

A Conformance Testing Relation for Symbolic Timed Automata*

Sabrina von Styp¹, Henrik Bohnenkamp¹, and Julien Schmaltz^{2,3}

¹ Software Modeling and Verification (i2)
Department of Computer Science

RWTH Aachen University Aachen, Germany

² School of Computer Science, Open University of the Netherlands
Heerlen, The Netherlands

³ Institute for Computing and Information Sciences
Radboud University Nijmegen, The Netherlands

Abstract. We introduce Symbolic Timed Automata, an amalgamation of symbolic transition systems and timed automata, which allows to express nondeterministic data-dependent control flow with inputs and outputs and real-time behaviour. In particular, input data can influence the timing behaviour. We define two semantics for STA, a concrete one as timed labelled transition systems and another one on a symbolic level. We show that the symbolic semantics is complete and correct w.r.t. the concrete one. Finally, we introduce symbolic conformance relation *stioco*, which is an extension of the well-known *ioco* conformance relation. Relation *stioco* is defined using FO-logic on a purely symbolic level. We show that *stioco* corresponds on the concrete semantic level to Krichen and Tripakis' implementation relation *tioco* for timed labelled transition systems.

Keywords: Real-time conformance testing, symbolic execution, implementation relation, semantics, FO logics.

1 Introduction

Specification-based testing is a branch of model-based testing, where test-cases are derived automatically from a formal specification (given as a labelled transition system or a related formalism) and executed against real-life implementations. The distinction to many other instances of model-based testing is that the whole test-case derivation and test-execution process is described formally. This rigorous definition enables the proof of soundness of the approach. In particular, it is possible to show that the execution of a derived test-case does

* This work has been performed as part of the "Quantitative System Properties in Model-Driven-Design of Embedded Systems" (Quasimodo) project, supported by the Seventh Research Framework Programme of the European Commission. Grant agreement number: INFSO-ICT-214755.

not yield false positives, *i.e.*, test-failures, when the implementation is actually correct. To achieve this form of rigorosity, a formal criterion is needed which relates specifications to its correct implementations: the *conformance relation*. Specification-based testing has developed over the last two decades. A well-known representative is the *ioco framework* [19], where the specification is given as a labelled transition system (*LTS*) with input and output actions. The conformance relation is called *ioco*. The *ioco* relation expresses that an implementation may only produce outputs if those outputs are also produced by the specification. Additionally, *ioco* possesses a notation for silence, denoted *quiescence*. An implementation that is *ioco*-correct may only be quiescent if this is allowed by the specification. There exists several tools for the derivation of *ioco* test-cases, e.g. TorX [3], TGV [14] and AGEDIS TOOL SET [11]. Based on *ioco*, recently more expressive formalisms have been considered to serve as specifications. *Timed Automata* have been proposed as specification formalisms in several approaches for testing real-time behaviours [16,4,5]. Different notions of conformance have been defined on the basis of timed *LTS* (*TLTS*), *i.e.*, only on the semantic level. *Symbolic Transition Systems* (*STS*) [8,9] have been introduced to specify systems with input- and output-data. *STS* are *LTS* extended with a notion of data and data-dependent control flow based on first order logic. The symbolic representation of data in *STS* allows for infinite data domains without facing the problems of infinite branching and infinite state space. For *STS*, the implementation relation *sioco* has been developed, which is defined solely within the FO-Logic framework on *STS* level [9].

What does not exist yet is a combination of real-time and data. In this paper we take first steps in the direction of specification-based testing for systems combining input/output data with real-time aspects in a non-orthogonal way, *i.e.*, the input data can influence the real-time behaviour. In particular, we introduce a conformance relation which takes data and real-time into account.

Our contributions are (1) a new formalism – called Symbolic Timed Automata (*STA*) – for modelling reactive real-time systems with data input and output; (2) a concrete operational semantics (in terms of timed labelled transition systems) and a symbolic trace semantics for this formalism; (3) a family of conformance relations $stioco_{\mathcal{F}_s}$, which expresses a correctness criterion of input-enabled implementations, formulated as *STA*, with specifications, also given as *STA*; (4) a theorem stating that $stioco_{\mathcal{F}_s}$ coincides on the concrete semantical level with *tioco* of Krichen and Tripakis [16]. Our formalism allows the real-time behaviour to be influenced by inputs. While allowing nondeterministic *STA* in general, we restrict ourselves in this paper to *branching nondeterministic STA*, *i.e.*, without τ -steps.

Related Work. A detailed comparison of the different notions of conformance for real-time testing is given by Schmaltz et al. [17]. In the testing tool UPPAAL Tron [13], a pragmatic approach to combine data and time is implemented. It is possible to let clock constraints depend on integer variables. Moreover, global integer variables can be designated as input or output parameters to input or output actions. With this a notion of value passing is implemented.

However, values are limited to finite sub-sets of integers, which allows an explicit representation of data as actions on Timed Automata level. This approach is hardly formally described, least of all on a symbolic level. A similar approach is taken in JTorX [2].

STS and the implementation relation *sioco* have been introduced in [9]. This implementation relation is used in the java testing tool JAMBITION introduced by Frantzen *et al.* [7]. JAMBITION uses a random and on-the-fly approach to automatically test web services based on *STS* specifications. In order to simulate the *STS* in JAMBITION the Java library STSIMULATOR has been developed [18].

Another tool for symbolic testing is STG [6], an extension of TGV. It automatically derives symbolic test-cases from a given formal model and a test purpose. The issue of symbolic test-case generation and selection has been addressed in [21] and [15].

