Summary of: On the expressiveness of modal transition systems with variability constraints

Maurice H. ter Beek$^{1[0000-0002-2930-6367]}$, Ferruccio Damiani$^{2[0000-0001-8109-1706]}$, Stefania Gnesi$^{1[0000-0002-0139-0421]}$, Franco Mazzanti$^{1[0000-0003-4562-8777]}$, and Luca Paolini$^{2[0000-0002-4126-0170]}$

[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 (MTSs) and featured transition systems (FTSs) are widely recognised as fundamental behavioural models for software product lines. This short paper summarises the contributions published in [3]: MTSs with variability constraints (MTSυs) are equally expressive as FTSs. This is proved by giving sound and complete transformations of the latter into the former, and of the former into the latter. The benefits of this result are twofold. First, it contributes to the expressiveness hierarchy of such basic models studied in the literature. Second, it provides an automatic algorithm from FTSs to MTSυs that preserves the original (compact) branching structure, thus paving the way for model checking FTSs with the variability model checker VMC.

Keywords: SPL · Variability · 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 [15]. 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.[3]

Behavioural models for SPLs are based on the superimposition of multiple labelled transition systems (LTSs), 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 product variant (ordinary LTS) can be derived from the enriched LTS by resolving this variability. This boils down to deciding which 'variable' behaviour to include in a specific product and which not, based on the combination of features defining the product.

[3] The reader is referred to [3] for a complete list of references

MTSs and FTSs are some of the best known enriched LTS models for SPLs.

An MTS is an LTS that distinguishes between admissible (may) and necessary (must) transitions. MTSs were originally introduced in [14] to capture the refinement of a partial description into a more detailed one, reflecting increased knowledge on the admissible (but not necessary) behaviour. In [5], MTSs were equipped with an additional set of variability constraints, resulting in MTSυs, to compactly model SPL behaviour, whose individual product behaviour, in the form of LTSs, can be obtained by means of a special-purpose refinement relation, or by an equivalent operational derivation procedure.

An FTS is an LTS equipped with a function that labels each transition with a feature expression which needs to hold for this specific transition to be part of executable (product) variant behaviour (according to some feature model). FTSs were introduced in [11, 10] to concisely model SPL behaviour, where again the behaviour of its (product) variants is modelled by LTSs. An FTS captures a family of LTSs, one per (product) variant, which can be obtained by projection (pruning away transitions not belonging to the variant).

In [9, 16, 17], these and other behavioural SPL models (with possibly infinite states) were compared with respect to their expressive power. The expressiveness results in [9] state that MTSs are less expressive than FTSs (with a generalised product-derivation relation), whereas in [17] FTSs are encoded into equivalent sets of multiple MTSs. In [3], we demonstrated instead that finite-state MTSυs are equally expressive as finite-state FTSs, by defining transformations of the latter into the former, of the former into the latter, and proving the soundness and completeness of both transformations, thus contributing to the expressiveness hierarchy of such basic behavioural SPL models studied in [9, 16, 17].

2 Contributions

In this section, we summarise the main results of [3] and mention their benefits.

We say that a behavioural SPL formalism M' is *at least as expressive as* a behavioural SPL formalism M, denoted by $M \leq M'$, iff there exists a transformation from M into M', denoted by $\tau\colon M \to M'$, such that for all models $\mathcal{M} \in M$, the sets of derived variants $\mathsf{lts}(\mathcal{M})$ and $\mathsf{lts}(\tau(\mathcal{M}))$ coincide. If also the vice-versa holds, we say that M' and M are *equally expressive*, denoted by $M' = M$.

As anticipated, in [3] we formally proved that finite-state MTSυs are equally expressive as finite-state FTSs. To this aim, we extended the automatic technique to transform an FTS into an MTSυ presented in [2], where we moreover merely sketched a proof of the soundness. More precisely, to show that finite-state MTSυs are equally expressive as finite-state FTSs we proved that:

1. FTSs $\leq$ MTSυs by defining an algorithm, FTS2MTSυ 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 LTSs as the original FTS).
2. MTSυs = FTSs by defining an algorithm, MTSυ2FTS, 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 LTSs as the original MTSυ).

Furthermore, in [3] we also pinpoint the specific features of MTSυs that make them at least as expressive as FTSs by illustrating the transformation into an MTSυ of an FTS that was introduced in [9] and for which it was demonstrated that it cannot be encoded as an MTS.

