# Input-Output Conformance Testing Based on Featured Transition Systems

Harsh Beohar
Center for Research on Embedded Systems
Halmstad University, Sweden
harsh.beohar@hh.se

Mohammad Reza Mousavi
Center for Research on Embedded Systems
Halmstad University, Sweden
m.r.mousavi@hh.se

## ABSTRACT

We extend the theory of input-output conformance testing to the setting of software product lines. In particular, we allow for input-output featured transition systems to be used as the basis for generating test suites and test cases. We introduce refinement operators both at the level of models and at the level of test suites that allow for projecting them into a specific product configuration (or product sub-line). We show that the two sorts of refinement are consistent and lead to the same set of test-cases.

## Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification—*Formal Methods*; D.2.5 [**Software Engineering**]: Testing and Debugging

## Keywords

Model based testing, Input-output conformance testing, Software product lines, Input-output featured transition systems

## 1. INTRODUCTION

### 1.1 Motivation

Software Product Lines (SPLs) have become common practice in software development and have been proven effective in mass production and customization of software. There have been several attempts to provide a structured discipline for testing SPLs. However, it appears from recent surveys [4, 5, 8, 7] that several fundamental approaches to model-based testing (based on finite state machines and labeled transition systems) are not yet fully adapted to and adopted in this domain. The theory of Input-Output Conformance (IOCO) [11], is one such fundamental approach, which uses labeled transition systems for model-based testing. We are not aware of any prior work in adapting the theory of IOCO to cater for variability in SPLs. The present paper addresses this gap by extending IOCO to the setting of SPLs. To this end, we propose Input-Output Featured Transition Systems (IOFTSs) as simple yet expressive behavioral models of SPLs and adapt the traditional IOCO theory to allow for using IOFTS (instead of plain input-output transition system models) as test models for model-based testing. We define the test suite and the test cases that are generated from an IOFTS, which can be used for checking conformance. We define two notions of refinement, one at the level of IOFTSs and another one at the level of test suites, that allow for focusing on particular sets of features and eventually on a particular product. We show that these two refinements interact nicely, in that they lead to the same set of test cases.


*SAC'14* March 24-28, 2014, Gyeongju, Korea.


### 1.2 Running Example

To illustrate the concepts throughout the paper, we formalize various aspects of the following SPL (due to Asirelli et al. [2]) and study its testing in the remainder of this paper.

*Example 1.* We model an SPL for vending machines, which accept one-Euro coins (1e) exclusively for the European market and one-Dollar coins (1d) exclusively for the American market. Then, a user has the choice of adding sugar or nosugar, after which the user is allowed to chose a beverage among coffee, tea, and cappuccino. Furthermore, the following constraints hold on each product:

1. Coffee must be offered by each and every variant of this product line.
2. Cappuccino is served only by the European machines and whenever cappuccino is served, a ring-tone must ring.
3. Tea is an optional feature for both markets.

### 1.3 Organization

In Section 2, we define the notion of input-output featured transition systems as our basic modeling language. In Section 3, a notion of refinement is proposed that allows for projecting the SPL behavior into the behavior of a product or a product sub-line. In Section 4, we define the notions of test suite and test cases. In Section 5, a notion of refinement is given on test suites, which allows for deriving more specific test suites from the more generic ones. In the same section, we show that the above-mentioned notions of refinement (i.e., on models and test suites) are consistent in that they lead to the same set of test cases. In Section 5, we also show that the intensional and extensional notions of conformance testing coincide, i.e., non-conformance can always be established by means of running test-cases. In

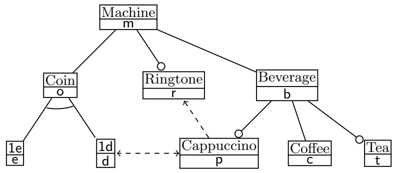



**Figure 1: Vending machine feature diagram [2].**

Section 6, we conclude the paper and present directions for future research.

## 2. INPUT-OUTPUT FEATURED TRANSITION SYSTEMS

Feature diagrams [6, 10] have been used to model variability constraints in SPLs using a graphical notation. A feature diagram represents all valid products of an SPL in terms of features that are arranged hierarchically. Usually, feature diagrams are represented by a directed acyclic graph, of which each node is a feature. There are different kinds of edges between a parent node (feature) and its children (sub-features), namely, the ones representing the mandatory sub-features, and the others representing the optional sub-features. Furthermore, a feature diagram can specify three additional type of constraints on features:

1. Alternative relationship, i.e., the designated sub-features can never be simultaneously present in any product.
2. Exclude relationship, i.e., different features at different levels of hierarchy can never be simultaneously present in any product.
3. Require relationship, i.e., if a feature is present in a product, the related feature should also be present in the same product.

*Example 2.* Consider the feature diagram depicted in Figure 1 [2], which represents the features and the feature constraints of Example 1. In this diagram every machine must consist of features machine (m), coin (o), and beverage (b) and may comprise an optional feature ring-tone (r). The coin feature is further decomposed into two alternative features euro (e) and dollar (d). Furthermore, Figure 1 also specifies that cappuccino (p) requires ring-tone (r), which is denoted by a uni-directional dashed line and cappuccino is absent in the machine that takes dollars, which is represented by a bi-directional dashed line.

A feature diagram only specifies the structural aspects of variability in an SPL; however, to formally analyze the behavior of an SPL, we follow the approach of [3] in annotating the transitions of a labeled transition system with logical constraints on the presence or absence of features; the features used in such logical constraints are assumed to be already specified in a feature diagram. We slightly extend the featured transition system of [3] to cater for the distinction between input and output actions. This is a necessary ingredient for extending the theories of testing, and particularly IOCO, to this setting.

Let $\mathbb{B} = \{\top, \bot\}$ be the set of Boolean constants and let $\mathbb{B}(F)$ be the set of all propositional formulae generated by interpreting the elements of the set $F$ as propositional variables. For instance, in the context of Example 2, formula $\mathsf{e} \wedge \neg\mathsf{d}$ asserts the presence of euro coin and the absence of dollar coin. We let $\varphi, \varphi'$ range over the set $\mathbb{B}(F)$.

*Definition 1.* A *input-output featured transition system* (IOFTS) is a 6-tuple $(S, s, A, F, T, \Lambda)$, where

1. $S$ is the set of *states*,
2. $s \in S$ is the *initial state*,
3. $A = A_{\mathcal{I}} \uplus A_{\mathcal{O}}$ is the set of *actions* partitioned into two disjoint sets of *input* $A_{\mathcal{I}}$ and *output* $A_{\mathcal{O}}$ actions,
4. $F$ is the set of *features*,
5. $T \subseteq S \times A_\tau \times \mathbb{B}(F) \times S$ is the *transition relation* ($A_\tau = A \cup \{\tau\}$ and $\tau \notin A$ is the so-called internal action) satisfying the following condition (for every $s_1, s_2 \in S, a \in A_\tau, \varphi, \varphi' \in \mathbb{B}(F)$):
$$(s_1, a, \varphi, s_2) \in T \wedge (s_1, a, \varphi', s_2) \in T \Rightarrow \varphi = \varphi',$$
6. $\Lambda \subseteq \{\lambda : F \to \mathbb{B}\}$ is the set of *product configurations.*

We write $s \xrightarrow{a}_{\varphi} s'$ to denote an element $(s, a, \varphi, s') \in T$ and drop the subscript $\varphi$ whenever it is clear from the context. We usually refer to an IOFTS by its initial state and graphically denote the initial state by an incoming arrow with no source state. Following the standard notation, we denote the *reachability* relation by $\twoheadrightarrow \ \subseteq S \times A^* \times S$, inductively defined as follows:

$$\frac{}{s \overset{\varepsilon}{\twoheadrightarrow} s} \qquad \frac{s \overset{\sigma}{\twoheadrightarrow} s', s' \xrightarrow{\tau} s''}{s \overset{\sigma}{\twoheadrightarrow} s''} \qquad \frac{s \overset{\sigma}{\twoheadrightarrow} s', s' \xrightarrow{a} s'', a \neq \tau}{s \overset{\sigma a}{\twoheadrightarrow} s''}.$$

The set of *reachable* states from a state $s$ by a trace $\sigma \in A^*$ is denoted by $\mathrm{Reach}(s, \sigma) = \{s' \mid s \overset{\sigma}{\twoheadrightarrow} s'\}$. Furthermore, we fix $\mathrm{Reach}(s) = \{s' \mid \exists_\sigma \ s \overset{\sigma}{\twoheadrightarrow} s'\}$.