2 Timed Transition Systems and *Tioco*

A *timed labelled transition system (TLTS)* is a tuple $\langle S, Act, s_0, \rightarrow \rangle$ with S a set of states, $Act = Act_I \cup Act_U$ a disjoint union of two sets of input- and output-actions, s_0 the starting state, and $\rightarrow \subseteq S \times (Act \cup \mathbb{R}_{\geq 0}) \times S$ a transition relation, where the following conditions must hold: $\forall s, s', s'' \in S$ (i) $s \xrightarrow{0} s$; (ii) $s \xrightarrow{d} s' \xrightarrow{d'} s''$ if and only if $s \xrightarrow{d+d'} s''$; (iii) $s \xrightarrow{d} s'$ and $s \xrightarrow{d} s''$ implies $s' = s''$. In the following we consider a fixed *TLTS* $\langle S, Act, s_0, \rightarrow \rangle$, and identify it with its starting state s_0 . The generalised transition relation $\Rightarrow \subseteq S \times (Act \cup \mathbb{R}_{\geq 0})^* \times S$ is defined as the least relation satisfying the following rules: (i) $s \xRightarrow{\epsilon} s \forall s \in S$; (ii) $s \xRightarrow{\sigma \cdot d} s'$, if $s \xrightarrow{\sigma} s'' \xrightarrow{d} s'$ for $d \in \mathbb{R}_{\geq 0}$; (iii) $s \xRightarrow{\sigma \cdot a} s'$, if $s \xrightarrow{\sigma} s'' \xrightarrow{a} s'$, for $a \in Act$. We consider normalised traces where actions and delays strictly alternate, starting with a delay. It has been shown that this set characterises the set of all traces [5]. A timed trace is thus a sequence $\sigma \in (\mathbb{R}_{\geq 0} \cdot Act)^* \cdot (\mathbb{R}_{\geq 0} + \epsilon)$ such that $s_0 \xrightarrow{\sigma} s'$ for some $s' \in S$. The set of traces of *TLTS* S is noted **traces**(S). The set of states that can be reached from state s via a trace σ is denoted as $s \mathbf{after}_t \sigma$.

Definition 1 (after_t). Let $\langle S, Act, s_0, \rightarrow \rangle$ be a *TLTS* and $\sigma \in (\mathbb{R}_{\geq 0} \cdot Act)^* \cdot (\mathbb{R}_{\geq 0} + \epsilon)$. Then $s \mathbf{after}_t \sigma =_{def} \{s' \mid s \xrightarrow{\sigma} s'\}$.

Crucial for the definition of *tioco* is the set of delay and output labels of the outgoing transitions of a state.

Definition 2 (elapse(s) and out_t(s)). We define $elapse(s) =_{def} \{d \mid s \xrightarrow{d}\}$, and $out_t(s) =_{def} \{o \in Act_U \mid s \xrightarrow{o}\} \cup elapse(s)$. For $S' \subseteq S$, $out_t(S') =_{def} \bigcup_{s \in S'} out_t(s)$.

As in the *ioco* theory, we assume that implementations under test are input-enabled.

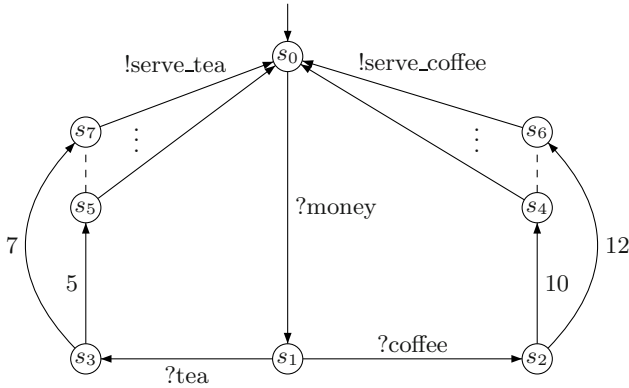


Fig. 1. TLTS Specification of a Beverage Vending Machine

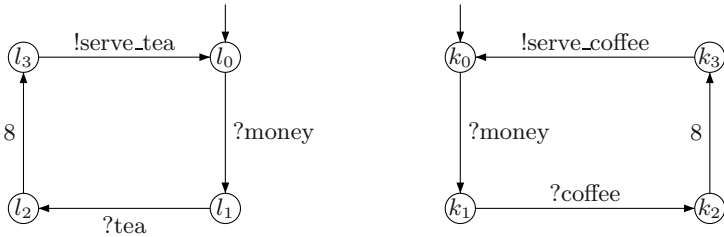


Fig. 2. Implementations of a Beverage Vending Machine

Definition 3 (Input-enabled TLTS). A TLTS $\langle S, Act, s_0, \rightarrow \rangle$ is called *input-enabled* if and only if for all $s \in S$ and all $i \in Act_I$: $s \xrightarrow{i}$.

With the introduced concepts we can define the family of implementation relations $tioco_{\mathcal{F}}$.

Definition 4 ($tioco_{\mathcal{F}}$). Let \mathcal{P} be an input-enabled TLTS, \mathcal{S} a TLTS, and $\mathcal{F} \subseteq \text{traces}(\mathcal{S})$. Then \mathcal{P} conforms to \mathcal{S} w.r.t. $tioco_{\mathcal{F}}$ (written $\mathcal{P} \text{ tioco}_{\mathcal{F}} \mathcal{S}$) if and only if the following holds: $\forall \sigma \in \mathcal{F} : \text{out}_t(\mathcal{P} \text{ after}_t \sigma) \subseteq \text{out}_t(\mathcal{S} \text{ after}_t \sigma)$.

Example 1. In Figure 1, a TLTS specifying a beverage vending machine is sketched. After inserting money, users can choose either tea or coffee. The former is produced between 5 and 7 time units after pushing the ?tea button. The latter is produced between 10 and 12 time units after pushing the ?coffee button (the time intervals are indicated by the dashed lines, which stand for a *continuum* of states). Figure 2 shows two implementations of beverage machines. None of these implementations conform to the specification according to the $tioco$ relation. The implementation on the left (starting state l_0) is too slow to produce tea. The implementation on the right (starting state k_0) is too fast to produce coffee. Formally, the reasons for non-conformance are (1) that $8 \in \text{elapsed}(l_0 \text{ after}_t ?money \cdot ?tea)$ but $8 \notin \text{elapsed}(s_0 \text{ after}_t ?money \cdot ?tea)$ and

(2) that $!serve_coffee \in \mathbf{out}_t(k_0 \mathbf{after}_t ?money \cdot ?coffee \cdot 8)$ but $!serve_coffee \notin \mathbf{out}_t(s_0 \mathbf{after}_t ?money \cdot ?coffee \cdot 8)$.

3 Symbolic Timed Automata

A symbolic timed automaton (*STA*) combines a timed automaton [1,12] with a *symbolic transition system* (*STS*), as introduced in [8,9]. *STS* extend labelled transition systems with *variables*, *input- and output-parameters* associated with input- and output-actions, *guards*, which are first-order logic formulas controlling the enabledness of transitions, and *variable updates*, which manipulate the values of variables while performing transitions. In *STA*, the concepts of timed automata, namely clocks, guards, invariants and clock resets, are integrated. In particular, clock constraints become just another sort of FO formulas.

