Available online at www.sciencedirect.com
ScienceDirect
Science of Computer Programming 00 (2018) 1–18

Basic Behavioral Models for Software Product Lines: Revisited

Mahsa Varshosaz[a], Harsh Beohar[b], Mohammad Reza Mousavi[c]

[a] *Center for Research on Embedded Systems (CERES), Halmstad University, Sweden*
[b] *University of Duisburg-Essen, Germany*
[c] *University of Leicester, UK*

Abstract

In "Basic Behavioral Models of Software Product Lines" (Science of Computer Programming, 123(1): 42–60, 2016), we established an expressiveness hierarchy and studied the notions of refinement and testing for three fundamental behavioral models for software product lines. These models were featured transition systems, product line labeled transition systems, and modal transition systems. It turns out that our definition of product line labeled transition systems is more restrictive than the one introduced by Gruler, Leucker, and Scheidemann. Adopting the original and more liberal notion changes the expressiveness results, as we demonstrate in this paper. Namely, we show that the original notion of product line labeled transition systems and featured transition systems are equally expressive. As an additional result, we show that there are featured transition systems for which the size of the corresponding product line labeled transition system, resulting from any sound encoding, is exponentially larger than the size of the original model. Furthermore, we show that each product line labeled transition system can be encoded into a featured transition system, such that the size of featured transition system is linear in terms of the size of the corresponding model. To summarise, featured transition systems are equally expressive as, but exponentially more succinct than, product line labeled transition systems.

Keywords:
Software Product Lines, Behavioral Model, Labeled Transition Systems, Featured Transition Systems, Calculus of Communicating Systems, Product Line Calculus of Communicating Systems, Product Line Labeled Transition Systems.

1. Introduction

Software Product Line (SPL) engineering is a software development technique enabling mass production and customisation. Using this technique, a family of software systems is efficiently developed based on a common core and by benefiting from systematic reuse of artifacts among products.

There are several quality assurance techniques such as model-based testing and model checking that require a model describing the behavior of the system. Hence, several behavioral models have been proposed that can be used for compactly and efficiently representing the behavior of the products in an SPL; examples of such models are Featured Transition Systems (FTSs) [1], Product Line Calculus of Communicating Systems (PL-CCSs) [2], and Modal Transition Systems (MTSs) [3] and different extensions of MTSs [4–7]. These formalisms are comparable in terms of the types of behavior that they can capture and also in terms of their underlying formal model, i.e., Labeled Transition Systems (LTSs).

Email addresses: mahsa.varshosaz@hh.se (Mahsa Varshosaz), harsh.beohar@uni-due.de (Harsh Beohar), mm789@le.ac.uk (Mohammad Reza Mousavi)

FTSs [1] are introduced as an extension of LTSs where the transitions are additionally labeled with feature expressions. Each feature expression is a propositional formula in which the variables represent the features of a product family. Feature expressions indicate the presence / absence of a transition in the model of each product (for more details see Section 2.2). Using FTSs, the behavior of all products is represented in a whole model and different types of analysis can be performed for all products at once using this model.

PL-CCSs [2] are an extension of Milner's Calculus of Communicating Systems (CCSs) [8]. Using PL-CCSs, it is possible to model alternative behavior. The syntax of PL-CCS is an extension of the syntax of CCS with a *variant* operator, which represents an alternative choice between its operands. A choice can be resolved once and for all. This means, in case of recursion, that if a variant choice is resolved in the first iteration, then it remains the same in the future iterations. In [2], Product Line Labeled Transition Systems (PL-LTSs) are defined as the semantic domain for PL-CCS models. In order to keep track of variant choices, a configuration vector is included in the state of PL-LTSs. In each PL-LTS, the size of the vector is equal to the number of variant choices in the corresponding PL-CCS term. The elements of the configuration vector can denote a choice that is either undecided or decided in favor of the left-hand side or right-hand side variant.

In [9], we studied the comparative expressiveness of three of the above formalisms, namely FTSs, PL-CCSs and MTSs, where as a part of the results, we concluded that the class of PL-LTSs is less expressive than the class of FTSs (see Theorem 4 in [9]). In this work, it was assumed that in PL-LTSs in each step, only one of the variant choices can be resolved. Based on this assumption each transition can change only one of the elements of the configuration vector in the target state. This turns out to be an overly restrictive assumption compared to the definition given for the PL-LTS transition rules in [2]. Considering this assumption, it was shown that PL-LTSs cannot capture some types of behavior such as three-way choices which can be captured by FTSs.

In this paper, we relax the above-mentioned restriction and adapt the result to the original and more liberal definition of PL-LTSs [2]. We revisit the comparative expressiveness of FTSs and PL-LTSs with respect to the products that they specify. We describe an encoding of FTSs into PL-LTSs and there by proving that for each FTS, the set of LTSs that implement the FTS are also valid implementations for the PL-LTS resulting from the encoding. The results show that the class of PL-LTSs is at least as expressive as the class of FTSs. We also show that the results provided in [9], specifying that the class of FTSs is at least as expressive as the class of PL-LTSs still holds. Hence, we conclude that the class of PL-LTSs and the class of FTSs are equally expressive. We also provide a comparative succinctness analysis of the size of the PL-LTSs resulting from any sound encoding in terms of the number of states of the corresponding FTS. The results of the succinctness analysis show that FTSs are more succinct formalisms compared to PL-LTSs to describe SPLs.

The rest of this paper is organized as follows. In Section 2, we review the basic definitions regarding FTSs and PL-CCSs. In Section 3, we provide encodings between FTSs and PL-LTSs. In Section 4, we show that the class of PL-LTSs, i.e., underlying semantic model of PL-CCSs, and the class of FTSs are equally expressive. In Section 5, we provide a comparative succinctness analysis for the models resulting from encoding of FTSs. In Section 6, we conclude the paper and present the directions of our ongoing and future work.

2. Preliminaries

In this section, we provide the definition of constructs and concepts that are used throughout the paper.

2.1. Feature Diagram

In SPL engineering, the commonalities and variabilities among products are described using *features*. A *feature* is defined as "a prominent or distinctive user-visible aspect, quality, or characteristic of a software system" [10]. Each product in an SPL is defined by a subset of features selected from the whole set of features of the SPL. There are different relations between the features in an SPL. *Feature models* [11] are a common means to compactly represent the set of products of an SPL in terms of its features.

A feature model is a hierarchical structure consisting of nodes and edges between them. Each node in a feature model represents a feature in the SPL. The structure of a feature model is tree like. Each node can have a set of child nodes. The features in an SPL can be *optional*, or *mandatory*. The mandatory features are present in all products of the SPL, while the optional features may be present in a subset of the products. A group of sibling features (nodes) can

have the *alternative* relation, which means only one of the features in the group can be included in a product in case that the parent feature is selected. Also, a group of sibling features can have the *or* relation, which means one or more features in the group can be included in a product if the parent feature is selected. There are cross tree relations such as *requires* (resp. *excludes*), where the inclusion of a feature results in inclusion (resp. exclusion) of other features. Each feature model can be represented by a propositional logic formula in which propositional variables represent the features in the SPL [12].

Example 1. *An example of a feature model is depicted in Fig. 1. The feature model corresponds to a vending machine product line (the vending machine in this example is a simplified version of the one given in [9]).*

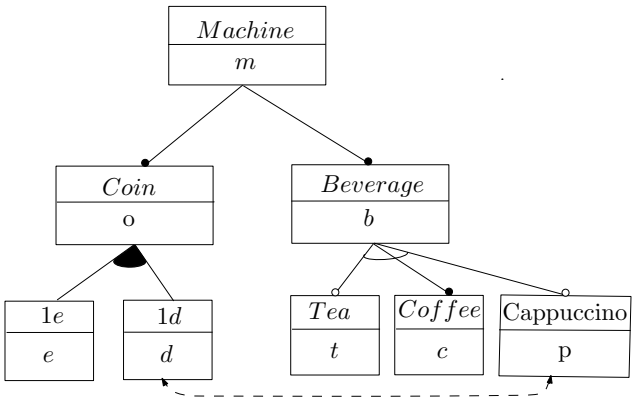

Figure 1. a feature model example.

In this feature model features such as coin (o)*, beverage* (b)*, and coffee* (c) *are mandatory and features tea* (t) *and cappuccino* (p) *are optional. (The single letters given under each feature are used later to represent the features in the propositional formulae.) The set of features coffee* (c)*, cappuccino* (p) *and tea* (t) *have the* or *relation. Also, features 1e* (e) *and 1d* (d) *have the* alternative *relation, which means the machine can take only one type of coin (euro or dollar). The dashed two headed arrow represents the* excludes *relation between the cappuccino* (p) *and the* $1d$ (d) *features.*

We assume that $\mathbb{B} = \{\top, \bot\}$ is the set of Boolean constants and $\mathbb{B}(F)$ denotes the set of all propositional formulae generated by considering the elements of the feature set F as propositional variables. Each propositional formula $\phi \in \mathbb{B}(F)$ is called a feature expression.

2.2. Featured Transition System

As mentioned before, in FTSs, the behavior of all products can be compactly depicted in one model by exploiting feature expressions as annotations. We give the formal definition of an FTS based on [1] as follows:

Definition 1 (FTS)**.** *A feature transition system is a 6-tuple* $(\mathbb{P}, A, F, \rightarrow, \Lambda, p_{init})$*, where*

1. $\mathbb{P}$ *is a set of* states*,*
2. A *is a set of* actions*,*
3. F *is a set of* features*,*
4. $\rightarrow \subseteq \mathbb{P} \times \mathbb{B}(F) \times A \times \mathbb{P}$ *is the* transition relation *satisfying the following condition:*

$$\forall_{P,a,P',\phi,\phi'} \left((P, \phi, a, P') \in \rightarrow \ \wedge\ (P, \phi', a, P') \in \rightarrow\ \right) \implies \phi = \phi',$$

5. $\Lambda \subseteq \{\lambda : F \rightarrow \mathbb{B}\}$ *is a set of* product configurations*,*

6. $p_{init} \in \mathbb{P}$ *is the* initial state[1].