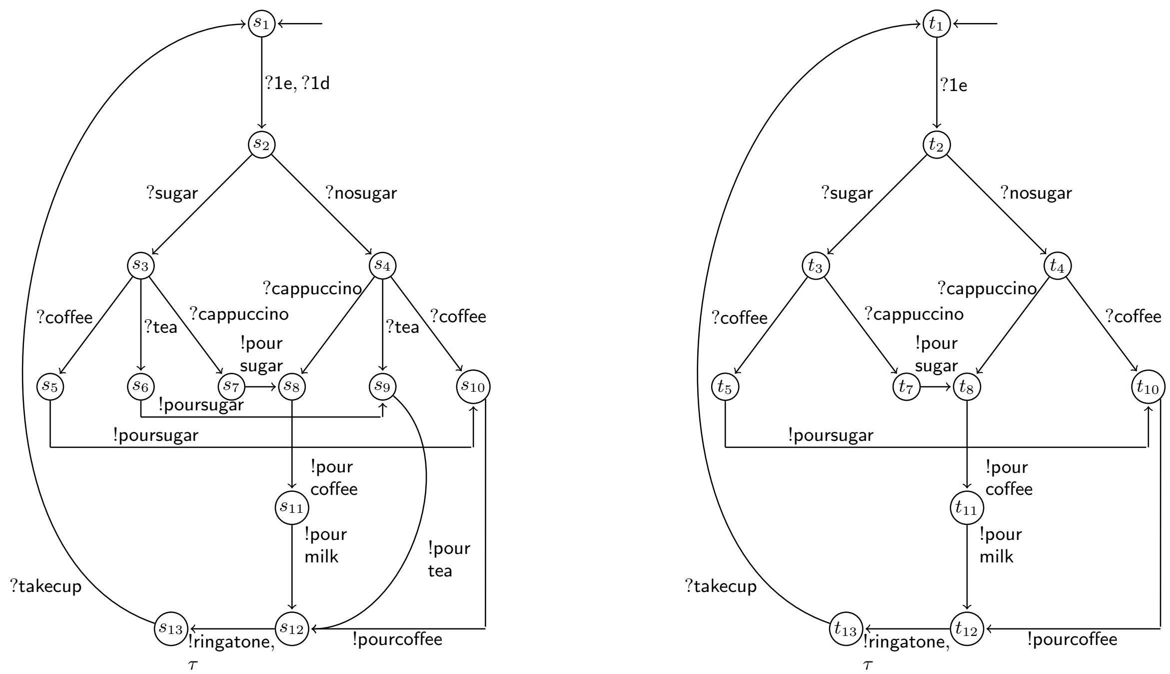
*Example 3.* Consider the FTS in Figure 2(a) with the associated annotation function $\gamma$ defined in the following way.

| Transitions | $\gamma(\_)$ |
|---|---|
| $s_1 \xrightarrow{\mathsf{1e}} s_2$ | e |
| $s_1 \xrightarrow{\mathsf{1d}} s_2$ | d |
| $s_2 \xrightarrow{\mathsf{coffee}} s_5$ | c |
| $s_2 \xrightarrow{\mathsf{tea}} s_6$ | t |

| Transitions | $\gamma(\_)$ |
|---|---|
| $s_2 \xrightarrow{\mathsf{cappuccino}} s_7$ | p |
| $s_{12} \xrightarrow{\mathsf{ringatone}} s_{13}$ | $\mathsf{p} \Rightarrow \mathsf{r}$ |
| remaining transitions | m |

In Figure 2(a), inputs and outputs are prefixed with symbols ? and !, respectively. The transition labeled with !ringatone, $\tau$ stands for two transitions. The set of product configurations of the IOFTS is the following set of 10 products specified by the feature diagram of Example 2 [1]:

$$\begin{aligned}\Lambda =\{&\{\mathsf{m,o,b,c,e}\}, \{\mathsf{m,o,b,c,e,r}\}, \{\mathsf{m,o,b,c,e,t}\},\\ &\{\mathsf{m,o,b,c,e,t,r}\}, \{\mathsf{m,o,b,c,e,p,r}\}, \{\mathsf{m,o,b,c,d}\},\\ &\{\mathsf{m,o,b,c,d,r}\}, \{\mathsf{m,o,b,c,d,t}\},\\ &\{\mathsf{m,o,b,c,d,t,r}\}, \{\mathsf{m,o,b,c,e,p,r,t}\}\}.\end{aligned}$$




(a) The vending machine product line.

(b) The behavior of all products in European market that do not serve tea.

**Figure 2: IOFTSs of the vending machine example [2].**

# 3. REFINEMENT OF MODELS

In [3], a family of operators, parameterized by product configuration, have been introduced to project an FTS into a labeled transition system describing the behavior of a specific product. In this paper, we generalize this approach by defining a family of product derivation operators (parameterized by feature constraints), which project the behavior of an IOFTS into another IOFTS representing a selection of products (a product sub-line).

*Definition 2.* Given a feature constraint $\varphi$ and an IOFTS $\mathcal{T} = (S, s, A, F, T, \Lambda)$, the projection of $\mathcal{T}$ with respect to $\varphi$, denoted by $\Delta_\varphi(\mathcal{T})$, is an IOFTS $(S', \Delta_\varphi(s), A_\delta, F, T', \Lambda')$, where

1. $S' = \{\Delta_\varphi(s') \mid s' \in S\}$ is the set of states,
2. $\Delta_\varphi(s)$ is the initial state,
3. $A_\delta = A \cup \{\delta\}$ is the set of actions, where $\delta$ is the special action label modeling quiescence (the absence of output and internal actions) [11],
4. $T'$ is the smallest relation satisfying the following rules:

$$\frac{\exists_{\lambda \in \Lambda}\ \lambda \models (\gamma(s,a,s') \wedge \varphi)}{\Delta_\varphi(s) \xrightarrow{a}_{\gamma(s,a,s')\wedge\varphi} \Delta_\varphi(s')} \quad (1)$$

$$\frac{\nexists_{\lambda,s',a}\ \lambda \models (\gamma(s,a,s') \wedge \varphi) \wedge a \in A_{\mathcal{O}} \cup \{\tau\}}{\Delta_\varphi(s) \xrightarrow{\delta}_{\varphi} \Delta_\varphi(s)} \quad (2)$$

5. $\Lambda' = \{\lambda \in \Lambda \mid \lambda \models \varphi\}$ is the set of product configurations.

