M., Mazzanti, F., Sulova, A.: VMC: A Tool for Product Variability Analysis. In: FM. LNCS, vol. 7436, pp. 450–454. Springer (2012)
11. Beohar, H., Varshosaz, M., Mousavi, M.: Basic behavioral models for software product lines: Expressiveness and testing pre-orders. Sci. Comput. Program. **123**, 42–60 (2016)
12. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. ACM Trans. Program. Lang. Sys. **8**(2), 244–263 (1986)
13. Classen, A., Cordy, M., Schobbens, P., Heymans, P., Legay, A., Raskin, J.: Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking. IEEE Trans. Softw. Eng. **39**(8), 1069–1089 (2013)
14. Classen, A., Heymans, P., Schobbens, P., Legay, A., Raskin, J.: Model Checking Lots of Systems: Efficient Verification of Temporal Properties in Software Product Lines. In: ICSE. pp. 335–344. ACM (2010)
15. Cordy, M., Classen, A., Heymans, P., Schobbens, P., Legay, A.: ProVeLines: A Product Line of Verifiers for Software Product Lines. In: SPLC. pp. 141–146. ACM (2013)
16. De Nicola, R., Vaandrager, F.W.: Action versus State based Logics for Transition Systems. In: Guessarian, I. (ed.) Semantics of Systems of Concurrent Processes. LNCS, vol. 469, pp. 407–419. Springer (1990)
17. Dimovski, A., Al-Sibahi, A.S., Brabrand, C., Wasowski, A.: Family-Based Model Checking Without a Family-Based Model Checker. In: SPIN. LNCS, vol. 9232, pp. 282–299. Springer (2015)
18. Erwig, M., Walkingshaw, E.: The Choice Calculus: A Representation for Software Variation. ACM Trans. Softw. Eng. Methodol. **21**(1), 6:1–6:27 (2011)
19. Fantechi, A., Gnesi, S.: A behavioural model for product families. In: Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE'07). pp. 521–524. ACM (2007). https://doi.org/10.1145/1287624.1287700
20. Fantechi, A., Gnesi, S., Lapadula, A., Mazzanti, F., Pugliese, R., Tiezzi, F.: A Logical Verification Methodology for Service-Oriented Computing. ACM Trans. Softw. Eng. Methodol. **21**(3), 16 (2012)
21. Fischbein, D., Uchitel, S., Braberman, V.A.: A Foundation for Behavioural Conformance in Software Product Line Architectures. In: Hierons, R.M., Muccini, H. (eds.) Proceedings of the ISSTA Workshop on Role of Software Architecture for Testing and Analysis (ROSATEA'06). pp. 39–48. ACM (2006)
22. Gruler, A., Leucker, M., Scheidemann, K.D.: Modeling and Model Checking Software Product Lines. In: Barthe, G., de Boer, F.S. (eds.) Proceedings of the 10th International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS'08). LNCS, vol. 5051, pp. 113–131. Springer (2008)
23. Larsen, K.G., Nyman, U., Wasowski, A.: Modal I/O Automata for Interface and Product Line Theories. In: De Nicola, R. (ed.) Proceedings of the 16th European Symposium on Programming (ESOP'07). LNCS, vol. 4421, pp. 64–79. Springer (2007)

24. Lauenroth, K., Pohl, K., Töhning, S.: Model Checking of Domain Artifacts in Product Line Engineering. In: Proceedings of the 24th International Conference on Automated Software Engineering (ASE'09). pp. 269–280. IEEE (2009)
25. Lochau, M., Mennicke, S., Baller, H., Ribbeck, L.: DeltaCCS: A Core Calculus for Behavioral Change. In: Margaria, T., Steffen, B. (eds.) Proceedings of the 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA'14). LNCS, vol. 8802, pp. 320–335. Springer (2014)
26. Muschevici, R., Proença, J., Clarke, D.: Feature Nets: behavioural modelling of software product lines. Softw. Sys. Model. **15**(4), 1181–1206 (2016)
27. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer (2005)
28. Tribastone, M.: Behavioral Relations in a Process Algebra for Variants. In: Proceedings of the 18th International Software Product Line Conference (SPLC'14). pp. 82–91. ACM (2014)
29. Varshosaz, M.: Test Models and Algorithms for Model-Based Testing of Software Product Lines, Licentiate thesis. Halmstad University Dissertations, vol. 30. Halmstad University Press (2017)