Testing Real-Time Multi Input-Output Systems

Laura Brandán Briones and Ed Brinksma

Faculty of Computer Science, University of Twente,
P.O.Box 217, 7500AE Enschede, The Netherlands
{brandanl, brinksma}@cs.utwente.nl

Abstract. In formal testing, the assumption of *input enabling* is typically made. This assumption requires all inputs to be enabled anytime. In addition, the useful concept of *quiescence* is sometimes applied. Briefly, a system is in a *quiescent* state when it cannot produce outputs.

In this paper, we relax the *input enabling* assumption, and allow some input sets to be enabled while others remain disabled. Moreover, we also relax the general bound M used in timed systems to detect *quiescence*, and allow different bounds for different sets of outputs.

By considering the $\mathbf{tioco}_M$ theory, an enriched theory for timed testing with repetitive *quiescence*, and allowing the partition of input sets and output sets, we introduce the $\mathbf{mtioco}_{\mathcal{M}}$ relation. A test derivation procedure which is nondeterministic and parameterized is further developed, and shown to be sound and complete wrt $\mathbf{mtioco}_{\mathcal{M}}$.

1 Introduction

Testing is the dominating validation activity in industry today. The necessity to improve it is urgent. The formal approach to testing and test generation, which aims to automatically generate test cases from models of the system under test (SUT), provides a structured way to improve and control the quality of testing.

Formal testing theory was introduced by De Nicola and Hennessy in their seminal paper [17], further elaborated in [16,9]. The first attempts to use De Nicola-Hennessy testing theory for finding algorithms to derive tests automatically from formal specifications were made by Brinksma in [3,4]. Tretmans [19] studied test generation for I/O transition systems. Building on this work, Heerink [8] extended the theory to deal with multiple channels, providing a more realistic scenario. These two approaches, depicted on Figure 1 (left) do not consider time in their models (i.e. are *untimed*). Recently, Tretmans' theory was extended to the timed setting by the authors [5], as shown in Figure 1 (top right).

It seems natural to ask whether a timed testing theory can also be extended to deal with multiple channels, thus completing Figure 1 (bottom right). In this paper we answer this question affirmatively, and we extend our theory [5] to account for multiple channels.

In black box, or functional, testing the model specifies the intended communication between the system and its environment, typically in terms of inputs (or *stimuli*) and outputs (or *responses*). In addition, the assumption of *input enabling* is commonly required. This assumption requests all inputs to a SUT

K.-K. Lau and R. Banach (Eds.): ICFEM 2005, LNCS 3785, pp. 264–279, 2005.

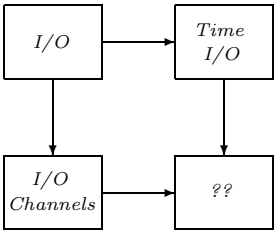


Fig. 1. Relation between test generation approaches

to be allowed at any time. A test case, then, provides inputs to the SUT and observes outputs from it. When it is not possible to recognize differences in the observable behaviour of two systems, it is concluded the systems are equal, i.e. a system is defined by its observable behaviour. In other words, the richer the observable behaviour is, the richer the distinguishing power of the test is. One way to improve this observable behaviour is by using the concept of *quiescence*. Briefly, a system is in a *quiescent* state when it cannot produce outputs without further inputs.

In [5] a real-time testing theory is formulated for *quiescent* time systems, which is parameterized by a bound M that is the explicit representation of the time a system should idle until *quiescence* can be concluded. Treating *quiescence* as a special sort of system output provides us with information to differentiate systems that have intuitively different deadlocking properties (cf. [5,12,19]).

In this paper we introduce the model of *timed multi input-output transition systems TMIOTS*. They model timed systems that communicate with the environment via multiple input and output channels. This allows us to consider *input enabling* and *quiescence* properties not only for an entire system but also on a per channel basis, thus relaxing global system assumptions.

Formally, channels are represented as a partitioning of the sets of input and output actions, each partition class defining the inputs (outputs) belonging to an individual input (output) channel. Following the ideas of Heerink [8] for the untimed case, we replace *input enabledness* by the requirement that for each input channel either all inputs are allowed, or they are all blocked. Often, this requirement is quite natural: a cash machine with a PIN card inserted would not accept the insertion of another card in the same slot.

In a similar way, we relax the treatment of *quiescence* by replacing the global bound M of $\mathbf{tioco}_M$, by a vector of bounds $\mathcal{M} = \langle M_1, \cdots, M_n \rangle$ for the different output channels. In $\mathbf{tioco}_M$ the global bound M is a parameter which inform for how long a system should wait before conclude *quiescent*. Relaxing the global bound M for a vector of bounds means that we will not have to wait for the slowest response time to conclude the *quiescence* of a faster channel.

The combination of these ideas is formalized as the $\mathbf{mtioco}_{\mathcal{M}}$ conformance relation. We develop a test derivation procedure for $\mathbf{mtioco}_{\mathcal{M}}$, which is shown to be sound and complete. Therefore, our work can be seen as a real-time extension of Heerink's **mioco** theory, which introduced the channel-based treatment of *input enabling/blocking* and *quiescence* in the untimed setting.

Organization of the Paper. The paper consists of two main parts: Models and Relations (Section 2) and Test Generation Framework (Section 3). In the first part, starting from a simple model and a simple conformance relation we build in three steps (Subsections: 2.1, 2.2 and 2.3) an extended model and its conformance relation $\mathbf{mtioco}_{\mathcal{M}}$. In the second part, we develop a parameterized nondeterministic test derivation procedure and prove that the set of test are sound and complete with respect to the $\mathbf{mtioco}_{\mathcal{M}}$ relation. Finally, Section 4 presents the conclusions of the paper. To save space we omitted the proof of lemmas and theorems in this paper, but they can be found in the extended version of this paper [6].