In the above-given rules $\lambda \models \varphi$, denotes that valuation $\lambda$ of features satisfies feature constraint $\varphi$. Intuitively, rule (1) describes the behavior of those valid products that satisfy the feature constraint $\varphi$ in addition to the original annotation of the transition emanating from $s$. Rule (2) models quiescence (the absence of outputs and internal actions) from the state $\Delta_\varphi(s)$. This ability to observe that no outputs are enabled is crucial in defining the input-output conformance relation between a specification $s$ and an implementation $i$ (see Section 4).

*Example 4.* Consider the vending machine product line and suppose we are interested in analyzing the behavior of all products in the European markets that do not serve tea. This can be formulated as $\Delta_\varphi(1)$, where $\varphi = \mathsf{1e} \wedge \neg\mathsf{tea}$ and 1 is the initial state in Figure 2(a). The behavior induced by this feature constraint is given in Figure 2(b). Notice that the one-dollar ($\mathsf{1d}$) transition does not occur at state 1 in Figure 2(b) even though this constraint is unspecified in $\varphi$.

In the sequel, we use the phrase "a feature specification $\Delta_\varphi(s)$" to mean an IOFTS $(\mathrm{Reach}(\Delta_\varphi(s)), \Delta_\varphi(s), A_\delta, F, T, \Lambda)$. Henceforth, we work only with feature specifications (we interpret the original IOFTS of Definition 1 as $\Delta_\top(s_0)$; this has the implicit advantage of always including quiescence in appropriate states). We end this section by the following proposition which states that for any sequence of transitions of the form $\Delta_{\varphi\wedge\varphi'}(s) \overset{\sigma}{\twoheadrightarrow} \Delta_{\varphi\wedge\varphi'}(s')$ such that $\sigma \in A^*$, there exists a sequence of transitions $\Delta_\varphi(s) \overset{\sigma}{\twoheadrightarrow} \Delta_\varphi(s')$ in the feature specification $\Delta_\varphi(s)$.

*Proposition 1.* If $\Delta_{\varphi\wedge\varphi'}(s) \overset{\sigma}{\twoheadrightarrow} \Delta_{\varphi\wedge\varphi'}(s')$ and $\sigma \in A^*$ then $\Delta_\varphi(s) \overset{\sigma}{\twoheadrightarrow} \Delta_\varphi(s')$.

PROOF. Straightforward by induction on $\sigma$, since for any product configurations $\lambda$ of the feature specification $\Delta_{\varphi\wedge\varphi'}(s)$ we have $\lambda \models \varphi \wedge \varphi' \Rightarrow \lambda \models \varphi$. $\square$

# 4. TEST SUITE AND TEST CASES

The ioco testing theory [11] formalizes model-based testing in terms of a conformance relation between a model and a system under test (SUT). This relation can be checked by constantly providing the SUT with inputs that are deemed relevant by the model (expressed as an IOTS: input-output labeled transition system) and observing outputs from the SUT and comparing them with the possible outputs prescribed by the model. The ioco theory is based on the *testing assumption* that the behavior of the system under test can be expressed by an IOTS, which is unknown to the tester. In addition to the above-sketched *extensional* definition of ioco, there is an equivalent *intensional* definition, which relies on comparing the traces of the underlying IOTSs.

In what follows, we first extend the intensional notion of conformance between any two feature specifications. Then, using a novel concept of an ioco test suite (Definition 5), we give an extensional definition of the class of test cases for a given specification $\Delta_\varphi(s)$.

To formally define both the intensional and the extensional notion of input-output conformance (ioco), we require the notion of *suspension traces* in an IOFTS. Informally, a suspension trace is a trace that may contain the action $\delta$ denoting quiescence [11].

*Definition 3.* The set of *suspension traces* of a feature specification $\Delta_\varphi(s)$ is defined as: $\mathrm{Straces}(\Delta_\varphi(s)) = \{\sigma \in {A_\delta}^* \mid \exists_{s'}\ \Delta_\varphi(s) \stackrel{\sigma}{\twoheadrightarrow} \Delta_\varphi(s')\}$.

Intuitively, the ioco relation asserts that the experiments derived from a feature specification (i.e, suspension traces of the specification) and executed on the implementation under test, result in outputs that are always allowed by the specification.

*Definition 4.* An implementation modeled as a feature specification $\Delta_{\varphi'}(s')$ is *input-output conforming* to a specification modeled as a feature specification $\Delta_\varphi(s)$, denoted $\Delta_\varphi(s) \sqsubseteq_{\mathbf{ioco}} \Delta_{\varphi'}(s')$, iff

$$\mathrm{out}(\mathrm{Reach}(\Delta_{\varphi'}(s'), \sigma)) \subseteq \mathrm{out}(\mathrm{Reach}(\Delta_\varphi(s), \sigma)),$$

for every suspension trace $\sigma \in \mathrm{Straces}(\Delta_\varphi(s))$, where $\mathrm{out}(X)$ denotes the set of output enabled from the states in the set $X$, i.e., $\mathrm{out}(X) = \{a \in A_\mathcal{O} \cup \{\delta\} \mid \forall_{s\in X} \exists_{s'} s \stackrel{a}{\twoheadrightarrow} s'\}$.

Conventionally, test cases are defined as deterministic input-output labeled transition systems having finite number of states (with no structure) and certain restrictions on the transitions (see [11, Definition 10]). In this paper, we define them operationally in the sense of [9] by endowing a structure on the states (see Definition 5). This allows for generating a test suite for a product line and refining it into test suites for more specific sub-lines (and eventually generating test cases for a specific product).

*Definition 5.* The *test suite* for an IOFTS $(\Delta_\varphi(S), \Delta_\varphi(s), A_\delta, F, T, \Lambda)$ is an IOFTS $(\mathbf{X}^\varphi_s \cup \{\mathbf{pass}, \mathbf{fail}\}, \hat{\mathbf{X}}^\varphi_s, A_\theta, F, T', \Lambda)$, where

1. $\mathbf{X}^\varphi_s = \big\{ \big(\{s' \mid \Delta_\varphi(s) \stackrel{\sigma}{\twoheadrightarrow} \Delta_\varphi(s')\}, \sigma\big) \mid \sigma \in \mathrm{Straces}(s) \big\}$ is the set of states and $\{\mathbf{pass}, \mathbf{fail}\}$ is the set of so-called *verdict states* [11],
2. $\hat{\mathbf{X}}^\varphi_s = \{(\{s' \mid \Delta_\varphi(s) \stackrel{\varepsilon}{\twoheadrightarrow} \Delta_\varphi(s')\}, \varepsilon)\}$ is the initial state of the test suite,
3. $A_\theta = A \cup \{\theta\}$ is the set of actions with the special label $\theta$ modelling quiescence in the test suite (cf [11]), and
4. the transition relation $T'$ is defined as the smallest relation satisfying the following rules.

$$\frac{X, Y \neq \emptyset \quad (X,\sigma), (Y, \sigma a) \in \mathbf{X}^\varphi_s}{(X,\sigma) \xrightarrow{f(a)}_\varphi (Y, \sigma a)} \quad (3) \qquad \frac{a \in A_\mathcal{O} \cup \{\theta\} \quad (X,\sigma) \xrightarrow{a}_\varphi (Y, \sigma')}{(X,\sigma) \xrightarrow{a}_\varphi \mathbf{pass}} \quad (4)$$

$$\frac{(X,\sigma) \stackrel{a}{\not\to}_\varphi \mathbf{pass}}{(X,\sigma) \xrightarrow{a}_\varphi \mathbf{fail}} \quad (5) \qquad \frac{a \in A_\mathcal{O} \cup \{\theta\}}{\mathbf{pass} \xrightarrow{a}_\varphi \mathbf{pass} \quad \mathbf{fail} \xrightarrow{a}_\varphi \mathbf{fail}} \quad (6)$$

