A Symbolic Framework for Model-Based Testing

L. Frantzen[1,2,⋆], J. Tretmans[2], and T.A.C. Willemse[2,⋆⋆]

[1] Instituto di Scienza e Tecnologie della Informazione "Alessandro Faedo"
Consiglio Nazionale delle Ricerche, Pisa – Italy
lars.frantzen@isti.cnr.it
[2] Institute for Computing and Information Sciences
Radboud University Nijmegen – The Netherlands
{lf, tretmans, timw}@cs.ru.nl

Abstract. The starting point for Model-Based Testing is an implementation relation that formally defines when a formal model representing the System Under Test conforms to a formal model constituting its specification. An implementation relation for the formalism of Labelled Transition Systems is **ioco**. For **ioco** several test generation algorithms and test tools have been built. In this paper we define a framework for the symbolic implementation relation **sioco** which lifts **ioco** to Symbolic Transition Systems. These are transition systems with an explicit notion of data and data-dependent control flow. The introduction of symbolism avoids the state-space explosion during test generation, and it preserves the information present in data definitions and constraints for use during the test selection process. We show the soundness and completeness of the symbolic notions w.r.t. their underlying Labelled Transition Systems' counterparts.

1 Introduction

Model-Based Testing (MBT) is a form of black-box testing where a System Under Test (SUT) is tested for conformance against a formally described specification, or model, of the SUT. Test cases can be automatically generated from this model, and test results can be automatically evaluated.

The starting point for MBT is a precise definition of what it means that an SUT conforms to its specification. Such a definition is expressed by an *implementation relation*: a formal relation between the specification formalism and the implementation formalism. Although such a relation is formally expressed, it cannot be used to directly verify the relation between an SUT and its specification. Since an SUT is a physical system that we observe as a black-box, we can only perform tests on the black-box to check the relation to its specification.

⋆ Lars Frantzen is supported by the Marie Curie Network TAROT (MRTN-CT-2004-505121) and by the Netherlands Organisation for Scientific Research (NWO) under project STRESS.

⋆⋆ Tim Willemse carried out this work as part of the TANGRAM project under the responsibility of the Embedded Systems Institute. Tangram is partially supported by the Netherlands Ministry of Economic Affairs under grant TSIT2026.

Yet, it is assumed that the SUT exhibits a behavior which *could* be expressed in the implementation formalism, even if we do not know this behavior in detail. This assumption is commonly referred to as the *test hypothesis*. By so doing we can consider SUTs as formal systems, and we can formally reason about the soundness and completeness of the testing process.

Many different implementation relations have been proposed; see [3] for an overview of the state-of-the-art. A prominent example of an implementation relation is the implementation relation **ioco** [16], which is based on the formalism of *Labelled Transition Systems* (LTSs). Several testing tools implement it, e.g. TorX [1] and TGV [10]. The LTS formalism is a powerful semantic model to describe systems. However, it has some drawbacks which make its direct use for MBT cumbersome. In particular, the use of data values and variables is not possible. None the less all state-of-the-art modeling formalisms allow for such a symbolic treatment of data and often have underlying LTS-semantics, e.g. Statecharts [9] or the data-enriched process algebra LOTOS [2]. To use such a model for serving as the input to an LTS-based testing tool all data must be encoded in action names representing concrete values; there is no symbolic treatment of data. This mapping of data values leads to the infamous state space explosion problem, which limits the usability of test generation tools. A second disadvantage of this mapping is that all structure and information available in the data definitions and constraints is lost. This information can be very useful in the test selection process (see e.g. [4]).

To overcome these problems we introduced *Symbolic Transition Systems* (STS) in [6]. An STS is a transition system incorporating an explicit notion of data and data-dependent control flow, such as guarded transitions, founded on first order logic. The underlying first order structure gives formal means to define both the data part algebraically, and the control flow part logically. The emphasis in [6] was on presenting an on-the-fly algorithm for generating **ioco** test cases derived directly from STSs.

In this paper we go a fundamental step ahead by lifting the **ioco** relation to the level of STSs: we give a fully symbolic version of **ioco**, called **sioco**. Hence, **sioco** relates symbolic specifications to symbolically modeled implementations. The goal is to have a complete formal framework for symbolic testing. By being sound and complete for **ioco** the framework allows to reason about all conformance aspects, for instance repetitive quiescence. It serves as a foundation to define further symbolic aspects like symbolic test cases, coverage criteria based on symbolic reachability, etc., and to gain insight into the underlying symbolic mechanisms. Studying the implementation relation **sioco** and the concepts needed to define it, also provides a necessary and well-defined basis for the development of symbolic test generation tools.

Overview. In Sect. 2, we recall the first order concepts underlying the STS formalism. The **ioco** relation is summarized in Sect. 3. Section 4 introduces STSs and the symbolic framework. Section 5 defines the symbolic variant **sioco**. An outlook at applications of the presented theory is given in Sect. 6, followed by conclusions and related work in Sect. 7.

2 First Order Logic

We use basic concepts from first order logic as our framework for dealing with data. For a general introduction into logic we refer to [5]. Throughout this paper we use the following conventions: for sets A and B, the set of all total functions from A to B is denoted B^A. For functions $f{:}B{\rightarrow}C$ and $g{:}A{\rightarrow}B$, we denote the composition of f and g by $f \circ g$. We sometimes treat a tuple $\langle x_1, \ldots, x_n\rangle$ as the set $\{x_1, \ldots, x_n\}$ when the context allows this.

From hereon we assume a first order structure $(\mathfrak{S}, \mathfrak{M})$ as given. $\mathfrak{S} = (F,\ P)$ is a logical signature with F being a set of *function symbols*. Each $f{\in}F$ has a corresponding arity $n{\in}\mathbb{N}$. P is a set of *predicate symbols*. Each $p{\in}P$ has a corresponding arity $n{\in}\mathbb{N}^+$. The model $\mathfrak{M} = (\mathfrak{U},\ (f_{\mathfrak{M}})_{f\in F},\ (p_{\mathfrak{M}})_{p\in P})$ consists of $\mathfrak{U}$ being a non-empty set called *universe*, and for all $f{\in}F$ with arity n, $f_{\mathfrak{M}}$ is a function of type $\mathfrak{U}^n{\rightarrow}\mathfrak{U}$. For every $p{\in}P$ with arity n we have $p_{\mathfrak{M}} \subseteq \mathfrak{U}^n$. For simplicity, and without loss of generality, we restrict to one-sorted signatures.

Let $\mathfrak{X}$ be a set of *variables*; we assume sets $X, Y \subseteq \mathfrak{X}$. *Terms* over X, denoted $\mathfrak{T}(X)$, are built from variables $x{\in}X$ and function symbols $f{\in}F$. The set of variables appearing in a term t is denoted $\mathsf{var}(t)$. A *term-mapping* is a function $\sigma{:}\mathfrak{X} \rightarrow \mathfrak{T}(\mathfrak{X})$. For a given tuple of variables $\langle x_1, \ldots, x_n\rangle$ we set $\sigma(\langle x_1, \ldots, x_n\rangle) = \langle \sigma(x_1), \ldots, \sigma(x_n)\rangle$. The identity term-mapping id is defined as $\mathsf{id}(x) = x$ for all $x{\in}\mathfrak{X}$. By σ_X, we denote a restricted term-mapping σ that is only to be applied on variables from X, i.e., $\sigma_X(x) = \sigma(x)$ if $x{\in}X$, and x otherwise. The set of all term-mappings $\sigma{\in}\mathfrak{T}(\mathfrak{X})^{\mathfrak{X}}$ for which hold that $\sigma(x){\in}\mathfrak{T}(Y)$ for all $x{\in}X$, and $\sigma(x) = x$ for all $x \notin X$, is denoted $\mathfrak{T}(Y)^{\mathfrak{X}\upharpoonright X}$. We will omit the mentioning of $\mathfrak{X}$ and just write $\mathfrak{T}(Y)^X$ in the remainder.