2 Models and Relations

This section presents three related models, and a conformance relation is formulated for each of them. First, we introduce timed transition systems and the **tmior** relation. Later on, the timed transition relation is extended with *quiescence* and *refusals* and a parameterized relation is defined: **mtiorf** relation. Finally, the concept of *observed outputs set* is introduced and the $\mathbf{mtioco}_{\mathcal{M}}$ relation is given. Throughout, a model of a cash machine is used as a running example.

2.1 A Basic Model and Relation

Basically, a *timed labelled transition system* is a *labelled transition system* extended with time delay transitions. This leads to three types of actions: *time-passage actions*, *visible actions* and the special *internal action* τ. All except the time-passage actions are thought of as occurring instantaneously, i.e. without consuming time. To specify time, a continuous dense time domain is used.

Definition 1. *A Timed Labelled Transition System (TLTS) is a 4-tuple* $\langle S, s_0, L_{\tau\mathcal{T}}, \rightarrow \rangle$*, where*

- S *is a non-empty set of states. With* $s_0 \in S$ *as the initial state.*
- $L_{\tau\mathcal{T}} \triangleq L \cup \{\tau\} \cup \mathcal{T}$ *are the actions* L *including the internal action* τ *and time-passage actions. Where* $\tau \notin L$ *and* $\mathcal{T} \triangleq \{d \mid d \in \mathbb{R}^{\geq 0}\}$ *with* $L \cup \{\tau\} \cup \mathcal{T} = \emptyset$
- $\rightarrow \subseteq (S \times L_{\tau\mathcal{T}} \times S)$ *is the timed transition relation with the following consistency constraints:* $\forall\ d, d_1, d_2 \in \mathcal{T}$*;* $\forall\ s, s', s'' \in S$
  - **Time Determinism** *whenever* $s \xrightarrow{d} s'$ *and* $s \xrightarrow{d} s''$ *then* $s' = s''$
  - **Time Additivity** $(\exists\ s' : s \xrightarrow{d_1} s' \xrightarrow{d_2} s'')$ *if and only if* $s \xrightarrow{d_1+d_2} s''$
  - **Null Delay** $s \xrightarrow{0} s'$ *if and only if* $s = s'$.

The labels in $L_{\mathcal{T}}$ ($L_{\mathcal{T}} \triangleq L \cup \mathcal{T}$) represent the observable actions of a system, i.e. labelled actions and passage of time. The τ label represents an unobservable internal action. A transition $(s, \mu, s') \in \rightarrow$ is denoted as $s \xrightarrow{\mu} s'$. A *computation* is a finite or infinite sequence of transitions:

$$s_0 \xrightarrow{\mu_1} s_1 \xrightarrow{\mu_2} s_2 \xrightarrow{\mu_3} \cdots \xrightarrow{\mu_{n-1}} s_{n-1} \xrightarrow{\mu_n} s_n (\rightarrow \ldots)$$

When a *timed labelled transition system* has the set of actions partitioned into input and output actions is called a *timed input-output transition system*,

denoted as $TIOTS(L_I, L_U)$ where L_I represents the set of inputs and L_U the set of outputs.

Our framework is based on *timed transitions systems* even though all examples we present are given as timed automata. In comparison, a timed automata have less expressiveness than a *timed transition systems* but they have a more compact representation. The relation between a timed automata and its corresponding semantics in terms of *timed transition system* can be found in [18].

Example 1. Our example is an adapted version of the cash machine in [8]. Figure 2 is the representation of a cash machine where a card can be inserted and for a limited period of time a PIN can be typed in. After the machine has decided if the PIN was correct, an amount of money can be requested. In case the machine has sufficient money, it will return the card and then give the requested money. If there is not enough money it will produce an error and return the card. In case the PIN or the amount of money are to late the machine return the card.

Throughout the paper we denote an input as a label followed with a ?-symbol and an output with a !- symbol. The example shows a system where there are inputs, outputs and real-time constraints. In terms of Definition 1 the cash machine is specified as a $TIOTS(L_I, L_U)$ where $\langle S, s_0, L_{\tau\mathcal{T}}, \rightarrow \rangle$, with $S = \{q_0, \cdots, q_{11}\}$, $s_0 = q_0$, $L_I = \{card?, PIN?, amount?\}$ and $L_U = \{card!, amount!, Ok!, Err-P!, Err-a!\}$.

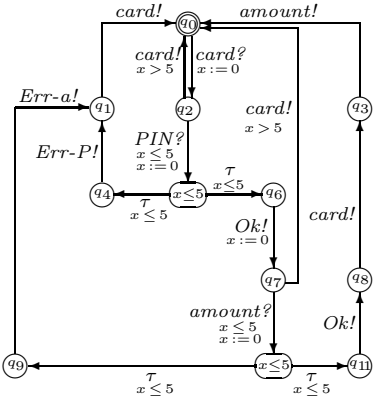


Fig. 2. A cash machine, a modified version of [8]

To explicitly encode the inability for a state to perform any action in a set A or any internal action τ, we extend the *timed transition relation* with self-loop transitions: $s \xrightarrow{A} s$, in case A is a *refusal* of s

$$s \xrightarrow{A} s' \triangleq \ \forall\, \mu \in (A \cup \{\tau\}) : s \overset{\mu}{\not\rightarrow} \wedge\ s = s'.$$

We use the well-known notation: $p \overset{\sigma}{\Rightarrow}$ to denote that there exists a reachable state q from p by performing σ while abstracting from the internal actions. In the rest of the paper, we do not always distinguish between p a *TLTS* and its initial state s_0, e.g. we write $p \overset{\sigma}{\Rightarrow}$ instead of $s_0 \overset{\sigma}{\Rightarrow}$.