Intuitively, the test suite for a feature specification is an IOFTS (possibly with infinite number of states) which contains all the possible test cases that can be generated. Rule (3) states that if $X$ and $Y$ are nonempty sets of reachable states from $s$ (under feature restriction $\varphi$) with the suspension traces $\sigma$ and $\sigma a$, respectively, then there exists a transition of the form $(X,\sigma) \xrightarrow{f(a)}_\varphi (Y, \sigma a)$ in the test suite. The function $f$ renames the quiescence action label $\delta$ into $\theta$ as done in [11], while the input and output actions are unchanged. Formally, the function $f$ is defined as follows:

$$f(a) = \begin{cases} a, & \text{if } a \neq \delta \\ \theta, & \text{if } a = \delta \end{cases} .$$

Rules (4) and (5) model the successful and unsuccessful observation of outputs and quiescence. Note input actions are not included in rules (4) and (5) because the implementation is assumed to be $A_\mathcal{I}$-input-enabled in ioco testing theory (cf. [11]); hence, they are only covered in rule (3). Rule (6) states that the verdict states contains self-loop for every output action and quiescence.

*Example 5.* Consider the feature specification $\Delta_\varphi(s_1)$ given in Figure 2(a). An illustration of the test suite (up to depth 2) for the specification $\Delta_\varphi(s_1)$ is shown in Figure 3. The edge $(\{s_1\}, \varepsilon) \xrightarrow{A_\mathcal{O}} \mathbf{fail}$ in Figure 3 denotes the transition $(\{s_1\}, \varepsilon) \xrightarrow{a} \mathbf{fail}$ for every output $a \in A_\mathcal{O}$.

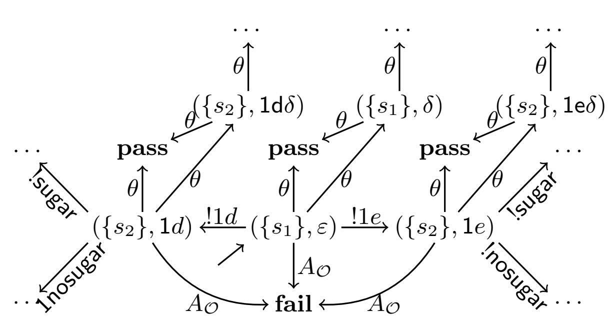



**Figure 3: Test suite of the vending machine**

The following properties are immediate from the rules given in Definition 5.

*Lemma 1.* If $(X, \sigma) \stackrel{\sigma'}{\twoheadrightarrow} (Y, \sigma'')$ then $\sigma' = \sigma''$.

*Lemma 2.* If $(\hat{\mathbf{X}}_s^{\varphi}, \varepsilon) \overset{\sigma}{\twoheadrightarrow} (X, \sigma)$ then

$$\forall_{s'}\ \Delta_\varphi(s) \overset{\sigma}{\twoheadrightarrow} \Delta_\varphi(s') \ \Leftrightarrow\ s' \in X.$$

*Lemma 3.* If $\Delta_\varphi(s) \overset{\sigma}{\twoheadrightarrow} \Delta_\varphi(s')$ for some $s'$ then

$$\exists_X\ (\hat{\mathbf{X}}_s^{\varphi}, \varepsilon) \overset{\sigma}{\twoheadrightarrow} (X, \sigma) \wedge s' \in X.$$

*Lemma 4.* If $(X, \sigma) \overset{\sigma'}{\twoheadrightarrow} (Y, \sigma')$ and $(X, \sigma) \overset{\sigma'}{\twoheadrightarrow} (Z, \sigma')$ then $Y = Z$.

PROOF. Proof of all the above-given lemmata is straightforward by induction on the corresponding trace ($\sigma'$ in Lemmata 1 and 4 and $\sigma$ in Lemmata 2 and 3). $\square$

Next, we formalise the intuition that a test case is a finite projection of a test suite, plus the restriction that at each moment of time at most one input can be fed into the system under test.

*Definition 6.* Given a test suit $\mathcal{T}$, the set of *test cases of* $\mathcal{T}$ *up depth* $n$, denoted by $t_n(\mathcal{T})$, is an IOFTS, of which the transition relation is the minimal relation satisfying both the following deduction rules,

$$\frac{(X, \sigma) \xrightarrow{a}_{\varphi} (Y, \sigma') \wedge |\sigma'| < n}{t_n(X, \sigma) \xrightarrow{a}_{\varphi} t_n(Y, \sigma')} \qquad (7)$$

$$\frac{(X, \sigma) \xrightarrow{a}_{\varphi} \mathcal{Y} \wedge (\mathcal{Y} = \mathbf{pass} \vee \mathcal{Y} = \mathbf{fail})}{t_n(X, \sigma) \xrightarrow{a}_{\varphi} \mathcal{Y}} \qquad (8)$$

and the following *Tretmans' restrictions*:

1. For every reachable state $\mathcal{X}$ such that $t_n(\hat{\mathbf{X}}_s^{\varphi}, \varepsilon) \overset{\sigma}{\twoheadrightarrow} \mathcal{X}$, either $\mathrm{init}(\mathcal{X}) = \{a\} \cup A_{\mathcal{O}}$ (for some $a \in A_{\mathcal{I}}$) or $\mathrm{init}(\mathcal{X}) = A_{\mathcal{O}} \cup \{\theta\}$, where $\mathrm{init}(\mathcal{X}) = \{a \mid \exists_{\mathcal{Y}}\ \mathcal{X} \xrightarrow{a} \mathcal{Y}\}$.

2. For every reachable state $\mathcal{X}$ such that $t_n(\hat{\mathbf{X}}_s^{\varphi}, \varepsilon) \overset{\sigma}{\twoheadrightarrow} \mathcal{X}$, if $\mathcal{X} \xrightarrow{a} \mathbf{pass}$ then $\forall_{\mathcal{Y}}\ \mathcal{X} \xrightarrow{a} \mathcal{Y} \Rightarrow \mathcal{Y} = \mathbf{pass}$.

A *test case* of depth $n$ for a feature specification $\Delta_\varphi(s)$ is $t_n(\hat{\mathbf{X}}_s^{\varphi}, \varepsilon)$.

*Example 6.* Consider the feature specification $\Delta_\varphi(s_1)$ given in Figure 2(a). A test case of depth 1 generated from the test suite of the feature specification $\Delta_\varphi(s_1)$ is shown in Figure 4.

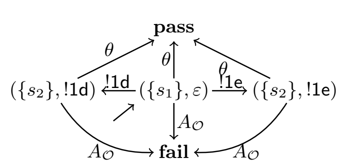



**Figure 4: A test case of the vending machine**

*Proposition 2.* A test case is always deterministic and $A_{\mathcal{O}} \cup \{\theta\}$-input enabled.

*Proposition 3.* A test case has no cycles except those in the verdict states **pass** and **fail**.

