Comparative Expressiveness of Product Line Calculus of Communicating Systems and 1-Selecting Modal Transition Systems

Mahsa Varshosaz[1] and Mohammad Reza Mousavi[2]

[1] Centre for Research on Embedded Systems
Halmstad University, Sweden,
mahsa.varshosaz@hh.se
[2] Department of Informatics,
University of Leicester, UK
mm789@le.ac.uk

Abstract. Product line calculus of communicating systems (PL-CCSs) is a process calculus proposed to model the behavior of software product lines. Modal transition systems (MTSs) are also used to model variability in behavioral models. MTSs are known to be strictly less expressive than PL-CCS. In this paper, we show that the extension of MTSs with hyper transitions by Fecher and Schmidt, called 1-selecting modal transition systems (1MTSs), closes this expressiveness gap. To this end, we propose a novel notion of refinement for 1MTSs that makes them more suitable for specifying variability for software product lines and prove its various essential properties.

Keywords: Product line calculus of communicating systems (PL-CCS), Modal transition system (MTSs), 1-selecting modal transition system (1MTS), Comparative expressiveness

1 Introduction

Variability modeling is a cornerstone of software product line (SPL) engineering, through which an inventory of commonalities and differences among different products are specified in a structured manner. Efficient analysis of variability-intensive systems is a major challenge due to the potentially large number of valid products. To this end, many techniques have been adapted, which exploit variability in different types of analysis. A basic building block of many of these techniques is a model for capturing variability at the structural or behavioral level. In this paper, we focus on formal behavioral models that can be used to capture variability; examples of such models include modal transition systems (MTSs) [18], product line calculus of communicating systems (PL-CCS) [14], and featured transition systems (FTSs) [10].

In a previous paper [9], we studied the comparative expressiveness of these formalisms with respect to the set of products (labeled transition systems (LTSs)) they can specify. There, we proved that MTSs are strictly less expressive than

PL-CCS (and its underlying semantic model, product line labeled transition systems (PL-LTSs)). A formalism that was not studied in our previous paper [9] is 1-Selecting Modal Transition System (1MTS) [12], which extends modal transition systems with (must/may) hyper transitions. Such hyper transitions bundle a number of possible behavior, of which exactly one will be included in each valid product. Using 1MTSs it is possible to model alternative behaviour (choices with XOR relation) in products, which cannot be modeled using MTSs. Intuitively, this seems the very missing modeling feature in order to fill the expressiveness gap between MTSs and PL-LTSs.

In this paper, we show that this extension is indeed sufficient to close the expressiveness gap between MTS and PL-LTS (see Section 5). Furthermore, we observed that by considering the current refinement relation provided for 1MTSs, some aspects of behavioral variability, such as persistent choices in recursive specifications, cannot be modeled satisfactorily (see Section 3). Hence, we propose a new refinement relation for 1MTSs which addresses these concerns (see Section 4) and also leads to more succinct models, and we show that the new refinement relation enjoys the same intuitive properties as the original one [12]. The other direction of comparison (from 1MTSs to PL-LTSs) is left as a future work. However, we conjecture that encoding 1MTSs into PL-LTSs is also possible.

2 Preliminaries

In this section, we explain some basic concepts regarding software product lines, 1-selecting modal transition systems, and product-line labeled transition systems that are used throughout the rest of the paper.

2.1 Software Product Lines

The products in a software product line are developed from a common core. The commonalities and variabilities among products are usually described in terms of features. A feature is a distinctive user-visible aspect or characteristic of the system [15]. The products in a product line can be described as sets of features. There are different types of relations between features in a product line. We explain some of these relations using an example of a vending machine product line. The product line includes three mandatory features, namely, *Coin*, *Drink*, and *Coffee*, which means all the products in this product line should include these three features. There are two types of coin, namely, *Dollar* and *Euro* which have *alternative* relation. This means that a product in this product line can either accept dollar or euro coins but not both. The *Drink* feature has two sub-features as well, namely, *Tea* and *Coffee*. The *Tea* is an optional feature. This means that a product in this product line can offer both tea and coffee or only coffee as drinks (since, *Coffee* is a mandatory feature).

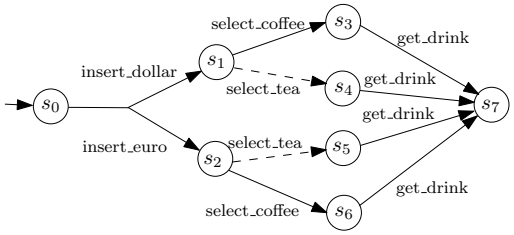


Fig. 1: 1MTS for vending machine product line.

2.2 1-Selecting Modal Transition Systems

Fecher and Schmidt [12] introduced the following definition of 1MTSs.