The set of free variables of a first order formula φ is denoted $\mathsf{free}(\varphi)$. The set of all first order formulas φ satisfying $\mathsf{free}(\varphi) \subseteq X$ is denoted $\mathfrak{F}(X)$. A tautology is represented by $\top$; we set $\neg\top = \bot$. We write $\overline{\exists}_X\varphi$ for the formula $\exists x_1 \exists x_2 \ldots \exists x_n :\ \varphi$, where $\{x_1, \ldots, x_n\} = X \cap \mathsf{free}(\varphi)$, referred to as the *existential closure* for X of φ. Analogously we define the *universal closure* $\overline{\forall}_X\varphi$.

Let σ be a term-mapping. Given a formula φ, the *substitution* of $\sigma(x)$ for $x{\in}\mathsf{free}(\varphi)$ is denoted $\varphi[\sigma]$. Substitutions are side-effect free, i.e. they do not add bound variables. This is achieved using an implicit proper renaming of bound variables. Likewise, for a term t, the *substitution* of $\sigma(x)$ for $x{\in}\,\mathsf{var}(t)$ is denoted $t[\sigma]$. Together we get $[\sigma] : \mathfrak{F}(\mathfrak{X}) \cup \mathfrak{T}(\mathfrak{X}) \rightarrow \mathfrak{F}(\mathfrak{X}) \cup \mathfrak{T}(\mathfrak{X})$.

A *valuation* is a function $\vartheta{\in}\mathfrak{U}^{\mathfrak{X}}$. For a given tuple of variables $\langle x_1, \ldots, x_n\rangle$ we set $\vartheta(\langle x_1, \ldots, x_n\rangle) = \langle \vartheta(x_1), \ldots, \vartheta(x_n)\rangle$. Let $*$ denote an arbitrary element of the set $\mathfrak{U}$. A *partial valuation* is a function $\vartheta_X{\in}\mathfrak{U}^X$; ϑ_X can be extended to a valuation ϑ as follows: $\vartheta(x) = \vartheta_X(x)$ if $x{\in}X$, and $\vartheta(x) = *$ when $x{\in}\mathfrak{X} \setminus X$. Having two partial valuations $\vartheta{\in}\mathfrak{U}^X$ and $\varsigma{\in}\mathfrak{U}^Y$, with $X \cap Y = \emptyset$, their union $(\vartheta{\cup}\varsigma){\in}\mathfrak{U}^{X\cup Y}$ is defined as $(\vartheta{\cup}\varsigma)(x) = \vartheta(x)$ if $x{\in}X$, and $(\vartheta{\cup}\varsigma)(x) = \varsigma(x)$ if $x{\in}Y$. The *satisfaction* of a formula φ w.r.t. a given valuation ϑ is denoted $\vartheta \models \varphi$. The extension to evaluate terms based on a valuation ϑ is called a *term-evaluation* and denoted $\vartheta_{\mathsf{eval}}{:}\mathfrak{T}(\mathfrak{X}) \rightarrow \mathfrak{U}$.

3 A Testing Relation for Labelled Transition Systems

We assume the reader has some basic familiarity with (**ioco**-based) model-based testing techniques as described in e.g. [16], and recall only those concepts and conventions relevant to this paper.

Definition 1. *A* Labelled Transition System (LTS) *is a tuple* $\mathcal{L} = \langle S, s_0, \Sigma, \rightarrow\rangle$, *where* S *is a set of* states *and* $s_0 \in S$ *is the* initial state. *The set* Σ *is a set of* observable action labels. *The action label* $\tau \notin \Sigma$ *denotes an* unobservable *action;* Σ_τ *abbreviates the set* $\Sigma \cup \{\tau\}$. *The relation* $\rightarrow\ \subseteq S \times \Sigma_\tau \times S$ *is the* transition relation*;* $s \xrightarrow{\mu} s'$ *abbreviates* $(s, \mu, s') \in \rightarrow$.

Let $\mathcal{L} = \langle S, s_0, \Sigma, \rightarrow\rangle$ be an LTS. The generalized transition relation $\Longrightarrow \subseteq S \times \Sigma^* \times S$ of $\mathcal{L}$ is obtained in the standard way, i.e. it is the smallest relation satisfying:

(Tϵ) $s \overset{\epsilon}{\Longrightarrow} s$, with $s \in S$,
(Tτ) $s \overset{\sigma}{\Longrightarrow} s'$ if $s \overset{\sigma}{\Longrightarrow} s''$ and $s'' \xrightarrow{\tau} s'$, with $s, s', s'' \in S$ and $\sigma \in \Sigma^*$,
(Tμ) $s \overset{\sigma\cdot\mu}{\Longrightarrow} s'$ if $s \overset{\sigma}{\Longrightarrow} s''$ and $s'' \xrightarrow{\mu} s'$, with $s, s', s'' \in S$, $\sigma \in \Sigma^*$ and $\mu \in \Sigma$.

We use the following shorthand notations and functions:

1. $s \xrightarrow{\mu}$ abbreviates $\exists s' \in S : \ s \xrightarrow{\mu} s'$, with $s \in S$ and $\mu \in \Sigma_\tau$,
2. $s \overset{\sigma}{\Longrightarrow}$ abbreviates $\exists s' \in S : \ s \overset{\sigma}{\Longrightarrow} s'$, with $s \in S$ and $\sigma \in \Sigma^*$,
3. $\mathit{traces}(s) =_{\mathit{def}} \{ \sigma \in \Sigma^* \mid s \overset{\sigma}{\Longrightarrow} \}$, with $s \in S$,
4. $\mathit{der}(s) =_{\mathit{def}} \{ s' \mid \exists \sigma \in \Sigma^* : s \overset{\sigma}{\Longrightarrow} s' \}$, with $s \in S$.

A specialization of the model of LTSs is the model of *Input-Output Labelled Transition Systems* (IOLTSs), which captures the notion of initiative of actions (i.e. whether the action is an input or an output).

Definition 2. *An IOLTS is a tuple* $\langle S, s_0, \Sigma_I, \Sigma_U, \rightarrow\rangle$, *such that* $\langle S, s_0, \Sigma_I \cup \Sigma_U, \rightarrow\rangle$ *is an LTS and* $\Sigma_I \cap \Sigma_U = \emptyset$*;* Σ_I *is the set of* inputs *and* Σ_U *is the set of* outputs.

Let $\mathcal{L} = \langle S, s_0, \Sigma_I, \Sigma_U, \rightarrow\rangle$ be an IOLTS. An *observation* from $\mathcal{L}$ is an output action $\mu \in \Sigma_U$ or the refusal of all outputs; we refer to such a refusal as *quiescence*. A state $s \in S$ in $\mathcal{L}$ is *quiescent*, denoted $\delta(s)$, iff $\forall \mu \in \Sigma_U \cup \{\tau\} : \ s \overset{\mu}{\not\rightarrow}$. Let δ be a constant not part of any action label set; Σ_δ abbreviates $\Sigma_I \cup \Sigma_U \cup \{\delta\}$, and Σ_δ^* is referred to as the set of *extended traces*. We define the *suspension transition relation* $\Longrightarrow_\delta \subseteq S \times \Sigma_\delta^* \times S$ as the smallest relation satisfying rules Tϵ, Tτ, Tμ (with $\Longrightarrow_\delta$ replacing $\Longrightarrow$) and Tδ, where Tδ is given as:

(Tδ) $s \overset{\sigma\cdot\delta}{\Longrightarrow}_\delta s'$ if $s \overset{\sigma}{\Longrightarrow}_\delta s'$ and $\delta(s')$, with $s, s' \in S$ and $\sigma \in \Sigma_\delta^*$.

We define the following functions for arbitrary $s \in S$, $C \subseteq S$ and $\sigma \in \Sigma_\delta^*$:

1. $\mathit{Straces}(s) =_{\mathit{def}} \{ \sigma \in \Sigma_\delta^* \mid s \overset{\sigma}{\Longrightarrow}_\delta \}$, is the set of *suspension traces*,
2. $C\ \mathbf{after}\ \sigma\ =_{\mathit{def}} \bigcup_{s \in C}\ s\ \mathbf{after}\ \sigma$, where $s\ \mathbf{after}\ \sigma\ =_{\mathit{def}} \{ s' \in S \mid s \overset{\sigma}{\Longrightarrow}_\delta s' \}$,

3. $out(C) =_{def} \bigcup_{s \in C} out(s)$, where $out(s) =_{def} \{\mu \in \Sigma_U \mid s \xrightarrow{\mu}\} \cup \{\delta \mid \delta(s)\}$.