3.1 An Example

Figure 3 shows an *STA* modelling a beverage vending machine. The machine accepts money in bills (parameter x of input $?money$) and returns change in coins (parameter x of output $!serve$). If there is not enough change in the machine on a bill, the bill is returned (output $!return$). Otherwise, the user can choose a beverage, the change is returned, and the beverage served (parameter y of output $!serve$). Variables x, y, t are *interaction variables*. Interaction variables represent the possible values that can be passed through input and output actions. Variables $i, q, \mathbf{change}, \mathbf{beverage}, \mathbf{money}, \mathbf{time}$ are *location variables* — *i.e.*, the store of the *STA* — and c is a clock, conceptually identical with a clock of a timed automaton. The switch from l_0 to l_1 is labelled with input action $?money$, with parameter x . Only if the value of x is larger or equal than 1 the switch can be executed. In that case, \mathbf{change} is mapped to $x - 1$ and \mathbf{money} to x . From location l_1 , the money is returned immediately, if $\mathbf{change} > q$, *i.e.*, the required change is not available. In that case, \mathbf{money} and \mathbf{change} are both mapped to 0. If $\mathbf{change} < q$, it is possible to choose a beverage via input $?choice$, where the type of beverage is communicated via y , and t stands for the time after which the machine shall serve the beverage. t could for example express the brewing time of a tea or whether an espresso should be short or long. Input t is assigned to location variable \mathbf{time} , which occurs in the invariant of location l_2 and the clock guard of the switch $l_2 \rightarrow l_0$. Together they ensure that location l_2 is left after exactly \mathbf{time} time units. In that case, output $!serve$ with output parameters y and x is sent, where $x = \mathbf{change}$ and $y = \mathbf{beverage}$ is required by the guard. Location variables \mathbf{money} and \mathbf{change} are then mapped to 0, and q is mapped on $q - \mathbf{change}$. Note that location variable i , used as bound in the invariant of l_1 and the guard leading back to l_0 , is never explicitly set in the *STA*. It is assumed that location variables (thus also i) are initialised when starting the *STA*. Different initialisations might define different behaviours of the same *STA*.

In this example, the time at which the transition from location l_2 to l_0 can be taken is controlled by input data t given in the previous transition (from l_1

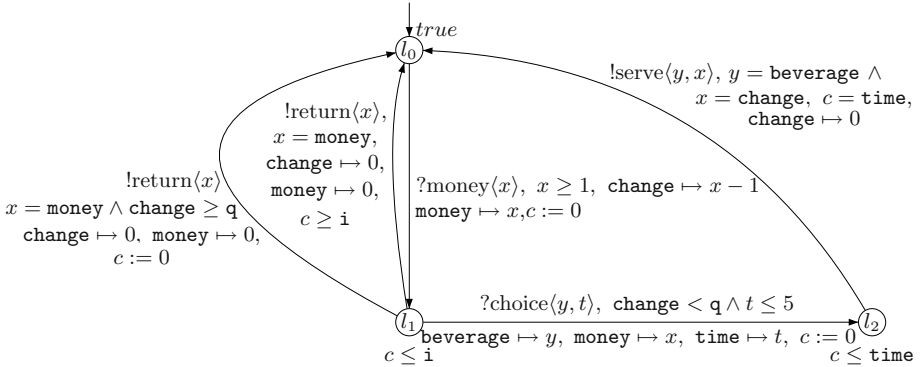


Fig. 3. STA Beverage Vending Machine

to l_2). In an STA, location variables are allowed to serve as bounds in clock guards and invariants. This is the only instance in which we allow an interaction between time (clocks) and data (variables). Interaction variables cannot be used to control time. This prevents the case where the time at which a transition can be taken depends on the value of the parameter of the action of that same transition. In our example, time t must first be stored in local variable `time` before it can be used to control time.

3.2 Definition and Concrete Semantics of STA

An STA is defined over a two sorted FO structure $FO_\Sigma = (\{\mathcal{U}_d, \mathcal{U}_t\}, \{r_\Sigma \mid r \in \mathcal{R}\}, \{f_\Sigma \mid f \in \mathcal{F}\})$, where \mathcal{U}_d denotes the universe for data and \mathcal{U}_t the time domain (e.g., the reals). The signature contains constant 0_t , binary addition $+$, and relations $\leq, <$, declared for all combinations $\{t, d\} \times \{t, d\}$, i.e., comparisons between time and data.

Clock-constraints are FO formulas using the above relations. Let \mathcal{C} be the set of clocks and Var be a set of variables. An *atomic clock-constraint* is a formula of the form $b_l \prec x \prec b_u$ for $x \in \mathcal{C}$, $\prec \in \{<, \leq\}$, and $b_l, b_u \in \mathbb{N}_{FO} \cup Var$, where \mathbb{N}_{FO} stands here for the representation of natural numbers on the logic level¹. Clock constraints are conjunctions of atomic clock constraints. The set of all clock constraints over clock set \mathcal{C} and variables Var is denoted by $\mathcal{B}(\mathcal{C}, Var)$, and $\mathcal{B}(\mathcal{C})$, if $Var = \emptyset$. Bounds of atomic clock constraints can be variables. We denote by $\mathfrak{T}(V)$ the set of all terms over a variable set V and by $\mathfrak{F}(V)$ the set of all FO formulas. Function $\rho : V \rightarrow \mathfrak{T}(V)$ is called a term mapping.

Definition 5 (Symbolic Timed Automaton). A symbolic timed automaton is a tuple $\mathcal{A} = \langle L, l_0, \mathcal{V}, \mathcal{I}, \mathcal{G}, \text{type}, \mathcal{C}, \text{Inv}, \rightarrow \rangle$ with L being a finite set of locations, $l_0 \in L$ is the initial location, \mathcal{V} and \mathcal{I} disjoint sets of location and interaction variables, \mathcal{G} is a set of gates, $\text{type} : \mathcal{G} \rightarrow 2^{\mathcal{I}}$ assigns sets of interaction

¹ Other literals are imaginable as bounds, depending on the choice of \mathcal{U}_d .

variables to gates, \mathcal{C} is a set of clocks, $Inv : L \rightarrow \mathcal{B}(\mathcal{C}, \mathcal{V})$ assigns a clock invariant to a location, and $\rightarrow \subseteq L \times \mathcal{G} \times \mathfrak{F}(Var) \times \mathcal{B}(\mathcal{C}, \mathcal{V}) \times \bigcup_{V \subseteq \mathcal{V}} \mathfrak{I}(Var)^V \times 2^{\mathcal{C}} \times L$ is the transition relation, where $Var = \mathcal{I} \cup \mathcal{V}$.