Example 2. *Consider the FTS given in Fig. 2. This FTS describes the behavior of the products in the vending machine product line. In this paper, we consider the finite behavior of systems. Hence, Fig. 2 represents a part of the finite behavior of the vending machine product line.*

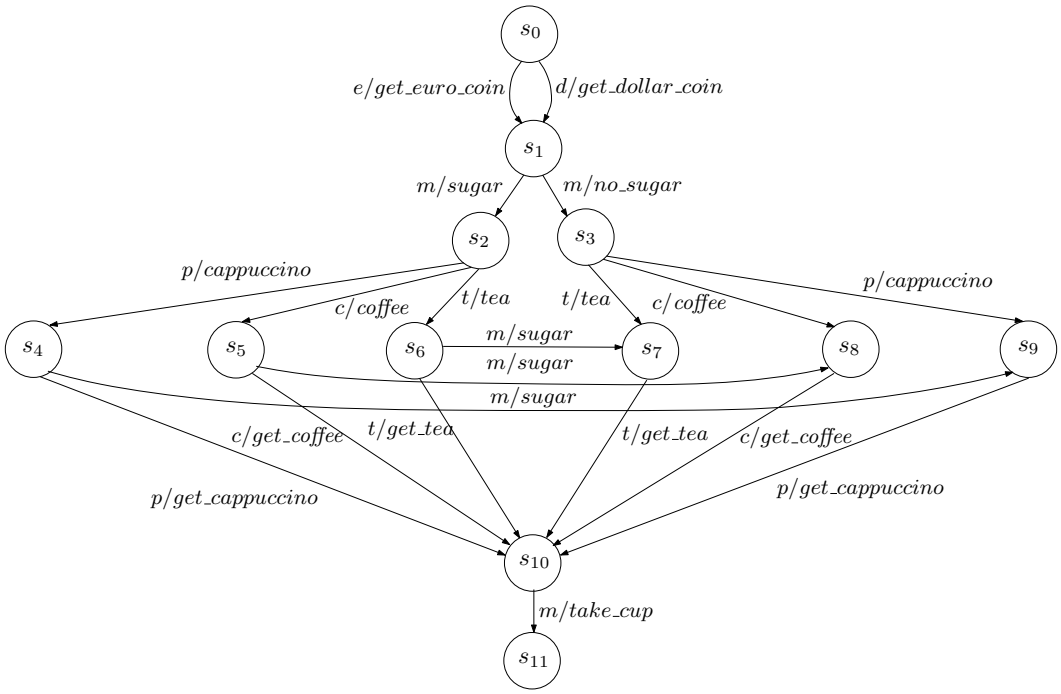

Figure 2. An FTS example.

The set of product configurations for this FTS is as follows:

$$\{\{m, o, b, e, c\}, \{m, o, b, d, c\}, \{m, o, b, e, c, t\}, \{m, o, b, d, c, t\}, \{m, o, b, e, c, p\}, \{m, o, b, e, c, p, t\}\}$$

Considering a feature expression $\phi \in \mathbb{B}(F)$ and a product configuration $\lambda \in \Lambda$, we say λ satisfies ϕ, denoted by $\lambda \models \phi$, if the result of every substitution of the value of the variables in the feature expression ϕ according to λ is satisfiable.

As mentioned above, each FTS represents the behavior of a set of products. We use LTSs as another formal structure in this paper to describe the behavior of single products. An LTS is defined as follows.

Definition 2 (LTS)**.** *A labeled transition system is a quadruple* $(\mathbb{S}, A, \rightarrow, s_{init})$*, where* $\mathbb{S}$ *is a set of states,* A *is a set of actions,* $\rightarrow \subseteq \mathbb{S} \times A \times \mathbb{S}$ *is the transition relation, and* s_{init} *is the initial state.*

Consider the LTS $(\mathbb{S}, A, \rightarrow, s_{init})$ and $s_{init} = s_0$; an initial finite path in this LTS is a sequence such as $\rho{=}s_0\ a_1\ s_1\ a_2\ \cdots\ a_n\ s_n$, where $\forall_{0 \le i < n} \cdot\ s_i \xrightarrow{a_{i+1}} s_{i+1}$. We denote the set of all initial finite paths in LTS T by $Paths(T)$. By $\rho(k)$, we denote the kth state in path ρ. For $\rho = s_0\ a_1\ \cdots\ a_n\ s_n$, we define $last(\rho) = s_n$. Furthermore, for a path ρ, $Trace(\rho)$ denotes the sequence of actions on the path. For example $Trace(s_0\ a_1\ s_1\ a_2\ \cdots\ a_n\ s_n) = a_1\ a_2\ \cdots\ a_n$. We Assume that $\mathbb{LTS}$ denotes the class of all LTSs.

[1] In the original definition of FTSs in [1], an FTS can have multiple initial states. Here, for the sake of a more succinct presentation we have considered a single initial state for FTSs; however it is straightforward to extend the results to multiple initial states.

2.2.1. Deriving Valid Products

Each FTS represents the behavior of a set of products. The behavior of each product can be represented using an LTS. Hence, each FTS has a set of valid LTS implementations. Intuitively, an LTS can be considered an FTS in which all the feature expressions on the transitions are true. To capture the behavior of a subset of products (or a single one), a refinement relation is defined. The refinement relation formalises the notion of product derivation as follows [9]:

Definition 3 (Product-Derivation Relation for FTSs)**.** *Given an FTS fts* $= (\mathbb{P}, A, F, \rightarrow, \Lambda, p_{init})$*, an LTS* $l = (\mathbb{S}, A, \rightarrow, s_{init})$*, and a product configuration* $\lambda \in \Lambda$*, a binary relation* $\mathcal{R}_\lambda \subseteq \mathbb{P} \times \mathbb{S}$ *is called* product-derivation relation *if and only if the following transfer properties are satisfied.*

1. $\forall_{P,Q,a,s,\phi}\ (P\,\mathcal{R}_\lambda\ s\ \wedge\ P \xrightarrow{\phi/a} Q \wedge \lambda \models \phi)\ \Rightarrow\ \exists_t \cdot\ s \xrightarrow{a} t\ \wedge\ Q\,\mathcal{R}_\lambda\ t,$

2. $\forall_{P,a,s,t}\ (P\,\mathcal{R}_\lambda\ s\ \wedge\ s \xrightarrow{a} t)\ \Rightarrow\ \exists_{Q,\phi} \cdot\ P \xrightarrow{\phi/a} Q\ \wedge\ \lambda \models \phi \wedge Q\,\mathcal{R}_\lambda\ t.$

A state $s \in \mathbb{S}$ *derives the product configuration* λ *from an FTS-specification* $P \in \mathbb{P}$*, denoted by* $P \vdash_\lambda s$*, if there exists a product-derivation relation* $\mathcal{R}_\lambda$ *such that* $P\,\mathcal{R}_\lambda\ s$*.*

We say that l is a valid implementation of *fts*, denoted by $fts \triangleright l$ if and only if there exists a product configuration $\lambda \in \Lambda$ such that $p_{init} \vdash_\lambda s_{init}$.

Example 3. *As an example, Fig. 3 depicts an LTS which implements the FTS in Fig. 2 and describes the behavior of a product in the vending machine product line serving coffee and tea with and without sugar.*

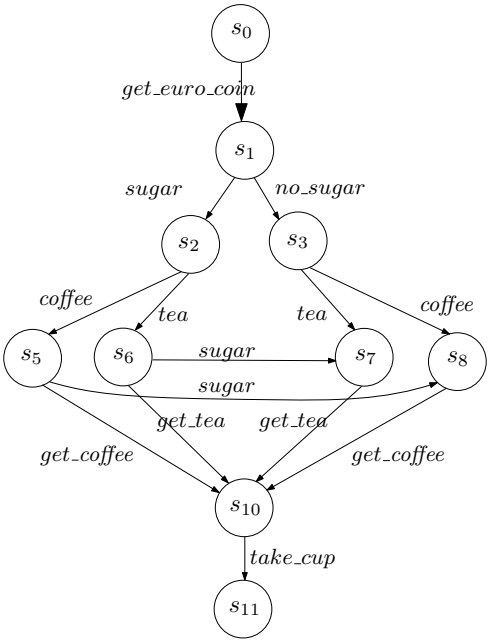

Figure 3. An LTS implementing the FTS in Fig. 2.

2.3. Product Line Process Algebras

PL-CCS is an extension of Milner's Calculus of Communicating Systems (CCS) [8] in which a new operator $\oplus$, called *binary variant*, is introduced to represent the alternative relation between features. The syntax of this process algebra is given in the following definition [2].

Definition 4 (PL-CCS). *Assuming the alphabet $A = \Sigma \cup \bar{\Sigma} \cup \{\tau\}$, where Σ is a set of symbols and $\bar{\Sigma} = \{\bar{a} \mid a \in \Sigma\}$ and $\tau \notin \Sigma$ is a symbol for internal actions. The syntax of PL-CCS terms e is defined by the following grammar:*

$$\mathbf{Nil} \mid \alpha.e \mid e + e' \mid e \oplus e' \mid e \parallel e' \mid e[f] \mid e\backslash L,$$

where **Nil** *denotes the terminating process, $\alpha.\_$ denotes the action prefixing for action $\alpha \in A$, $\_ + \_$ and $\_ \parallel \_$, respectively, denote non-deterministic choice and parallel composition, $\_[f]$ denotes renaming by means of a function f where $f : A \to A$, for each $L \subseteq A$, $\_\backslash L$ denotes the restriction operator (blocking actions in L), and finally $\_ \oplus \_$ denotes a family of binary operators.*

The difference between the introduced binary variant operator $\oplus$ and the ordinary alternative composition operator + in CCS is that the binary variant choice is made once and for all. As an example, consider the process terms $P = b.P + c.P$ and $Q = b.Q \oplus c.Q$; recursive process P keeps making choices between b and c in each recursion, while process Q makes a choice between b and c in the first recursion, and in all the following iterations the choice is respected. This means that process Q behaves deterministically after the first iteration with respect to the choice between b and c. For the sake of simplicity in the formal development of the theory, Gruler et al. assume that the $\oplus$ operators in each term are uniquely indexed with natural numbers. This means 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 [2], using a structural operational semantics. We refer to [2] for the formal semantics of PL-CCS. Each state in a PL-LTS comprises a pair of an ordinary state, e.g., a process term, and a *configuration vector*. The transitions of a PL-LTS are also labeled with configuration vectors. The configuration vectors are used to keep track of the choices made about alternative behavior and are of type $\{L, R, ?\}^I$ with I being an index set. The formal definition of a PL-LTS is as follows:

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 quintuple $(\mathbb{P} \times \{L, R, ?\}^I, A, I, \to, p_{init})$ consisting of a set of states $\mathbb{P} \times \{L, R, ?\}^I$, a set of actions A, an index set I, a transition relation $\to \subseteq (\mathbb{P} \times \{L, R, ?\}^I) \times (A \times \{L, R, ?\}^I) \times (\mathbb{P} \times \{L, R, ?\}^I)$, and an initial state p_{init}, 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. $\forall_{P_0,\nu_0,a,Q_0,\nu'_0,i,P_1,\nu_1,b,Q_1,\nu'_1,i}\ (P_0, \nu_0) \xrightarrow{a,\nu'_0} (Q_0, \nu'_0) \wedge (P_1, \nu_1) \xrightarrow{b,\nu'_1} (Q_1, \nu'_1) \wedge \nu_0(i) = \nu_1(i) = ? \wedge \nu'_0(i) \neq ? \neq \nu'_1(i) \implies (P_0, \nu_0) = (P_1, \nu_1)$.

The first condition indicates that each transition in the model is labeled with the configuration vector in the target state of the transition. The second condition shows that after making a variant choice which leads to assigning the value of an element in the configuration vector to L or R, that value remains the same in the following steps. The third condition indicates that the same choice cannot be resolved in multiple states in the model. This follows from the definition of the semantics for PL-CCS terms in [2], where each variant operator is labeled with a unique index. Assuming that in the above defined PL-LTS, $p_{init} = (P_0, \nu_0)$; an initial finite path in this PL-LTS is a sequence such as $(P_0, \nu_0)\ \{a_1, \nu_1\}\ (P_1, \nu_1)\ \cdots\ \{a_n, \nu_n\}\ (P_n, \nu_n)$ where $\forall_{0 \leq i < n} \cdot\ (P_i, \nu_i) \xrightarrow{a_{i+1}, \nu_{i+1}} (P_{i+1}, \nu_{i+1})$. We denote the set of all such paths for a PL-LTS *plt* by *Paths*(*plt*). We define the following relations between configuration vectors in a PL-LTS which are used in the rest of the paper.