The *testing hypothesis* [16] states that implementations can be modeled as *input-enabled* IOLTSs, where an IOLTS $\langle S, s_0, \Sigma_I, \Sigma_U, \rightarrow\rangle$ is input-enabled if and only if:

$\forall s \in der(s_0) \forall \mu \in \Sigma_I : \; s\, \mathbf{after}\, \mu \neq \emptyset$.

The conformance testing relation **ioco** is defined as follows:

Definition 3. *Let* $\mathcal{S} = \langle S, s_0, \Sigma_I, \Sigma_U, \rightarrow_{\mathcal{S}}\rangle$ *be a specification IOLTS, and let* $\mathcal{F} \subseteq Straces(s_0)$*. An input-enabled IOLTS* $\mathcal{P} = \langle P, p_0, \Sigma_I, \Sigma_U, \rightarrow_{\mathcal{P}}\rangle$ *is* $\mathbf{ioco}_{\mathcal{F}}$*-conform to* $\mathcal{S}$*, denoted by* $\mathcal{P}\; \mathbf{ioco}_{\mathcal{F}}\; \mathcal{S}$*, iff*

$\forall \sigma \in \mathcal{F} : out(\, p_0\, \mathbf{after}\, \sigma\,) \subseteq out(\, s_0\, \mathbf{after}\, \sigma\,)$

4 The Symbolic Framework

In practical situations, LTSs lack the required level of abstraction for modeling complex, data-intensive systems. This problem is solved by the model of *Symbolic Transition Systems* (see e.g. [15,6]), which we introduce in this section.

4.1 Syntax and Semantics for Symbolic Transition Systems

The STS model extends the model of LTSs by incorporating an explicit notion of data and data-dependent control flow (such as guarded transitions), founded on first order logic.

Definition 4. *An STS is a tuple* $\mathcal{S} = \langle L, l_0, \mathcal{V}, \mathcal{I}, \Lambda, \rightarrow\rangle$*, where* L *is a set of* locations *and* $l_0 \in L$ *is the* initial location*.* $\mathcal{V}$ *is a set of* location variables *and* $\mathcal{I}$ *is a set of* interaction variables*;* $\mathcal{V} \cap \mathcal{I} = \emptyset$*, and we set* $Var =_{def} \mathcal{V} \cup \mathcal{I}$*.* Λ *is the set of* gates*; constant* $\tau \notin \Lambda$ *denotes an* unobservable gate*;* Λ_τ *abbreviates* $\Lambda \cup \{\tau\}$*. The relation* $\rightarrow \subseteq L \times \Lambda_\tau \times \mathfrak{F}(Var) \times \mathfrak{T}(Var)^{\mathcal{V}} \times L$ *is the* switch relation*;* $l \xrightarrow{\lambda,\varphi,\rho} l'$ *abbreviates* $(l, \lambda, \varphi, \rho, l') \in \rightarrow$*, where* φ *is the* switch restriction *and* ρ *is the* update mapping*. We use the following functions and vocabulary:*

1. $\mathsf{arity} : \Lambda_\tau \rightarrow \mathbb{N}$ *is the* arity function*,*
2. $\mathsf{type}(\lambda)$ *yields a tuple of size* $\mathsf{arity}(\lambda)$ *of interaction variables for gate* λ*,*
3. $\mathcal{S}$ *is* well-defined *iff* $\mathsf{arity}(\tau) = 0$*,* $\mathsf{type}(\lambda)$ *yields a tuple of* distinct *interaction variables, and* $l \xrightarrow{\lambda,\varphi,\rho} l'$ *implies* $\mathsf{free}(\varphi) \subseteq \mathcal{V} \cup \mathsf{type}(\lambda)$ *and* $\rho \in \mathfrak{T}(\mathcal{V} \cup \mathsf{type}(\lambda))^{\mathcal{V}}$*,*
4. $\mathcal{S}(\iota)$ *is an* initialized STS*, where* $\iota \in \mathfrak{U}^{\mathcal{V}}$ *initializes all variables from* $\mathcal{V}$ *in* l_0*.*

We only consider well-defined STSs in this paper.

Example 1. The STS $\langle \{l_i \mid 0 \leq i \leq 5\}, l_0, \{\mathsf{rp}, \mathsf{q}, \mathsf{r}\}, \{\mathsf{prod}, \mathsf{quant}, \mathsf{ref}\}, \Lambda, \rightarrow\rangle$, with $\Lambda = \{?\mathsf{rq}, !\mathsf{gq}, ?\mathsf{ord}, !\mathsf{confirm}, !\mathsf{cancel}\}$ is depicted in Fig. 1; $\rightarrow$ is given by the directed edges linking the locations. We have e.g. $\mathsf{arity}(?\mathsf{rq}) = 2$ and $\mathsf{type}(?\mathsf{rq}) = <\mathsf{prod},\mathsf{quant}>$. The underlying first order structure is based on a natural number universe with the common "less-than" predicate $<$. The STS specifies a simplified supplier system which can be requested for a quote for a given product prod and

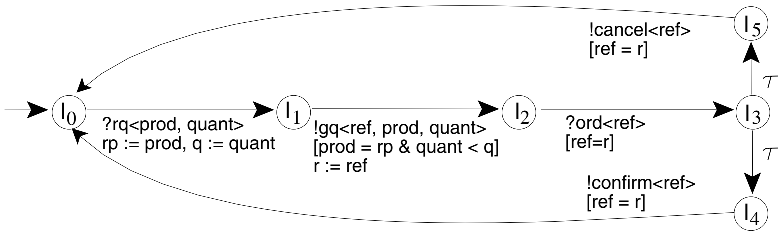

Fig. 1. An STS specifying a simplified *Supplier*

quantity quant via gate ?rq located on the switch from l_0 to l_1. The requested product and quantity are stored in the location variables rp and q, respectively. Next a quote is returned via gate !gq which must deal with the same product and with a quantity less than the requested one. Subsequently, the quote can be ordered via gate ?ord by giving the correct reference number from the received quote. Finally the supplier nondeterministically communicates a cancellation of the order via the !cancel gate, or confirms the order via the !confirm gate. As a convention, switch constraints $\top$ and update-mappings id are not explicitly drawn. We will refer to this STS in the following examples as the Supplier STS. □

The interpretation of an STS is defined in terms of LTSs.

Definition 5. *Let $\mathcal{S} = \langle L, l_0, \mathcal{V}, \mathcal{I}, \Lambda, \rightarrow\rangle$ be an STS. Its interpretation $[\![\mathcal{S}]\!]_\iota$ in the context of $\iota \in \mathfrak{U}^{\mathcal{V}}$, is defined as $[\![\mathcal{S}]\!]_\iota = \langle L \times \mathfrak{U}^{\mathcal{V}}, (l_0, \iota), \Sigma, \rightarrow\rangle$ for all $\iota \in \mathfrak{U}^{\mathcal{V}}$, where*

– $\Sigma = \bigcup_{\lambda \in \Lambda}(\{\lambda\} \times \mathfrak{U}^{\mathsf{arity}(\lambda)})$, *is the set of actions.*
– $\rightarrow \subseteq (L \times \mathfrak{U}^{\mathcal{V}}) \times (\Sigma \cup \{\tau\}) \times (L \times \mathfrak{U}^{\mathcal{V}})$ *is defined by the following rule:*

$$\frac{l \xrightarrow{\lambda,\varphi,\rho} l' \quad \varsigma \in \mathfrak{U}^{\mathsf{type}(\lambda)} \quad \vartheta \cup \varsigma \models \varphi \quad \vartheta' = (\vartheta \cup \varsigma)_{\mathsf{eval}} \circ \rho}{(l, \vartheta) \xrightarrow{(\lambda,\, \varsigma(\mathsf{type}(\lambda)))} (l', \vartheta')}$$

The semantics of an initialized STS $\mathcal{S}(\iota)$ is given by the LTS $[\![\mathcal{S}]\!]_\iota$.