Definition 2. *Let p be an $TLTS(L)$, with P a set of states in p, then*

$\text{f-ttraces}(p) = \{\sigma \in (\mathcal{P}(L) \cup L_{\mathcal{T}})^* \mid p \stackrel{\sigma}{\Rightarrow}\}$, *$\mathcal{P}(L)$ denotes the power set of L*

$\text{ttraces}(p) = \text{f-ttraces}(p) \cap L_{\mathcal{T}}^*$

$\text{init}(p) = \{\mu \in L_{\tau\mathcal{T}} \mid \exists\ p' : p \stackrel{\mu}{\rightarrow} p'\}$

$\text{der}(p) = \{p' \mid \exists\ \sigma \in L_{\mathcal{T}}^* : p \stackrel{\sigma}{\Rightarrow} p'\}$

P **after** $\sigma = \{p' \mid \exists\ p \in P : p \stackrel{\sigma}{\Rightarrow} p'\}$

p *is* deterministic *if and only if* $\forall\ \sigma \in L_{\mathcal{T}}^* : |\{p\}$ **after** $\sigma| \leq 1$.

As expected, a *timed trace* (ttrace) is a standard trace extended with time. A *failure ttrace* (f-ttrace) is a *ttrace* extended with sets of actions that can not be performed, in other words actions that are *refused*. The *init* is the set of all possible actions from a given state, the *der* is the set of all reachable states from a given state, and the **after** is the set of all states reachable after a given *ttrace*. We call a system *deterministic* if for all *ttrace*'s it has at most one reachable state.

In Figure 2 we can observe that the $init(q_0) = \{card?\}$, the $der(q_0)$ is the set of all states $\{q_1, \cdots, q_{11}\}$ and $(\{q_{11}, q_9\}$ **after** Ok!$) = \{q_8\}$. Moreover, we can recognize the cash machine is not *deterministic*.

As we already anticipated in the introduction, the novelty of this paper is to consider *real-time transition systems* where the input set and output set are partitioned into subsets, called channels. More precisely, a $TIOTS(\mathcal{L}_I, \mathcal{L}_U)$ is a *timed input-output transition system* $TIOTS(L_I, L_U)$ where the set of inputs and outputs are partitioned into channels $\mathcal{L}_I = \{L_I^1, \cdots, L_I^n\}$ and $\mathcal{L}_U = \{L_U^1, \cdots, L_U^m\}$.

The partition in channels gives us the possibility to introduce the first relation : **tmior** (Definition 3). This relation refers to the inclusion of *f-ttraces* where the *refusals* can only be full channels.

Definition 3. *Let p and q be $TIOTS(\mathcal{L}_I, \mathcal{L}_U)$, then*

$$q \sqsubseteq_{\mathbf{tmior}} p \ \triangleq\ \text{f-ttraces}(q) \cap (L_{\mathcal{T}} \cup \mathcal{L}_I \cup \mathcal{L}_U)^* \subseteq \text{f-ttraces}(p).$$

2.2 An Extended Model and Relation

The **tmior** relation, from Section 2.1 induced us to define an extension of *TIOTS* where the input and output sets are subdivides in channels. Then, a *timed multi input-output transition system* ($TMIOTS(\mathcal{L}_I, \mathcal{L}_U)$) is a $TIOTS(L_I, L_U)$ where, in each reachable state each input channel is either blocked or all inputs of that channel are accepted (*input enabling* for particular channels). More formally:

Definition 4. *For $\mathcal{L}_I = \{L_I^1, \cdots, L_I^n\}$ and $\mathcal{L}_U = \{L_U^1, \cdots, L_U^m\}$ a Timed Multi Input-Output Transition System p $TMIOTS(\mathcal{L}_I, \mathcal{L}_U)$ is a TIOTS with $L_I = \bigcup_{1 \leq i \leq n} L_I^i$ and $L_U = \bigcup_{1 \leq j \leq m} L_U^j$, where*

$$\forall\ s \in der(p) : (\forall\ \mu \in L_I^i : s \stackrel{\mu}{\not\rightarrow}) \vee (\forall\ \mu \in L_I^i : s \stackrel{\mu}{\rightarrow})$$

Moreover, whenever a channel L_I^i is blocked in state s, it is denoted $\gamma^i(s)$.

Example 2. It is possible to see the cash machine of Figure 2 as a *TMIOTS* $(\mathcal{L}_I, \mathcal{L}_U)$ $\langle S, s_0, L_{\tau\mathcal{T}}, \rightarrow\rangle$, where $S = \{q_0, \cdots, q_{11}\}$, $s_0 = q_0$, $\mathcal{L}_I = \{L_I^1, L_I^2\}$ and

$\mathcal{L}_U = \{L_U^1, L_U^2, L_U^3\}$ with $L_I^1 = \{card?\}$, $L_I^2 = \{PIN?, amount?\}$ and $L_U^1 = \{card!\}$, $L_U^2 = \{amount!\}$, $L_U^3 = \{Ok!, Err\text{–}P!, Err\text{–}a!\}$. With the corresponding saturation for each channel (i.e. every state with an outgoing transition labeled by input from a channel, is assumed, to have the rest of the inputs from that channel as self-loop transitions. Even when this might not be explicit).

Since the definition of TMIOTS$(\mathcal{L}_I, \mathcal{L}_U)$ implies *input enabling* or no input at all for each channel, we use it only when the *input enabling* property is necessary. Otherwise, we use the more general notation TIOTS$(\mathcal{L}_I, \mathcal{L}_U)$, implying TIOTS with the input and output sets partitioned in channels.

The notion of *quiescence* is crucial, since some systems can only be distinguished by their *quiescent* states. Intuitively, the underlying idea is that the environment may observe not only output actions, but also the *absence* of output actions (i.e. in a given state, the system does not emit any output for the environment to observe).