Definition 1 (1MTS). *A 1-selecting modal transition system, is a tuple* $(\mathbb{S}, A, \rightarrow, \dashrightarrow, s_{init})$*, where:*

- $\mathbb{S}$ *is a set of states or processes,*
- A *is a set of actions,*
- $\rightarrow \subseteq \mathbb{S} \times (2^{A \times \mathbb{S}} \setminus \emptyset)$*, is the must hyper transition relation,*
- $\dashrightarrow \subseteq \mathbb{S} \times (2^{A \times \mathbb{S}} \setminus \emptyset)$ *is the may hyper transition relation,*
- $s_{init} \subseteq \mathbb{S}$*, is a non-empty set of initial states.*

In each 1MTS, the relation $\rightarrow \subseteq \dashrightarrow$ *holds between the sets of may- and must hyper transitions. This means that must hyper transitions also implicitly represent may hyper transitions.*

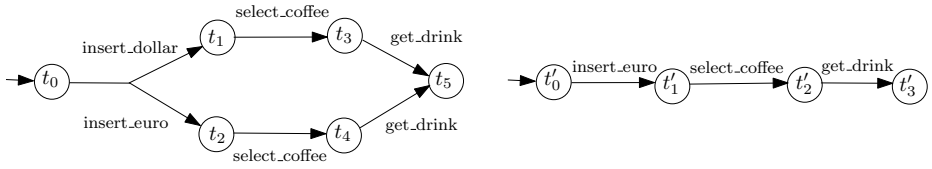
We use $\mathbb{1MTS}$, to denote the class of all 1MTSs.

Based on the above definition, there are two types of hyper transitions in a 1MTS, called may- and must hyper transitions. A may hyper transition represents a set of alternative choices which are optional (at most one of the choices can be selected). On the other hand, a must hyper transition represents a set of alternative choices where selecting one of the choices is obligatory. Furthermore, we assume that for each state s, $(s \dashrightarrow) = \{\gamma \mid (s, \gamma) \in \dashrightarrow\}$. A simple example of a 1MTS is provided in Fig. 1. This 1MTS represents the behavior of products in the vending machine product line.

In order to define how a transition among those in a hyper transition is chosen, the following notion of choice function is used.

Definition 2 (Choice Function). *Let* A *be a set, and* $B \subseteq 2^A$ *and* $\gamma : B \rightarrow A$*. Then* γ *is a choice function if* $\forall_{b \in B} : \gamma(b) \in b$*. The set of all choice functions on* B *is defined by choice*(B)

As 1MTSs are abstract models, one can associate with each 1MTS a set of 1MTSs that refine it by allowing for fewer optional choices. The refinement relation on 1MTSs is defined as follows [12].



(1) (2)

Fig. 2: (1) 1MTS and (2) LTS refining the model in Fig. 1 and Fig. 2(1).

Definition 3 (Refinement for 1MTSs). *A refinement relation between two 1MTSs such as* $M = (\mathbb{S}, L, \rightarrow, \dashrightarrow, s_{init})$ *and* $\bar{M} = (\bar{\mathbb{S}}, L, \bar{\rightarrow}, \bar{\dashrightarrow}, \bar{s}_{init})$*, is defined as a relation* $\mathcal{R}_{1MTS} \subseteq \mathbb{S} \times \bar{\mathbb{S}}$ *such that* $\forall_{s \in s_{init}} \cdot \exists_{\bar{s} \in \bar{s}_{init}} \cdot s \ \mathcal{R}_{1MTS} \ \bar{s}$ *and* $\forall (s, \bar{s}) \in \mathcal{R}_{1MTS} \cdot \ \forall \gamma \in choice(s \dashrightarrow) \cdot \ \exists \bar{\gamma} \in choice(\bar{s} \bar{\dashrightarrow})$*, such that:*

1. $\forall\ \omega \in (s \dashrightarrow) \cdot\ \exists\ \bar{\omega} \in (\bar{s} \bar{\dashrightarrow})\ \cdot\ \ \exists\ a \in L, s' \in \mathbb{S}, \bar{s}' \in \bar{\mathbb{S}}\ \cdot\ \ \gamma(\omega) = (a, s') \wedge \bar{\gamma}(\bar{\omega}) = (a, \bar{s}') \wedge s'\ \mathcal{R}_{1MTS}\ \bar{s}'$,
2. $\forall\ \bar{\omega} \in (\bar{s} \bar{\rightarrow}) \cdot\ \exists\ \omega \in (s \rightarrow) \cdot\ \ \exists\ a \in L, s' \in \mathbb{S}, \bar{s}' \in \bar{\mathbb{S}}\ \cdot\ \ \gamma(\omega) = (a, s') \wedge \bar{\gamma}(\bar{\omega}) = (a, \bar{s}') \wedge s'\ \mathcal{R}_{1MTS}\ \bar{s}'$.

M is said to refine $\bar{M}$, written as $M \triangleright \bar{M}$, when there exists a refinement relation $\mathcal{R}_{1MTS}$ relating each of the initial states of M to one of the initial states of $\bar{M}$.

As a simple example in Fig. 2(1), a 1MTS is shown which refines the 1MTS in Fig. 1. In this 1MTS, the may hyper transitions are not present.

We define the concrete implementations of a 1MTS as labeled transition systems, defined below.

Definition 4 (LTS). *An LTS is a tuple* $(S, A, \rightarrow, s_{init})$*, where S is a set of states, A is a set of actions,* $\rightarrow: S \times A \times S$ *is the transition relation, and* s_{init} *is the initial state. We denote the class of LTSs by* $\mathbb{LTS}$*. (We follow the definition given for LTSs as implementations of 1MTSs with single initial states in [12])*

As a simple example in Fig. 2(2), an LTS is shown which refines the 1MTSs in Fig. 1 and 2(1). In this LTS, the may hyper transitions are not present and the alternative choice among the *insert_ euro* and *insert_ dollar* is resolved by choosing the former.