As usual, we write $l \xrightarrow{\gamma, \varphi, g, \rho, r} l'$ to denote $(l, \gamma, \varphi, g, \rho, r, l') \in \rightarrow$, where γ is the gate, φ the data guard, g the clock guard, ρ the update function, and r the clock reset.

A variable valuation is a function $\vartheta : Var \rightarrow \mathfrak{U}_d$ from variables to concrete values in the universe \mathfrak{U}_d . A clock valuation is a function $u : \mathcal{C} \rightarrow \mathfrak{U}_t$. We denote with $[\mathcal{C} \mapsto 0]$ the constant 0 clock valuation. For $d \in \mathfrak{U}_t$, we define $(u + d)(c) = u(c) + d$. If ϑ is a variable valuation and $C \in \mathcal{B}(\mathcal{C}, Var)$, then we denote with $C[\vartheta]$ the clock constraint, where every occurrence of a variable $x \in Var$ is replaced by $\vartheta(x)$; thus, $C[\vartheta] \in \mathcal{B}(\mathcal{C})$. We write $u \models C$, if clock valuation u satisfies clock constraint $C \in \mathcal{B}(\mathcal{C})$, i.e., if the relational expression obtained by replacing all occurrences of clock names c by $u(c)$ evaluates to true. If $r \subseteq \mathcal{C}$, then $u[r \mapsto 0](c) = 0$, if $c \in r$, and $u[r \mapsto 0](c) = u(c)$, otherwise. The semantics of an STA is given as a TLTS, defined as follows.

Definition 6. Let $\mathcal{A} = \langle L, l_0, \mathcal{V}, \mathcal{I}, \mathcal{G}, \mathbf{type}, \mathcal{C}, Inv, \rightarrow \rangle$ be an STA. Its TLTS semantics in the context of an initial valuation $\iota \in \mathfrak{U}^{\mathcal{V}}$ of location variables and $\zeta_0 = [\mathcal{C} \mapsto 0]$ for clocks, is a TLTS $\llbracket \mathcal{A} \rrbracket_{\iota} = \langle S, Act, s_0, \rightarrow \rangle$, where $S = L \times \mathfrak{U}_d^{\mathcal{V}} \times \mathfrak{U}_t^{\mathcal{C}}$, $s_0 = (l_0, \iota, \zeta_0)$, $Act = \bigcup_{\gamma \in \mathcal{G}} (\{\gamma\} \times \mathfrak{U}^{\mathbf{type}(\gamma)})$, and \rightarrow is defined as the least set of transitions derivable by the following rules:

$$\langle Delay \rangle \frac{\zeta \models Inv(l)[\vartheta] \quad \forall d' \leq d : \zeta + d' \models Inv(l)[\vartheta]}{(l, \vartheta, \zeta) \xrightarrow{d} (l, \vartheta, \zeta + d)} (d \in \mathbb{R}_{\geq 0})$$

$$\langle Action \rangle \frac{l \xrightarrow{\gamma, \varphi, \rho} l' \quad \vartheta \cup \varsigma \models \varphi \quad \zeta' \models Inv(l')[\vartheta']}{(l, \vartheta, \zeta) \xrightarrow{(\gamma, \varsigma(\mathbf{type}(\gamma)))} (l', \vartheta', \zeta')}$$

with $\vartheta' = ((\vartheta \cup \varsigma)_{ev} \circ \rho)_{\mathcal{V}}$ and $\zeta' = ((\zeta)_{ev} \circ \rho)_{\mathcal{C}}$.

With $(\cdot)_{ev}$ we denote the lifting of a variable/clock valuation to terms. With $(\cdot)_{\mathcal{V}}$ and $(\cdot)_{\mathcal{C}}$ we denote the restriction of a valuation or term mapping to sets \mathcal{V} and \mathcal{C} . In Definition 6 we see that we can only delay if all clock valuations ζ before the delay and all clock valuations ζ' after the delay satisfy the invariant of location l . Delaying has no influence on the location itself or on the valuation of location variables ϑ . To take a switch from l to l' with gate γ , the constraint φ over variables and clocks and the invariant of the next location l' have to be satisfied. Important is that the invariant of l' has to be satisfied after the clocks in ρ have been set to zero.

4 Symbolic Trace Semantics for STA

Inspired by Frantzen *et al.* [9], we define a symbolic trace semantics for STA, which is sound and complete *w.r.t.* the TLTS semantics described before. We use *delay-variable* d to represent time symbolically. Variable d is of sort t and

represents delaying in a location. We define the following two notations. First, for $r \subseteq \mathcal{C}$, we denote with $FO(r)$ a term-mapping that maps all clocks $c \in r$ to zero, *i.e.*, $FO(r) : c \mapsto 0_t$ for $c \in r$ and $c \mapsto c$ otherwise. Second, $\varrho : \mathcal{C} \rightarrow \mathfrak{T}(\mathcal{C} \cup \mathcal{T})$ is a partial term-mapping that expresses the passing of time: $\varrho : c \mapsto c + d$, *i.e.*, delaying is described by adding delay variable d to all clocks $c \in \mathcal{C}$.

The symbolic trace semantics is expressed by transition relation $l \xrightarrow{\sigma, \varphi, \rho} l'$, where φ denotes the logical condition which needs to be fulfilled to reach location l' from l with symbolic trace σ , and ρ is a term mapping that denotes symbolically the possible variable and clock valuations after l' has been reached with σ . Relation \Rightarrow thus describes the symbolic execution of the *STA*. To define φ and ρ , we introduce history variables, *i.e.*, variables that allow to distinguish between different input- and output values communicated over the gates, and time delays spent in locations. The history variables are an infinite number of “copies” i_1, i_2, i_3, \dots of each interaction variable $i \in \mathcal{I}$, and delay history variables d_1, d_2, d_3, \dots for variable d . We define $\mathcal{I}_n = \{i_n \mid i \in \mathcal{I}\}$ for $n \geq 1$ and $\widehat{\mathcal{I}} =_{\text{def}} \bigcup_{n=1}^{\infty} \mathcal{I}_n$. Similarly, we consider sets $\mathcal{T}_n = \{d_n\}$ for $(n \geq 1)$ and $\widehat{\mathcal{T}}$ defined analogously to $\widehat{\mathcal{I}}$. Let $\mathcal{H}_n =_{\text{def}} \mathcal{I}_n \cup \mathcal{T}_n$ and $\widehat{\mathcal{H}} =_{\text{def}} \widehat{\mathcal{I}} \cup \widehat{\mathcal{T}}$, $\mathcal{H} =_{\text{def}} \mathcal{I} \cup \mathcal{T}$. We define renaming bijections $r_n : \mathcal{H} \rightarrow \mathcal{H}_n$ with $r_n(v) = v_n$ for all $v \in \mathcal{H}$, and $n \in \mathbb{N}$. Function $s^{\gg i} : \widehat{\mathcal{H}} \rightarrow \widehat{\mathcal{H}}$ is defined as $s^{\gg i} : x \mapsto x_{n+i}$ for all $x = x_n \in \widehat{\mathcal{H}}$. In the following, the entirety of all relevant variables is the set $\widehat{\text{Var}} =_{\text{def}} \mathcal{V} \cup \mathcal{C} \cup \mathcal{H} \cup \widehat{\mathcal{H}}$, and $\text{Var} =_{\text{def}} \widehat{\text{Var}} \setminus \widehat{\mathcal{H}}$.