There are two possible ways to deal with *quiescence*. First, we may consider the situation in which the environment can only observe one channel. In this case, it is not relevant for the notion of *quiescence* whether the remaining channels stay silent or not. Second, we may consider the environment to be able to observe all possible channels. In this case, to conclude *quiescence* in one particular channel L_U^j must imply that the remaining channels stay silent for at least the period of time L_U^j stayed silent. We adopt the latter direction, assuming an environment which can observe simultaneously all channels. This choice fits well with the testing framework of [5], where tests synchronize on all output actions. Partial observations of system output can be dealt with by considering modified SUTs where the unobservable channels have become internal actions to the system.

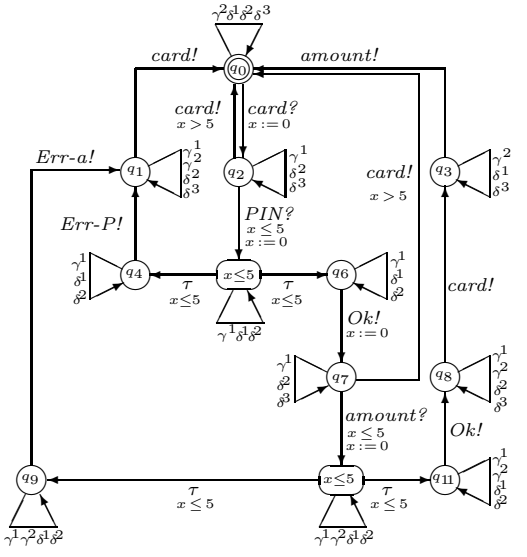
Definition 5. *Let p be a TIOTS$(\mathcal{L}_I, \mathcal{L}_U)$ with s an state of p is called L_U^j-*quiescent*, denoted $\delta^j(s)$, if and only if $\forall\ \mu \in L_U^j : \forall\ d \in \mathbb{R}^+ : s \overset{\mu(d)}{\not\Rightarrow}$. Where $s \overset{d}{\Rightarrow}$ is used as the syntactic sugar for $\exists\ s' : s \overset{d}{\Rightarrow} s' \overset{\mu}{\Rightarrow}$, and $s \overset{\mu(d)}{\not\Rightarrow}$ its corresponding negation.*

We would like to point out that in the non-timed framework, the *quiescence* definition uses a single arrow notation $\rightarrow$, namely without abstracting from τ transitions. In the timed case this is not possible. For example, take the definition above with a single arrow, and the following system, with $o! \in L_U^j$: $s \overset{d}{\rightarrow} s' \overset{\tau}{\rightarrow} s'' \overset{d'}{\rightarrow} s''' \overset{o!}{\rightarrow}$. Then, with the new definition the state s is *quiescent* because is not possible to reach from s'' from s with a single arrow. Consequently, in *timed systems* it is essential that the definition of *quiescent* have double arrow.

With the definition of L_U^j*-quiescence*, we extend the *timed transition relation* to include self-loop transitions for *refusals* and *quiescence*. Therefore, $s \overset{\gamma^i}{\rightarrow} s$ if and only if s refuses L_I^i, and $s \overset{\delta^j}{\rightarrow} s$ if and only if s is L_U^j*-quiescent*. We denote p a *TIOTS*$(\mathcal{L}_I, \mathcal{L}_U)$ with the extended timed transition relation for *refusals* and *quiescence* as $\Delta(p)$. A consequence of this extension is: f-ttraces$(p) \cap (L_{\mathcal{T}} \cup \mathcal{L}_I \cup$

$\mathcal{L}_U)^* = \text{ttraces}(\Delta(p))$. Therefore, using this notation we can re-write the **tmior** relation as: $q \sqsubseteq_{\mathbf{tmior}} p$ if and only if: $ttraces(\Delta(q)) \subseteq ttraces(\Delta(p))$.

Example 3. Figure 3 illustrates the cash machine with the extended timed transition relation with $refusals(\gamma^i)$ and $quiescence(\delta^j)$. To avoid too much detail, it is assumed that in each state without a self-loop for *refusal* of an input channel, all the absent inputs of that channel have self-loop transitions in that state.



$L_I^1 = \{card?\} \Rightarrow \gamma^1$

$L_I^2 = \{PIN?,\ amount?\} \Rightarrow \gamma^2$

$L_U^1 = \{card!\} \Rightarrow \delta^1$

$L_U^2 = \{amount!\} \Rightarrow \delta^2$

$L_U^3 = \{Ok!,\ Err\text{-}P!,\ Err\text{-}a!\} \Rightarrow \delta^3$

Fig. 3. A cash machine, a modified version of [8]

An immediate problem, in black box testing, is how to detect *quiescence* in implementations. Given that a *quiescent* state in an implementation is only recognizable after a period of time where there was no output observations, it is necessary to fix for how long a test should be waiting before concluding *quiescence*. Therefore, we define three properties. First, we define what it means for a state to be *quiescent* with respect to a channel and a particular time bound. Intuitively, for a state to be *quiescent* on a channel wrt a particular bound means that all reachable states after delaying by the given bound are *quiescent* on that particular channel. More precisely, a state of a system is M_j-*quiescent*, for an output channel j, if and only if all reachable states from that state after M_j are *quiescent*. Second, the definition is extended to all state in the system. Third, the definition is extended to include all output channels.

Definition 6. *Let p be a TIOTS$(\mathcal{L}_I, \mathcal{L}_U)$ with S as its states, $s \in S$ and $\mathcal{M}$ an ordered set of bounds $\mathcal{M} = \langle M_1, \cdots, M_m \rangle : \forall\ 1 \leq j \leq m : M_j \in \mathbb{R}^{\geq 0}$, then*

- *s is $\mathcal{M}_j$-*quiescent *if and only if* $\forall\ s' \in (s \textbf{ after } M_j) : s' \in L_U^j$-quiescent
- *p is $\mathcal{M}_j$-*quiescent *if and only if* $\forall\ s \in S : \mathcal{M}_j$-quiescent$(s)$
- *p is $\mathcal{M}$-*quiescent *if and only if* $\forall\ 1 \leq j \leq m : \mathcal{M}_j$-quiescent$(q)$.