Definition 6 (Configuration Ordering). *The preorder $\sqsubseteq$ 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 $\nu, \nu' \in \{L, R, ?\}^I$.

Using this relation we can specify if a configuration vector is more refined compared to the other (i.e. has less undecided choices).

Definition 7 (Configuration Conflict)**.** *The relation* $\bowtie$ *on the set* $\{L, R, ?\}$ *is defined as:*

$$\bowtie = \{(L, R), (R, L)\}.$$

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

Using this relation we can specify if there is a conflict between two configuration vectors (i.e. there is at least one element which is decided in both configuration vectors and the decision is not the same).

In order to define the set of LTS implementations of a PL-LTS, the product-derivation relation for PL-LTSs is given as follows.

Definition 8 (Product-Derivation Relation for PL-LTSs)**.** *Let* $plt = (\mathbb{P} \times \{L, R, ?\}^I, A, I, \rightarrow, p_{init})$ *be a PL-LTS and let* $l = (\mathbb{S}, A, \rightarrow, s_{init})$ *be an LTS. A binary relation* $\mathcal{R}_\theta \subseteq (\mathbb{P} \times \{L, R, ?\}^I) \times \mathbb{S}$ *(parameterized by product configuration* $\theta \in \{L, R\}^I$*) is a* product-derivation relation *if and only if the following transfer properties are satisfied:*

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

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

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* $\theta \in \{L, R\}^I$*, denoted by* $(P, \nu) \vdash_\theta s$*, if* $\nu \sqsubseteq \theta$ *and there exists a product-derivation relation* $\mathcal{R}_\theta$ *such that* $(P, \nu)\ \mathcal{R}_\theta\ s$.

We say that l is a valid implementation of the PL-LTS plt, denoted by $plt \prec l$ if and only if there exists a configuration vector $\theta \in \{L, R, ?\}^I$ such that $p_{init} \vdash_\theta s_{init}$.

2.4. Encoding

In order to compare the expressiveness power between different modeling formalisms for SPLs, we give the following definitions, respectively, for *product line structure* and *encoding*.

Definition 9 (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 FTSs and PL-LTSs) and* $[\![\ ]\!] : \mathbb{M} \rightarrow \mathbb{LTS}$ *is the semantic function mapping a product line model to a set of product LTSs that can be derived from the product line model.*

Consider the tuple $(\mathbb{FTS}, [\![]\!])$, which is a product line structure defined for the class of FTSs. For an arbitrary FTS fts and arbitrary LTS l, it holds $l \in [\![fts]\!] \Leftrightarrow fts \triangleright l$ (see definition of $fts \triangleright l$ in Section 2.2.1). Similarly, consider the tuple $(\mathbb{PL} - \mathbb{LTS}, [\![]\!])$, which is the product line structure defined for the class of PL-LTSs. For an arbitrary PL-LTS plt and arbitrary LTS l it holds $l \in [\![plt]\!] \Leftrightarrow plt \prec l$ (see definition of $plt \prec l$ in Section 2.3).

Definition 10 (Encoding)**.** *An encoding from a product line structure* $\mathbf{M} = (\mathbb{M}, [\![\ ]\!])$ *into* $\mathbf{M}' = (\mathbb{M}', [\![\ ]\!]')$*, is defined as a function* $E : \mathbf{M} \rightarrow \mathbf{M}'$ *satisfying the following correctness criterion:* $[\![\ ]\!] = [\![\ ]\!]' \circ E$.

We say that a product line structure $\mathbf{M}'$ *is at least as expressive as* $\mathbf{M}$ *if and only if there exists an encoding* $E : \mathbf{M} \rightarrow \mathbf{M}'$*. Also, we say that two product line structures* $\mathbf{M}$ *and* $\mathbf{M}'$ *are equally expressive if and only if there exists an encoding from* $\mathbf{M}$ *to* $\mathbf{M}'$ *and vice versa.*

3. Encodings between FTSs and PL-LTSs

In this section, we provide an encoding from FTSs to PL-LTSs and thereby show that PL-LTSs are at least as expressive as FTSs. Furthermore, we provide a slight variation of the encoding from PL-LTSs into FTSs given in [9], based on the more liberal definition of PL-LTSs. We show that based on the latter encoding, the class of FTSs is at least as expressive as the class of PL-LTSs; thus, reinstating the results of [9] for the more liberal definition of PL-LTS. The combination of these two encodings shows that PL-LTSs and FTSs are equally expressive.

Definition 11 (FTS to PL-LTS Encoding)**.** *Consider an FTS such as fts* $= (\mathbb{P}, A, F, \rightarrow, \Lambda, p_{init})$*. The PL-LTS resulting from the encoding, denoted by E(fts), is a quintuple* $(\bar{\mathbb{P}}, A, I, \rightarrow, \bar{p}_{init})$*, where:*

- $\bar{\mathbb{P}} \subseteq \mathbb{P} \times \{L, R, ?\}^I$*,*
- *A is the set of actions,*
- $I = \{0, 1, \cdots, |\Lambda| - 1\}$ *is the index set,*
- $\bar{p}_{init} = (p_{init}, \{?\}^I)$ *is the initial state,*
- $\rightarrow \subseteq \bar{\mathbb{P}} \times A \times \{L, R, ?\}^I \times \bar{\mathbb{P}}$*, is the transition relation which is defined as follows.*

  Consider an arbitrary bijective function $X : \Lambda \rightarrow [0 \cdots |\Lambda| - 1]$*, where* $[0 \cdots |\Lambda| - 1]$ *is the set of all natural numbers not greater than* $|\Lambda| - 1$*. For each product configuration* $\lambda \in \Lambda$*, we define* ν_λ *to be the configuration vector such that* $\forall_{0 \leq j \leq |\Lambda|} \cdot (j < X(\lambda) \Rightarrow \nu_\lambda(j) = R) \wedge (j = X(\lambda) \Rightarrow \nu_\lambda(j) = L) \wedge (j > X(\lambda) \Rightarrow \nu_\lambda(j) =?)$*. Then, the transition relation is the smallest set satisfying the following two conditions:*

$$\forall_{\lambda \in \Lambda} \cdot P \xrightarrow{\phi/a} Q \wedge P = p_{init} \wedge \lambda \models \phi \Rightarrow (p_{init}, \{?\}^I) \xrightarrow{a, \nu_\lambda} (Q, \nu_\lambda)$$

$$\forall_{\lambda \in \Lambda} \cdot P \xrightarrow{\phi/a} Q \wedge P \neq p_{init} \wedge \lambda \models \phi \Rightarrow (P, \nu_\lambda) \xrightarrow{a, \nu_\lambda} (Q, \nu_\lambda)$$

In the above definition, all the choices are resolved in the first step and through the transitions emanating from the initial state. After that, the configuration vectors remain the same.

Example 4. *An example of encoding an FTS into a PL-LTS is depicted in Fig. 4. In this figure, part (a) represents an FTS resulting from removing feature* tea *from the FTS in Fig. 2 and part (b) represents the PL-LTS resulting form the encoding.*

As can be seen the encoding results in a blow up of the size of the model. In the remainder of the paper, we show that for some FTSs, such exponential blow up of the size after encoding, regardless of the applied encoding, is unavoidable. Next, we show that the conditions of Definition 5 are satisfied by the PL-LTSs resulting after the encoding.

Theorem 1. *Each PL-LTS resulting from encoding an FTS, using the encoding given in Definition 11, satisfies the conditions of Definition 5.*

Proof. Consider an arbitrary FTS $fts = (\mathbb{P}, A, F, \rightarrow, \Lambda, p_{init})$ and the PL-LTS resulting from the encoding, $E(fts) = (\bar{\mathbb{P}}, A, I, \rightarrow, \bar{p}_{init})$. The first condition of Definition 5, is as follows: $\forall_{(P,\nu),(Q,\nu'') \in \bar{\mathbb{P}}, a \in A, \nu' \in \{R,L,?\}^I} (P, \nu) \xrightarrow{a,\nu'} (Q, \nu'') \implies \nu' = \nu''$. It is trivial to see that the first condition in Definition 5 holds, due to the construction of the transition relation in Definition 11.

Next, we consider the second condition in Definition 5, that is: $\forall_{(P,\nu),(Q,\nu') \in \bar{\mathbb{P}}, a \in A, i \in I} (P, \nu) \xrightarrow{a,\nu'} (Q, \nu') \wedge \nu(i) \neq? \implies \nu'(i) = \nu(i)$.

According to Definition 11, the transitions in $E(fts)$ are defined using the following two rules:

1. $\forall_{\lambda \in \Lambda} \cdot P \xrightarrow{\phi/a} Q \wedge P = p_{init} \wedge \lambda \models \phi \Rightarrow (p_{init}, \{?\}^I) \xrightarrow{a, \nu_\lambda} (Q, \nu_\lambda)$
2. $\forall_{\lambda \in \Lambda} \cdot P \xrightarrow{\phi/a} Q \wedge P \neq p_{init} \wedge \lambda \models \phi \Rightarrow (P, \nu_\lambda) \xrightarrow{a, \nu_\lambda} (Q, \nu_\lambda)$

If the transition is due to rule 1, then $\nu(i) \neq?$ cannot hold and hence this condition holds trivially. If the transition is due to rule 2, the configuration vector in the target state of a transition is the same as the configuration vector in the source state of the transition. Hence, the second condition in Definition 5 is satisfied.

Finally, we consider the third condition in Definition 5, that is: $\forall_{(P_0,\nu_0),(Q_0,\nu'_0),(P_1,\nu_1),(Q_1,\nu'_1) \in \bar{\mathbb{P}}, a,b \in A, i \in I} (P_0, \nu_0) \xrightarrow{a,\nu'_0} (Q_0, \nu'_0) \wedge (P_1, \nu_1) \xrightarrow{b,\nu'_1} (Q_1, \nu'_1) \wedge \nu_0(i) = \nu_1(i) =? \wedge \nu'_0(i) \neq? \neq \nu'_1(i) \implies (P_0, \nu_0) = (P_1, \nu_1)$. According to Definition 11, only the transitions emanating from the initial state of $E(fts)$ have source and target states with different configuration vectors. Since, $E(fts)$ has a single initial state, the third condition in Definition 11 is preserved by $E(fts)$. □