2.3 Product Line Process Algebras

Milner's Calculus of Communicating Systems (CCS) [20] is extended by Gruler et al. [14] into PL-CCS by introducing a new operator, called *binary variant*, to represent the alternative behavior. The introduced binary variant operator $\oplus_i$ is different from the ordinary alternative composition operator $+$ in CCS in that the binary variant choice is made once and for all. As an example, consider the process terms $s = a.(b.s + c.s)$ and $t = a.(b.t \oplus_1 c.t)$; recursive process s

keeps making choices between b and c in each recursion, while process t makes a choice between b and c in the first recursion after performing a, and the choice is recorded and respected in all the following iterations. This means that process t behaves deterministically after the first iteration with respect to the choice between b and c. To simplify the formal development of the theory, Gruler et. al. assume that in every PL-CCS term, there is at most one appearance of the operator $\oplus_i$ for each and every index i. We use the same assumption throughout the rest of the paper, as well.

The semantics of a PL-CCS term is defined based on PL-LTSs [14], using a structural operational semantics, which is explained informally next. The states of a product line labeled transition system are pairs of ordinary states, i.e., process terms, and *configuration vectors*. The transitions of a PL-LTS are also labeled with configuration vectors. These vectors are of type $\{L, R, ?\}^I$ with I being an index set, L and R, respectively, denoting that the choice has been made in favor of the left- or right-hand-side term and ? denoting that the choice has not been made yet.

Definition 5 (PL-LTS). *Let $\{L, R, ?\}^I$ denote the set of all total functions from an index set I to the set $\{L, R, ?\}$. A* product line labeled transition system *is a 5-tuple $(\mathbb{P} \times \{L, R, ?\}^I, A, I, \rightarrow, p_{init})$ consisting of a set of states $\mathbb{P} \times \{L, R, ?\}^I$, a set of actions A, and a transition relation $\rightarrow \subseteq (\mathbb{P} \times \{L, R, ?\}^I) \times (A \times \{L, R, ?\}^I) \times (\mathbb{P} \times \{L, R, ?\}^I)$, and an initial state $p_{init} \in \mathbb{P} \times \{L, R, ?\}^I$, satisfying the following restrictions:*

1. $\forall_{P,\nu,a,Q,\nu',\nu''}\ (P, \nu) \xrightarrow{a,\nu'} (Q, \nu'') \implies \nu' = \nu''$.
2. $\forall_{P,\nu,a,Q,\nu',i}\ (P, \nu) \xrightarrow{a,\nu'} (Q, \nu') \wedge \nu(i) \neq ? \implies \nu'(i) = \nu(i)$.
3. $\forall_{P_0,\nu_0,a,Q_0,\nu_0',i,P_1,\nu_1,b,Q_1,\nu_1',i}\ (P_0, \nu_0) \xrightarrow{a,\nu_0'} (Q_0, \nu_0') \wedge (P_1, \nu_1) \xrightarrow{b,\nu_1'} (Q_1, \nu_1') \wedge \nu_0(i) = \nu_1(i) = ? \wedge \nu_0'(i) \neq ? \neq \nu_1'(i) \implies (P_0, \nu_0) = (P_1, \nu_1)$.

In Definition 5, the conditions follow from the operational rules given by Gruler et al. [14]. The first condition indicates that the change in the configuration is identically reflected in the label and the target. The second condition indicates that a decision made on a choice is recorded as L or R in the configuration vector and would not change in the future. The third condition reflects that the configuration at index i can be resolved in at most one state; this follows immediately from the uniqueness of indices in PL-CCS terms.

In order to define the valid implementations of a PL-LTS, we start with the following relation between the configuration vectors [9].

Definition 6 (Configuration Ordering). *The ordering relation $\sqsubseteq$ on the set $\{L, R, ?\}$ is defined as $\sqsubseteq = \{(?, ?), (L, L), (R, R), (?, L), (?, R)\}$. We lift this ordering relation to the level of configuration vectors by defining $\nu \sqsubseteq \nu' \iff \forall_{i \in I}\ \nu(i) \sqsubseteq \nu'(i)$, for any $\nu, \nu' \in \{L, R, ?\}^I$.*

Considering the above definition, for each $\nu, \nu' \in \{R, L, ?\}^I$, we say $\nu(i) \bowtie \nu(j) \Leftrightarrow \nu(i) \sqsubseteq \nu(j) \vee \nu(j) \sqsubseteq \nu(i)$, for each $i, j \in I$. We lift this ordering relation

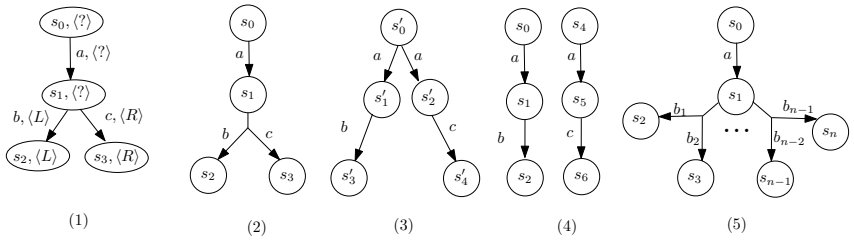


Fig. 3: (1) A PL-LTS example (2) A 1MTS with an LTS implementation (3). (4) A 1MTS modeling the same behavior as the PL-LTS in (1). (5) An example of a 1MTS to demonstrate conciseness problem.

to the level of configuration vectors by defining $\nu \bowtie \nu' \iff \forall_{i \in I}\ \nu(i) \bowtie \nu'(i)$, for any $\nu, \nu' \in \{L, R, ?\}^I$.