If $\rho \in (\mathfrak{T}(\widehat{\text{Var}}))^{\widehat{\text{Var}}}$ and $t \in \mathfrak{T}(\widehat{\text{Var}})$, we denote by $t[\rho]$ the term obtained by substituting all variables x occurring in t by $\rho(x)$. Analogously for all formulas $\varphi \in \mathfrak{F}(\widehat{\text{Var}})$, where all *free* variables are substituted.

4.1 Symbolic Trace Semantics

The symbolic trace semantics of an *STA* $\langle L, l_0, \mathcal{V}, \mathcal{I}, \mathcal{G}, \text{type}, \mathcal{C}, \text{Inv}, \rightarrow \rangle$ is then given by the transition relation $\Rightarrow \subseteq L \times ((d \cdot \mathcal{G})^* \cdot \{d, \varepsilon\}) \times \mathfrak{F}(\widehat{\text{Var}}) \times (\bigcup_{V \subseteq \mathcal{V}} \mathfrak{T}(\widehat{\text{Var}})^V \cup \bigcup_{\mathcal{C} \subseteq \mathcal{C}} \mathfrak{T}(\mathcal{C} \cup \widehat{\mathcal{T}})^{\mathcal{C}}) \times L$, which is defined as follows:

Definition 7 (Generalised Switch Relation for *STA*).

$$(d) \frac{}{l \xrightarrow{d, \kappa[\varrho][r_1], \varrho[r_1]} l}$$

$$(Sd) \frac{l \xrightarrow{\sigma \cdot \gamma, \varphi, \rho} l'}{\sigma \cdot \gamma \cdot d, \varphi \wedge \kappa'[\varrho][r_n][\rho], \Theta(\rho, r_n, \varrho)} l' \quad (S\gamma) \frac{l \xrightarrow{\sigma \cdot d, \varphi, \rho} l'' \quad l'' \xrightarrow{\gamma, \psi, \pi} l'}{\sigma \cdot d \cdot \gamma, \varphi \wedge \psi[r_n][\rho] \wedge \kappa'[\pi][\rho], \Theta(\rho, r_n, \pi)} l'$$

where $n = |\sigma| + 2$, $\Theta(a, b, c) = c[b][a]$, $\kappa \equiv \text{Inv}(l)$, and $\kappa' \equiv \text{Inv}(l')$.

Rule (d) states that delaying in a location l is possible as long as formula $\kappa[\varrho][r_n]$, *i.e.*, the historised and updated invariant of l , is satisfied. The clocks change according to term mapping $\varrho[r_1]$.

Rule (Sd) states that if location l' is reached from l with trace $\sigma \cdot \gamma$ under condition φ , then the condition to delay further in l' is the conjunction of φ and $\kappa[\varrho][r_n][\rho]$. As an example, we assume that $n = 7$, $\kappa \equiv c \leq v$, where c is

a clock, and v a location variable, and $\rho(c) = d_1 + d_3 + d_5$, $\rho(v) = 4$. Then $\kappa[\varrho] \equiv c + d \leq v$, $\kappa[\varrho][r_7] \equiv c + d_7 \leq v$, and $\kappa[\varrho][r_7][\rho] \equiv d_1 + d_3 + d_5 + d_7 \leq 4$.

Similarly, Rule $(S\gamma)$ states that if l'' is reached from l with $\sigma \cdot d$ under condition φ , then the condition to reach l' from l with trace $\sigma \cdot d \cdot \gamma$ is the conjunction of φ and $\psi[r_n][\rho] \wedge \kappa[\pi][\rho]$. Formula $\psi[r_n][\rho]$ is the ‘‘historised’’ enabling condition of switch $l'' \xrightarrow{\gamma, \psi, \pi} l'$, with all variables and clocks substituted according to ρ . Formula $\kappa[\pi][\rho]$ is the historised invariant of l' and expresses an extra condition on the symbolic clock valuations under which l' may be entered.

In both rules (Sd) and $(S\gamma)$, the new update mapping Θ for variables and clocks is obtained by the concatenation of the current update mapping ρ with the renaming function r_n and the variable update π for $(S\gamma)$ or the clock update ϱ for (Sd) . If, for example, we assume $\rho(c) = d_1 + d_3$, $r_n = 5$, then $\varrho[r_n][\rho](c) = ([\rho] \circ [r_n] \circ \varrho)(c) = d_1 + d_3 + d_5$.

4.2 Symbolic States and Relation to *TLTS* Semantics

To record under which conditions a location can be reached and with what potential variable valuations, we introduce *symbolic states*. Whenever $l \xrightarrow{\sigma, \varphi, \rho} l'$, tuple (l', φ, ρ) is a symbolic state. If in φ and ρ only history variables with an index up to at most i occur, we note this fact by indexing the symbolic state with i , *i.e.*, in this example $(l', \varphi, \rho)_i$.

A symbolic state (l, φ, ρ) defines implicitly a set of concrete states $\llbracket (l, \varphi, \rho) \rrbracket_\iota$ for $\iota \in \mathfrak{U}_d^V$ (as defined in the *TLTS* semantics above). Let $v \in \mathfrak{U}_d^T$ and $\varpi \in \mathfrak{U}_t^T$. Then $\llbracket (l, \varphi, \rho) \rrbracket_{\iota, v, \varpi \cup [C \mapsto 0]} =_{\text{def}} \{(l, ((\iota \cup v)_{\text{ev}} \circ \rho)_{\mathcal{V}}, ((\varpi \cup [C \mapsto 0])_{\text{ev}} \circ \rho)_{\mathcal{C}}) \mid \iota \cup v \cup \varpi \cup [C \mapsto 0] \models \varphi\}$. Note that $\llbracket (l, \varphi, \rho) \rrbracket_{\iota, v, \varpi}$ is either a singleton or empty.