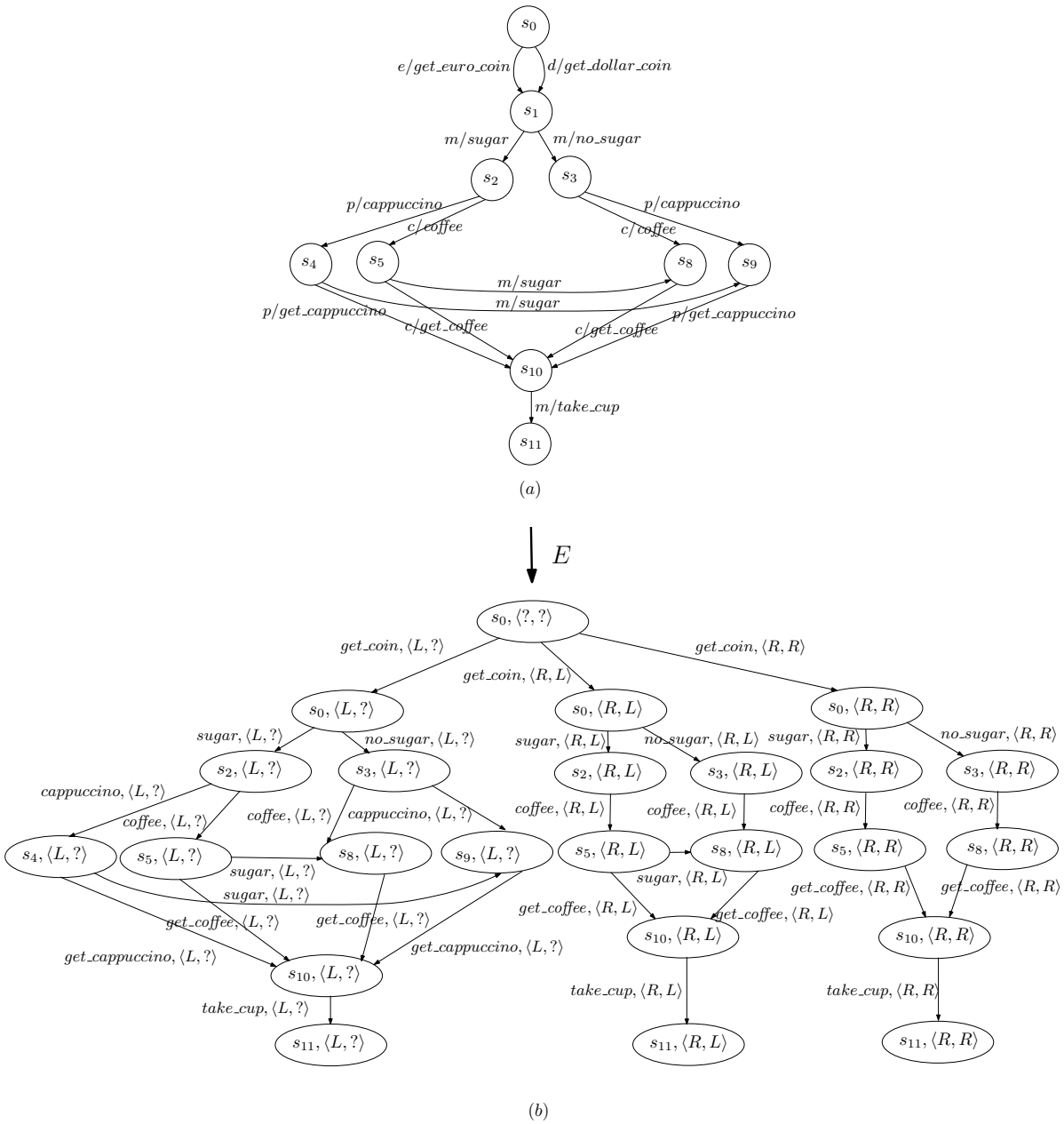

Figure 4. Example of encoding an FTS to a PL-LTS.

Next, we give a slight variation of the encoding from PL-LTSs into FTSs given in [9].

Definition 12 (PL-LTS to FTS Encoding)**.** *Consider a PL-LTS* $plt = (\mathbb{P}, A, I, \rightarrow, p_{init})$ *with the set of product configurations* Θ*; the FTS resulting from the encoding, denoted by* $E(plt)$*, is a 6-tuple* $(\mathbb{P}, A, F, \rightarrow', \Lambda, p_{init})$*, where:*

- $F = \bigcup_{i \in I} \{L_i, R_i\}$.
- $\Lambda = \bigcup_{\theta \in \Theta} \{\bigwedge_{i \in I} \theta(i)_i\}$

- *The transition relation $\rightarrow'$ is defined in the following way:*

$$\frac{(P,\nu) \xrightarrow{a,\nu} (Q,\nu)}{(P,\nu) \overset{a,\top}{\rightarrow}' (Q,\nu)} \qquad \frac{(P,\nu) \xrightarrow{a,\nu'} (Q,\nu') \quad \phi = \bigwedge_{i\in I} \nu'(i)_i \quad \Xi(I,\nu,\nu')}{(P,\nu) \overset{a,\phi}{\rightarrow}' (Q,\nu')},$$

where $\Xi(I,\nu,\nu') \iff \forall i \in I \cdot\ \nu'(i) \neq \nu(i) \wedge \forall_{j\notin I}\ \nu'(j) = \nu(j)$. (In the above definition we assume the notations $\nu(i)_i$ and $\theta(i)_i$, which are used in construction of feature expressions and the product configurations, can be also used for representing variables that stand for features, i.e., L_i or R_i for $i \in I$.)

The difference between the encoding given in Definition 12 and the one given in Theorem 3 in [9], is in the definition of the transition relation. In the definition of transition relation given in Theorem 3 in [9], in the description of function Ξ it is assumed that only one of the elements of ν changes in each step. In Definition 12, we relax this assumption according to the original and more liberal definition of PL-LTSs given in Definition 5.

4. Comparative Expressiveness

In this section, we first prove that the class of PL-LTSs is at least as expressive as the class of FTSs. Then, we show that the result of the proof provided in [9], which shows that the class of FTSs is at least as expressive as the class of PL-LTSs, remains the same considering the more liberal definition of PL-LTSs. Thus, we conclude that the class of PL-LTSs and the class of FTSs are equally expressive.

Theorem 2. *The class of PL-LTSs is at least as expressive as the class of FTSs.*

Proof. It suffices to show that each LTS l that implements an arbitrary FTS *fts* is also a valid implementation of $E(fts)$, the PL-LTS resulting from applying the encoding given in Definition 11 to *fts*, and vice versa, i.e., $\forall l \in \mathbb{LTS} \cdot fts \triangleright l \Leftrightarrow E(fts) \prec l$. This means the proof of the theorem can be reduced to proving $[\![fts]\!] = [\![E(fts)]\!]'$ (see Definition 9).

Consider $fts = (\mathbb{P}, A, F, \rightarrow, \Lambda, p_{init})$ and $E(fts) = (\bar{\mathbb{P}}, A, I, \rightarrow, \bar{p}_{init})$; we separate the bi-implication in the proof obligation into the following two implications:

- $[\![fts]\!] \subseteq [\![E(fts)]\!]'$: In order to prove $[\![fts]\!] \subseteq [\![E(fts)]\!]'$, we show that $\forall l \in \mathbb{LTS} \cdot\ l \in [\![fts]\!] \Rightarrow l \in [\![E(fts)]\!]'$.

  Consider an arbitrary LTS $l = (\mathbb{S}, A, \rightarrow, s_{init})$ s.t. $l \in [\![fts]\!]$, which means that $fts \triangleright l$ (see Section 2.4). We prove $E(fts) \prec l$ and hence, $l \in [\![E(fts)]\!]'$.

  Let Θ denote the set of product configuration vectors derived from the set Λ, i.e., $\Theta = \{\nu_\lambda \mid \lambda \in \Lambda\}$, where ν_λ has the same definition as given in Definition 11. In order to prove $E(fts) \prec l$, it suffices to show that for some product configuration vector $\theta \in \Theta$, it holds that $\bar{p}_{init} \vdash_\theta s_{init}$ (see Section 2.3).

  Next, we show that the above statement is satisfied by l and $E(fts)$. Considering Definition 8, for any two arbitrary states, $(P,\nu) \in \bar{\mathbb{P}}$ and $s \in \mathbb{S}$, $(P,\nu) \vdash_\theta s$ holds if a product-derivation relation such as $\mathcal{R}_\theta$ exists such that $(P,\nu)\ \mathcal{R}_\theta\ s$ and $\mathcal{R}_\theta$ satisfies the following properties:

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

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

  Hence, in the next step we prove that such a relation exists and that the initial states are related by it.

  Based on Definition 3, the assumption $fts \triangleright l$ implies that for some $\lambda \in \Lambda$ a product-derivation relation $\mathcal{R}_\lambda \subseteq \mathbb{P}\times\mathbb{S}$ exists which satisfies the following properties:

  1. $\forall_{P,Q,a,s,\phi} \cdot\ (P\ \mathcal{R}_\lambda\ s\ \wedge\ P \xrightarrow{\phi/a} Q \wedge \lambda \models \phi) \Rightarrow \exists_t \cdot\ s \xrightarrow{a} t\ \wedge\ Q\ \mathcal{R}_\lambda\ t$
  2. $\forall_{P,a,s,t} \cdot\ (P\ \mathcal{R}_\lambda\ s\ \wedge\ s \xrightarrow{a} t) \Rightarrow \exists_{Q,\phi} \cdot\ P \xrightarrow{\phi/a} Q\ \wedge\ \lambda \models \phi \wedge Q\ \mathcal{R}_\lambda\ t$,

We define a binary relation $\mathcal{R}_\theta \subseteq \bar{\mathbb{P}} \times \mathbb{S}$ (where θ is a configuration vector in Θ) such that:

$$\forall P \in \mathbb{P}, s \in \mathbb{S} \cdot \ \exists \lambda \in \Lambda \cdot \big((P = p_{init} \ \wedge \ P \,\mathcal{R}_\lambda \, s) \Leftrightarrow (p_{init}, \{?\}^I) \,\mathcal{R}_\theta \, s \ \wedge \ \theta = \nu_\lambda\big) \wedge \\ \big((P \neq p_{init} \ \wedge \ P \,\mathcal{R}_\lambda \, s) \Leftrightarrow (P, \nu_\lambda) \,\mathcal{R}_\theta \, s \ \wedge \ \theta = \nu_\lambda\big)$$

Consider the configuration vector λ that derives LTS l from FTS *fts* (based on Definition 8); let $\theta = \nu_\lambda$ and $\mathcal{R}_\theta$ be a member of the above defined relation. Next, we prove that $\mathcal{R}_\theta$ satisfies the properties of a product-derivation relation (statements (1) and (2)).

Consider an arbitrary pair of states in $\mathcal{R}_\theta$, such as $(P, \nu) \,\mathcal{R}_\theta \, s$; based on the definition given above for $\mathcal{R}_\theta$, it holds $P \,\mathcal{R}_\lambda \, s$, where we distinguish the following two cases: $(P, \nu) = \bar{p}_{init}$ and $(P, \nu) \neq \bar{p}_{init}$.