In order to compare the expressiveness of 1MTS with PL-LTS, we define product derivation relation for a PL-LTS as follows [9].

Definition 7 (Refinement for PL-LTSs). *Let* $(\mathbb{P} \times \{L, R, ?\}^I, A, \rightarrow, p_{init})$ *be a PL-LTS and let* $(\mathbb{S}, A, \rightarrow, s_{init})$ *be an LTS. A binary relation* $\mathcal{R}_\theta \subseteq \mathbb{S} \times (\mathbb{P} \times \{L, R, ?\}^I)$ *(parameterized by every product configuration* $\theta \in \{L, R\}^I$*) is a* product-derivation relation *if and only if the following transfer properties are satisfied:*

(a) $\forall_{P,Q,a,\nu,\nu',s}\ s\ \mathcal{R}_\theta\ (P,\nu) \wedge (P,\nu) \xrightarrow{a,\nu'} (Q,\nu') \wedge \nu' \sqsubseteq \theta \Rightarrow \exists_t\ s \xrightarrow{a} t \wedge t\ \mathcal{R}_\theta\ (Q,\nu')$,

(b) $\forall_{P,a,\nu,s,t}\ s\ \mathcal{R}_\theta\ (P,\nu) \wedge s \xrightarrow{a} t \Rightarrow \exists_{Q,\nu'}\ (P,\nu) \xrightarrow{a,\nu'} (Q,\nu') \wedge \nu' \sqsubseteq \theta \wedge t\ \mathcal{R}_\theta\ (Q,\nu')$.

A state $s \in \mathbb{S}$ *in an LTS is (the initial state of) a product of a PL-LTS* (P,ν) *with respect to a configuration vector* θ*, denoted by* $(P,\nu) \vdash_\theta s$*, if* $\nu \sqsubseteq \theta$ *and there exists an* $\mathcal{R}_\theta$ *product-derivation relation such that* $s\ \mathcal{R}_\theta\ (P,\nu)$*.*

We say an LTS $T = (\mathbb{S}, A, \rightarrow, s_{init})$ is a valid implementation of a PL-LTS $P = (\mathbb{P} \times \{L, R, ?\}^I, A, I, \rightarrow, p_{init})$, denoted by $T \prec P$ if and only if there exists a configuration $\theta \in \{L, R, ?\}^I$ such that $p_{init} \vdash_\theta s_{init}$.

3 Design Decisions

In this section, we study the refinement relation provided for 1MTSs by Fecher and Schmidt [12] (see Definition 3) and use some examples to point out a few issues in using this notion of refinement for product derivation. These issues lead us to design decisions for a new notion of refinement, introduced in the next section, that is more suitable for the setting of software product lines.

The first example concerns alternative behavior. Consider the PL-CCS terms $s_0 = a.s_1$ and $s_1 = b.s_2 \oplus_1 c.s_3$. The corresponding underlying PL-LTS is represented in Fig. 3(1). Then, consider the 1MTS shown in Fig. 3(2). Intuitively,

this model may be considered as a solution to represent the same set of products using 1MTSs: it bundles the choice between the b- and c-labeled transitions into a must hyper transition. (Recall from Definition 1 that must hyper transitions intuitively represent mandatory choices.) However, in Fig. 3(3), a valid implementation of this 1MTS based on the refinement relation in Definition 3 is depicted. (The dashed arrows show how the states of the LTS and 1MTS are related using the refinement relation.) In the LTS implementation, both the b- and c-labeled transitions are included. A 1MTS that has the same implementations as the PL-LTS in Fig. 3(1), is given in Fig. 3(4); namely, the choice has been lifted to the initial states. This way, the exclusive behavior can be separated among the two parts of the model initiated in these two states.

The process of lifting choices to the initial states can lead to an exponential blow up in 1MTS representation of product lines. This is already hinted at by the 1MTS given in Fig. 3(4) and can be generalized as follows. Consider the 1MTS shown in Fig. 3(5). This model is similar to the 1MTS given in Fig. 3(2) with $k = n/2$ independent exclusive choices (modeled by k must hyper transitions). There are 2^k possible combinations of all choices. This model suffers from the same problem as described above, namely, the alternative transitions can be included simultaneously in some LTS implementations. As mentioned above, in order to model alternative behavior the solution is to use a model with several initial states where each part of the model includes one of the possible combinations. Hence, the model should include 2^k separate parts each with a different initial state. This issue severely compromises succinctness in 1MTS representation of product lines.

Another issue in using 1MTSs for modeling product lines concerns persistent choices. Assume that we add the term $s_3 = d.s_1$ to the aforementioned PL-CCS process term. This will lead to having a new state in the PL-LTS $(s_1, \langle R \rangle)$ and a transition from $(s_3, \langle R \rangle)$ to this state. As mentioned in Section 2.3, the decisions made about the exclusive choices are stored in configuration vectors. Hence, when going back again to s_1, the choice that was made before, which is R, will not change. Using the current notion of refinement for 1MTSs, it is not possible to keep track of the choices that are made in the past. Assume that we want to model the same behavior (as in Fig. 3(1)) using 1MTSs. Assume a transition from state s_3 to state s_1 with label d is added to the 1MTS represented in Fig. 3(2). One of the valid implementations of such 1MTS is an LTS where b is chosen the first time reaching state s_1 and then c is chosen the next time that this state is reached. The solution to solve this problem, is the same as above (using several initial states) in addition to unrolling loops.