The defined FTS2MTSυ transformation algorithm leads to an MTSυ that generally could be optimised in several ways without changing its behaviour nor its variants, such as reducing the so-called descriptional complexity of the MTSυ (like the number of variability constraints) or improving the efficiency of model checking properties over the MTSυ or deriving variants from it. Notably, the FTS2MTSυ transformation algorithm preserves the original (compact) branching structure of the FTS.[4] This paves the way for using (optimised) versions of the MTSυs for family-based SPL model checking of FTSs with the variability model checker VMC [7, 6], which currently accepts only MTSυ. VMC is a tool for modeling and analysing behavioural SPL models, which accepts MTSυs 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 Work

In [3], we proved that finite-state MTSυs are equally expressive as finite-state FTSs by defining an algorithm to transform any MTSυ into an FTS and vice-versa and proving the soundness and completeness of these transformations. This result complements the expressiveness results that were reported in [9, 16] for behavioural SPL formalisms with possibly infinite states, viz. MTSs are less expressive than FTSs (with a generalised product-derivation relation).

Since one of the aims of [3] was to complement the expressiveness hierarchy of fundamental behavioural models for SPLs studied in [9, 16], the theoretical result of [3] that MTSυs are equally expressive as FTSs is of interest by itself. However, since the FTS2MTSυ transformation algorithm preserves the original (compact) branching structure of FTSs, this result moreover paves the way for using (optimised) versions of the resulting MTSυs for family-based SPL model checking of FTSs with the variability model checker VMC [7, 6].

A related approach, outlined in [4], uses the fact that any FTS can be transformed into an MTS and if the FTS is unambiguous, then the corresponding MTS is live, thus allowing to carry over a result from [5] to facilitate family-based model checking of such FTSs for a rich temporal logic.

In the future, we plan to implement an optimised version of the FTS2MTSυ transformation algorithm (e.g. by creating must transitions whenever possible) as a front-end of VMC and to further explore the approach from [4], to offer SPL model checking of rich temporal logic properties against either FTSs or MTSυs.

Currently, efficient SPL model checking over FTSs is possible by using dedicated family-based model checkers like ProVeLines [12] or, alternatively, by using highly optimised off-the-shelf model checkers like SPIN or mCRL2, which were recently made amenable to family-based SPL model checking over FTSs [13, 8].

[4] The resulting MTSυ has one additional state and dummy transitions to that state [3].

References

1. Apel, S., Batory, D.S., Kästner, C., Saake, G.: Feature-Oriented Software Product Lines. Springer (2013). https://doi.org/10.1007/978-3-642-37521-7
2. ter Beek, M.H., 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). https://doi.org/10.1007/978-3-319-22969-0_24
3. ter Beek, M.H., 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
4. ter Beek, M.H., Damiani, F., Lienhardt, M., Mazzanti, F., Paolini, L.: Static Analysis of Featured Transition Systems. In: SPLC. pp. 39–51. ACM (2019). https://doi.org/10.1145/3336294.3336295
5. ter Beek, M.H., 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). https://doi.org/10.1016/j.jlamp.2015.11.006
6. ter Beek, M.H., Mazzanti, F.: VMC: Recent Advances and Challenges Ahead. In: SPLC. pp. 70–77. ACM (2014). https://doi.org/10.1145/2647908.2655969
7. ter Beek, M.H., Mazzanti, F., Sulova, A.: VMC: A Tool for Product Variability Analysis. In: FM. LNCS, vol. 7436, pp. 450–454. Springer (2012). https://doi.org/10.1007/978-3-642-32759-9_36
8. ter Beek, M.H., de Vink, E.P., Willemse, T.A.C.: Family-Based Model Checking with mCRL2. In: FASE. LNCS, vol. 10202, pp. 387–405. Springer (2017). https://doi.org/10.1007/978-3-662-54494-5_23
9. 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). https://doi.org/10.1016/j.scico.2015.06.005
10. 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). https://doi.org/10.1109/TSE.2012.86
11. Classen, A., Heymans, P., Schobbens, P., Legay, A., Raskin, J.: Model Checking Lots of Systems. In: ICSE. pp. 335–344. ACM (2010). https://doi.org/10.1145/1806799.1806850
12. 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). https://doi.org/10.1145/2499777.2499781
13. 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). https://doi.org/10.1007/978-3-319-23404-5_18
14. Larsen, K., Thomsen, B.: A Modal Process Logic. In: LICS. pp. 203–210. IEEE (1988). https://doi.org/10.1109/LICS.1988.5119
15. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering. Springer (2005). https://doi.org/10.1007/3-540-28901-1
16. Varshosaz, M., Beohar, H., Mousavi, M.: Basic behavioral models for software product lines: Revisited. Sci. Comput. Program. **168**, 171–185 (2018). https://doi.org/10.1016/j.scico.2018.09.001
17. Varshosaz, M., Luthmann, L., Mohr, P., Lochau, M., Mousavi, M.: Modal transition system encoding of featured transition systems. J. Log. Algebr. Meth. Program. **106**, 1–28 (2019). https://doi.org/10.1016/j.jlamp.2019.03.003