Science of Computer Programming 169 (2019) 1–17

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

On the expressiveness of modal transition systems with variability constraints

Maurice H. ter Beek [a,*], Ferruccio Damiani [b], Stefania Gnesi [a], Franco Mazzanti [a], Luca Paolini [b]

[a] ISTI–CNR, Pisa, Italy
[b] Università di Torino, Italy

ARTICLE INFO

Article history:
Received 7 December 2017
Received in revised form 20 September 2018
Accepted 24 September 2018
Available online xxxx

Keywords:
Software product lines
Formal specification
Behavioural specification
Modal transition systems
Featured transition systems

ABSTRACT

We demonstrate that modal transition systems with variability constraints are equally expressive as featured transition systems, by defining a transformation of the latter into the former, a transformation of the former into the latter, and proving the soundness and completeness of both transformations. Modal transition systems and featured transition systems are widely recognised as fundamental behavioural models for software product lines and our results thus contribute to the expressiveness hierarchy of such basic models studied in other papers published in this journal.

1. Introduction

Modern software systems are often developed and managed as software product lines (SPLs) to allow for mass customisation of many individual product variants [1]. The variability among the instances of such 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 [2]. Foundational formal models for the specification and verification of SPL behaviour have been the subject of extensive research throughout the last decade [3–16]. Most fundamental 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 one single LTS enriched with feature-based variability (a family model). Consequently, a family's products, i.e. ordinary LTSs, can be derived from the enriched LTS by *resolving* this variability. This boils down to deciding which 'variable' (i.e. optional) behaviour to include in a specific product and which not, based on the combination of features defining the product.

In [17], some of the most fundamental behavioural models for SPLs were compared with respect to their expressiveness, which was defined as the set of (product) variants (modelled as LTSs) that can be derived from these models according to some (product derivation) refinement relation. In particular, it was demonstrated that modal transition systems (MTSs) are less expressive than featured transition systems (FTSs). Furthermore, an FTS was provided for which it was demonstrated that it cannot be encoded as an MTS.

* Corresponding author at: Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Via G. Moruzzi 1, 56124 Pisa, Italy.
E-mail addresses: maurice.terbeek@isti.cnr.it (M.H. ter Beek), ferruccio.damiani@unito.it (F. Damiani), stefania.gnesi@isti.cnr.it (S. Gnesi), franco.mazzanti@isti.cnr.it (F. Mazzanti), luca.paolini@unito.it (L. Paolini).

https://doi.org/10.1016/j.scico.2018.09.006
0167-6423/

In [18], we informally presented an automatic technique to transform an FTS into an MTS with variability constraints (MTSυ), which is an extension of MTSs introduced in [15], and we sketched a proof of the soundness of this model transformation (cf. Theorem 1 in [18]). In this paper, we contribute to the expressiveness hierarchy of fundamental behavioural models for SPLs studied in [17], by proving that finite-state MTSυs are equally expressive as finite-state FTSs:

- We first prove that MTSυs are at least as expressive as FTSs by defining an algorithm that transforms any FTS into an MTSυ and proving its soundness and completeness (i.e. an MTSυ results with the same set of variant LTSs as the original FTS)—we thus formalise and improve the procedure sketched informally in [18]. Moreover, to illustrate our result, we transform both the aforementioned FTS from [17], reproduced in Example 25, and a more elaborate SPL example from [11], into MTSυs.
- Next, we prove that MTSυs are equally expressive as FTSs by defining an algorithm that transforms any MTSυ into an FTS and proving its soundness and completeness (i.e. an FTS results with the same set of variant LTSs as the original MTSυ). We illustrate this by an example.

Moreover, the transformation algorithm from FTS to MTSυ preserves the original (compact) branching structure, thus paving the way for using an (optimised) version for family-based SPL model checking of FTSs with the variability model checker VMC [19,20], which currently accepts only MTSυ.

The outline of the paper is as follows. In Section 2, we define LTSs and a few standard notions used in the sequel, after which we define FTSs and MTSυs in Sections 3 and 4, respectively. Our main contributions are presented next: in Section 5, we present an algorithm to transform any FTS into an MTSυ with a proof of soundness and completeness, followed in Section 6 by an algorithm to transform any MTSυ into an FTS together with its soundness and completeness proof. In Section 7, we embed our results in the literature, after which Section 8 concludes the paper and mentions possible future work.

2. Labelled transition systems

We start by introducing LTSs which are the common underlying (semantic) structure for FTSs and MTSs.

Definition 1 *(Labelled transition system).* A *labelled transition system* is a quadruple $(Q, \Sigma, \bar{q}, \delta)$, where Q is a finite (non-empty) set of states, Σ is a set of actions, $\bar{q} \in Q$ is an initial state, and $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation. We call $(q, a, q') \in \delta$ an a-transition (from source q to target q') and we may also write it as $q \xrightarrow{a} q'$.

We formalise two standard notions concerning LTSs in the next definition.

Definition 2 *(Path, reachable).* Let $\mathcal{L} = (Q, \Sigma, \bar{q}, \delta)$ be an LTS. Then σ is a *path* of $\mathcal{L}$ if $\sigma = \bar{q}$ (empty path) or $\sigma = q_1 a_1 q_2 a_2 q_3 \cdots$ with $q_1 = \bar{q}$ and $q_i \xrightarrow{a_i} q_{i+1}$ for all $i > 0$ (possibly infinite path); its ith state is denoted by $\sigma(i)$ and its ith action is denoted by $\sigma\{i\}$. A state $q \in Q$ is *reachable* in $\mathcal{L}$ if there exists a path σ such that $\sigma(i) = q$ for some $i > 0$. An action $a \in \Sigma$ is *reachable* in $\mathcal{L}$ if there exists a path σ such that $\sigma\{i\} = a$, for some $i > 0$.

Example 3. In Fig. 1, we depict an LTS with 7 actions (E, x, a, m, p, ℓ, e). Paths start from initial state 1, including infinite path $1E2x3a6m7p8\ell9e1\cdots$ which implies that all states and all actions are reachable.

Since we will be studying the expressiveness of behavioural models, we restrict our attention to LTSs without unreachable states. In particular, when deriving an LTS from an FTS we will drop all the unreachable states and both their ingoing and their outgoing transitions. Note, however, that we will admit LTSs including unreachable actions (i.e. not labelling transitions) as is done, e.g., in [21]. This is because we will study sets of LTSs (i.e. product models) generated from a common set of actions (viz. of the family model).

We define an action relabelling for LTSs, which will be used in the sequel.

Definition 4 *(Action relabelling).* Let $\mathcal{L} = (Q, \Sigma_1, \bar{q}, \delta)$ be an LTS and let $\rho : \Sigma_1 \to \Sigma_2$ be a relabelling function. The ρ-relabelling of $\mathcal{L}$ is the LTS $\rho(\mathcal{L}) = (Q, \Sigma_2, \bar{q}, \{(q, \rho(a), q') \mid (q, a, q') \in \delta\})$.

Relabelling is commonly adopted to reuse a given specification (model) with different action names.

It is worth noting that the above relabelling function is not required to be injective, in accord with similar operators defined in [21–23]. This choice allows us to collapse different actions to the same action (e.g. it is quite usual to collapse different actions on a generic (irrelevant) internal action).