An interpretation of this definition is that for a tester to check for *quiescence* in channel j, it is enough with wait a period of time equal to M_j, without observing outputs. There are two important principles involved in this definition. We are spending different times for detecting *quiescence* for different channels. Moreover, we assume that after delaying by the corresponding bound of a channel there will not be any spontaneous output on that channel.

Lemma 1. *If a system* $p \in TIOTS(\mathcal{L}_I, \mathcal{L}_U)$ *is* $\mathcal{M}_j$-quiescent *with* S *as its states and* $s \in S$*, then:* $\delta^j(s) \triangleq \forall\, \mu \in L_U^j : \forall\, d \in \mathbb{R}^{\geq 0} : d \leq M_j : s \overset{\mu(d)}{\not\Rightarrow}$.

Corollary 1. *Let* $p \in TIOTS(\mathcal{L}_I, \mathcal{L}_U)$ *be* $\mathcal{M}$-quiescent *with* S *as its states,* $s \in S$*, and* $\mathcal{M} = \langle M_1, \cdots, M_m \rangle$*, then*

$$\forall\, j = 1, \cdots, m : (\delta^j(s) \Leftrightarrow (\forall\, \mu \in L_U^j : \forall\, d \in \mathbb{R}^{\geq 0} : d \leq M_j : s \overset{\mu(d)}{\not\Rightarrow})).$$

Considering the cash machine in Figure 3 for $\mathcal{M} = \langle M_1, M_2, M_3 \rangle$ with $M_1 = 6$, $M_2 = 6$ and $M_3 = 6$ we can recognize that state q_0 is M_1-*quiescent*.

Since in an implementation we can detect *quiescence* only with the observation of absence of outputs for a period of time, and using the property of $\mathcal{M}$-*quiescence* for a system, we define the **mtiorf** relation, parameteraized by $\mathcal{M}$. In the *traces* considered in **mtiorf** a δ^j can only occur after M_j timed units.

Definition 7. *Let* p *be a* $TIOTS(\mathcal{L}_I, \mathcal{L}_U)$ *and* q *be a* $\mathcal{M}$-*quiescent* $TMIOTS$ $(\mathcal{L}_I, \mathcal{L}_U)$*, then*

$$q \sqsubseteq^{\mathcal{M}}_{\mathbf{mtiorf}} p \quad \textit{if and only if} \quad \Delta_{\mathcal{M}}(q) \subseteq \Delta_{\mathcal{M}}(p)$$

where for $r \in TIOTS(\mathcal{L}_I, \mathcal{L}_U)$*, with* ϵ *as the empty word:*

$\Delta_{\mathcal{M}}(r) \triangleq ttraces(\Delta(r)) \cap \bigcup_i \bigcup_j (((\mathcal{T} \cup \{\epsilon\}) \cdot (L \cup \gamma^i)) \cup M_j \cdot \delta^j)^*$.

2.3 The Relation: $mtioco_{\mathcal{M}}$

Up to now, we considered relations built up from information based on knowledge of the behaviour of both specifications and implementations. A relation that uses information from the behaviour of only the specification is more desirable in the context of black box testing, which is our main goal in the present paper. To this end, we now define the *observed output set*, which condenses the whole information as perceived by the environment, and a more practical notation in the form of *nttraces*.

Similarly to the definition of *nttraces* for $\mathbf{tioco}_M$ theory in [5], we present the *normalized ttraces* for *TMIOTS*.

Definition 8. *Let* p *be* $\mathcal{M}$-*quiescent and* σ *be a* ttraces *in* $\Delta(p)$*, then*

- σ *is a normalized* ttrace *if and only if* $\sigma \in \bigcup_i \bigcup_j (\mathcal{T} \cdot (L \cup \gamma^i \cup \delta^j))^*$
- nttraces$(p) = \{\sigma \in \bigcup_i \bigcup_j (\mathcal{T} \cdot (L \cup \gamma^i \cup \delta^j))^* \mid p \overset{\sigma}{\Rightarrow}\}$
- *for* nttraces $\sigma = d_0 \delta^1 d_1 \gamma^1 d_2 a!$ *we also write* $\hat{\sigma} = \delta^1(d_0)\gamma^1(d_1)a!(d_2)$.

Moreover, the definition of *nttraces* already assumes that TIOTS$(\mathcal{L}_I, \mathcal{L}_U)$ systems have the *timed transition relation* extended, implying

$$nttraces(\Delta(r)) = nttraces(r).$$

An example of an *nttrace* in the cash machine is:

$$card?(3)PIN?(2)Err{-}P!(5)\gamma^1(6)card!(0).$$

For consistency, we need to prove that with this new notation we are not losing expressiveness, as it is crucial to have that the inclusion of *nttraces* for two systems is equal to the inclusion of *ttraces* for the corresponding extended systems. This result is given in the following lemma.

Lemma 2. *Let* $p_1, p_2 \in TIOTS(\mathcal{L}_I, \mathcal{L}_U)$*, then*
$\text{ttraces}(\Delta(p_1)) \subseteq \text{ttraces}(\Delta(p_2))$ *if and only if* $\text{nttraces}(p_1) \subseteq \text{nttraces}(p_2)$.

The *observed output set* of a given set of states P, denoted $obsOut_{\mathcal{M}}(P)$, is defined as the union of two sets: the set of output actions enriched with *quiescent*, denoted $obsOut^o_{\mathcal{M}}$, and the set of *refusals*, denoted $obsOut^r_{\mathcal{M}}$. Hence, $obsOut^o_{\mathcal{M}}$ is the set of outputs that could happen after a period of time plus the special symbol $\delta^j(M_j)$ expressing *quiescence* on output channel j in case a reachable state after M_j is *quiescent* on channel j. And, the set $obsOut^r_{\mathcal{M}}$ is the set of *refusals* $\gamma^i(d)$ for each input channel i that is refused after d timed units.