Also traces $\sigma \in (d \cdot \mathcal{G})^*$ have an interpretation on the semantic level. Let $\chi \in \mathfrak{F}(\widehat{\mathcal{H}} \cup \mathcal{V} \cup \mathcal{C})$. Then we call (σ, χ) an extended trace. χ can be chosen freely. The set of all symbolic extended traces of an *STA* \mathcal{S} is defined by the following set: $\mathcal{E}Traces(\mathcal{S}) = \{(\sigma, \chi) \mid l_0 \xrightarrow{\sigma, \varphi, \rho} l', \chi \in \mathfrak{F}(\widehat{\mathcal{H}} \cup \mathcal{V} \cup \mathcal{C})\}$. Defining ι, v, ϖ as above, $\llbracket (\sigma, \chi) \rrbracket_{\iota, v, \varpi \cup [C \mapsto 0]} = \{\mathbf{etraces}_{v, \varpi}(\sigma) \mid \iota \cup v \cup \varpi \models \chi\}$, where $\mathbf{etraces}_{v, \varpi}$ is defined inductively as follows:

$$\mathbf{etraces}_{v, \varpi}(\epsilon) = \epsilon$$

$$\mathbf{etraces}_{v, \varpi}(\sigma \cdot g) = \mathbf{etraces}_{v, \varpi}(\sigma) \cdot (g, v(r_{\text{length}(\sigma)+1}(\text{type}(g))))$$

$$\mathbf{etraces}_{v, \varpi}(\sigma \cdot d) = \mathbf{etraces}_{v, \varpi}(\sigma) \cdot \varpi(d_{\text{length}(\sigma)+1}).$$

The following two theorems state that the interpretations of symbolic states and extended traces are correct *w.r.t.* the *TLTS* semantics. Variable mapping id is defined as $\text{id}(x) = x$.

Theorem 1 (Soundness). *For all $\varpi \in \mathfrak{U}_t^T$, $\iota \in \mathfrak{U}_d^V$ and $v \in \mathfrak{U}_d^T$ it holds that $l \xrightarrow{\sigma, \varphi, \rho} l'$ and $\iota \cup v \cup \varpi \cup [C \mapsto 0] \models \varphi$ implies $\llbracket (l, \top, \text{id}) \rrbracket_{\iota, v, \varpi \cup [C \mapsto 0]} \xrightarrow{\llbracket (\sigma, \varphi) \rrbracket_{\iota, v, \varpi \cup [C \mapsto 0]}} \llbracket (l', \varphi, \rho) \rrbracket_{\iota, v, \varpi \cup [C \mapsto 0]}$*

Theorem 2 (Completeness). *For all semantical states $(l, v, \zeta) \in L \times \mathfrak{U}_d^V \times \mathfrak{U}_t^C$ such that $(l_0, \iota, [C \mapsto 0]) \xrightarrow{\bar{\sigma}} (l, v, \zeta)$ for $\iota \in \mathfrak{U}_d^V$ and some timed trace $\bar{\sigma}$, there is a*

valuation $v' \in \mathfrak{U}_d^{\widehat{T}}$, $\varpi \in \mathfrak{U}_t^{\widehat{T}}$ and a transition $l_0 \xrightarrow{\sigma, \varphi, \rho} l$ such that $\iota \cup v' \cup \varpi \cup [\mathcal{C} \mapsto 0] \models \varphi$, $\bar{\sigma} = \llbracket (\sigma, \varphi) \rrbracket_{l, v', \varpi \cup [\mathcal{C} \mapsto 0]}$ and $(l, v, \zeta) = \llbracket (l, \varphi, \rho) \rrbracket_{l, v', \varpi \cup [\mathcal{C} \mapsto 0]}$.

The proofs of Theorems 1 and 2 can be found in [20].

5 *Stioco* - A Symbolic Timed Implementation Relation

Taking the symbolic trace semantics in Definition 7 as a foundation, we define symbolic implementation relation *stioco*. This definition is based on two central notions: **after**, a function which returns the symbolic states reachable with an extended trace, and **out**, the outputs that can potentially be observed from a set of symbolic states. The big difference is that outputs are accompanied by FO formulas which state the conditions under which outputs can be observed.

Definition 8 (after). Let $(l, \varphi, \rho)_i$ be a symbolic state with index i and (σ, χ) an extended trace with n the length of σ . Then **after** is defined as

$$(l, \varphi, \rho)_i \mathbf{after}(\sigma, \chi) =_{\text{def}} \{(l', \varphi'(\psi), \rho'(\pi)) \mid l \xrightarrow{\sigma, \psi, \pi} l'\},$$

where $\varphi'(\psi) = \varphi \wedge ((\psi \wedge \chi)[s^{\gg i}])[\rho]$, and $\rho'(\pi) = ([\rho] \circ [s^{\gg i}] \circ \pi)$.

$\varphi'(\psi)$ is the conjunction of φ and the condition $\psi \wedge \chi$, where every index of a history variable is increased by i and every clock and location variable is substituted according to ρ . $\rho'(\pi)$ is the symbolic variable valuation of the location variables and clocks after (σ, χ) has been executed. Note that the symbolic states in $(l, \varphi, \rho)_i \mathbf{after}(\sigma, \chi)$ have index $i + n$.

Symbolic observations are tuples (γ, φ, ψ) , where $\gamma \in \mathcal{G}_{U_a}$, φ is a general enabling condition for γ and ψ is a special enabling condition. φ and ψ are thus both formulas, which are however defined over different variable sets: $\varphi \in \mathfrak{F}(\widehat{Var})$ and $\psi \in \mathfrak{F}(Var)$. The set of symbolic observations, denoted as \mathcal{O} , is thus defined as $\mathcal{O} =_{\text{def}} \mathcal{G}_{U_a} \times \mathfrak{F}(\widehat{Var}) \times \mathfrak{F}(Var)$.

The **out**-set of a symbolic state is defined, similar to \mathbf{out}_t for *TLTS*, as the union of delays and output actions that can be made in the symbolic state. We use symbolic observations in order to symbolically represent the conditions under which an output or a delay may be observed.