Next, we show that the intensional and the extensional notion of testing coincides. To do so, we recall the definition of the *synchronous parallel composition* operator $\rceil\!\lceil$ that allows us to model a test run on an implementation (cf. [11]). This synchronous parallel composition operator $\rceil\!\lceil$ is defined over a test suite and an IOFTS (the intended implementation) as follows. Note that the calligraphic letters $\mathcal{X}, \mathcal{Y}$ in the following rules range over the states of a test suite.

$$\frac{\mathcal{X} \xrightarrow{a} \mathcal{Y} \quad \Delta_\varphi(s) \xrightarrow{a} \Delta_\varphi(s') \quad a \in A}{\mathcal{X} \rceil\!\lceil \Delta_\varphi(s) \xrightarrow{a}_{\top} \mathcal{Y} \rceil\!\lceil \Delta_\varphi(s')} \qquad (9)$$

$$\frac{\Delta_\varphi(s) \xrightarrow{\tau} \Delta_\varphi(s')}{\mathcal{X} \rceil\!\lceil \Delta_\varphi(s) \xrightarrow{\tau}_{\top} \mathcal{X} \rceil\!\lceil \Delta_\varphi(s')} \qquad (10)$$

$$\frac{\mathcal{X} \xrightarrow{\theta} \mathcal{Y} \quad \Delta_\varphi(s) \xrightarrow{\delta} \Delta_\varphi(s')}{\mathcal{X} \rceil\!\lceil \Delta_\varphi(s) \xrightarrow{\theta}_{\top} \mathcal{Y} \rceil\!\lceil \Delta_\varphi(s')} \qquad (11)$$

By having a notion of running a test suite on a feature specification (modeling the intended implementation), we can now define what it means for a feature specification to pass the test suite. Informally, a test suite is passed by a feature specification if and only if every synchronous interaction between the test suite and the feature specification does not leads to the **fail** verdict state.

*Definition 7.* A feature specification $\Delta_{\varphi'}(s')$ passes the test suite $(\hat{\mathbf{X}}_s^{\varphi}, \varepsilon)$ iff

$$\forall_{\sigma \in A_\theta{}^*, s''} \nexists_{\mathcal{X}}\ (\hat{\mathbf{X}}_s^{\varphi}, \varepsilon) \rceil\!\lceil \Delta_{\varphi'}(s') \overset{\sigma}{\twoheadrightarrow} \mathcal{X} \rceil\!\lceil \Delta_{\varphi'}(s'') \wedge \mathcal{X} = \mathbf{fail}$$

Next we prove that the intensional and the internal characterization of the $\sqsubseteq_{\mathbf{ioco}}$ relation coincide, i.e., $\sqsubseteq_{\mathbf{ioco}}$ can always be checked by means of the generated test suite.

*Theorem 1.* $\Delta_\varphi(s) \sqsubseteq_{\mathbf{ioco}} \Delta_{\varphi'}(s')$ iff $\Delta_{\varphi'}(s')$ passes the test suite $(\hat{\mathbf{X}}_s^{\varphi}, \varepsilon)$.

PROOF SKETCH. ($\Leftarrow$) Suppose the feature specification $\Delta_{\varphi'}(s')$ passes the test suite $(\hat{\mathbf{X}}_s^{\varphi}, \varepsilon)$. Then, we show by contradiction that $\Delta_\varphi(s) \sqsubseteq_{\mathbf{ioco}} \Delta_{\varphi'}(s')$ holds. So suppose $a \in \mathrm{out}(\mathrm{Reach}(\Delta_{\varphi'}(s'), \sigma))$ and let $a \notin \mathrm{out}(\mathrm{Reach}(\Delta_\varphi(s), \sigma))$, for some $\sigma \in \mathrm{Straces}(\Delta_\varphi(s)), a \in A_{\mathcal{O}}$. Then, $\exists_{\mathcal{X}}\ (\hat{\mathbf{X}}_s^{\varphi}, \varepsilon) \overset{\sigma a}{\twoheadrightarrow} \mathcal{X} \wedge \mathcal{X} = \mathbf{fail}$. But, $\Delta_{\varphi'}(s')$ passes the test suite $(\hat{\mathbf{X}}_s^{\varphi}, \varepsilon)$, i.e., $\nexists_{\mathcal{X}}\ (\hat{\mathbf{X}}_s^{\varphi}, \varepsilon) \overset{\sigma a}{\twoheadrightarrow} \mathcal{X} \wedge \mathcal{X} = \mathbf{fail}$, which leads to a contradiction.

($\Rightarrow$) Suppose $\Delta_\varphi(s) \sqsubseteq_{\mathbf{ioco}} \Delta_{\varphi'}(s')$. Then we prove by contradiction that the feature specification $\Delta_{\varphi'}(s')$ passes the test suite $(\hat{\mathbf{X}}_s^{\varphi}, \varepsilon)$. Wlog, suppose that $\exists_X\ (\hat{\mathbf{X}}_s^{\varphi}, \varepsilon) \rceil\!\lceil \Delta_{\varphi'}(s') \overset{\sigma}{\twoheadrightarrow} (X, \sigma) \rceil\!\lceil \Delta_{\varphi'}(s_1') \xrightarrow{a} \mathbf{fail} \rceil\!\lceil \Delta_{\varphi'}(s_2')$, for some $\sigma, s_1', s_2'$, and $a \in A_{\mathcal{O}}$. Clearly, $\sigma \in \mathrm{Straces}(\Delta_\varphi(s))$, $a \notin \mathrm{out}(\mathrm{Reach}(\Delta_\varphi(s), \sigma))$, and $a \in \mathrm{out}(\mathrm{Reach}(\Delta_{\varphi'}(s'), \sigma))$. But $\Delta_\varphi(s) \sqsubseteq_{\mathbf{ioco}} \Delta_{\varphi'}(s')$ implies that $\mathrm{out}(\mathrm{Reach}(\Delta_{\varphi'}(s'), \sigma)) \subseteq \mathrm{out}(\mathrm{Reach}(\Delta_\varphi(s), \sigma))$ which again leads to a contradiction. $\square$

# 5. REFINEMENT OF TEST SUITES

In this section, we define the notion of refinement on test suites, to project them into more specific product sub-lines and eventually into products. As the main result of this section, we show that the two notion of refinements (the one on IOFTS as models defined in Section 2 and the other defined in this section) are consistent. More precisely, we show that restricting a test suite of the feature specification $\Delta_\varphi(s)$ by a feature constraint $\varphi'$ is isomorphic to the test suite of the feature specification $\Delta_{\varphi \wedge \varphi'}(s)$.

*Definition 8.* Two states $\mathcal{X}, \mathcal{Y}$ are *isomorphic*, denoted $\mathcal{X} \cong \mathcal{Y}$, if there exists a bijection $f : \mathrm{Reach}(\mathcal{X}) \rightarrow \mathrm{Reach}(\mathcal{Y})$ such that $f$ preserves the transition structure, i.e.,

$$\forall_{\mathcal{X}_1, \mathcal{X}_2 \in \mathrm{Reach}(\mathcal{X}), a} \ \mathcal{X}_1 \xrightarrow{a} \mathcal{X}_2 \Leftrightarrow f(\mathcal{X}_1) \xrightarrow{a} f(\mathcal{X}_2).$$

Nest, we introduce the projection operator $\Delta^{\mathbf{t}}_{\varphi}$ that restricts the behavior of the test suite of the feature specification $\Delta_{\varphi}(s)$ by $\varphi'$.