- First, we prove that statement (1) is satisfied by $\mathcal{R}_\theta$.
  - $(P, \nu) = \bar{p}_{init}$.
    Thus $\nu = \{?\}^I$ and $P = p_{init}$ (see Definition 11). Consider an arbitrary transition of the form $(p_{init}, \{?\}^I) \xrightarrow{a,\nu_{\lambda'}} (Q, \nu_{\lambda'})$; based on Definition 11, such a transition is resulting from encoding one of the outgoing transitions from P in *fts*, i.e.:

    $$\forall_{a,Q,\lambda'} \cdot \ (p_{init}, \{?\}^I) \xrightarrow{a,\nu_{\lambda'}} (Q, \nu_{\lambda'}) \Leftrightarrow \exists \phi, P \cdot \ P \xrightarrow{\phi/a} Q \ \wedge \lambda' \models \phi \ \wedge P = p_{init} \tag{1.i}$$

    Considering property 1 satisfied by relation $\mathcal{R}_\lambda$, $P \,\mathcal{R}_\lambda \, s$ implies the following statement:

    $$\forall_{a,Q,\phi} \cdot \ \big(P \xrightarrow{\phi/a} Q \wedge \lambda \models \phi\big) \ \Rightarrow \ \exists t \ \cdot \ s \xrightarrow{a} t \ \wedge \ Q \,\mathcal{R}_\lambda \, t \tag{1.ii}$$

    Based on the definition of $\mathcal{R}_\theta$, $Q \,\mathcal{R}_\lambda \, t \Leftrightarrow (Q, \nu_\lambda) \,\mathcal{R}_\theta \, t \ \wedge \theta = \nu_\lambda$. Hence, $(Q, \nu_{\lambda'}) \,\mathcal{R}_\theta \, t$ holds only in case of $\lambda' = \lambda$. Given that $\theta = \nu_\lambda$, based on the definition of the relation $\sqsubseteq$ (see Definition 6) it holds $\nu_\lambda \sqsubseteq \theta$ (notice that for any $\lambda' \in \Lambda$ such that $\lambda \neq \lambda'$, based on the definition of ν_λ it holds $\nu_\lambda \bowtie \nu_{\lambda'}$ and hence, $\nu_{\lambda'} \not\sqsubseteq \theta$). Thus, from (1.*i*) and (1.*ii*), the following statement is derived:

    $$\forall_{a,Q} \cdot \ (p_{init}, \{?\}^I) \xrightarrow{a,\nu_\lambda} (Q, \nu_\lambda) \ \wedge \ \nu_\lambda \sqsubseteq \theta \Rightarrow \ \exists t \ \cdot \ s \xrightarrow{a} t \ \wedge \ (Q, \nu_\lambda) \,\mathcal{R}_\theta \, t \tag{1.iii}$$

  - $(P, \nu) \neq \bar{p}_{init}$.
    Thus, (based on the definition of $\mathcal{R}_\theta$) $\nu = \nu_\lambda$. Consider an arbitrary transition emanating from (P, ν_λ) of the form $(P, \nu_\lambda) \xrightarrow{a,\nu_\lambda} (Q, \nu_\lambda)$; based on Definition 11, the configuration vector in the target state of the outgoing transitions from (P, ν_λ) is ν_λ and such transition is resulting from encoding one of the outgoing transitions from P in *fts*, i.e.:

    $$\forall_{a,Q} \cdot \ (P, \nu_\lambda) \xrightarrow{a,\nu_\lambda} (Q, \nu_\lambda) \Leftrightarrow \exists \phi \cdot \ P \xrightarrow{\phi/a} Q \ \wedge \lambda \models \phi \ \wedge P \neq p_{init} \tag{1.iv}$$

    Considering property 1 satisfied by relation $\mathcal{R}_\lambda$, $P \,\mathcal{R}_\lambda \, s$ implies the following statement:

    $$\forall_{a,Q,\phi} \cdot \ \big(P \xrightarrow{\phi/a} Q \wedge \lambda \models \phi\big) \ \Rightarrow \ \exists t \ \cdot \ s \xrightarrow{a} t \ \wedge \ Q \,\mathcal{R}_\lambda \, t \tag{1.v}$$

    Based on the definition of $\mathcal{R}_\theta$, $Q \,\mathcal{R}_\lambda \, t \Leftrightarrow (Q, \nu_\lambda) \,\mathcal{R}_\theta \, t \ \wedge \theta = \nu_\lambda$. Using the same reasoning as in the previous case, from (1.*iv*) and 1.(*v*), the following statement is derived:

    $$\forall_{a,Q} \cdot \ (P, \nu_\lambda) \xrightarrow{a,\nu_\lambda} (Q, \nu_\lambda) \ \wedge \ \nu_\lambda \sqsubseteq \theta \Rightarrow \ \exists t \ \cdot \ s \xrightarrow{a} t \ \wedge \ (Q, \nu_\lambda) \,\mathcal{R}_\theta \, t \tag{vi}$$

  Considering (1.*iii*) and (1.*vi*), the following statement holds:

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

  which means $\mathcal{R}_\theta$ satisfies statement (1).

– Next, we prove that statement (2) is satisfied by $\mathcal{R}_\theta$. Again we distinguish the two cases: $(P, \nu) = \bar{p}_{init}$ and $(P, \nu) \neq \bar{p}_{init}$.

  * $(P, \nu) = \bar{p}_{init}$.
    Thus, $\nu = \{?\}^I$ and $P = p_{init}$ (see Definition 11).
    Consider an arbitrary transition emanating from $(p_{init}, \{?\}^I)$ of the form $(p_{init}, \{?\}^I) \xrightarrow{a,\nu_{\lambda'}} (Q, \nu_{\lambda'})$; based on Definition 11, such a transition is resulting from encoding an outgoing transition from P, i.e.:

$$\forall_{a,Q,\lambda'} \cdot \ (p_{init}, \{?\}^I) \xrightarrow{a,\nu_{\lambda'}} (Q, \nu_{\lambda'}) \Leftrightarrow \exists \phi, P \cdot \ \ P \xrightarrow{\phi/a} Q \ \wedge \lambda \models \phi \wedge P = p_{init} \tag{2.i}$$

    Considering property 2 satisfied by relation $\mathcal{R}_\lambda$, $P \, \mathcal{R}_\lambda \, s$ implies the following statement:

$$\forall_{a,t} \cdot \ (s \xrightarrow{a} t) \ \Rightarrow \ \exists_{Q,\phi} \cdot \ P \xrightarrow{\phi/a} Q \ \wedge \ \lambda \models \phi \wedge Q \, \mathcal{R}_\lambda \, t \tag{2.ii}$$

    Based on the definition of $\mathcal{R}_\theta$, $Q \, \mathcal{R}_\lambda \, t \Leftrightarrow (Q, \nu_\lambda) \, \mathcal{R}_\theta \, t \ \wedge \theta = \nu_\lambda$. Hence, $(Q, \nu_{\lambda'}) \, \mathcal{R}_\theta \, t$ holds only in case $\lambda' = \lambda$. Given that $\theta = \nu_\lambda$, based on the definition of the relation $\sqsubseteq$ (see Definition 6) it holds $\nu_\lambda \sqsubseteq \theta$ and for all $\lambda' \in \Lambda$ such that $\lambda \neq \lambda'$ it holds $\nu_{\lambda'} \not\sqsubseteq \theta$. Considering (2.*i*) and (2.*ii*), the following statement holds:

$$\forall_{a,t} \cdot \ s \xrightarrow{a} t \Rightarrow \ \exists a, Q \ \cdot \ (p_{init}, \{?\}^I) \xrightarrow{a,\nu_\lambda} (Q, \nu_\lambda) \ \wedge \ \nu_\lambda \sqsubseteq \theta \ \wedge \ (Q, \nu_\lambda) \, \mathcal{R}_\theta \, t \tag{2.iii}$$

  * $(P, \nu) \neq \bar{p}_{init}$.
    Thus, (based on the definition of $\mathcal{R}_\theta$) $\nu = \nu_\lambda$. Consider an arbitrary transition emanating from (P, ν_λ) of the form $(P, \nu_\lambda) \xrightarrow{a,\nu_\lambda} (Q, \nu_\lambda)$, based on the Definition 11, the configuration vector in the target state of the outgoing transitions from (P, ν_λ) is ν_λ; such a transition is resulting from encoding an outgoing transition from P, i.e.:

$$\forall_{a,Q} \cdot \ (P, \nu_\lambda) \xrightarrow{a,\nu_\lambda} (Q, \nu_\lambda) \Leftrightarrow \exists \phi \cdot \ \ P \xrightarrow{\phi/a} Q \ \wedge \lambda \models \phi \ \wedge P \neq p_{init} \tag{2.iv}$$

    Furthermore, consider property 2 satisfied by relation $\mathcal{R}_\lambda$; $P \, \mathcal{R}_\lambda \, s$ implies the following statement:

$$\forall_{a,t} \cdot \ (s \xrightarrow{a} t) \ \Rightarrow \ \exists_{Q,\phi} \cdot \ P \xrightarrow{\phi/a} Q \ \wedge \ \lambda \models \phi \wedge Q \, \mathcal{R}_\lambda \, t \tag{2.v}$$

    Given the definition of $\mathcal{R}_\theta$, $Q \, \mathcal{R}_\lambda \, t \Leftrightarrow (Q, \nu_\lambda) \, \mathcal{R}_\theta \, t \ \wedge \theta = \nu_\lambda$. Since, $\theta = \nu_\lambda$ and based on Definition 6 it holds $\nu_\lambda \sqsubseteq \theta$. Considering (2.*iv*) and (2.*v*), the following statement holds:

$$\forall a, t \cdot \ s \xrightarrow{a} t \Rightarrow \ \exists a, Q \ \cdot \ (P, \nu_\lambda) \xrightarrow{a,\nu_\lambda} (Q, \nu_\lambda) \ \wedge \ \nu_\lambda \sqsubseteq \theta \ \wedge \ (Q, \nu_\lambda) \, \mathcal{R}_\theta \, t \tag{2.vi}$$

  Considering (2.*iii*) and (2.*vi*), the following statement holds:

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

  which means that $\mathcal{R}_\theta$ satisfies the second property of a product derivation relation.

Based on the assumption $fts \triangleright l$, it holds $p_{init} \vdash_\lambda s_{init}$. As shown above, $\mathcal{R}_\theta$ satisfies the properties of a product derivation relation given in Definition 8. Hence, based on the definition of $\mathcal{R}_\theta$ it holds that $p_{init} \vdash_\lambda s_{init} \Rightarrow (p_{init}, \{?\}^I) \vdash_\theta s_{init}$. Thus, $\bar{p}_{init} \vdash_\theta s_{init}$. This means $E(fts) \prec l$ and subsequently $l \in [\![E(fts)]\!]'$.

Hence, we conclude that $[\![fts]\!] \subseteq [\![E(fts)]\!]'$.

- $[\![E(fts)]\!] \subseteq [\![fts]\!]$.

Proof. In order to prove $[\![E(fts)]\!]' \subseteq [\![fts]\!]$, we show that $\forall l \in \mathbb{LTS} \cdot \ l \in [\![E(fts)]\!]' \Rightarrow l \in [\![fts]\!]$.

Consider an arbitrary LTS $l = (\mathbb{S}, A, \rightarrow, s_{init})$, s.t., $l \in [\![E(fts)]\!]'$ and subsequently $E(fts) \prec l$ (see Definition 10). We prove $fts \triangleright l$ and hence, $l \in [\![fts]\!]$.

To prove $fts \triangleright l$, it suffices to show that for some product configuration $\lambda \in \Lambda$ the following statement holds: $p_{init} \vdash_\lambda s_{init}$ (see Definition 3).

Next, we show that the above statement is satisfied by l and fts. According to Definition 3, $P \vdash_\lambda s$ holds if a product-derivation relation such as $\mathcal{R}_\lambda$ exists such that $P\,\mathcal{R}_\lambda\,s$ and $\mathcal{R}_\lambda$ satisfies the following properties:

(1) $\forall_{P,Q,a,s,\phi} \cdot (P\,\mathcal{R}_\lambda\,s \ \wedge\ P \xrightarrow{\phi/a} Q \wedge \lambda \models \phi) \ \Rightarrow\ \exists_t \cdot\ s \xrightarrow{a} t \ \wedge\ Q\,\mathcal{R}_\lambda\,t$

(2) $\forall_{P,a,s,t} \cdot (P\,\mathcal{R}_\lambda\,s \ \wedge\ s \xrightarrow{a} t) \ \Rightarrow\ \exists_{Q,\phi} \cdot\ P \xrightarrow{\phi/a} Q \ \wedge\ \lambda \models \phi \wedge Q\,\mathcal{R}_\lambda\,t$

Hence, in the next step we prove the existence of a relation between states of l and fts that satisfies the above properties.