To address these 3 issues, namely, alternative behavior, succinct representation of choice, and persistence choice, we introduce a new notion of refinement for 1MTSs in the next section.

4 Revisiting the Refinement Relation

In this section, we propose a new refinement relation for 1MTSs to address the issues pointed out in the previous section regarding the original refinement relation [12]. Then, we show that our new refinement relation preserves the intuitive properties posed for the original one [12].

4.1 New Refinement Relation

We revisit the refinement relation in Definition 3, and provide a new refinement relation for 1MTSs as follows. First, we define an auxiliary function, namely, the choice resolution function.

Definition 8 (Choice Resolution Function). *Consider a 1MTS* $M=(\mathbb{S}, L, \rightarrow, \dashrightarrow, s_{init})$. *A choice resolution function is a total function* $\Gamma : \mathbb{S} \rightarrow \bigcup_{s\in\mathbb{S}} choice(s\dashrightarrow)$. *We denote the set of all choice resolution functions of the 1MTS* M *by* Γ_M.

The purpose of defining the choice resolution function is to assign a choice function to each state of the 1MTS once and for all. Next, we give the refinement relation for 1MTSs as follows.

Definition 9 (New Refinement for 1MTS). *Consider two arbitrary 1MTSs* $M= (\mathbb{S}, L, \rightarrow, \dashrightarrow, s_{init})$ *and* $M'= (\mathbb{S}', L, \rightarrow', \dashrightarrow', s'_{init})$, *we say* M *refines* M', *denoted by* $M \triangleright M'$, *iff there exists a refinement relation* $\mathcal{R}_{1MTS} \subseteq S \times S' \times \Gamma_M \times \Gamma_{M'}$ *such that* $\forall f \in \Gamma_M\ \exists f' \in \Gamma_{M'}\ \forall s_0 \in s_{init}\ \exists s'_0 \in s'_{init} \cdot (s_0, s'_0, f, f') \in \mathcal{R}_{1MTS}$ *and* $\forall (s, s', f, f') \in \mathcal{R}_{1MTS}$, *the following conditions hold:*

(i) $\forall\ \omega \in (s\dashrightarrow) \cdot\ \exists\ \omega' \in (s'\dashrightarrow')\ \cdot\ \exists\ a \in L, s'' \in \mathbb{S}, s''' \in \mathbb{S}'\ \cdot\ f(s)(\omega) = (a, s'') \wedge f'(s')(\omega') = (a, s''') \wedge (s'', s''', f, f') \in \mathcal{R}_{1MTS}$, *and*

(ii) $\forall\ \omega' \in (s' \rightarrow') \cdot\ \exists\ \omega \in (s \rightarrow) \cdot\ \exists\ a \in L, s'' \in \mathbb{S}, s''' \in \mathbb{S}'\ \cdot\ f(s)(\omega) = (a, s'') \wedge f'(s')(\omega') = (a, s''') \wedge (s'', s''', f, f') \in \mathcal{R}_{1MTS}$.

(iii) Additionally, $\forall s_1 \in S, f'' \in \Gamma_{M'} \cdot (s_1, s', f, f'') \in \mathcal{R}_{1MTS} \Rightarrow f' = f''$.

In the rest of the paper, we use $\mathcal{R}^{f,f'}_{1MTS}$ to denote a 1MTS refinement relation that follows the above definition (that uses choice resolution functions f and f'). In Fig. 4(1), an example of a 1MTS is given. Based on the Definition 3, the 1MTS in Fig. 4(2) is refining this 1MTS. However, based on the Definition 9, this is not a valid refinement for the 1MTS in Fig. 4(1). Hence, the problem with modeling alternative behavior that was mentioned in Section 3 is solved in the new definition. Similarly the problems with modeling the conciseness and the persistent behavior are solved.

4.2 Refinement Relation Properties

We prove a set of properties for the new refinement relation as follows. This is the same set of properties proven for the original 1MTS refinement relation by Fecher and Schmidt in [12]. (Due to space limitation, the proofs are omitted and we will include them in an extended version of the paper.) First, we show that the new refinement relation is a preorder.

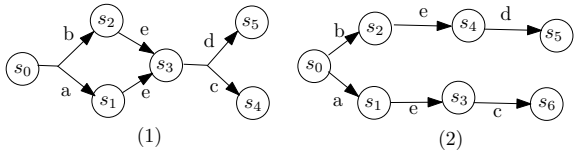


Fig. 4: (1) A 1MTS example. (2) A 1MTS refining (1).

Proposition 1. *The refinement relation given in Definition 9, is a preorder.*

Next, we show that all the LTS implementations of a 1MTS also implement the 1MTSs that are refined by this 1MTS.

Proposition 2. *Consider two 1MTSs M and M' such that $M \triangleright M'$. Then $\forall lts \in \mathbb{LTS} \cdot lts \triangleright M \Rightarrow lts \triangleright M'$.*

Next, we prove that the bisimulation relation satisfies the properties of the refinement relation in Definition 9.

Proposition 3. *Consider two arbitrary LTSs lts_1 and lts_2 such that $lts_1 \sim lts_2$, where $\sim$ denotes strong bisimilarity; it follows that $lts_1 \triangleright lts_2$.*

5 Encoding PL-LTSs into 1MTSs

In order to compare the expressiveness of PL-LTSs with 1MTSs, following the approach provided by Beohar et al. in [9], we define an encoding from PL-LTSs into 1MTSs. The main idea of giving an encoding is to define a transformation from one class of models into the other class of models that is semantic preserving. First, we give the following auxiliary definitions taken from [9].