*Definition 9.* Let $(\mathbf{X}^{\varphi}_s \cup \{\mathbf{pass}, \mathbf{fail}\}, \hat{\mathbf{X}}^{\varphi}_s, A_{\theta}, F, T', \Lambda)$ be the test suite for a feature specification $\Delta_{\varphi}(s)$. For a feature constraint $\varphi'$, the *test-projection operator* $\Delta^{\mathbf{t}}_{\varphi'}(\_)$ induces an IOFTS $(\Delta^{\mathbf{t}}_{\varphi'}(\mathbf{X}^{\varphi}_s) \cup \{\mathbf{pass}, \mathbf{fail}\}, \Delta^{\mathbf{t}}_{\varphi'}(\hat{\mathbf{X}}^{\varphi}_s), A_{\theta}, F, T', \Lambda')$, where the transition relation $T'$ is defined as the smallest relation satisfying the following rules.

$$\frac{(X,\sigma) \xrightarrow{a}_{\varphi} (Y,\sigma') \quad \lambda \models \varphi'}{\Delta^{\mathbf{t}}_{\varphi'}(X,\sigma) \xrightarrow{a}_{\varphi} \Delta^{\mathbf{t}}_{\varphi'}(Y,\sigma')} \quad (12)$$

$$\frac{\begin{array}{c} a \in A_{\mathcal{O}} \cup \theta \\ \Delta^{\mathbf{t}}_{\varphi'}(X,\sigma) \xrightarrow{a}_{\varphi} \Delta^{\mathbf{t}}_{\varphi'}(Y,\sigma') \end{array}}{\Delta^{\mathbf{t}}_{\varphi'}(X,\sigma) \xrightarrow{a}_{\varphi} \mathbf{pass}} \quad (13) \qquad \frac{\Delta^{\mathbf{t}}_{\varphi'}(X,\sigma) \not\xrightarrow{a}_{\varphi} \mathbf{pass}}{\Delta^{\mathbf{t}}_{\varphi'}(X,\sigma) \xrightarrow{a}_{\varphi} \mathbf{fail}} \quad (14)$$

$$\frac{\begin{array}{c} (X,\sigma) \xrightarrow{\theta}_{\varphi} \mathbf{fail} \\ \exists_{s' \in X} \ \Delta_{\varphi \wedge \varphi'}(s) \overset{\sigma}{\twoheadrightarrow} \Delta_{\varphi \wedge \varphi'}(s') \xrightarrow{\delta} \Delta_{\varphi \wedge \varphi'}(s') \end{array}}{\Delta^{\mathbf{t}}_{\varphi'}(X,\sigma) \xrightarrow{\theta}_{\varphi} \Delta^{\mathbf{t}}_{\varphi'}(X,\sigma\delta)} \quad (15)$$

The component $\Lambda'$ is defined as $\Lambda' = \{\lambda \in \Lambda \mid \lambda \models \varphi\}$.

Intuitively, rule (12) states that if an $a$-transition can be executed in the test suite for the specification $\Delta_{\varphi}(s)$ (i.e., $(X,\sigma) \xrightarrow{a} (Y,\sigma a)$) and there exists a product configuration in the test suite that satisfies $\varphi \wedge \varphi'$ then the $a$-transition can be executed in the restricted test suite. Rules (13) and (14) model the successful and the unsuccessful observations of outputs and quiescence.

Rule (15) handles the situation when new quiescence is detected due to the feature constraint $\varphi'$. Namely, a disabled action at a state of $\Delta_{\varphi}(s)$ will remain disabled at the corresponding state in $\Delta_{\varphi \wedge \varphi'}(s)$; however, a non-quiescent state of $\Delta_{\varphi}(s)$ can become a quiescent state of $\Delta_{\varphi \wedge \varphi'}(s)$ after the restriction under $\varphi'$. Consequently, such states in $\Delta_{\varphi \wedge \varphi'}(s)$ will have new suspension traces that are not present in $\Delta_{\varphi}(s)$.

*Example 7.* Consider an IOFTS $(\{s, s'\}, \{a\}, \{f\}, \{s \xrightarrow{a}_f s'\}, \{\lambda \mid \lambda \models f\})$ with $a \in A_{\mathcal{O}}$. Clearly, when we restrict the state $s$ with a feature constraint $\neg f$, the $a$-transition will become disabled. As a result, the transition $\Delta_{f \wedge \neg f}(s) \xrightarrow{\delta} \Delta_{f \wedge \neg f}(s)$ will cause new suspension traces $\delta^*$ that were never possible from the state $\Delta_f(s)$.

We now prove some properties on the restricted test suite of the specification $\Delta_{\varphi}(s)$ under $\varphi'$. Lemma 5 is similar to Lemma 4, which states that a unique state is always reachable for every trace in the restricted test suite.

*Lemma 5.* If $\Delta^{\mathbf{t}}_{\varphi'}(\hat{\mathbf{X}}^{\varphi}_s, \varepsilon) \overset{\sigma}{\twoheadrightarrow} \Delta^{\mathbf{t}}_{\varphi'}(X,\sigma)$, $\Delta^{\mathbf{t}}_{\varphi'}(\hat{\mathbf{X}}^{\varphi}_s, \varepsilon) \overset{\sigma}{\twoheadrightarrow} \Delta^{\mathbf{t}}_{\varphi'}(Y,\sigma)$ then $X = Y$.

Proof. We prove this lemma by induction on $\sigma$. The base case, when $\sigma = \varepsilon$, is trivial because there are no $\tau$

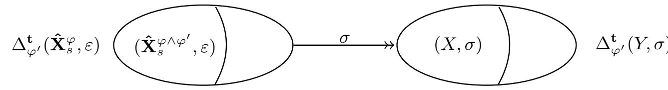



**Figure 5: An illustration of Lemma 6**

steps present in a test suite. For the inductive case, assume the following transitions

$$\begin{array}{l} \Delta^{\mathbf{t}}_{\varphi'}(X,\sigma) \xrightarrow{a} \Delta^{\mathbf{t}}_{\varphi'}(Y,\sigma') \\ \Delta^{\mathbf{t}}_{\varphi'}(X,\sigma) \xrightarrow{a} \Delta^{\mathbf{t}}_{\varphi'}(Z,\sigma'), \end{array} \quad (1)$$

for some $\Delta^{\mathbf{t}}_{\varphi'}(X,\sigma) \in \mathrm{Reach}(\Delta^{\mathbf{t}}_{\varphi'}(\hat{\mathbf{X}}^{\varphi}_s, \varepsilon))$. We observe that if $a \in A$ then the two transitions in (1) are due to the application of rule (12), i.e., $(X,\sigma) \xrightarrow{a} (Y,\sigma')$ and $(X,\sigma) \xrightarrow{a} (Z,\sigma')$. Furthermore, from Lemma 4 it follows that $Y = Z$. However, when $a = \theta$ the two transitions in (1) can be the result of rule (12) or (15) or mix of both. To this end, we show that if $\Delta^{\mathbf{t}}_{\varphi'}(X,\sigma) \xrightarrow{\theta} \Delta^{\mathbf{t}}_{\varphi'}(Y,\sigma')$ is due to the application of rule (12) then the transition $\Delta^{\mathbf{t}}_{\varphi'}(X,\sigma) \xrightarrow{\theta} \Delta^{\mathbf{t}}_{\varphi'}(Z,\sigma')$ is also due to the application of rule (12).