Definition 9 (out). Let (l, φ, ρ) be a symbolic state. Then $\mathbf{out}((l, \varphi, \rho))$ is a set of symbolic observations, defined as follows.

$$\mathbf{out}((l, \varphi, \rho)) =_{\text{def}} \{(\gamma, \varphi, \psi[\rho] \wedge \text{Inv}(l')[\pi][\rho]) \in \mathcal{O} \mid \exists \gamma \in \mathcal{G}_U, \psi, \pi, l' : \\ l \xrightarrow{\gamma, \psi, \pi} l'\} \cup \{(d, \varphi, \text{Inv}(l)[\varrho][\rho]) \in \mathcal{O}\}.$$

We define $\mathbf{out}(\mathcal{Q}) =_{\text{def}} \bigcup_{(l, \varphi, \rho) \in \mathcal{Q}} \mathbf{out}((l, \varphi, \rho))$, for \mathcal{Q} a set of symbolic states.

The following definition of *stioco* is in essence very similar to the definition of *tioco*: there, an implementation conforms to the specification *w.r.t.* *tioco*, if, whenever the implementation, after the execution of a certain timed trace σ , produces an output of a certain kind or a delay of a certain length, then also the

specification, after the execution of σ , must be able to produce the same output or delay. In the case of *stioco*, however, this condition is expressed logically in terms of an FO formula, which says the following: the implementation conforms to the specification *w.r.t. stioco*, if, whenever the logical conditions are satisfied for the implementation to produce an output or delay, then also the specification satisfies the conditions to produce the same output or delay. It is therefore necessary to collect all conditions that lead to an observation. This is done by formula ϕ defined as follows.

Definition 10. *Let $\gamma \in \mathcal{G}_{U_d}$, $l \in L$, and σ a trace. Then $\phi(\gamma, l, \sigma)$ is an FO formula defined as*

$$\phi(\gamma, l, \sigma) =_{def} \bigvee \{ \varphi \wedge \psi \mid (\gamma, \varphi, \psi) \in \mathbf{out}((l, \top, \mathbf{id}) \mathbf{after} (\sigma, \top)) \}.$$

Formula ϕ is a disjunction of all conditions that lead to an observation after a trace σ . There is a valuation such that (1) at least one sub-term $\varphi \wedge \psi$ of the disjunction is satisfied, (2) formula ϕ holds, and (3) we know that there is a least one symbolic observation that can be observed after executing σ .

The *tioco*-relation is defined for input-enabled implementations. We define an STA \mathcal{S} to be input-enabled if and only if its semantics $\llbracket \mathcal{S} \rrbracket_i$ is an input-enabled TLTS.

Definition 11 (*stioco*). *Let $\mathcal{S}(\iota_s) = (L_s, l_s, \mathcal{V}_s, \mathcal{I}, \mathcal{G}, \mathcal{C}_S, \mathit{Inv}, \rightarrow)$ be an initialised STA (the specification) $\mathcal{A}_S, \mathcal{F}_s \subseteq \mathcal{ETraces}(\mathcal{S})$ and let $\mathcal{P}(\iota_p) = (L_p, l_p, \mathcal{V}_p, \mathcal{I}, \mathcal{G}, \mathcal{C}_P, \mathit{Inv}, \rightarrow)$ be an input-enabled implementation given as an STA, with $\mathcal{V}_p \cap \mathcal{V}_s = \emptyset$ and $\mathcal{C} = \mathcal{C}_S \cup \mathcal{C}_p$ the set of all clocks. \mathcal{P} conforms to \mathcal{S} with respect to $\mathit{stioco}_{\mathcal{F}_s}$ (written as $\mathcal{P} \mathit{stioco}_{\mathcal{F}_s} \mathcal{S}$.) if and only if the following holds. $\forall (\sigma, \chi) \in \mathcal{F}_s, \gamma \in \mathcal{G}_{U_d} :$*

$$(\iota_p)_{\mathcal{V}_p} \cup (\iota_s)_{\mathcal{V}_s} \cup [\mathcal{C} \mapsto 0] \models \underbrace{\bar{\forall}_{\hat{\mathcal{H}} \cup \mathcal{H}} (\phi(l_p, \gamma, \sigma) \wedge \chi \rightarrow \phi(l_s, \gamma, \sigma))}_{(*)}.$$

The heart of this definition lies in the universally quantified formula (*). At the symbolic level, we do not look at concrete outputs but at symbolic constraints defining a set of possible concrete actions. On the left-hand side of the implication we have for the implementation a conjunction of a disjunction of the symbolic constraints accumulated after trace σ and restriction χ which can be used to prune the symbolic execution. On the right-hand side, we have the disjunction of all accumulated symbolic constraints for the specification. The implication states that the constraints accumulated for the implementation imply the constraints accumulated by the specification.

Example 2. We consider two instances, \mathcal{S}_1 and \mathcal{S}_2 , of the STA shown in Figure 3. Assuming the following instantiations for variable i : $i := 8$ for \mathcal{S}_1 and $i := 10$ for \mathcal{S}_2 we get that $\mathcal{S}_1 \mathit{stioco} \mathcal{S}_2$. Since whenever the conditions for \mathcal{S}_1 to produce an output or for delaying are satisfied the conditions for \mathcal{S}_2 are also satisfied. However this does not hold for $\mathcal{S}_2 \mathit{stioco} \mathcal{S}_1$. The condition for an output of location l_1 for STA \mathcal{S}_2 is $\phi_{\mathcal{S}_2} := (d_3 \leq 10 \wedge (x = \mathbf{money} \wedge (\mathbf{change} \geq \mathbf{q} \vee d_3 \geq 10)))$ and $\phi_{\mathcal{S}_1} := (d_3 \leq 8) \wedge (x = \mathbf{money} \wedge (\mathbf{change} \geq \mathbf{q} \vee d_3 \geq 8))$ for \mathcal{S}_1 . Therefore $\phi_{\mathcal{S}_2} \not\rightarrow \phi_{\mathcal{S}_1}$. Thus, formula * in Definition 11 for *stioco* is violated.

The following theorem states that *stioco* corresponds to *tioco* on the semantical level. The proof of Theorem 3 is given in [20].

Theorem 3.