Definition 10 (Product Line Structure). *A* product line structure *is a tuple $\mathbf{M} = (\mathbb{M}, [\![\ ]\!])$, where $\mathbb{M}$ is the class of the intended product line models (in this paper 1MTSs and PL-LTSs) and $[\![\ ]\!] : \mathbb{M} \to \mathbb{LTS}$ is the semantic function mapping a product formalism to a set of product LTSs that can be derived from each product line model.*

Next, we give the formal definition of an encoding.

Definition 11 (Encoding). *An encoding from a product line structure $\mathbf{M} = (\mathbb{M}, [\![\ ]\!])$ into $\mathbf{M'} = (\mathbb{M}', [\![\ ]\!]')$, is defined as a function $E : \mathbf{M} \to \mathbf{M'}$ satisfying the following correctness criterion: $[\![\ ]\!] = [\![\ ]\!]' \circ E$. We say a product line structure $\mathbf{M'}$ is at least as expressive as $\mathbf{M}$ if and only if there exists an encoding $E : \mathbf{M} \to \mathbf{M'}$.*

Before elaborating on the proposed encoding, we give two auxiliary definitions which are used for encoding the transitions of a PL-LTS into must/may hyper transitions of a 1MTS. As (hyper) transitions in a 1MTS are transitions with multiple targets (see Definition 1), we need to group some of the transitions in a

PL-LTS, which correspond to resolving the same alternative choice, and encode them as a (may/must) hyper transition. To this end, we consider the type of changes that is made by a transition to the configuration vector of a PL-LTS. A transition for which the configuration vectors in the source and target states are not identical, is corresponding to resolving a choice (making a decision about one of the variant operators). We formally define the hyper must closed set and hyper may closed set as follows.

Definition 12 (Hyper Must Closed Set). *Consider a state (P,ν) of a PL-LTS such as $(\mathbb{P}\times\{L,R,?\}^I,A,I,\rightarrow,p_{init})$; we assume that $Out^{(P,\nu)}$ denotes the set of all outgoing transitions from state (P,ν) and $Out_\delta^{(P,\nu)}$ denotes the set of outgoing transitions form (P,ν) that make a change in at least one of the elements of the configuration vector of the source state, i.e., for each $(P,\nu)\xrightarrow{a,\nu'}(P',\nu')\in Out_\delta^{(P,\nu)}$, there exists an $i\in I$ s.t. $\nu(i)=?\wedge\nu'(i)\neq?$. A set $T\subseteq\rightarrow$ of transitions is hyper must-closed for (P,ν) when it is a maximal subset of $Out_\delta^{(P,\nu)}$ such that:*

- *For each $(P,\nu)\xrightarrow{a_0,\nu_0}(Q_0,\nu_0)\in T$, and each $i\in I$ s.t. $\nu(i)\neq\nu_0(i)$ there exists a $(P,\nu)\xrightarrow{a_1,\nu_1}(Q_1,\nu_1)\in T$ s.t. $\neg(\nu_0(i)\bowtie\nu_1(i))$ and for all $j\neq i$, $\nu_0(j)\bowtie\nu_1(j)$.*
- *For each two different transitions $(P,\nu)\xrightarrow{a_0,\nu_0}(Q_0,\nu_0)\in T$ and $(P,\nu)\xrightarrow{a_1,\nu_1}(Q_1,\nu_1)\in T$, exists $i\in I$ s.t. $\neg(\nu_0(i)\bowtie\nu_1(i))$.*

We denote the set of all such maximal subsets for a state (P,ν), by $\mathcal{T}_{\rightarrow}^{(P,\nu)}$.

Definition 13 (Hyper May Closed Set). *The hyper may closed set for a state (P,ν), denoted by $\mathcal{T}_{\dashrightarrow}^{(P,\nu)}$, is defined the same as the hyper must closed set as given in Definition 12, with the only difference that the first condition is replaced with the following condition.*

- *For each $(P,\nu)\xrightarrow{a_0,\nu_0}(Q_0,\nu_0)\in T$, for some $i\in I$ s.t. $\nu(i)\neq\nu_0(i)$ there exists a $(P,\nu)\xrightarrow{a_1,\nu_1}(Q_1,\nu_1)\in T$ s.t. $\neg(\nu_0(i)\bowtie\nu_1(i))$ and for all $j\neq i$, $\nu_0(j)\bowtie\nu_1(j)$.*

Next, we formalise the encoding of a PL-LTS into a 1MTS.

Definition 14 (PL-LTS to 1MTS Encoding). *Let $(\mathbb{P},A,I,\rightarrow,p_{init})$ be a PL-LTS. We construct a 1MTS $M=(\mathbb{S},A,\rightarrow,\dashrightarrow,s_{init})$ as an encoding of such a PL-LTS as follows.*