Assume otherwise, the transitions $\Delta^{\mathbf{t}}_{\varphi'}(X,\sigma) \xrightarrow{\theta} \Delta^{\mathbf{t}}_{\varphi'}(Y,\sigma')$ and $\Delta^{\mathbf{t}}_{\varphi'}(X,\sigma) \xrightarrow{\theta} \Delta^{\mathbf{t}}_{\varphi'}(Z,\sigma')$ are due to the application of rules (12) and (15), respectively. From rule (15) we have $(X,\sigma) \xrightarrow{\theta} \mathbf{fail}$, which further implies that $\nexists_{Y,\sigma'} \ (X,\sigma) \xrightarrow{\theta} (Y,\sigma')$. But, from rule (12) we have $\exists_{Y,\sigma'} \ (X,\sigma) \xrightarrow{\theta} (Y,\sigma')$, which is clearly a contradiction. A similar contradiction can be obtained if the transitions in (1) are due to rules (15) and (12), respectively. Thus, the application of rules (12) and (15) is mutually exclusive.

Lastly, if the transitions in (1) is due to rule (12), then from Lemma 4 we can conclude that $Y = Z$ (just like in the case of $a \in A$). On the other hand, if the transitions in (1) is due to rule (15) then the result $X = Y = Z$ follows directly from the conclusion of rule (15). $\square$

Lemma 6 states that any reachable state in the test suite of the specification $\Delta_{\varphi \wedge \varphi'}(s)$ is a subset of a reachable state in the restricted test suite (see Figure 5 for an illustration, where the subset relationship is denoted by the partition).

*Lemma 6.* If $(\hat{\mathbf{X}}^{\varphi \wedge \varphi'}_s, \varepsilon) \overset{\sigma}{\twoheadrightarrow} (X,\sigma)$ then

$$\exists_Y \ \Delta^{\mathbf{t}}_{\varphi'}(\hat{\mathbf{X}}^{\varphi}_s, \varepsilon) \overset{\sigma}{\twoheadrightarrow} \Delta^{\mathbf{t}}_{\varphi'}(Y,\sigma) \wedge X \subseteq Y.$$

Proof. We prove this lemma by induction on $\sigma$. We identify the following cases:

1. Let $\sigma = \varepsilon$. We need to show that $\hat{\mathbf{X}}^{\varphi \wedge \varphi'}_s \subseteq \hat{\mathbf{X}}^{\varphi}_s$.

$$\begin{array}{ll} s' \in \hat{\mathbf{X}}^{\varphi \wedge \varphi'}_s & \text{(Assumption)} \\ \Rightarrow \Delta_{\varphi \wedge \varphi'}(s) \overset{\varepsilon}{\twoheadrightarrow} \Delta_{\varphi \wedge \varphi'}(s') & \text{(Lemma 2)} \\ \Rightarrow \Delta_{\varphi}(s) \overset{\varepsilon}{\twoheadrightarrow} \Delta_{\varphi}(s') & \text{(Proposition 1)} \\ \Rightarrow s' \in \hat{\mathbf{X}}^{\varphi}_s & \text{(Lemma 2)} \ . \end{array}$$

2. Let $\sigma \neq \varepsilon$. Suppose $(\hat{\mathbf{X}}^{\varphi \wedge \varphi'}_s, \varepsilon) \overset{\sigma}{\twoheadrightarrow} (X,\sigma) \xrightarrow{a} (X',\sigma a)$. By the induction hypothesis we have

$$\exists_Y \ \Delta^{\mathbf{t}}_{\varphi'}(\hat{\mathbf{X}}^{\varphi}_s, \varepsilon) \overset{\sigma}{\twoheadrightarrow} \Delta^{\mathbf{t}}_{\varphi'}(Y,\sigma) \wedge X \subseteq Y.$$

We identify the following cases based on the type of $a$.

$$
\begin{tikzcd}
\Delta_{\varphi}(s) \arrow[rr, "\text{test generation}"] \arrow[d, "\Delta_{\varphi'}(\_)"'] & & (\hat{\mathbf{X}}_s, \varepsilon) \arrow[d, "\Delta^{\mathbf{t}}_{\varphi'}(\_)"] \\
\Delta_{\varphi\wedge\varphi'}(s) \arrow[rr, "\text{test generation}"'] & & (\hat{\mathbf{X}}_s^{\varphi\wedge\varphi'}, \varepsilon) \cong \Delta^{\mathbf{t}}_{\varphi'}(\hat{\mathbf{X}}_s, \varepsilon)
\end{tikzcd}
$$

**Figure 6: An illustration of Theorem 2**

(a) Let $a \in A$. Then, by rule (1) we have $X, X' \neq \emptyset$. Furthermore,

$$\exists_{s_1 \in X, s_2 \in X'} \Delta_{\varphi\wedge\varphi'}(s_1) \xrightarrow{a} \Delta_{\varphi\wedge\varphi'}(s_2)$$
$$\Rightarrow s_1 \in Y \wedge \Delta_{\varphi}(s_1) \xrightarrow{a} \Delta_{\varphi}(s_2)$$
$$(X \subseteq Y \text{ and Proposition 1})$$
$$\Rightarrow \exists_{Y'}\ (Y, \sigma') \xrightarrow{a} (Y', \sigma' a) \wedge s_2 \in Y' (\text{Lemma 3}).$$

Next, we need to show that $X' \subseteq Y'$. Let $s'_2 \in X'$, for some $s'_2 \in S$. Then there is a transition $\Delta_{\varphi\wedge\varphi'}(s_1) \xrightarrow{a} \Delta_{\varphi\wedge\varphi'}(s'_2)$, for some $s_1 \in X$. And from Proposition 1 we get $\Delta_{\varphi}(s_1) \xrightarrow{a} \Delta_{\varphi}(s'_2)$. But, $X \subseteq Y$ and from Lemma 2 we have $s'_2 \in Y'$; whence, $X' \subseteq Y'$.

(b) Let $a = \theta$. Then we have $\Delta_{\varphi\wedge\varphi'}(s_1) \xrightarrow{\delta} \Delta_{\varphi\wedge\varphi'}(s_1)$, for some $s_1 \in X, s_1 \in X'$. Now there are two possibilities:

i. Either $\exists_{s_1 \in Y}\ \Delta_{\varphi}(s_1) \xrightarrow{\delta} \Delta_{\varphi}(s_1)$. The remainder of the proof is similar to the previous case.

ii. Or, $\nexists_{s_1 \in Y}\ \Delta_{\varphi}(s_1) \xrightarrow{\delta} \Delta_{\varphi}(s_1)$. Then, $(Y, \sigma') \xrightarrow{\theta}$ **fail**. Then from rule (15) we get $\Delta^{\mathbf{t}}_{\varphi'}(Y, \sigma) \xrightarrow{\theta} \Delta^{\mathbf{t}}_{\varphi'}(Y, \sigma\delta)$. Furthermore, if $s_2 \in X'$ then $s_2 \in X$ since $\Delta_{\varphi\wedge\varphi'}(s_2) \xrightarrow{\delta} \Delta_{\varphi\wedge\varphi'}(s_2)$. Thus, $X' \subseteq X$ and from the induction hypothesis we have $X \subseteq Y$; whence, $X' \subseteq Y$. ☐

*Lemma 7.* If $\Delta^{\mathbf{t}}_{\varphi'}(\hat{\mathbf{X}}^{\varphi}_s, \varepsilon) \overset{\sigma}{\twoheadrightarrow} \Delta^{\mathbf{t}}_{\varphi'}(X, \sigma)$ then

$$\forall_{s'}\ \Delta_{\varphi\wedge\varphi'}(s) \overset{\sigma}{\twoheadrightarrow} \Delta_{\varphi\wedge\varphi'}(s') \Rightarrow s' \in X.$$

We are now ready to prove the main result (Figure 6) of this section which states restricting a test suite leads to an isomorphic test suite by restricting a feature specification.