4.2 Symbolic Executions and Symbolic States

The notion of a trace of an STS can be defined by appealing to the semantics of an initialized STS. This, however, suffers from the disadvantage that all high-level information and structure about the data that is communicated over gates is lost. Therefore, we choose to define a notion of traces on the level of *symbolic executions.*

Symbolic execution as a technique was initially developed to symbolically execute imperative programs with the aim of proving correctness. This can be a hard task since the symbolic execution tree can for instance be of infinite size due to loops in the program. For this reason already early approaches suggested to just partially generate the execution tree for testing the program against a given specification, see e.g. [12].

Even if our domain of interest, i.e. reactive systems, has some fundamental originalities like nondeterminism, non-termination, etc., many of the classical symbolic execution techniques can be recovered in our setting. We do not adopt the symbolic tree representation, though, instead we use a more compact, linear representation which fits better to the standard notions we have introduced in Sect. 3. Whereas in a symbolic tree the ordering of events is encoded in its depth, we do so explicitly via so called *history variables*, which represent possible interactions. These variables provide a representation for the data that could have been communicated over a particular gate appearing at some point in a symbolic execution.

Example 2. Starting in location l_0 we can let the Supplier symbolically move to location l_1. Here the gate ?rq requests a product prod and a quantity quant. These values are stored in the location variables rp and q, respectively. All we know after executing this switch symbolically is that rp equals the value of prod, and q equals the value of quant. Proceeding symbolically we may encounter again the interaction variables prod or quant, hence it is necessary to make explicit that we are referring to the first occurrence of these variables within the symbolic execution. We do so by introducing the history variables prod_1 and quant_1. Hence we can, after moving from l_0 to l_1, formally record that $\mathsf{rp} \mapsto \mathsf{prod}_1$ and $\mathsf{q} \mapsto \mathsf{quant}_1$. Proceeding now from l_1 to l_2 the gate !gq returns a quote which also consists of a quantity, represented again by the interaction variable quant. This variable is now constrained by $\mathsf{quant} < \mathsf{q}$. In our symbolic context this equals $\mathsf{quant} < \mathsf{quant}_1$. Also here we have to refer to the correct occurrence of the interaction variable, so we introduce another history variable quant_2 and record here $\mathsf{quant}_2 < \mathsf{quant}_1$. Analogously we get $\mathsf{prod}_2 = \mathsf{prod}_1$ and $\mathsf{r} \mapsto \mathsf{ref}_2$. □

For the remainder of this section we assume an STS $\mathcal{S} = \langle L, l_0, \mathcal{V}, \mathcal{I}, \Lambda, \rightarrow \rangle$. Henceforth, we assume to have *history variable sets* $\mathcal{I}_1, \mathcal{I}_2, \ldots$ which are disjoint from each-other and from the set *Var* of $\mathcal{S}$. We set $\widehat{\mathcal{I}} =_{def} \bigcup_j \mathcal{I}_j$, and $\widehat{Var} =_{def} \mathcal{V} \cup \widehat{\mathcal{I}}$. In addition, we assume to have bijective *variable-renamings* $r_n \in \mathcal{I}_n^{\mathcal{I}}$.

The generalized switch relation $\Longrightarrow \subseteq L \times \Lambda^* \times \mathfrak{F}(\widehat{Var}) \times \mathfrak{T}(\widehat{Var})^{\mathcal{V}} \times L$, is defined as the smallest relation satisfying the following three rules:

(Sϵ) $l \xRightarrow{\epsilon,\ \top,\ \mathsf{id}} l$,

(Sτ) $l \xRightarrow{\sigma,\ \varphi\wedge\psi[\rho],\ [\rho]\circ\pi} l'$ if $l \xRightarrow{\sigma,\ \varphi,\ \rho} l''$ and $l'' \xrightarrow{\tau,\ \psi,\ \pi} l'$,

(Sλ) $l \xRightarrow{\sigma\cdot\lambda,\ \varphi\wedge(\psi[r_n])[\rho],\ ([\rho]\circ([r_n]\circ\pi))_{\mathcal{V}}} l'$ if $l \xRightarrow{\sigma,\ \varphi,\ \rho} l''$ and $l'' \xrightarrow{\lambda,\psi,\pi} l'$ and $n = \mathit{length}(\sigma){+}1$.

Analogously to the generalized transition relation $\Longrightarrow$ for LTSs, the generalized switch relation hides unobservable events without affecting the observable events that can follow it. The intuition behind a generalized switch $l \xRightarrow{\sigma,\varphi,\rho} l'$ is that location l' can be reached from location l via a series of interactions over gates, the sequence of which is dictated by σ, and the values that are passed over these gates satisfy the conditions collected in φ (called *attainment constraint*[1]); the values for the location variables $\mathcal{V}$ are specified by *update-mapping* ρ.

Using a *variable-shifting* for the involved attainment constraints and term-mappings, generalized switches can be composed to yield larger generalized switches. A variable-shifting function re-indexes sets of history variables: having renamings $r_{(j,k)} \in \mathcal{I}_k^{\mathcal{I}_j}$ between all pairs $\mathcal{I}_j$ and $\mathcal{I}_k$, defined as $r_{(j,k)} =_{def} r_k \circ r_j^{-1}$, for all $i, j, k \in \mathbb{N}^+$ (where $r_k \in \mathcal{I}_k^{\mathcal{I}}$ is a bijective renaming function for history variables), we define a *variable-shifting* function $s^{\gg i} \in \widehat{\mathcal{I}}^{\widehat{\mathcal{I}}}$ for all $i \in \mathbb{N}$ as follows:

$$s^{\gg i}(x) =_{def} \begin{cases} r_{(j,j+i)}(x) & \text{if } x \in \mathcal{I}_j \text{ for some } j, \\ x & \text{otherwise} \end{cases}$$

Proposition 1. *If* $l \xrightarrow{\sigma_1,\ \varphi_1,\ \rho_1} l''$ *and* $l'' \xrightarrow{\sigma_2,\ \varphi_2,\ \rho_2} l'$ *and* $n = length(\sigma_1)$, *then also* $l \xrightarrow{\sigma_1 \cdot \sigma_2,\ \varphi_1 \wedge (\varphi_2[s^{\gg n}])[\rho_1],\ ([\rho_1] \circ ([s^{\gg n}] \circ \rho_2))_{\mathcal{V}}} l'$.

Note that there may be a large number of different executions (generalized switches) to get from l to l'. Each of these may have different effects on the values for the location variables at location l'. Therefore, given a location, we have no means to deduce what the possible values for the location variables are. These values are required to compute the semantical states of an STS, which in turn is required for defining the implementation relation **ioco**. To solve this issue, we introduce the concept of *symbolic states*. Symbolic states provide a finite characterization of (possibly infinite) sets of semantical states of an STS.

Definition 6. *A* symbolic state *is a tuple* $(l, \varphi, \rho) \in L \times \mathfrak{F}(\widehat{Var}) \times \mathfrak{T}(\widehat{Var})^{\mathcal{V}}$. *When the history variables referenced by attainment constraint* φ *and update-mapping* ρ *are from a set not above some* $i \in \mathbb{N}$, *we may add an* index *to the symbolic state* (l, φ, ρ) *and refer to it as an* indexed *symbolic state, denoted* $(l, \varphi, \rho)_i$. *We require that* $(l, \varphi, \rho)_i$ *satisfies:*

1. $\varphi \in \mathfrak{F}(\mathcal{V} \cup \bigcup_{j \le i} \mathcal{I}_j)$, *and*
2. $\rho \in \mathfrak{T}(\mathcal{V} \cup \bigcup_{j \le i} \mathcal{I}_j)^{\mathcal{V}}$.

The interpretation of a symbolic state in the context of location variable valuation ι and history variable valuation υ is a set of states of $[\![\mathcal{S}]\!]_\iota$.