Based on Definition 8, the assumption $E(fts) \prec l$ implies that for some $\theta \in \Theta$, a product-derivation relation $\mathcal{R}_\theta \subseteq \bar{\mathbb{P}} \times \mathbb{S}$ exists, which satisfies the following properties:

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

Next, we define a binary relations $\mathcal{R}_\lambda$ (where λ is a product configuration in Λ) such that:

$$\forall_{P\in\mathbb{P},s\in\mathbb{S}} \cdot\ \forall_{\theta\in\Theta} \cdot \big((p_{init},\{?\}^I)\,\mathcal{R}_\theta\,s \Leftrightarrow \big((P = p_{init} \ \wedge\ P\,\mathcal{R}_\lambda\,s \ \wedge\ \nu_\lambda = \theta)\big) \wedge$$
$$\big((P,\nu_\lambda)\,\mathcal{R}_\theta\,s \Leftrightarrow (P \neq p_{init} \ \wedge\ P\,\mathcal{R}_\lambda\,s \ \wedge \nu_\lambda = \theta)\big)$$

Assume that LTS l is derived from PL-LTS $E(fts)$ with regards to the product configuration vector $\theta = \nu_\lambda$. Let $\mathcal{R}_\lambda$ be a relation defined as above. Next, we prove that $\mathcal{R}_\lambda$ satisfies the properties of a product-derivation relation (statements (1) and (2)).

– First, we prove that $\mathcal{R}_\lambda$ satisfies statement (1).
Consider an arbitrary pair (P, s) of states where $P \in \mathbb{P}$ and $s \in \mathbb{S}$ such that $P\,\mathcal{R}_\lambda\,s$. Based on the definition given for $\mathcal{R}_\lambda$; it holds $(P,\nu)\,\mathcal{R}_\theta\,s$, where $\theta = \nu_\lambda$, and $\nu = \{?\}^I$ if $P = p_{init}$ and $(P,\nu)\,\mathcal{R}_\theta\,s$ where $\nu = \nu_\lambda$ and $\theta = \nu_\lambda$ if $P \neq p_{init}$. We distinguish the following two cases: $P = p_{init}$ and $P \neq p_{init}$.
  * First, we consider the case where $P = p_{init}$:
  Based on Definition 11, each transition emanating from P such as $P \xrightarrow{\phi/a} Q$, is encoded as a transition in PL-LTS $E(fts)$, i.e.:

$$\forall_{a,Q,\lambda'} \cdot\ P \xrightarrow{\phi/a} Q \ \wedge\ P \in p_{init} \ \wedge\ \lambda' \models \phi \Leftrightarrow\ (p_{init},\{?\}^I) \xrightarrow{a,\nu_{\lambda'}} (Q,\nu_{\lambda'}) \tag{1.i}$$

  Considering property 1 satisfied by relation $\mathcal{R}_\theta$, $(p_{init},\{?\}^I)\,\mathcal{R}_\theta\,s$ implies the following statement:

$$\forall_{a,Q,\lambda'} \cdot ((p_{init},\{?\}^I) \xrightarrow{a,\nu_{\lambda'}} (Q,\nu_{\lambda'}) \ \wedge\ \nu_{\lambda'} \sqsubseteq \theta) \ \Rightarrow\ \exists_t \cdot\ s \xrightarrow{a} t \ \wedge\ (Q,\nu_{\lambda'})\,\mathcal{R}_\theta\,t \tag{1.ii}$$

  Based on the definition of $\mathcal{R}_\lambda$, $(Q,\nu_{\lambda'})\,\mathcal{R}_\theta\,t \Leftrightarrow Q\,\mathcal{R}_{\lambda'}\,t \ \wedge \nu_{\lambda'} = \theta$. Hence, $Q\,\mathcal{R}_\lambda\,t$ holds only in case $\lambda' = \lambda$. Thus, from (1.*i*) and (1.*ii*), the following statement is derived:

$$\forall_{a,Q} \cdot (P \xrightarrow{\phi/a} Q \ \wedge\ \lambda \models \phi) \ \Rightarrow\ \exists_t \cdot\ s \xrightarrow{a} t \ \wedge\ Q\,\mathcal{R}_\lambda\,t \tag{1.iii}$$

  * Next, we assume $P \neq p_{init}$:
  Based on Definition 11, each transition emanating from P such as $P \xrightarrow{\phi/a} Q$, is encoded as a transition in PL-LTS $E(fts)$, i.e.:

$$\forall_{a,Q,\lambda} \cdot\ P \xrightarrow{\phi/a} Q \ \wedge\ P \neq p_{init} \ \wedge\ \lambda \models \phi \Leftrightarrow\ (P,\nu_\lambda) \xrightarrow{a,\nu_\lambda} (Q,\nu_\lambda) \tag{1.iv}$$

  Considering property 1 satisfied by relation $\mathcal{R}_\theta$, $(P,\nu_\lambda)\,\mathcal{R}_\theta\,s$ implies the following statement:

$$\forall_{a,Q,\nu_\lambda} \cdot \left((P,\nu_\lambda)\,\mathcal{R}_\theta\; s \wedge (P,\nu_\lambda) \xrightarrow{a,\nu_\lambda} (Q,\nu_\lambda)\ \wedge\ \nu_\lambda \sqsubseteq \theta\right) \Rightarrow\ \exists_t \cdot\ s \xrightarrow{a} t\ \wedge\ (Q,\nu_\lambda)\,\mathcal{R}_\theta\, t \tag{1.v}$$

Based on the definition of $\mathcal{R}_\lambda$, $(Q,\nu_\lambda)\,\mathcal{R}_\theta\; t \Leftrightarrow Q\,\mathcal{R}_\lambda\; t\ \wedge \nu_\lambda = \theta$. Thus, from (1.*iv*) and (1.*v*), the following statement is derived:

$$\forall_{a,Q} \cdot\ \left(P\,\mathcal{R}_\theta\; s \wedge P \xrightarrow{\phi/a} Q\ \wedge\ \lambda \models \phi\right) \Rightarrow\ \exists_t \cdot\ s \xrightarrow{a} t\ \wedge\ (Q,\nu_\lambda)\,\mathcal{R}_\theta\, t \tag{1.vi}$$

Considering (1.*iii*) and (1.*vi*), the following statement holds:

$$\forall_{P,Q,a,s,\phi} \cdot\ \left(P\,\mathcal{R}_\lambda\; s\ \wedge\ P \xrightarrow{\phi/a} Q \wedge \lambda \models \phi\right) \Rightarrow\ \exists_t \cdot\ s \xrightarrow{a} t\ \wedge\ Q\,\mathcal{R}_\lambda\, t,$$

which means that the relation $\mathcal{R}_\lambda$ satisfies statement (1).

– Next, we prove that $\mathcal{R}_\lambda$ satisfies statement (2).

Considering an arbitrary pair of states in $\mathcal{R}_\lambda$, such as $P\,\mathcal{R}_\lambda\; s$. Based on the definition given for $\mathcal{R}_\lambda$, it holds $(P,\nu)\,\mathcal{R}_\theta\; s$, where $\theta = \nu_\lambda$, and $\nu = \{?\}^I$ if $P = p_{init}$ and $(P,\nu)\,\mathcal{R}_\theta\; s$ where $\theta = \nu_\lambda$ and $\nu = \nu_\lambda$ if $P \neq p_{init}$. For the sake of clarity we distinguish the following two cases: $P = p_{init}$ and $P \neq p_{init}$.

∗ First, we consider the case that $P = p_{init}$:

Based on Definition 11, each transition emanating from P, such as $P \xrightarrow{\phi/a} Q$ is encoded as an outgoing transition from $(p_{init}, \{?\}^I)$, i.e.:

$$\forall_{a,Q,\lambda'} \cdot\ \ P \xrightarrow{\phi/a} Q\ \wedge\ P = p_{init}\ \wedge\ \lambda' \models \phi \Leftrightarrow\ (p_{init}, \{?\}^I) \xrightarrow{a,\nu_{\lambda'}} (Q,\nu_{\lambda'}) \tag{2.i}$$

Considering property 2 of relation $\mathcal{R}_\theta$, $(p_{init}, \{?\})\,\mathcal{R}_\theta\; s$ implies the following statement:

$$\forall_{a,s,t} \cdot\ (s \xrightarrow{a} t)\ \Rightarrow\ \exists_{Q,\nu_{\lambda'}} \cdot\ (p_{init}, \{?\}^I) \xrightarrow{a,\nu_{\lambda'}} (Q,\nu_{\lambda'})\ \wedge\ \nu_{\lambda'} \sqsubseteq \theta\ \wedge\ (Q,\nu_{\lambda'})\mathcal{R}_\theta\, t \tag{2.ii}$$

Based on the definition of $\mathcal{R}_\lambda$, $(Q,\nu_{\lambda'})\,\mathcal{R}_\theta\; t \Leftrightarrow Q\,\mathcal{R}_{\lambda'}\; t\ \wedge \nu_{\lambda'} = \theta$. Hence, $Q\,\mathcal{R}_\lambda\; t$ holds only in case $\lambda' = \lambda$. Thus, from (2.*i*) and (2.*ii*), the following statement is derived:

$$\forall_{a,s,t} \cdot\ \left(P\,\mathcal{R}_\lambda\; s\ \wedge\ s \xrightarrow{a} t\right)\ \Rightarrow\ \exists_{Q,\phi} \cdot\ P \xrightarrow{\phi/a} Q\ \wedge\ \lambda \models \phi \wedge Q\,\mathcal{R}_\lambda\, t \tag{2.iii}$$

∗ Next, we assume $P \neq p_{init}$:

Based on Definition 11, each transition emanating from P, such as $P \xrightarrow{\phi/a} Q$, is encoded as an outgoing transition from P, i.e.:

$$\forall_{a,Q,\lambda} \cdot\ \ P \xrightarrow{\phi/a} Q\ \wedge\ P \neq p_{init}\ \wedge\ \lambda \models \phi \Leftrightarrow\ (P,\nu_\lambda) \xrightarrow{a,\nu_\lambda} (Q,\nu_\lambda) \tag{2.iv}$$

Considering property 2 satisfied by relation $\mathcal{R}_\theta$, $(P,\nu_\lambda)\,\mathcal{R}_\theta\; s$ implies the following statement:

$$\forall_{a,\nu_\lambda,s,t} \cdot\ (s \xrightarrow{a} t)\ \Rightarrow\ \exists_{Q,\nu_\lambda} \cdot\ (P,\nu_\lambda) \xrightarrow{a,\nu_\lambda} (Q,\nu_\lambda)\ \wedge\ \nu_\lambda \sqsubseteq \theta\ \wedge\ (Q,\nu_\lambda)\,\mathcal{R}_\theta\, t \tag{2.v}$$

Based on the definition of $\mathcal{R}_\lambda$, $(Q,\nu_\lambda)\,\mathcal{R}_\theta\; t \Leftrightarrow Q\,\mathcal{R}_\lambda\; t\ \wedge \nu_\lambda = \theta$. Thus, from (2.*iv*) and (2.*v*), the following statement is derived:

$$\forall_{a,s,t} \cdot\ (s \xrightarrow{a} t)\ \Rightarrow\ \exists_{Q,\phi} \cdot\ P \xrightarrow{\phi/a} Q\ \wedge\ \lambda \models \phi \wedge Q\,\mathcal{R}_\lambda\, t \tag{2.vi}$$

Considering two derived statements (2.*iii*) and (2.*vi*), the following statement holds:

$$\forall_{P,a,s,t} \cdot\ \left(P\,\mathcal{R}_\lambda\; s\ \wedge\ s \xrightarrow{a} t\right)\ \Rightarrow\ \exists_{Q,\phi} \cdot\ P \xrightarrow{\phi/a} Q\ \wedge\ \lambda \models \phi \wedge Q\,\mathcal{R}_\lambda\, t$$

Hence, we conclude that $\mathcal{R}_\lambda$ satisfies statement (2).