*Theorem 2.* $\Delta^{\mathbf{t}}_{\varphi'}(\hat{\mathbf{X}}^{\varphi}_s, \varepsilon) \cong (\hat{\mathbf{X}}^{\varphi\wedge\varphi'}_s, \varepsilon)$.

Proof. To show this isomorphism, we define the function $g : \text{Reach}(\Delta^{\mathbf{t}}_{\varphi'}(\hat{\mathbf{X}}^{\varphi}_s), \varepsilon) \to \text{Reach}(\hat{\mathbf{X}}^{\varphi\wedge\varphi'}_s, \varepsilon)$ as follows:
$g(\Delta^{\mathbf{t}}_{\varphi'}(X, \sigma)) = (Y, \sigma)$ if

$$\Delta^{\mathbf{t}}_{\varphi'}(\hat{\mathbf{X}}^{\varphi}_s, \varepsilon) \overset{\sigma}{\twoheadrightarrow} \Delta^{\mathbf{t}}_{\varphi'}(X, \sigma) \Rightarrow (\hat{\mathbf{X}}^{\varphi\wedge\varphi'}_s, \varepsilon) \overset{\sigma}{\twoheadrightarrow} (Y, \sigma);$$

$g(\mathbf{pass}) = \mathbf{pass}$; and $g(\mathbf{fail}) = \mathbf{fail}$. The function $g$ is well-defined follows from Lemma 4. The injectivity of $g$ follows from Lemma 5. Furthermore, $g$ is surjective follows from Lemmas 4 and 6.

Next, we show that $g$ preserves the transition structure. Let $\mathcal{X} \xrightarrow{a} \mathcal{Y}$, for some $\mathcal{X}, \mathcal{Y} \in \text{Reach}(\Delta^{\mathbf{t}}_{\varphi'}(\hat{\mathbf{X}}^{\varphi}_s, \varepsilon))$. The case when $\mathcal{X}$ is either **pass** or **fail** is trivial. However, the interesting case is when $\mathcal{X} = \Delta^{\mathbf{t}}_{\varphi'}(X, \sigma)$. We further identify the following cases:

1. Let $\mathcal{Y} = \Delta^{\mathbf{t}}_{\varphi'}(Y, \sigma')$. Then, from Lemma 7 we know that $\sigma' \in \text{Straces}(\Delta_{\varphi\wedge\varphi'}(s))$; thus, there exists $Y'$ such that $(\hat{\mathbf{X}}^{\varphi\wedge\varphi'}_s, \varepsilon) \overset{\sigma'}{\twoheadrightarrow} (Y', \sigma')$. And by the construction of $g$ we have $g(\mathcal{Y}) = (Y', \sigma')$. For the converse, suppose $g(\mathcal{X}) \xrightarrow{a} (Y', \sigma')$, for some $(Y', \sigma') \in \text{Reach}(\hat{\mathbf{X}}^{\varphi}_s, \varepsilon)$. From Lemmas 5 and 6 we conclude $g(\mathcal{Y}) = (Y', \sigma')$.

2. Let $\mathcal{Y} = \mathbf{pass}$. Then,

$$
\begin{aligned}
&\mathcal{X} \xrightarrow{a} \mathbf{pass} \\
&\Leftrightarrow \exists_{Y,\sigma'} \mathcal{X} \xrightarrow{a} \Delta^{\mathbf{t}}_{\varphi'}(Y, \sigma') && \text{(rule (13))} \\
&\Leftrightarrow g(\mathcal{X}) \xrightarrow{a} g(\Delta^{\mathbf{t}}_{\varphi'}(Y, \sigma')) && \text{(Case 1)} \\
&\Leftrightarrow g(\mathcal{X}) \xrightarrow{a} \mathbf{pass} && \text{(rule (4))}.
\end{aligned}
$$

3. Let $\mathcal{Y} = \mathbf{fail}$. Suppose otherwise $g(\mathcal{X}) \xrightarrow{a} \mathbf{pass}$. Then, from rule (4) we know that there exists $Y', \sigma'$ such that $g(\mathcal{X}) \xrightarrow{a} (Y', \sigma')$. And by Lemma 6 we have $\exists Y\ \mathcal{X} \xrightarrow{a} (Y, \sigma)$. But, $\mathcal{X} \xrightarrow{a} \mathbf{fail}$; hence, a contradiction.

   For the converse, suppose $\mathcal{X} \xrightarrow{a} \mathbf{pass}$ and $g(\mathcal{X}) \xrightarrow{a} \mathbf{fail}$. Then, from rule (13) we know that there exists $Y, \sigma'$ such that $\mathcal{X} \xrightarrow{a} \Delta^{\mathbf{t}}_{\varphi'}(Y, \sigma')$. And from Case 1 we know that $g(\mathcal{X}) \xrightarrow{a} g(Y, \sigma')$, which again leads to a contradiction. ☐

# 6. CONCLUSIONS

In this paper, we extended the notion of input-output conformance testing to the setting of product line, by allowing for models that are annotated with feature constraints. Such models are called input-output featured transition systems. In addition to the theory of conformance testing, we defined notions of refinement both on models and on test suite that allow for projecting, respectively, the behavior and the test suites into a specific set of features and eventually into a specific product.

We have two main items in our research agenda in this area: we would like to extend our theoretical framework to allow for coordinated and incremental testing of various products such that the effort in testing common features is factored out as much as possible. Secondly, we would like to implement our theoretical framework and perform empirical research on its effectiveness and efficiency.

# 7. REFERENCES


[1] P. Asirelli, M. H. Beek, A. Fantechi, and S. Gnesi. A compositional framework to derive product line behavioural descriptions. In *Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change*, volume 7609 of *LNCS*, pages 146–161. Springer, 2012.

[2] P. Asirelli, M. H. ter Beek, S. Gnesi, and A. Fantechi. Formal description of variability in product families. In *Proc. of 15th International Software Product Line Conference*, pages 130–139. IEEE, 2011.

[3] A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay, and J.-F. Raskin. Featured transition systems: Foundations for verifying variability-intensive systems and their application to LTL model checking (to appear). *IEEE Trans Software Eng (TSE)*, 2012.

[4] P. A. da Mota Silveira Neto, I. do Carmo Machado, J. D. McGregor, E. S. de Almeida, and S. R. de Lemos Meira. A systematic mapping study of software product lines testing. *Inf. Softw. Technol.*, 53(5):407–423, 2011.

[5] E. Engström and P. Runeson. Software product line testing - a systematic mapping study. *Information & Software Technology*, 53(1):2–13, 2011.

[6] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, 1990.

[7] B. P. Lamancha, M. P. Usaola, and M. P. Velthius. Systematic review on software product line testing. In *Software and Data Technologies*, volume 170 of *Comm. in Computer and Information Science*, pages 58–71. Springer, 2013.

[8] S. Oster, A. Wübbeke, G. Engels, and A. Schürr. Model-based software product lines testing survey. In *Model-based Testing for Embedded Systems*, pages 339–381. CRC Press, 2011.

[9] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.

[10] P.-Y. Schobbens, P. Heymans, and J.-C. Trigaux. Feature diagrams: A survey and a formal semantics. In *Proc. of the 14th IEEE International Conference on Requirements Engineering*, pages 136–145. IEEE, 2006.

[11] J. Tretmans. Model based testing with labelled transition systems. In *Formal Methods and Testing*, volume 4949 of *LNCS*, pages 1–38. Springer, 2008.