Definition 7. *Let* $\iota \in \mathfrak{U}^{\mathcal{V}}$ *and let* $\upsilon \in \mathfrak{U}^{\widehat{\mathcal{I}}}$. *The interpretation of a symbolic state* (l, φ, ρ) *with respect to* ι *and* υ *is defined by:*

$$[\![(l, \varphi, \rho)]\!]_{\iota,\upsilon} =_{def} \{(l,\ (\iota \cup \upsilon)_{\mathsf{eval}} \circ \rho) \mid \iota \cup \upsilon \models \varphi\}$$

Remark that $|[\![(l, \varphi, \rho)]\!]_{\iota,\upsilon}| \le 1$; as a convention we identify the singleton set with its only element, omitting the set notation at our convenience. For sets of symbolic states $\mathcal{C} \subseteq L \times \mathfrak{F}(\widehat{Var}) \times \mathfrak{T}(\widehat{Var})^{\mathcal{V}}$, we define the following shorthands:

[1] The attainment constraint φ corresponds to what is called a *path condition* in the literature for symbolic execution of programs.

1. $[\![\mathcal{C}]\!]_{\iota,\upsilon} =_{def} \bigcup_{(l,\varphi,\rho)\in\mathcal{C}} [\![(l,\varphi,\rho)]\!]_{\iota,\upsilon}$,
2. $[\![\mathcal{C}]\!]_{\iota} =_{def} \bigcup_{\upsilon\in\mathfrak{U}^{\hat{\mathcal{I}}}} [\![\mathcal{C}]\!]_{\iota,\upsilon}$.

Example 3. The history variables for the Supplier are $\mathcal{I}_j = \{\mathsf{prod}_j, \mathsf{quant}_j, \mathsf{ref}_j\}$ with $j\in\mathbb{N}^+$. A generalized switch is $l_0 \xRightarrow{\mathsf{?rq}\cdot\mathsf{!gq}\cdot\mathsf{?ord}\cdot\mathsf{!cancel},\ \varphi,\ \rho} l_0$ with $\varphi = (\mathsf{prod}_2 = \mathsf{prod}_1)\wedge(\mathsf{quant}_2 < \mathsf{quant}_1)\wedge(\mathsf{ref}_3 = \mathsf{ref}_2)\wedge(\mathsf{ref}_4 = \mathsf{ref}_2)$ and $\rho = \{\mathsf{rp} \mapsto \mathsf{prod}_1,\ \mathsf{q} \mapsto \mathsf{quant}_1,\ \mathsf{r} \mapsto \mathsf{ref}_2\}$. The symbolic state (l_0, φ, ρ) can be indexed by 4 or greater, and $[\![\{(l_0,\varphi,\rho)\}]\!]_\iota = \{(l_0,\ \{\mathsf{rp} \mapsto x,\ \mathsf{q} \mapsto y,\ \mathsf{r} \mapsto z\}) \mid x, z\in\mathbb{N}, y\in\mathbb{N}^+\}$ for all $\iota\in\mathfrak{U}^{\mathcal{V}}$. □

5 A Symbolic Implementation Relation for STSs

In this section, we introduce the necessary concepts to define the implementation relation **sioco** on the level of STSs, which we prove to be equivalent to **ioco** on LTSs. We specialism the model of STSs by recognizing input-gates and output-gates. The resulting model is called *Input-Output Symbolic Transition Systems* (IOSTSs).

Definition 8. *An IOSTS is a tuple* $\langle L, l_0, \mathcal{V}, \mathcal{I}, \Lambda_I, \Lambda_U, \rightarrow\rangle$ *with* $\langle L, l_0, \mathcal{V}, \mathcal{I}, \Lambda_I \cup \Lambda_U, \rightarrow\rangle$ *being an STS and* $\Lambda_I \cap \Lambda_U = \emptyset$*;* Λ_I *is the set of* input gates *and* Λ_U *is the set of* output gates.

Throughout this section we assume a given IOSTS $\mathcal{S} = \langle L, l_0, \mathcal{V}, \mathcal{I}, \Lambda_I, \Lambda_U, \rightarrow\rangle$. The interpretation of $\mathcal{S}$ is a function from initialization functions to IOLTSs; it is a straightforward adaptation of Def. 5, in which Σ_I is the set of actions $(\lambda, \_)$ with $\lambda\in\Lambda_I$, and Σ_U is the set of actions $(\lambda, \_)$ with $\lambda\in\Lambda_U$. Distinguishing between input- and output interactions at the symbolic level allows us to define a symbolic analogue to quiescence. Since quiescence of a location $l\in L$ depends on the values for the location variables and the existence of proper values for interaction variables, we are primarily interested in the condition under which location l is quiescent. This symbolic quiescence condition is denoted $\Delta(l)\in\mathfrak{F}(\mathcal{V})$, and is defined as follows:

$$\Delta(l) =_{def} \bigwedge\{\neg\exists_{\mathsf{type}(\lambda)}\psi \ \mid \exists l', \pi \ :\ l \xrightarrow{\lambda,\psi,\pi} l' \text{ with } \lambda\in\Lambda_U \cup \{\tau\}\}$$

Example 4. To transform the Supplier STS into an IOSTS we set $\Lambda_I = \{\mathsf{?rq},\ \mathsf{?ord}\}$ and $\Lambda_U = \{\mathsf{!gq},\ \mathsf{!confirm},\ \mathsf{!cancel}\}$. We get $\Delta(l_1) = \neg\big(\exists\mathsf{ref}\exists\mathsf{prod}\exists\mathsf{quant}\ :\ \mathsf{prod} = \mathsf{rp} \wedge \mathsf{quant} < \mathsf{q}\big)$ for the Supplier. In the underlying natural numbers model the satisfiability of this formula boils down to $\mathsf{q} = 0$, i.e. l_1 is quiescent given that the requested quote has a zero quantity. The switch restriction from l_1 to l_2 is unsolvable, resulting in deadlock. □

Communications over output gates $\lambda\in\Lambda_U$, or the refusals δ of any output communication are the observables of an IOSTS. In contrast to the semantic framework of LTSs, these communications may depend on values that were communicated at an earlier stage, meaning that the observations are conditional. The combination of such conditions and the communications over a gate is referred to as a *symbolic observation.* Let the set of symbolic observations $\mathcal{O}$ for a given IOSTS $\mathcal{S}$ be defined as the set $\mathcal{O} =_{def} (\Lambda_U \cup \{\delta\}) \times \mathfrak{F}(\widehat{Var}) \times \mathfrak{F}(\widehat{Var} \cup \mathcal{I})$ with $\mathsf{free}(\psi) \subseteq \mathsf{type}(\lambda_\delta) \cup \widehat{Var}$ for all $(\lambda_\delta, \varphi, \psi) \in \mathcal{O}$ (assuming $\mathsf{type}(\delta) = \emptyset$). We interpret a symbolic observation in terms of semantic actions:

Definition 9. *Let $(\lambda_\delta, \varphi, \psi)$ be a symbolic observation. The interpretation $[\![(\lambda_\delta, \varphi, \psi)]\!]_{\iota,\upsilon}$ of $(\lambda_\delta, \varphi, \psi)$ is given in the context of $\iota \in \mathfrak{U}^{\mathcal{V}}$ and $\upsilon \in \mathfrak{U}^{\widehat{\mathcal{I}}}$:*

$$[\![(\delta, \varphi, \psi)]\!]_{\iota,\upsilon} = \{\delta \mid \iota \cup \upsilon \models \varphi \wedge \psi\}$$
$$[\![(\lambda, \varphi, \psi)]\!]_{\iota,\upsilon} = \{(\lambda, \varsigma(\mathsf{type}(\lambda))) \mid \iota \cup \upsilon \cup \varsigma \models \varphi \wedge \psi \text{ with } \varsigma \in \mathfrak{U}^{\mathsf{type}(\lambda)}\}$$

The interpretation of a set $O \subseteq \mathcal{O}$ in the context of $\iota \in \mathfrak{U}^{\mathcal{V}}$ and $\upsilon \in \mathfrak{U}^{\widehat{\mathcal{I}}}$ is defined as follows: $[\![O]\!]_{\iota,\upsilon} =_{def} \bigcup_{(\lambda_\delta, \varphi, \psi) \in O} [\![(\lambda_\delta, \varphi, \psi)]\!]_{\iota,\upsilon}$

The function $\mathbf{out}_s$ is defined on symbolic states, yielding a set of observations.