Definition 9. *Let* P *be a set of states of a* $TIOTS(\mathcal{L}_I, \mathcal{L}_U)$ *with timed transition relation extended, then:*

$$obsOut_{\mathcal{M}}(P) = \bigcup_{p \in P} obsOut^o_{\mathcal{M}}(p) \bigcup \bigcup_{p \in P} obsOut^r_{\mathcal{M}}(p)$$

$$\text{where:} \quad obsOut^o_{\mathcal{M}}(p) = \{\mu(d) \mid \mu \in L_U \wedge p \overset{\mu(d)}{\Rightarrow}\} \cup \bigcup_j \{\delta^j(M_j) \mid p \overset{\delta^j(M_j)}{\Rightarrow}\}$$

$$obsOut^r_{\mathcal{M}}(p) = \bigcup_i \{\gamma^i(d) \mid \forall\, \mu \in L^i_I :\ p \overset{\mu(d)}{\not\Rightarrow}\}$$

A immediate and useful consequence of this definition is that a system has an *nttrace* if and only if the *observed output set*, $obsOut_{\mathcal{M}}$, of the system after that *nttrace* is not empty.

Lemma 3. *Let* $p \in TIOTS(\mathcal{L}_I, \mathcal{L}_U)$ *and* $\sigma \in \text{nttraces}$*, then*
$obsOut_{\mathcal{M}}(p \textbf{ after } \sigma) = \emptyset$ *if and only if* $\sigma \notin nttraces(p)$.

We also prove that the parameterized **mtiorf** relation is equal to checking the inclusion of *observed output set* for all *nttraces* that only have δ^j after M_j timed units.

Lemma 4. *Let* p *be* $TIOTS(\mathcal{L}_I, \mathcal{L}_U)$ *and* q *be* $\mathcal{M}$*-quiescent* $TMIOTS(\mathcal{L}_I, \mathcal{L}_U)$*, then* $q \sqsubseteq^{\mathcal{M}}_{\textbf{tmiorf}} p$ *if and only if* $\forall\, \sigma \in \Delta_{\mathcal{M}}$:

$$obsOut_{\mathcal{M}}(q \textbf{ after } \sigma) \subseteq obsOut_{\mathcal{M}}(p \textbf{ after } \sigma).$$

Finally, we are in position to define the $\textbf{mtioco}_{\mathcal{M}}$ relation, based solely on information from the *observed output set* and the specification. Particularly, without any internal knowledge of the implementation, which complies with the requirement of black box testing.

For p a specification in $TIOTS(\mathcal{L}_I, \mathcal{L}_U)$ and q an implementation in $TMIOTS(\mathcal{L}_I, \mathcal{L}_U)$: q will be $\mathbf{mtioco}_{\mathcal{M}}$ to p if and only if the *observed output set* of q, after every *nttrace* of p is a subset of the *observed output set* of p after the same *nttrace*.

Definition 10. *Let p be a $TIOTS(\mathcal{L}_I, \mathcal{L}_U)$ and q be $\mathcal{M}$-quiescent $TMIOTS(\mathcal{L}_I, \mathcal{L}_U)$, then:*

q $\mathbf{mtioco}_{\mathcal{M}}$ $p \triangleq \forall \sigma \in \Delta_{\mathcal{M}}(p)$: $obsOut_{\mathcal{M}}(q \textbf{ after } \sigma) \subseteq obsOut_{\mathcal{M}}(p \textbf{ after } \sigma)$.

The $\mathbf{mtioco}_{\mathcal{M}}$ relation is a parameterized timed relation that consider *quiescent* for each particular channel. Moreover, in the next section, we use this relation to build our test derivation framework over $TMIOTS(\mathcal{L}_I, \mathcal{L}_U)$.

3 Test Generation Framework

In this section we define the concept of real-time test cases, the nature of their execution, and the evaluation of their success or failure. Later, a test generation procedure is presented for $\mathbf{mtioco}_{\mathcal{M}}$ relation. Moreover, it is shown that this procedure is sound and complete.

A *test case* $\mathbf{t}$ is a $TIOTS(\mathcal{L}_I, \mathcal{L}_U)\langle S, s_0, L_{\mathcal{T}} \cup \{\delta\}, \rightarrow\rangle$ such that is deterministic and has *bounded behaviour*, in the sense that all computations have finitely many action occurrences and its accumulative time is bounded. The set of states also contains the terminal states **pass** and **fail** without outgoing transitions. For any state different from **pass** and **fail** there exists a bounded time to observe *quiescence* or to be able to make an input action. Moreover, tests under consideration are deterministic and therefore τ-transitions are not allowed. The class of test cases over $\mathcal{L}_I$ and $\mathcal{L}_U$ is denoted as $TTEST(\mathcal{L}_I, \mathcal{L}_U)$. A *test suite* $\mathbf{T}$ is a set of test cases: $\mathbf{T} \subseteq TTEST(\mathcal{L}_I, \mathcal{L}_U)$. Again, to simplify notation we represent tests as timed automata.

A test run of an implementation with a test case is modelled by the synchronous parallel execution of the test case together with the implementation under test. This run continues until no more interactions are possible, i.e. until a deadlock occurs.