- *The set $\mathbb{S}$ of states is defined as $\mathbb{P}$, i.e., the set of states in the PL-LTS, $p_{init}=s_{init}$, A is the same set of actions,*
- *We construct the $\rightarrow$ and $\dashrightarrow$, which, respectively, denote the must and may hyper transition relations for each state of the the encoding 1MTSs as follows. Given Definition 12 and Definition 13, we define the following transition rules:*

$$((P,\nu)\rightarrow)=\bigcup_{\Lambda\in\mathcal{T}_{\rightarrow}^{(P,\nu)}_{|\Lambda|>1}}\{\bigcup_{1\leq i\leq|\Lambda|}\{(a_i,(P_i,\nu_i))\}|\ (P,\nu)\xrightarrow{a_i,\nu_i}(P_i,\nu_i)\in\Lambda\}\cup$$

$$\bigcup_{1\leq i\leq |Out^{(P,\nu)}\setminus Out_{\delta}^{(P,\nu)}|} \{\{(a_i,(P_i,\nu))\}|\ (P,\nu)\xrightarrow{a_i,\nu}(P_i,\nu)\in Out^{(P,\nu)}\setminus Out_{\delta}^{(P,\nu)}\}$$

$$((P,\nu)\text{-}\blacktriangleright) = \bigcup_{\Lambda\in\mathcal{T}_{\text{-}\blacktriangleright}^{(P,\nu)}} \{\bigcup_{1\leq i\leq|\Lambda|}\{(a_i,(P_i,\nu_i))\}|\ (P,\nu)\xrightarrow{a_i,\nu_i}(P_i,\nu_i)\in\Lambda\}\cup$$

$$\{(a,(P',\nu'))|\ (P,\nu)\xrightarrow{a,\nu'}(P',\nu')\in\big(Out_{\delta}^{(P,\nu)}\setminus(\mathcal{T}_{\rightarrow}^{(P,\nu)}\cup\mathcal{T}_{\text{-}\blacktriangleright}^{(P,\nu)})\big)\}$$

Given the above encoding, we prove that the class of 1MTSs is at least as expressive as the class of PL-LTSs. (Due to space limitation, the proofs are omitted and we will include them in an extended version of the paper.)

Theorem 1. *The class of 1MTSs is at least as expressive as the class of PL-LTSs.*

6 Related Work

In this section, we discuss related work regarding formalisms used for modeling product lines and the comparison of their expressiveness. We limit our consideration to the models which have LTSs as the semantic domain.

Considering the comparison of the expressiveness of the formalisms used for modeling variability, Beohar et al. in [9] provide a comparison between the expressiveness of three fundamental models, namely, MTSs, PL-CCSs, and Feature Transition Systems (FTSs). (FTSs [11] are extensions of LTSs with propositional formulas called feature expressions.) A novel notion of encoding, based on the set of implementing LTSs, from one class of models to the other is provided. The existence of mutual encodings between two classes of models is described as having the same expressiveness. As a result a hierarchy of formalisms based on their expressiveness is provided. Furthermore, Benduhn et al. in [7], provide a survey on formalisms focusing on the suitability of these models in applying different analysis techniques.

Considering the formalisms proposed for modeling product lines; In [13], Fischbein et al. for the first time argued that MTSs are adequate for modeling variability. In several works, MTSs have been used for modeling variability in the behavior of product lines [2, 1, 3, 19, 16]. As shown in [9], MTSs are the least expressive in the provided hierarchy. In order to tackle the limited expressiveness of MTSs, several extensions of such models have been proposed. In a set of works, MTSs are used with variability constraints [6], which are constraints expressed in Modal-Hennessy-Milner-Logic (MHML) [2, 1, 3]. In [17], an extension of MTSs, namely, Disjunctive Modal Transition Systems (DTMSs) are introduced which provides the possibility to model an *or* relation between choices in the behavior using hyper transitions. Fecher and Schmidt in [12], introduce 1MTSs, which (as mentioned in Section 2) can be used for modeling alternative choices. Furthermore, in this work, a comparison between the expressiveness of these two models is provided, which shows that the two classes of models have

the same expressiveness concerning the sets of implementing LTSs. Benes et al. in [8], introduce an extension of MTSs, namely, parametric modal transition systems in which the concept of obligation functions is used. The obligation functions are defined upon atomic propositions of states, the transitions, and a set of parameters, which can be used for representing features. By setting the valuation of parameters the presence or absence of states and transitions in a specific product model can be specified. Moreover, an extension of contract automata with modality [5] is introduced by Basile et al. in [4]. In this extension of the model, permitted and necessary requests are distinguished using feature constraints. There have been other approaches introduced that use some interface theories principles to indicate the set of derivable variants from an MTS as the ones that are compatible under parallel composition with regards to a given environmental specification [16, 19].

As mentioned in Section 2, PL-CCS [14], introduced by Gruler et al. [14], is an extension of Milner's CCS [20] by means of an alternative choice operator called "binary variant". This operator provides the possibility of modeling persistent choices in the behavior. The validity of variants can be further restricted using the multi-valued modal *mu*-calculus [21].

To the best of our knowledge, the provided encoding from PL-LTSs into 1MTSs, the results regarding the expressiveness, and the provided refinement relation for 1MTSs that addresses the limitations of such models in modeling variability in the behavior in this paper are novel.

7 Conclusion

In this paper, we compared the expressiveness of PL-LTSs and 1MTSs. To this end, we defined the set of products for specifications in both formalisms, of which the behaviors are commonly specified in the domain of LTSs. We then showed that 1MTSs can capture all products that can be specified by the product line calculus of communicating systems. Furthermore, we provided a set of observations regarding the limitations in modeling variability in the behavior which are enforced by the refinement relation given for 1MTSs. We proposed a new refinement relation for 1MTSs to tackle these limitations and proved a set of properties for the new refinement relation.