Definition 10. *Let (l, φ, ρ) be a symbolic state. We define:*

$$\mathbf{out}_s((l, \varphi, \rho)) =_{def} \{(\lambda, \varphi, \psi[\rho]) \in \mathcal{O} \mid \exists l', \pi : l \xrightarrow{\lambda,\psi,\pi} l'\} \cup \{(\delta, \varphi, \Delta(l)[\rho])\}$$

Let $\mathcal{C}$ be a set of symbolic states. Here we set:

$$\mathbf{out}_s(\mathcal{C}) =_{def} \bigcup_{(l,\varphi,\rho) \in \mathcal{C}} \mathbf{out}_s((l, \varphi, \rho))$$

Lemma 1. *For all $\iota \in \mathfrak{U}^{\mathcal{V}}$, $\upsilon \in \mathfrak{U}^{\widehat{\mathcal{I}}}$ and sets $\mathcal{C}$ of symbolic states we have:*

$$[\![\, \mathbf{out}_s(\mathcal{C})]\!]_{\iota,\upsilon} = \mathbf{out}([\![\mathcal{C}]\!]_{\iota,\upsilon})$$

From hereon, we set $\Lambda_\delta =_{def} \Lambda_I \cup \Lambda_U \cup \{\delta\}$. We define the *symbolic suspension switch relation* $\Longrightarrow_\delta \subseteq L \times \Lambda_\delta^* \times \mathfrak{F}(\widehat{Var}) \times \mathfrak{T}(\widehat{Var})^{\mathcal{V}} \times L$ as the smallest relation satisfying rules Sϵ, Sτ, Sλ (with $\Longrightarrow_\delta$ replacing $\Longrightarrow$) and Sδ, given as:

(Sδ) $l \xRightarrow{\sigma \cdot \delta,\ \varphi \wedge \Delta(l')[\rho],\ \rho}_\delta l'$ if $l \xRightarrow{\sigma,\varphi,\rho}_\delta l'$.

The rule Sδ reveals the fact that quiescence is an intrinsic semantical property. During a symbolic execution we can at any step just *hypothesize* that the system is quiescent and add a corresponding logical statement to the attainment constraint (that is what rule Sδ does). Solving the constraint semantically means to compute the conditions under which quiescence really occurs (i.e. the traces which lead to a quiescent state).

The history variables that are allowed to be addressed in a sequence $\sigma \in \Lambda_\delta^*$ are given by $\mathsf{var}(\sigma)$, where $\mathsf{var} : \Lambda_\delta^* \rightarrow 2^{\widehat{\mathcal{I}}}$ is defined inductively as:

$$\begin{cases} \mathsf{var}(\epsilon) &= \emptyset \\ \mathsf{var}(\sigma \cdot \delta) &= \mathsf{var}(\sigma) \\ \mathsf{var}(\sigma \cdot \lambda) &= \mathsf{var}(\sigma) \cup \{r_{length(\sigma)+1}(\nu) \mid \nu \in \mathsf{type}(\lambda)\} \end{cases}$$

Lemma 2. *If* $l \xRightarrow{\sigma,\ \varphi,\ \rho}_\delta l'$ *then we have* $\varphi \in \mathfrak{F}(\mathcal{V} \cup \mathsf{var}(\sigma))$ *and* $\rho \in \mathfrak{T}(\mathcal{V} \cup \mathsf{var}(\sigma))^{\mathcal{V}}$ *and* $(l', \varphi, \rho)_{length(\sigma)}$ *is an indexed symbolic state.*

Let $\mathcal{E}$ denote the set of *symbolic extended traces* $\{(\sigma, \varphi) \in \Lambda_\delta^* \times \mathfrak{F}(\widehat{Var}) \mid \mathsf{free}(\varphi) \subseteq \mathcal{V} \cup \mathsf{var}(\sigma)\}$. The interpretation of symbolic extended traces is given below:

Definition 11. *Let* $\iota \in \mathfrak{U}^{\mathcal{V}}$ *and let* $\upsilon \in \mathfrak{U}^{\widehat{\mathcal{I}}}$. *The interpretation of a symbolic extended trace* (σ, φ) *with respect to* ι *and* υ *is an extended trace, defined by:*

$$[\![(\sigma, \varphi)]\!]_{\iota,\upsilon} =_{def} \{\mathbf{etrace}_\upsilon(\sigma) \mid \iota \cup \upsilon \models \varphi\}$$

where $\mathbf{etrace}_\upsilon(\sigma)$ *is inductively defined as follows:*

$$\begin{cases} \mathbf{etrace}_\upsilon(\epsilon) &= \epsilon \\ \mathbf{etrace}_\upsilon(\sigma \cdot \delta) &= \mathbf{etrace}_\upsilon(\sigma) \cdot \delta \\ \mathbf{etrace}_\upsilon(\sigma \cdot \lambda) &= \mathbf{etrace}_\upsilon(\sigma) \cdot (\lambda, \upsilon(r_n(\mathsf{type}(\lambda)))) \qquad \textit{with } n = length(\sigma) + 1 \end{cases}$$

Note that $|[\![(\sigma, \varphi)]\!]_{\iota,\upsilon}| \leq 1$; as a convention, we identify the singleton set with its only element. For sets $E \subseteq \mathcal{E}$, we define the following shorthands:

1. $[\![E]\!]_{\iota,\upsilon} =_{def} \bigcup_{(\sigma,\varphi) \in E} [\![(\sigma, \varphi)]\!]_{\iota,\upsilon}$,
2. $[\![E]\!]_{\iota} =_{def} \bigcup_{\upsilon \in \mathfrak{U}^{\widehat{\mathcal{I}}}} [\![E]\!]_{\iota,\upsilon}$.

To complete the set of symbolic counterparts for the relevant semantical notions we define a symbolic $\mathbf{after}_s$ function, mapping pairs of indexed symbolic states and symbolic extended traces to new indexed symbolic states.

Definition 12. *Let* $(l, \varphi, \rho)_i$ *be an indexed symbolic state and let* $(\sigma, \chi) \in \mathcal{E}$ *be a symbolic extended trace. We define the binary function* $\mathbf{after}_s$ *as follows:*

$$\begin{aligned} &(l, \varphi, \rho)_i \, \mathbf{after}_s (\sigma, \chi) \\ =_{def} &\{(l', \varphi \wedge ((\psi \wedge \chi)[s^{\gg i}])[\rho], ([\rho] \circ ([s^{\gg i}] \circ \pi))_{\mathcal{V}})_{i+length(\sigma)} \mid l \xRightarrow{\sigma,\psi,\pi}_\delta l'\} \end{aligned}$$

Let $\mathcal{C}$ *be a set of indexed symbolic states. Here we set*
$\mathcal{C} \, \mathbf{after}_s(\sigma, \chi) =_{def} \bigcup_{(l,\varphi,\rho)_i \in \mathcal{C}} (l, \varphi, \rho)_i \, \mathbf{after}_s(\sigma, \chi)$.

Lemma 3. *Let* $(l, \varphi, \rho)_i$ *be an indexed symbolic state and let* $(\sigma, \chi) \in \mathcal{E}$ *be a symbolic extended trace. Then for all* $\iota \in \mathfrak{U}^{\mathcal{V}}$ *and* $\upsilon \in \mathfrak{U}^{\widehat{\mathcal{I}}}$, *we have:*

$$[\![(l, \varphi, \rho)_i \, \mathbf{after}_s(\sigma, \chi)]\!]_{\iota,\upsilon} = [\![(l, \varphi, \rho)_i]\!]_{\iota,\upsilon} \, \mathbf{after} \, [\![(\sigma, \chi)]\!]_{(\iota \cup \upsilon)_{\mathsf{eval}} \circ \rho,\ \upsilon \circ s^{\gg i}}$$

Example 5. For the $\mathsf{Supplier}$ we get $(l_2, \mathsf{r} > \mathsf{prod}_3, \mathsf{id})_3 \, \mathbf{after}_s(?\mathsf{ord}, \mathsf{ref}_1 < 42) = \{(l_i, \xi, \mathsf{id})_4 \mid i = 3, 4, 5\}$ with $\xi = \mathsf{r} > \mathsf{prod}_3 \wedge \mathsf{ref}_4 = \mathsf{r} \wedge \mathsf{ref}_4 < 42$. If we call the latter set M and apply common first order equalities we get $\mathbf{out}_s(M) = \{(\delta, \xi, \bot), (!\mathsf{confirm}, \xi, \mathsf{ref} = \mathsf{r}), ((!\mathsf{cancel}, \xi, \mathsf{ref} = \mathsf{r}))\}$. □

The symbolic concepts that have been introduced so far provide a characterization of the semantically relevant concepts that were introduced in Section 3. The precise connection is established in the following two theorems.