Definition 11. *Let $\mathbf{t}$ be a test in $TTEST(\mathcal{L}_I, \mathcal{L}_U)$ and imp be a $\mathcal{M}$-quiescent implementation in $TMIOTS(\mathcal{L}_I, \mathcal{L}_U)$, then*

- *Running a test case $\mathbf{t}$ with an implementation imp is modelled by the parallel operator $||$: $TTEST(\mathcal{L}_I, \mathcal{L}_U) \times TMIOTS(\mathcal{L}_I, \mathcal{L}_U) \rightarrow TIOTS(\mathcal{L}_I, \mathcal{L}_U)$ which is defined by the following inference rules:*

$$
\begin{array}{lll}
imp \xrightarrow{\tau} imp' & \vdash & \mathbf{t}||imp \xrightarrow{\tau} \mathbf{t}||imp' \\
\mathbf{t} \xrightarrow{\delta^j} \mathbf{t}' & \vdash & \mathbf{t}||imp \xrightarrow{\delta^j} \mathbf{t}'||imp \\
\mathbf{t} \xrightarrow{\gamma^i} \mathbf{t}', imp \overset{\mu}{\not\longrightarrow} imp', \mu \in L_I^i & \vdash & \mathbf{t}||imp \xrightarrow{\gamma^i} \mathbf{t}'||imp \\
\mathbf{t} \xrightarrow{\mu} \mathbf{t}', imp \xrightarrow{\mu} imp', \mu \in L & \vdash & \mathbf{t}||imp \xrightarrow{\mu} \mathbf{t}'||imp' \\
\mathbf{t} \xrightarrow{d} \mathbf{t}', imp \xrightarrow{d} imp', d \in \mathbb{R}^{\geq 0} & \vdash & \mathbf{t}||imp \xrightarrow{d} \mathbf{t}'||imp'
\end{array}
$$

- *A test run of* $\mathbf{t}$ *with an implementation imp, is a* σ *in* $\Delta_M(\mathbf{t}||imp)$ *leading to a terminal state of* $\mathbf{t}$*. Then, an implementation imp* **passes** *test case* $\mathbf{t}$*, if all their test runs lead to the* **pass** *state of* $\mathbf{t}$*. Moreover, an implementation imp* **passes** *a test suite* $\mathbf{T}$*, if it* **passes** *all test cases in* $\mathbf{T}$*. And finally, if imp does not* **pass** *the test suite, it* **fails**.

$$
\begin{array}{lcl}
\textit{test run of } \mathbf{t} \textit{ and } imp & \triangleq & \exists\, imp' : (\mathbf{t}||imp \stackrel{\sigma}{\Rightarrow} \mathbf{pass}||imp') \textit{ or} \\
 & & \quad (\mathbf{t}||imp \stackrel{\sigma}{\Rightarrow} \mathbf{fail}||imp') \\
imp \textbf{ passes } \mathbf{t} & \triangleq & \forall\, \sigma \in \Delta_M : \forall\, imp' : \mathbf{t}||imp \stackrel{\sigma}{\not\Rightarrow} \mathbf{fail}||imp' \\
imp \textbf{ passes } \mathbf{T} & \triangleq & \forall\, \mathbf{t} \in \mathbf{T} : imp \textbf{ passes } \mathbf{t} \\
imp \textbf{ fails } \mathbf{T} & \triangleq & \exists\, \mathbf{t} \in \mathbf{T} : imp \textbf{ passes } \mathbf{t}.
\end{array}
$$

If an implementation can behave nondeterministically, then different test runs of the same test case may lead to different terminal states with different verdicts. This implies that an implementation **passes** a test case if an only if all possible test runs lead to the verdict **pass**.

For the description of test cases we use a process algebraic behavioural notation with a syntax inspired by LOTOS [10]:

$$B \triangleq a;B \mid B+B \mid \Sigma\, \mathcal{B}$$

where $a \in L_{\mathcal{T}\gamma\delta}$ ($L_{\mathcal{T}\gamma\delta} \triangleq L_{\mathcal{T}} \cup \{\gamma^i \mid 1 \leq i \leq n\} \cup \{\delta^j \mid 1 \leq j \leq m\}$), $\mathcal{B}$ is a countable set of behaviour expressions, and axioms plus inference rules are:

$$
\begin{array}{ll}
a \in L & \vdash\ a;B \stackrel{a}{\rightarrow} B' \\
a = d, d' < d & \vdash\ d;B \stackrel{d'}{\longrightarrow} d-d';B \\
a = d & \vdash\ B \stackrel{d}{\longrightarrow} B' \\
B_1 \stackrel{\mu}{\rightarrow} B_1',\ \mu \in L_{\mathcal{T}\gamma\delta} & \vdash\ B_1+B_2 \stackrel{\mu}{\rightarrow} B_1' \\
B_2 \stackrel{\mu}{\rightarrow} B_2',\ \mu \in L_{\mathcal{T}\gamma\delta} & \vdash\ B_1+B_2 \stackrel{\mu}{\rightarrow} B_2' \\
B \stackrel{\mu}{\rightarrow} B', B \in \mathcal{B}, \mu \in L_{\mathcal{T}\gamma\delta} & \vdash\ \Sigma\, \mathcal{B} \stackrel{\mu}{\rightarrow} B'
\end{array}
$$

Here, we use $\mu(d)$ as syntactic sugar for $d;\mu$, following Definition 8.

3.1 Test Case Generation Procedure

We define a procedure to generate test cases from a given specification in *TIOTS* $(\mathcal{L}_I, \mathcal{L}_U)$. Similar to [19,5] test cases result from the nondeterministic, recursive application of three test generation steps: (1) *termination*, (2) *inputs* (including *refusals*), and (3) *waiting for outputs* (including *quiescence*).

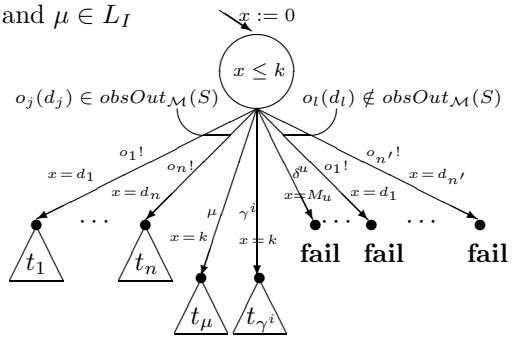
The construction steps involve negations of predicates of the form: $o(d) \in obsOut_{\mathcal{M}}(S)$ or $\gamma^i(d) \in obsOut_{\mathcal{M}}(S)$; which on the general level of *TMIOTS* are undecidable. Then, the procedure given here, should be seen as a meta-algorithm that can be used to generate tests effectively for subclasses of *TMIOTS* for which these predicates are decidable, such as timed automata [11,13] with sub partitioning of the input and output sets.