An immediate question to ask is whether the two formalism have the same expressive power or not. We conjecture that the answer is positive and leave this for immediate future work. We also would like to combine the results of this paper with our earlier results in [9] and present a comprehensive lattice of expressive power among all fundamental behavioral models for software product lines. As another part of our future work, we plan to provide a stronger relation between PL-LTSs and PL-CCS terms by introducing a set of conditions (on the configuration vectors of states) in a PL-LTS which guarantee that the PL-LTS is induced from a PL-CCS term.

References

1. Asirelli, P., ter Beek, M.H., Fantechi, A., Gnesi, S.: A model-checking tool for families of services. In: Proc. on Formal techniques for distributed systems. pp. 44–58. FMOODS'11/FORTE'11, Springer (2011)
2. Asirelli, P., ter Beek, M.H., Gnesi, S., Fantechi, A.: Formal description of variability in product families. In: Proceedings of the 15th International Software Product Line Conference (SPLC '11). pp. 130–139. IEEE (2011)
3. Asirelli, P., ter Beek, M.H., Fantechi, A., Gnesi, S.: A compositional framework to derive product line behavioural descriptions. In: Proceedings of the 5th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change (ISoLA '12), Lecture Notes in Computer Science, vol. 7609, pp. 146–161. Springer (2012)
4. Basile, D., ter Beek, M.H., Di Giandomenico, F., Gnesi, S.: Orchestration of dynamic service product lines with featured modal contract automata. In: Proceedings of the 21st International Systems and Software Product Line Conference - Volume B. pp. 117–122. SPLC '17, ACM, New York, NY, USA (2017), http://doi.acm.org/10.1145/3109729.3109741
5. Basile, D., Di Giandomenico, F., Gnesi, S., Degano, P., Ferrari, G.L.: Specifying variability in service contracts. In: Proceedings of the Eleventh International Workshop on Variability Modelling of Software-intensive Systems. pp. 20–27. VAMOS '17, ACM, New York, NY, USA (2017), http://doi.acm.org/10.1145/3023956.3023965
6. 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. Journal of Logical and Algebraic Methods in Programming 85(2), 287 – 315 (2016)
7. Benduhn, F., Thüm, T., Lochau, M., Leich, T., Saake, G.: A survey on modeling techniques for formal behavioral verification of software product lines. In: Proceedings of the Ninth International Workshop on Variability Modelling of Software-intensive Systems. pp. 80:80–80:87. VaMoS '15, New York, NY, USA (2015)
8. Beneš, N., Křetínský, J., Larsen, K.G., Møller, M.H., Srba, J.: Parametric modal transition systems. In: Bultan, T., Hsiung, P.A. (eds.) Automated Technology for Verification and Analysis. pp. 275–289. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
9. Beohar, H., Varshosaz, M., Mousavi, M.R.: Basic behavioral models for software product lines: Expressiveness and testing pre-orders. Science of Computer Programming (2015), in Press, available online.
10. Classen, A., Cordy, M., Schobbens, P.Y., Heymans, P., Legay, A., Raskin, J.F.: Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking. Software Engineering, IEEE Transactions on 39(8), 1069–1089 (Aug 2013)
11. Classen, A., Heymans, P., Schobbens, P.Y., Legay, A., Raskin, J.F.: Model checking lots of systems: efficient verification of temporal properties in software product lines. In: Proceedings of the 32nd International Conference on Software Engineering (ICSE '10). vol. 1, pp. 335–344. ACM (2010)
12. Fecher, H., Schmidt, H.: Comparing disjunctive modal transition systems with an one-selecting variant. The Journal of Logic and Algebraic Programming 77(1-2), 20–39 (2008)

13. Fischbein, D., Uchitel, S., Braberman, V.: A foundation for behavioural conformance in software product line architectures. In: Proceedings of the ISSTA Workshop on Role of software architecture for testing and analysis. pp. 39–48. ACM (2006)
14. Gruler, A., Leucker, M., Scheidemann, K.: Modeling and model checking software product lines. In: Proceedings of the Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS '08), Lecture Notes in Computer Science, vol. 5051, pp. 113–131. Springer (2008)
15. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Tech. Rep. CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University (1990)
16. Larsen, K.G., Nyman, U., Wąsowski, A.: Modal I/O automata for interface and product line theories. In: Nicola, R.D. (ed.) Proceedings of the 16th European Symposium on Programming Languages and Systems, 16th European Symposium on Programming (ESOP '07), Lecture Notes in Computer Science, vol. 4421, pp. 64–79. Springer (2007)
17. Larsen, K.G., Xinxin, L.: Equation solving using modal transition systems. In: Proc. of the Fifth Annual Symposium on Logic in Computer Science (LICS '90). pp. 108–117. IEEE Computer Society (1990), http://dx.doi.org/10.1109/LICS.1990.113738
18. Larsen, K., Thomsen, B.: A modal process logic. In: Proc. of the 3rd Annual Symposium on Logic in Computer Science (LICS '88). pp. 203–210. IEEE (1988)
19. Lochau, M., Kamischke, J.: Parameterized preorder relations for model-based testing of software product lines. In: Proceedings of the 5th Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change (ISOLA' 12), Lecture Notes in Computer Science, vol. 7609, pp. 223–237. Springer (2012)
20. Milner, R.: A Calculus of Communicating Systems, vol. 92. Springer (1982)
21. Shoham, S., Grumberg, O.: Multi-valued model checking games. J. Comput. Syst. Sci. 78(2), 414–429 (2012)