Let $\mathcal{P}(\iota_{\mathcal{P}})$ be an input-enabled initialised STA $\mathcal{S}(\iota_{\mathcal{S}})$ be an initialised STA. Let \mathcal{F}_s be a set of delayed symbolic extended traces for \mathcal{S} . Then:

$$\mathcal{P}(\iota_{\mathcal{P}}) \text{ stioco}_{\mathcal{F}_s} \mathcal{S}(\iota_{\mathcal{S}}) \Leftrightarrow \llbracket \mathcal{P} \rrbracket_{\iota_{\mathcal{P}}} \text{ tioco}_{\llbracket \mathcal{F}_s \rrbracket_{\iota_{\mathcal{S}}}} \llbracket \mathcal{S} \rrbracket_{\iota_{\mathcal{S}}}.$$

6 Conclusions

We presented a symbolic framework for timed automata combined with symbolic transitions systems. We defined an implementation relation *stioco* on STA which coincides with *tioco* [16] on the semantical level. To define such an implementation relation we provided symbolic trace executions and symbolic observations. It must be noted that, since timed automata are a subclass of STA, *stioco* is also a symbolic implementation relation expressing *tioco* on a symbolic level for timed automata.

The interaction between time and data in this paper is restricted to the influence that data inputs can have on the timing behaviour of the considered STA. This was expressed by allowing location variables to serve as bounds in clock constraints and invariants. More and different interactions between time and data are imaginable, for example, by assigning clock valuations to location variables, *i.e.*, by keeping historic information about the occurrence time of events in the STA. In principle, this extension could also be encoded in the first-order logical framework. However, even for the more restricted case considered in this paper, it is necessary to investigate first whether the obtained formalism is not already too expressive to be useful for practical testing, in terms of decidability of the forward reachability problem. A suitable subclass of FO logic might have to be identified to ensure this and to be able to apply the provided theory on practical applications. This would encompass the development of an algorithm for automatic test-case generation and test-execution.

Our theory is restricted to systems without τ -transitions. Adding those would be straightforward, in a similar way as it has been done for STS in [9], although special care has to be taken to combine successive delays. This will be part of future investigations.

Acknowledgments. We thank Lars Frantzen for helpful discussions and a preliminary chapter of his PhD thesis on STS.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. TCS 126(2), 183–235 (1994)
2. Belinfante, A.: JTorX: A tool for on-line model-driven test derivation and execution. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 266–270. Springer, Heidelberg (2010)

3. Belinfante, A., Feenstra, J., de Vries, R.G., Tretmans, J., Goga, N., Feijs, L.M.G., Mauw, S., Heerink, L.: Formal test automation: A simple experiment. In: IWTCS, pp. 179–196 (1999)
4. Bohnenkamp, H., Belinfante, A.: Timed testing with TorX. In: Fitzgerald, J., Hayes, I.J., Tarlecki, A. (eds.) FM 2005. LNCS, vol. 3582, pp. 173–188. Springer, Heidelberg (2005)
5. Brandán Briones, L., Brinksma, H.: A test generation framework for *quiescent* real-time systems. In: Grabowski, Nielsen (eds.) [10], pp. 64–78
6. Clarke, D., Jéron, T., Rusu, V., Zinovieva, E.: STG: a symbolic test generation tool. In: Katoen, J.-P., Stevens, P. (eds.) TACAS 2002. LNCS, vol. 2280, p. 470. Springer, Heidelberg (2002)
7. Frantzen, L., Huerta, M.N., Kiss, Z.G., Wallet, T.: On-The-Fly Model-Based Testing of Web Services with Jambition. In: Bruni, R., Wolf, K. (eds.) WS-FM 2008. LNCS, vol. 5387, pp. 143–157. Springer, Heidelberg (2009)
8. Frantzen, L., Tretmans, J., Willemse, T.A.C.: Test generation based on symbolic specifications. In: Grabowski, Nielsen (eds.) [10], pp. 1–15
9. Frantzen, L., Tretmans, J., Willemse, T.A.C.: A Symbolic Framework for Model-Based Testing. In: Havelund, K., Núñez, M., Roşu, G., Wolff, B. (eds.) FATES 2006 and RV 2006. LNCS, vol. 4262, pp. 40–54. Springer, Heidelberg (2006)
10. Grabowski, J., Nielsen, B. (eds.): FATES 2004. LNCS, vol. 3395. Springer, Heidelberg (2005)
11. Hartman, A., Nagin, K.: The AGEDIS tools for model based testing. SIGSOFT Softw. Eng. Notes 29(4), 129–132 (2004)
12. Henzinger, T.A., Nicollin, X., Sifakis, J., Yovine, S.: Symbolic model checking for real-time systems. Inf. and Comp. 111(2), 193–244 (1994)
13. Hessel, A., Larsen, K.G., Mikucionis, M., Nielsen, B., Pettersson, P., Skou, A.: Testing real-time systems using UPPAAL. In: Hierons, R.M., Bowen, J.P., Harman, M. (eds.) FORTEST 2000. LNCS, vol. 4949, pp. 77–117. Springer, Heidelberg (2008)
14. Jard, C., Jéron, T.: TGV: theory, principles and algorithms: A tool for the automatic synthesis of conformance test cases for non-deterministic reactive systems. J. STTS 7(4), 297–315 (2005)
15. Jéron, T.: Symbolic model-based test selection. In: Machado, P., Andrade, A., Duran, A. (eds.) SBMF 2008, pp. 17–32 (2008)
16. Krichen, M., Tripakis, S.: Black-box conformance testing for real-time systems. In: Graf, S., Mounier, L. (eds.) SPIN 2004. LNCS, vol. 2989, pp. 109–126. Springer, Heidelberg (2004)
17. Schmaltz, J., Tretmans, J.: On conformance testing for timed systems. In: Cassez, F., Jard, C. (eds.) FORMATS 2008. LNCS, vol. 5215, pp. 249–263. Springer, Heidelberg (2008)
18. STSimulator homepage, <http://www.cs.ru.nl/~1f/tools/stsimulator/>
19. Tretmans, J.: Test generation with inputs, outputs and repetitive quiescence. Software - Concepts and Tools 17(3), 103–120 (1996)
20. von Styp-Rekowski, S.: Towards a testing theory for timed symbolic systems. Diplomarbeit, RWTH Aachen University (November 2009), http://moves.rwth-aachen.de/dl/vstyp/styp_da.pdf
21. Rusu, V., du Bousquet, L., Jéron, T.: An approach to symbolic test generation. In: Grieskamp, W., Santen, T., Stoddart, B. (eds.) IFM 2000. LNCS, vol. 1945, pp. 338–357. Springer, Heidelberg (2000)