Theorem 1 (Soundness). *Let $\mathcal{S} = \langle L, l_0, \mathcal{V}, \mathcal{I}, \Lambda_I, \Lambda_U, \rightarrow\rangle$ be an IOSTS. Then for all $\iota \in \mathfrak{U}^{\mathcal{V}}$ and all $\upsilon \in \mathfrak{U}^{\widehat{\mathcal{I}}}$ we have: if both $l \xRightarrow{\sigma,\varphi,\rho}_{\delta} l'$ and $\iota \cup \upsilon \models \varphi$ then also $[\![(l, \top, \mathsf{id})]\!]_{\iota,\upsilon} \xRightarrow{[\![(\sigma,\varphi)]\!]_{\iota,\upsilon}}_{\delta} [\![(l', \varphi, \rho)]\!]_{\iota,\upsilon}$.*

Theorem 2 (Completeness). *Let $\mathcal{S} = \langle L, l_0, \mathcal{V}, \mathcal{I}, \Lambda_I, \Lambda_U, \rightarrow\rangle$ be an IOSTS. For all states $(l, \iota), (l', \iota')$ we have: $(l, \iota) \xRightarrow{\overline{\sigma}}_{\delta} (l', \iota')$ implies there is a valuation $\upsilon \in \mathfrak{U}^{\widehat{\mathcal{I}}}$ and a suspension switch $l \xRightarrow{\sigma,\varphi,\rho}_{\delta} l'$ satisfying $\iota \cup \upsilon \models \varphi$, $\overline{\sigma} = [\![(\sigma, \varphi)]\!]_{\iota,\upsilon}$ and $(l', \iota') = [\![(l', \varphi, \rho)]\!]_{\iota,\upsilon}$.*

The set of *symbolic suspension traces* of a location l of an IOSTS $\mathcal{S}$ is denoted $\mathit{Straces}_s(l)$, which is defined as $\mathit{Straces}_s(l) =_{def} \{(\sigma, \varphi) \in \mathcal{E} \mid \exists l', \rho : \ l \xRightarrow{\sigma,\varphi,\rho}_{\delta} l'\}$.

Corollary 1. *Let $\mathcal{S}(\iota) = \langle L, l_0, \mathcal{V}, \mathcal{I}, \Lambda_I, \Lambda_U, \rightarrow\rangle$ be an initialized IOSTS. Then we have $[\![\mathit{Straces}_s(l_0)]\!]_{\iota} = \mathit{Straces}((l_0, \iota))$.*

Definition 13. *Let $\mathcal{S}(\iota)$ be an initialized IOSTS. We set: $\mathcal{S}(\iota)$ is input enabled $\Leftrightarrow_{def}$ $[\![\mathcal{S}]\!]_{\iota}$ is an input-enabled IOLTS.*

Now we are in the position to give the symbolic **sioco** variant of the **ioco** relation, based on the notions introduced so far.

Definition 14 (sioco). *Let $\mathcal{F}_s$ be a set of symbolic extended traces for an initialized specification IOSTS $\mathcal{S}(\iota_S) = \langle L_S, l_S, \mathcal{V}_S, \mathcal{I}, \Lambda, \rightarrow_S\rangle$, satisfying $[\![\mathcal{F}_s]\!]_{\iota_S} \subseteq \mathit{Straces}((l_0, \iota_S))$. An implementation, given as an input-enabled IOSTS $\mathcal{P}(\iota_P) = \langle L_P, l_P, \mathcal{V}_P, \mathcal{I}, \Lambda, \rightarrow_P\rangle$, with $\mathcal{V}_S \cap \mathcal{V}_P = \emptyset$, is $\mathbf{sioco}_{\mathcal{F}_s}$-conform to $\mathcal{S}(\iota_S)$ (written $\mathcal{P}(\iota_P)\, \mathbf{sioco}_{\mathcal{F}_s}\, \mathcal{S}(\iota_S)$) iff*

$$\forall(\sigma, \chi) \in \mathcal{F}_s\ \forall \lambda_\delta \in \Lambda_U \cup \{\delta\} : \iota_P \cup \iota_S \models \overline{\forall}_{\widehat{\mathcal{I}} \cup \mathcal{I}}\big(\Phi(l_P, \lambda_\delta, \sigma) \wedge \chi \rightarrow \Phi(l_S, \lambda_\delta, \sigma)\big)$$
$$\textit{where } \Phi(\varkappa, \lambda_\delta, \sigma) = \bigvee\{\varphi \wedge \psi \mid (\lambda_\delta,\ \varphi,\ \psi) \in \mathbf{out}_s((\varkappa, \top, \mathsf{id})_0\ \mathbf{after}_s(\sigma, \top))\}$$

The following theorem expresses that **sioco** coincides with **ioco**.

Theorem 3. *Let $\mathcal{S}(\iota_S) = \langle L, l_0, \mathcal{V}, \mathcal{I}, \Lambda_I, \Lambda_U, \rightarrow\rangle$ be an initialized IOSTS and let $\mathcal{P}(\iota_P)$ be an input-enabled IOSTS. Let $\mathcal{F}_s$ be a set of symbolic extended traces for $\mathcal{S}$, satisfying $[\![\mathcal{F}_s]\!]_{\iota_S} \subseteq \mathit{Straces}((l_0, \iota_S))$. Then*

$$\mathcal{P}(\iota_P)\, \mathbf{sioco}_{\mathcal{F}_s}\, \mathcal{S}(\iota_S) \ \textit{ iff } \ [\![\mathcal{P}]\!]_{\iota_P}\, \mathbf{ioco}_{[\![\mathcal{F}_s]\!]_{\iota_S}}\, [\![\mathcal{S}]\!]_{\iota_S}$$

6 Application

The concepts that we have defined can be employed to define relations such as *symbolic state inclusion*, which allow one to efficiently prune symbolic executions (see e.g. [7]). Another example of how our theory contributes in improving and studying practically relevant testing problems is given in this section. We first define a naive (but often used) coverage measure that is based on reachability of states, and show that in the presence of data and control, coverage measures that are based on the concepts of locations and symbolic states are much more appropriate. In practice, a coverage measure can be used to fuel the test selection process; such a test selection process could e.g. be combined with the on-the-fly test derivation algorithm we presented in [6]. Note that the coverage measures described in this section are defined on the basis of the specification, rather than on the implementation (which is considered to be a black box). The underlying assumption is that a higher coverage value is an indication of a higher test quality; as such, one would always aim at a coverage value of 1 (i.e. full coverage).

We assume that the execution of a set of *test cases* (see e.g. [16] for a definition) on an implementation has resulted in a number of *test runs*, which we assume can be represented by a prefix-closed set of extended traces. Let $\mathcal{L} = \langle S, s_0, \Sigma_I, \Sigma_U, \rightarrow\rangle$ be an LTS-specification; a state-coverage measure $P_s(R)$ of a set of executed test runs $R \subseteq \Sigma_\delta^*$ can be defined as the ratio between states that have potentially been covered by test runs from R, and the total number of reachable states:

$$P_s(R) =_{def} |\frac{| \bigcup_{\rho\in R} \;\; s_0 \,\textbf{after}\, \rho \;|}{| \; der(s_0) \; |}$$

State-coverage quickly becomes impractical when data plays a role, since the set of reachable states becomes extremely large or even infinite. This is exemplified by the Supplier STS: there is an infinite number of initial transitions leading to an infinite number of reachable states, since the underlying LTS model of the Supplier STS is infinitely branching in its initial state, effectively giving $P_s(R) = 0$ for all sets of test runs R. Note that this problem persists, even when we consider an alternative definition of P_s which relies on the total number of states that can be reached within a finite (known) number of steps.