The assumption $E(fts) \prec l$ implies that $\bar{p}_{init} \vdash_\theta s_{init}$. Based on the above proof, $\mathcal{R}_\lambda$ satisfies the properties of a product derivation relation given in Definition 3. Hence, based on the definition of $\mathcal{R}_\lambda$ it holds that $(p_{init}, \{?\}^I) \vdash_\theta s_{init} \Rightarrow p_{init} \vdash_\lambda s_{init}$. This means $fts \triangleright l$ and subsequently $l \in [\![fts]\!]$.

Hence, we conclude that $[\![E(fts)]\!]' \subseteq [\![fts]\!]$. □

As is shown above $[\![fts]\!] \subseteq [\![E(fts)]\!]'$ and $[\![E(fts)]\!]' \subseteq [\![fts]\!]$. Thus, $[\![fts]\!] = [\![E(fts)]\!]$, which means the class of PL-LTSs is at least as expressive as the family of FTSs. □

In the next theorem, we show that the class of FTSs is at least as expressive as class of PL-LTSs, which is the same result as provided in [9], but based on a modified version of encoding given in the proof of Theorem 3, in [9] (see Definition 12). The subsequent proof is almost identical to the proof of Theorem 3 in [9], as well.

Theorem 3. *The class of FTSs is at least as expressive as the class of PL-LTSs.*

Proof. Consider the encoding from PL-LTSs into FTSs given in Definition 12. The proof for the above theorem remains the same as the proof of theorem 3, in [9] (by considering the modified encoding), which is as follows. Assume $plt = (\mathbb{P} \times \{L, R, ?\}^I, A, I, \rightarrow, p_{init})$ is a PL-LTS and the FTS $E(plt) = (\mathbb{P} \times \{L, R, ?\}^I, A, F, \rightarrow', \Lambda, p_{init})$, is the result of applying encoding E on plt, based on Definition 12. For any $(P, \nu) \in \mathbb{P} \times \{L, R, ?\}^I$, we fix $E(P, \nu) = (P, \nu)$. We need to show that $[\![plt]\!] = [\![E(plt)]\!]'$. We divide the proof obligation into the following two cases:

- $([\![plt]\!] \subseteq [\![E(plt)]\!]')$: Let $l \in [\![plt]\!]$, where $l = (\mathbb{S}, A, \rightarrow, s_{init})$. Then $p_{init} \vdash_\theta s_{init}$, for some $\theta \in \{L, R\}^I$. Define a configuration $\lambda_\theta \in \Lambda$ as follows: $\lambda_\theta(L_i) = \top \iff \theta(i) = L$ and $\lambda_\theta(R_i) = \top \iff \theta(i) = R$. Furthermore, consider the following relation $\mathcal{R}_{\lambda_\theta}$ such that $\forall_{(Q,\nu')\in\mathbb{P}, s\in\mathbb{S}} \cdot (Q, \nu')\mathcal{R}_{\lambda_\theta} s \iff (Q, \nu') \vdash_\theta s$. It is straightforward to show that $\mathcal{R}_{\lambda_\theta}$ is a product derivation relation with regards to Definition 3.

- $([\![(P, \nu)]\!]' \subseteq [\![(P, \nu)]\!])$: Let $l \in [\![E(plt)]\!]'$, where $l = (\mathbb{S}, A, \rightarrow, s_{init})$. Then $p_{init} \vdash_\lambda s_{init}$ for some $\lambda \in \Lambda$. Let $\theta_\lambda \in \{L, R\}^I$ be a configuration vector defined as $\theta_\lambda(i) = L \iff \lambda(L_i) = \top$ and $\theta_\lambda(i) = R \iff \lambda(R_i) = \top$. Define a relation $\mathcal{R}_{\theta_\lambda}$ such that $\forall_{(Q,\nu')\in\mathbb{P}, s\in\mathbb{S}} \cdot (Q, \nu')\mathcal{R}_{\theta_\lambda} s \iff (Q, \nu') \vdash_{\theta_\lambda} s$. It is straightforward to verify that $\mathcal{R}_{\theta_\lambda}$ is a product derivation relation for PL-LTSs (see Definition 8).

□

Based on Theorem 2 and Theorem 3, we give the following corollary.

Corollary 1. *The class of PL-LTSs and the class of FTSs are equally expressive.*

Proof. Considering Theorem 2 the class of PL-LTSs is at least as expressive as the class of FTSs. Based on Theorem 3, the class of FTSs is at least as expressive as the class of PL-LTSs. Hence, considering Definition 10, we conclude that the class of PL-LTSs and the class of FTSs are equally expressive. □

5. Succinctness Analysis

In this section, we provide an analysis of the succinctness (the number of states and the configuration vector size included in the states) of PL-LTSs resulting from encoding FTSs. We prove that for some FTSs the size of the PL-LTS which is resulting from any sound encoding, is exponential in terms of the number of states of the FTS. Furthermore, we show that for each PL-LTS a sound encoding into FTSs exists such that the size of the resulting FTS is linear in terms of the size of PL-LTS. Hence, as a result we conclude that FTSs are in general exponentially more succinct than PL-LTSs. In the rest of this section, we assume that $\mathbb{E}$ denotes all sound encodings from the class of FTSs into the class of PL-LTSs. We consider the FTS *fts* depicted in Fig. 5. In this FTS, in each state s_i there is a variant choice between features f_{x_i} and f_{y_i} i.e, in each valid product either the transition labeled f_{x_i} or the one with f_{y_i} (but not both) must be present. We assume $E(fts) = (\bar{\mathbb{P}}, A, I, \rightarrow, \bar{p}_{init})$ is the PL-LTS resulting from encoding *fts* using an arbitrary encoding $E \in \mathbb{E}$. The FTS *fts* has 2^n non-trace equivalent LTS implementations each of which has exactly one path. We assume *Imp* denotes the set of all such implementing LTSs.

First, we prove the following statement which is used to compute the least possible size of the configuration vector in the states of $E(fts)$, i.e., $\Omega(\mid I \mid)$. Consider two distinct LTS implementations derived from the PL-LTS $E(fts)$; at

least one state in $E(fts)$ exists such that each of the considered LTSs implements a distinct outgoing transition from that state. This in turn means that for each two distinct valid products (implementations) of the above mentioned model, there should be at least one configuration vector corresponding to each of these products in the PL-LTS which is refined by that product's vector such that these configuration vectors are conflicting in at least one bit. We formalize this in terms of the following lemma.

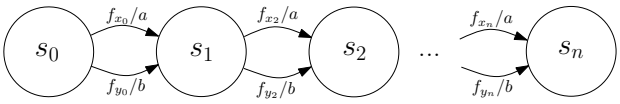

Figure 5. FTS FT.

Lemma 1. *Assuming two non-trace-equivalent LTSs, $l_1, l_2 \in Imp$, such as $l_1 = (\mathbb{S}_1, A, \rightarrow_1, s_0^1)$ and $l_2 = (\mathbb{S}_2, A, \rightarrow_2, s_0^2)$, where $\bar{p}_{init} \vdash_{\theta_1} s_0^1$ and $\bar{p}_{init} \vdash_{\theta_2} s_0^2$. It holds that:*

$$\exists_{P,Q\in\bar{\mathbb{P}},\alpha\in A,\nu,\nu'\in I} \cdot\ (P,\nu) \xrightarrow{\alpha,\nu'} (Q,\nu')\ \wedge\ (\nu' \sqsubseteq \theta_1\ \wedge \nu' \not\sqsubseteq \theta_2) \wedge$$

$$\exists_{P,Q\in\bar{\mathbb{P}},\alpha\in A,\nu,\nu'\in I} \cdot\ (P,\nu) \xrightarrow{\alpha,\nu'} (Q,\nu')\ \wedge\ (\nu' \not\sqsubseteq \theta_1\ \wedge \nu' \sqsubseteq \theta_2)$$

Proof. Assuming that $Paths(l_1) = \rho_1$, $Paths(l_2) = \rho_2$, $\Sigma = pref(Trace(\rho_1)) \cap pref(Trace(\rho_1))$, where $pref(.)$ denotes the set of the finite prefixes of a sequence. We consider $\sigma \in \Sigma$ such that $\nexists_{\sigma'\in\Sigma} \cdot\ |\sigma| < |\sigma'|$ i.e., a maximal trace σ in Σ. Assume $|\sigma| = k$; let $\rho_1(k) = s_k^1$, and $\rho_2(k) = s_k^2$, given that $l_1, l_2 \in Imp$ are distinct it holds that: $s_k^1 \xrightarrow{\alpha}_1 s_{k+1}^1$ and $s_k^2 \xrightarrow{\beta}_2 s_{k+1}^2$ where $\alpha \neq \beta$. Based on the condition (2) in Definition 8, it holds:

$$s_k^1 \xrightarrow{\alpha}_1 s_{k+1}^1 \Rightarrow \exists_{P_1,Q_1,\nu_1,\nu_1'} \cdot (P_1,\nu_1) \xrightarrow{\alpha,\nu_1'} (Q_1,\nu_1')\ \wedge\ \nu_1' \sqsubseteq \theta_1$$

$$s_k^2 \xrightarrow{\beta}_2 s_{k+1}^2 \Rightarrow \exists_{P_2,Q_2,\nu_2,\nu_2'} \cdot (P_2,\nu_2) \xrightarrow{\beta,\nu_2'} (Q_2,\nu_2')\ \wedge\ \nu_2' \sqsubseteq \theta_2,$$

Given that $l_1, l_2 \in Imp$ it holds $|Out(s_k^1)| = 1$ and $|Out(s_k^2)| = 1$; hence $\nu_2' \not\sqsubseteq \theta_2$ and $\nu_1' \not\sqsubseteq \theta_1$ (otherwise, s_k^1 and s_k^2 should have more than one outgoing transitions). Thus, it can be concluded that:

$$\exists_{P,Q,\alpha,\nu,\nu'} \cdot\ (P,\nu) \xrightarrow{\alpha,\nu'} (Q,\nu')\ \wedge\ (\nu' \sqsubseteq \theta_1\ \wedge \nu' \not\sqsubseteq \theta_2) \wedge$$

$$\exists_{P,Q,\alpha,\nu,\nu'} \cdot\ (P,\nu) \xrightarrow{\alpha,\nu'} (Q,\nu')\ \wedge\ (\nu' \not\sqsubseteq \theta_1\ \wedge \nu' \sqsubseteq \theta_2)$$

□

Next, we provide a lower bound for the size of the configuration vector in the states of the PL-LTSs resulting from encoding the FTS represented in Fig. 5.