A coverage measure that side-steps this problem is *location-coverage* for STSs. Let $\mathcal{S}(\iota) = \langle L, l_0, \mathcal{V}, \mathcal{I}, \Lambda_I, \Lambda_U, \rightarrow\rangle$ be an STS-specification (we assume it has semantics $\langle S, s_0, \Sigma_I, \Sigma_U, \rightarrow\rangle$); a location-coverage $P_l(R)$ of a set of executed test runs $R \subseteq \Sigma_\delta^*$ is defined as the ratio between locations that have potentially been covered by test runs from R, and the total set of reachable locations of $\mathcal{S}$:

$$P_l(R) =_{def} \frac{| \{l' {\in} L \mid \exists \rho {\in} R : \; \exists \iota' {\in} \mathfrak{U}^{\mathcal{V}} : \; (l', \iota') {\in} \, s_0 \,\textbf{after}\, \rho \} \;|}{| \; \{l' {\in} L \mid \exists \iota' {\in} \mathfrak{U}^{\mathcal{V}} : \; (l', \iota') {\in} der(s_0))\} \;|}$$

While the (in)finiteness of a state space is irrelevant for the location-coverage (in the usual case that L is finite), a major drawback of location-coverage is that e.g. a full coverage largely relies on control-flow; data is not considered on equal footing. In the Supplier STS, this means that $P_l(R) = 1$ does not imply that R has a test run $?\mathsf{rq}\langle p, 0\rangle$ (where $p {\in} \mathbb{N}$ is some instantiation), leading to a data-dependent quiescence observation.

A refinement of location-coverage that *does* treat data and control on equal footing is *symbolic state-coverage*. Let $n {\in} \mathbb{N}$ be the maximal length of a test run. The symbolic state-coverage $P_{ss}(R, n)$ of a set of executed test runs $R \subseteq \Sigma_\delta^*$ of length at most n is defined as the ratio between the symbolic states that have been covered by test runs from R, and the total set of (semantically) reachable symbolic states of $\mathcal{S}$ (using experiments of length n at most):

$$P_{ss}(R,n) =_{def} \frac{\mid \{(l,\varphi,\rho) \mid \exists\sigma{\in}\Lambda_\delta^{\leq n} :\ l_0 \xRightarrow{\sigma,\varphi,\rho}_\delta l \text{ and } [\![\{(\sigma,\varphi)\}]\!]_\iota \cap R \neq \emptyset\} \mid}{\mid \{(l,\varphi,\rho) \mid \exists\sigma{\in}\Lambda_\delta^{\leq n} : l_0 \xRightarrow{\sigma,\varphi,\rho}_\delta l \text{ and } [\![\{(\sigma,\varphi)\}]\!]_\iota \neq \emptyset\} \mid}$$

We leave it to the reader to check that in order to achieve $P_{ss}(R,n) = 1$, with $n > 1$ for the Supplier STS, the set R must also contain a test run $?\mathsf{rq}\langle p, 0\rangle$ (for some $p{\in}\mathbb{N}$). A test selection process aiming at a particular coverage using coverage measure P_{ss} could employ (subsets of) the set appearing in the denominator of P_{ss} to select test cases that reach symbolic states in this set.

7 Conclusions and Related Work

We have presented a symbolic implementation relation **sioco**, and proven its soundness and completeness w.r.t. the semantical **ioco** relation. The symbolic concepts that were needed to define **sioco** are not mere artefacts of the definition of **sioco**, but they have their own merits. We illustrated this by defining a test coverage measure that is based on symbolic states, which has advantages over coverage measures based on locations or semantic states. Similar advantages were found when investigating symbolic test case generation (not discussed in this paper), and, we expect to be able to reuse these concepts in e.g. test data selection.

To the best of our knowledge, this is the first approach that gives a fully symbolic implementation relation including quiescence. A closely related approach is described in [15], that uses a variant of a symbolic transition system and a weaker relation, e.g. they do not deal with quiescence. In [11] the problem of symbolic reachability analysis is approached with over-approximation techniques.

Also [7] presents a symbolic variation of the theme which is more focused on implementation issues. Their models are syntactically less expressive, e.g. inputs cannot directly be constrained, and the underlying implementation relation is not fully **ioco** (repetitive quiescence is missing). By having a simpler model without dedicated interaction variables, some computational tasks are easier to solve, for instance symbolic quiescence becomes quantifier-free.

Symbolic transitions systems are somewhat similar to *Statecharts* [9], and to their UML-variant called *State Machines* [14]. State Machines, though, tend to be applied in a synchronous setting, where inputs and outputs appear together on a single transition. This has consequences for compositionality issues, nondeterminism, etc., and corresponds to the semantical model of a *Mealy Machine* (also called *Finite State Machine* (FSM)). There is an important branch of formal testing which is based on Mealy Machines and their symbolic variant called *Extended Finite State Machine*, see [13] for a survey. Also the approach to testing in general differs, see e.g. [8] for a comparison. The testing approaches which are based on LTSs have instead an asynchronous nature, inputs and outputs appear here isolated on transitions. We hope that the presented framework can aid in embedding and reasoning about the many variations of LTS-based testing approaches which have been defined.

It is one of our main current research directions to investigate efficient implementations of the presented framework. One concrete instance is a Java-based test system for testing web services implementing the on-the-fly algorithm of [6] together with the symbolic coverage criteria as being indicated in Sect. 6.

References

1. A. Belinfante, J. Feenstra, R.G. de Vries, J. Tretmans, N. Goga, L. Feijs, S. Mauw, and L. Heerink. Formal test automation: A simple experiment. In G. Csopaki, S. Dibuz, and K. Tarnay, editors, *IWTCS'99*, pages 179–196. Kluwer Academic Publishers, 1999.
2. T. Bolognesi and E. Brinksma. Introduction to the ISO specification language Lotos. *Computer Networks*, 14(1):25–59, January 1988.
3. M. Broy, B. Jonsson, J.P. Katoen, M. Leucker, and A. Pretschner, editors. *Model-based Testing of Reactive Systems: Advanced Lectures*, volume 3472 of *LNCS*. Springer, 2005.
4. A.D. Brucker and B. Wolff. Symbolic test case generation for primitive recursive functions. In *FATES 2004*, volume 3395 of *LNCS*, pages 16–32. Springer-Verlag, 2005.
5. H. Ehrig, B. Mahr, F. Cornelius, M. Große-Rhode, and P. Zeitz. *Mathematisch-strukturelle Grundlagen der Informatik*. Springer, 2nd edition, 2001.
6. L. Frantzen, J. Tretmans, and T.A.C. Willemse. Test generation based on symbolic specifications. In *FATES 2004*, volume 3395 of *LNCS*, pages 1–15. Springer-Verlag, 2005.
7. C. Gaston, P. Le Gall, N. Rapin, and A. Touil. Symbolic execution techniques for test purpose definition. In M. Ü. Uyar, A. Y. Duale, and M. A. Fecko, editors, *TestCom 2006*, volume 3964 of *LNCS*, pages 1–18. Springer, 2006.
8. N. Goga. Comparing torx, autolink, tgv and uio test algorithms. In *SDL '01: Proceedings of the 10th International SDL Forum Copenhagen on Meeting UML*, pages 379–402, London, UK, 2001. Springer-Verlag.
9. D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.
10. C. Jard and T. Jéron. TGV: theory, principles and algorithms. In *IDPT '02*. Society for Design and Process Science, 2002.
11. B. Jeannet, T. Jéron, V. Rusu, and E. Zinovieva. Symbolic test selection based on approximate analysis. In *TACAS*, pages 349–364, 2005.
12. J. C. King. A new approach to program testing. In *Proceedings of the international conference on Reliable software*, pages 228–233, New York, NY, USA, 1975. ACM Press.
13. D. Lee and M. Yannakakis. Principles and methods of testing finite state machines - A survey. In *Proceedings of the IEEE*, volume 84, pages 1090–1126, 1996.
14. Object Management Group. *UML 2.0 Superstructure Specification*, ptc/03-08-02 edition. Adopted Specification.
15. V. Rusu, L. du Bousquet, and T. Jéron. An Approach to Symbolic Test Generation. In W. Grieskamp, T. Santen, and B. Stoddart, editors, *IFM'00*, volume 1945 of *LNCS*, pages 338–357. Springer-Verlag, 2000.
16. J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software—Concepts and Tools*, 17(3):103–120, 1996.