Lemma 2. *Let $E \in \mathbb{E}$ be an arbitrary encoding. The size of the configuration vector included in the states of $E(fts)$ (i.e., $\Omega(|\ I\ |)$) is at least n.*

Proof. Consider two non-trace-equivalent LTSs $l_1, l_2 \in Imp$, such as $l_1 = (\mathbb{S}_1, A, \rightarrow_1, s_0^1)$ and $l_2 = (\mathbb{S}_2, A, \rightarrow_2, s_0^2)$, where $\bar{p}_{init} \vdash_{\theta_1} s_0^1$ and $\bar{p}_{init} \vdash_{\theta_2} s_0^2$. According to Lemma 1, it holds that:

$$\exists_{P,Q,\alpha,\nu,\nu'} \cdot\ (P,\nu) \xrightarrow{\alpha,\nu'} (Q,\nu')\ \wedge\ (\nu' \sqsubseteq \theta_1\ \wedge \nu' \not\sqsubseteq \theta_2) \wedge$$

$$\exists_{P,Q,\alpha,\nu,\nu'} \cdot\ (P,\nu) \xrightarrow{\alpha,\nu'} (Q,\nu')\ \wedge\ (\nu' \not\sqsubseteq \theta_1\ \wedge \nu' \sqsubseteq \theta_2),$$

which means for any two arbitrary LTSs $l_1, l_2 \in Imp$ it holds that: $\exists_{\rho\in Paths(E(fts))} \cdot last(\rho) = (Q,\nu')\ \wedge\ (\nu' \sqsubseteq \theta_1\ \wedge \nu' \not\sqsubseteq \theta_2)$ and $\exists_{\rho\in Paths(E(fts))} \cdot last(\rho) = (Q,\nu')\ \wedge\ (\nu' \not\sqsubseteq \theta_1\ \wedge \nu' \sqsubseteq \theta_2)$. Thus, $\exists_{(Q_1,\nu_1'),(Q_2,\nu_2')\in\bar{\mathbb{P}}} \cdot\ \nu_1' \bowtie \nu_2'$ (see Definition 7). This means $\forall_{l_1,l_2\in Imp}\ \exists_{(Q_1,\nu_1'),(Q_2,\nu_2')\in\bar{\mathbb{P}}}\ \exists_{i\in I} \cdot \nu_1'(i) \bowtie \nu_2'(i)$, hence for each two products selected from Imp, there are two states in the PL-LTS that the configuration vectors in these states are conflicting. As $|Imp| = 2^n$, the minimum size of the configuration vector included in the state of the PL-LTS is $log(2^n) = n$. □

Next, we prove the following theorem regarding the succinctness of the PL-LTSs resulting from encoding FTSs.

Lemma 3. *Consider the class of all possible encodings from FTSs into PL-LTSs, denoted by $\mathbb{E}$. There exists an FTS such that the size of the encoded PL-LTS (the number of states) is exponential in the number of the states in that FTS, regardless of which encoding is selected.*

Proof. Let $E \in \mathbb{E}$ be an arbitrary encoding and $E(fts) = (\bar{\mathbb{P}}, A, I, \rightarrow, \bar{p}_{init})$ be the PL-LTS resulting from the encoding. Consider two distinct LTSs, $l_1, l_2 \in Imp$, such as $l_1 = (\mathbb{S}_1, A, \rightarrow_1, s_0^1)$ and $l_2 = (\mathbb{S}_2, A, \rightarrow_2, s_0^2)$, where $\bar{p}_{init} \vdash_{\theta_1} s_0^1$ and $\bar{p}_{init} \vdash_{\theta_2} s_0^2$; according to Lemma 1, it holds that:

$$\exists_{P,Q,\alpha,\nu,\nu'} \cdot (P,\nu) \xrightarrow{\alpha,\nu'} (Q,\nu') \ \wedge \ (\nu' \sqsubseteq \theta_1 \ \wedge \nu' \not\sqsubseteq \theta_2) \wedge$$

$$\exists_{P,Q,\alpha,\nu,\nu'} \cdot (P,\nu) \xrightarrow{\alpha,\nu'} (Q,\nu') \ \wedge \ (\nu' \not\sqsubseteq \theta_1 \ \wedge \nu' \sqsubseteq \theta_2),$$

Hence, for each two arbitrary LTSs l_1, l_2 it holds that $\exists_{(Q_1,\nu'_1),(Q_2,\nu'_2)\in\bar{\mathbb{P}}} \cdot \nu'_1 \bowtie \nu'_2$. As $|Imp| = 2^n$ it holds that the size of the set of states in $E(fts)$ is at least 2^n.

Hence, we conclude that the total number of the states in $E(fts)$ is exponential in terms of the number of states in *fts*. □

Next, we prove that each PL-LTS can be encoded into an FTS using a sound encoding such that the size of the FTS is linear in terms of the size of the PL-LTS.

Theorem 4. *An encoding $E \in \mathbb{E}$ from PL-LTSs into FTSs exists such that for any PL-LTS P, the size of the model resulting from the encoding of P is linear in terms of the size of P, i.e., $|E(P)| = O(|P|)$ where $|\ .\ |$ represents the number of states.*

Proof. Let $(\mathbb{P} \times \{L, R, ?\}^I, A, \rightarrow, p_{init})$ be an arbitrary PL-LTS. We consider the encoding given in Definition 12. The corresponding FTS is denoted by $(\mathbb{P} \times \{L, R, ?\}^I, A, F, \rightarrow', \Lambda, p_{init})$. The result of encoding a state in the PL-LTS such as $(P, \nu) \in \mathbb{P} \times \{L, R, ?\}^I$ is state (P, ν) in the FTS. Considering the transition relation given in Definition 12, the result of encoding each transition $(P, \nu) \xrightarrow{a,\nu'} (Q, \nu')$, for either $\nu = \nu'$ or $\nu \neq \nu'$, is one transition in the FTS. Hence, it is straightforward to see that the size of the FTS resulting from the encoding is linear in terms of the size of the original PL-LTS. □

6. Conclusion

In this paper, we compared the expressiveness of the PL-CCSs and FTSs. To this end, we used a more liberal definition for PL-LTSs (which are considered as the semantic domain for PL-CCS terms) in comparison with our previous work [9]. We described an encoding from the class of FTSs into the class of PL-LTSs. Then, we proved that the set of LTSs that implement an FTS, are also valid implementations of the PL-LTS resulting from encoding the FTS and vice versa. Furthermore, we showed that the class of FTSs is at least as expressive as the class of PL-LTSs, which is the same result as provided in [9]. Thus, we conclude that the class of PL-LTSs and the class of FTSs are equally expressive. We also provided a succinctness analysis of the models resulting from the encoding. The results show that for some FTSs the size of the PL-LTS resulting from encoding the FTS (using any sound encoding) is exponential in terms of the number of the states of the FTS. Furthermore, the results show that there exists an encoding from PL-LTSs to FTSs for which the size of the FTS resulting from the encoding is linear in terms of the size of the original PL-LTS. Hence, as a result we conclude that FTSs are in general exponentially more succinct than PL-LTSs.

Both in the present paper and in [9], we have only considered models with finite behavior; considering infinite behavior in our study of comparative expressiveness is among the future work that we aim to pursue. Completing the lattice of expressive power and succinctness given in [9] and in the present paper, by comparing the expressiveness and succinctness of other formalisms, such as MTSs (for succinctness) and their extensions such as 1MTSs [4], DMTSs [5], PMTSs [6] and MTSs with variability constraints [7], and also variations of process algebras such as Variant Process Algebra [13] and DeltaCCS [14] with formalisms included in the lattice, is another avenue for our future work.

Acknowledgments. This work has been supported by grants from the Swedish Knowledge Foundation (Stiftelsen för Kunskaps- och Kompetensutveckling) in the context of the AUTO-CAAS HoGproject (number: 20140312), Swedish Research Council (Vetenskapsradet) award number: 621-2014-5057 (Effective Model-Based Testing of Concurrent Systems), and the ELLIIT Strategic Research Environment.

7. References

[1] A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay, J.-F. Raskin, Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking, Software Engineering, IEEE Transactions on 39 (8) (2013) 1069–1089. doi:10.1109/TSE.2012.86.

[2] A. Gruler, M. Leucker, K. Scheidemann, Modeling and model checking software product lines, in: Proceedings of the Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS '08), Vol. 5051 of Lecture Notes in Computer Science, Springer, 2008, pp. 113–131.

[3] K. Larsen, B. Thomsen, A modal process logic, in: Proc. of the 3rd Annual Symposium on Logic in Computer Science (LICS '88), IEEE, 1988, pp. 203–210.

[4] H. Fecher, H. Schmidt, Comparing disjunctive modal transition systems with an one-selecting variant, Journal of Logic and Algebraic Programming 77 (1-2) (2008) 20–39. doi:http://dx.doi.org/10.1016/j.jlap.2008.05.003.

[5] K. G. Larsen, L. Xinxin, Equation solving using modal transition systems, in: Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), 1990, pp. 108–117. doi:10.1109/LICS.1990.113738.

[6] N. Beneš, J. Křetínský, K. G. Larsen, M. H. Møller, J. Srba, Parametric modal transition systems, in: Proceedings of Automated Technology for Verification and Analysis, ATVA, Springer, Berlin, Heidelberg, 2011, pp. 275–289. doi:10.1007/978-3-642-24372-1-20.

[7] M. H. ter Beek, A. Fantechi, S. Gnesi, F. Mazzanti, 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) (2016) 287 – 315. doi:https://doi.org/10.1016/j.jlamp.2015.11.006.
URL http://www.sciencedirect.com/science/article/pii/S2352220815001431

[8] R. Milner, A Calculus of Communicating Systems, Vol. 92 of Lecture Notes in Computer Science, Springer, 1982.

[9] H. Beohar, M. Varshosaz, M. R. Mousavi, Basic behavioral models for software product lines: Expressiveness and testing pre-orders, Sci. Comput. Program. 123 (2016) 42–60.

[10] K. Kang, S. Cohen, J. Hess, W. Novak, S. Peterson, Feature-Oriented Domain Analysis (FODA) Feasibility Study, Tech. Rep. CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University (1990).

[11] D. Benavides, S. Segura, A. Ruiz-Cortés, Automated analysis of feature models 20 years later: A literature review, Vol. 35, 2010, pp. 615–636.

[12] D. Batory, Feature models, grammars, and propositional formulas, in: Proceedings of the 9th International Conference on Software Product Lines, SPLC'05, 2005, pp. 7–20.

[13] M. Tribastone, Behavioral relations in a process algebra for variants, in: Proceedings of the 18th International Software Product Line Conference - Volume 1, SPLC '14, ACM, New York, NY, USA, 2014, pp. 82–91. doi:10.1145/2648511.2648520.
URL http://doi.acm.org/10.1145/2648511.2648520

[14] M. Lochau, S. Mennicke, H. Baller, L. Ribbeck, Incremental model checking of delta-oriented software product lines, Journal of Logical and Algebraic Methods in Programming 85 (1, Part 2) (2016) 245 – 267, formal Methods for Software Product Line Engineering. doi:https://doi.org/10.1016/j.jlamp.2015.09.004.
URL http://www.sciencedirect.com/science/article/pii/S2352220815000863