1. *termination*

pass

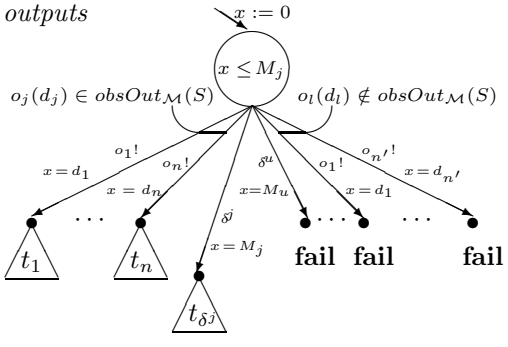
2. *inputs*

choose $k \in [0, Max\{M_1, \cdots, M_m\})$
and $\mu \in L_I$



3. *waiting for outputs*

choose j



1. *termination*

$t :=$ **pass**

It is possible to stop the recursion at any time using this step.

2. *inputs*

$$\begin{aligned} t := \quad & \Sigma\{o_j(d_j); t_j \mid o_j \in L_U \wedge d_j < k \wedge o_j(d_j) \in obsOut_{\mathcal{M}}(S)\} \\ & + \{\mu(k); t_\mu \mid \mu \in L_I^i \wedge \exists\ s \in S : \gamma^i(k) \notin obsOut_{\mathcal{M}}(s)\} \\ & + \{\mu(k); \mathbf{fail} \mid \mu \in L_I^i \wedge \forall\ s \in S : \gamma^i(k) \in obsOut_{\mathcal{M}}(s)\} \\ & + \{\gamma^i(k); \mathbf{fail} \mid \mu \in L_I^i \wedge \gamma^i(k) \notin obsOut_{\mathcal{M}}(S)\} \\ & + \{\gamma^i(k); t_{\gamma^i} \mid \mu \in L_I^i \wedge \gamma^i(k) \in obsOut_{\mathcal{M}}(S)\} \\ & + \Sigma\{\delta^u(M_u); \mathbf{fail} \mid M_u \in \mathcal{M} \wedge M_u < k \wedge \delta^u(M_u) \notin obsOut_{\mathcal{M}}(S)\} \\ & + \Sigma\{o_l(d_l); \mathbf{fail} \mid o_l \in L_U \wedge o_l(d_l) \notin obsOut_{\mathcal{M}}(S)\} \end{aligned}$$

where x is a clock, k is a timed variable and t_j, t_μ and t_{γ^i} are obtained by recursively applying the algorithm to $(S$ **after** $o_j(d_j))$, $(S$ **after** $\mu(k))$ and $(S$ **after** $\gamma^i(k))$, respectively.

3. *waiting for outputs*

$$\begin{aligned} t := \quad & \Sigma\{o_j(d_j); t_j \mid o_j \in L_U \wedge o_j(d_j) \in obsOut_{\mathcal{M}}(S)\} \\ & + \Sigma\{\delta^j(M_j); t_{\delta^j} \mid \delta^j \in obsOut_{\mathcal{M}}(S \textbf{ after } M_j)\} \\ & + \Sigma\{\delta^j(M_j); \mathbf{fail} \mid \delta^j \notin obsOut_{\mathcal{M}}(S \textbf{ after } M_j)\} \\ & + \Sigma\{\delta^u(M_u); \mathbf{fail} \mid M_u \in \mathcal{M} \wedge M_U < M_j \wedge \delta^u(M_u) \notin obsOut_{\mathcal{M}}(S)\} \\ & + \Sigma\{o_l(d_l); \mathbf{fail} \mid o_l \in L_U \wedge o_l(d_l) \notin obsOut_{\mathcal{M}}(S)\} \end{aligned}$$

where x is a clock and t_j and t_{δ_j} are obtained by recursively applying the algorithm for $(S \ \textbf{after}\ o_j(d_j))$ and $(S \ \textbf{after}\ \delta^j(M_j))$, respectively.

Note 1. Case 2: *inputs* and case 3: *waiting for outputs* are overlapping. If in a derivation of the *input* case test there exists an arrow for a δ^u, then it is clear that the test will never succeed to make the input or check for γ^i. This knowledge could be used, once it is known that an arrow for δ^u exists for the *inputs* case, the test could be forced to choose the *waiting for outputs* case with $j = u$. On the other hand, this overlapping can improve the speed of an error detection.

Note 2. In case 2: *inputs*, to check γ^i seams to mean that we should check that for all μ in L_I^i the impossibility to do μ at a precise time. However, this is not feasible in practice, at least in one step. It is possible to try with any input in that channel, thanks to the *input enabling* assumption.

Example 4. Figure 4 shows a test for the cash machine. The test checks that it is not possible to ask for money before a card is authenticated. Then a card and a PIN are inserted and if the PIN was correct it is possible to ask for money.

For simplicity, in the figure the outputs are represented as follows: *card*! as $c!$, *amount*! as $a!$, *Ok*! as $o!$, $Err{-}P!$ as $eP!$ and $Err{-}a!$ as $ea!$.

Soundness. The test generation procedure presented is sound with respect to $\textbf{mtioco}_{\mathcal{M}}$ relation. This very important property is shown in the following theorem.

Theorem 1. *Let spec be a specification in TIOTS*$(\mathcal{L}_I, \mathcal{L}_U)$*, then for all* $\mathcal{M}$*-*quiescent *implementations imp in TMIOTS*$(\mathcal{L}_I, \mathcal{L}_U)$ *and all test cases* $\mathbf{t}$ *obtained from spec by the above procedure:*

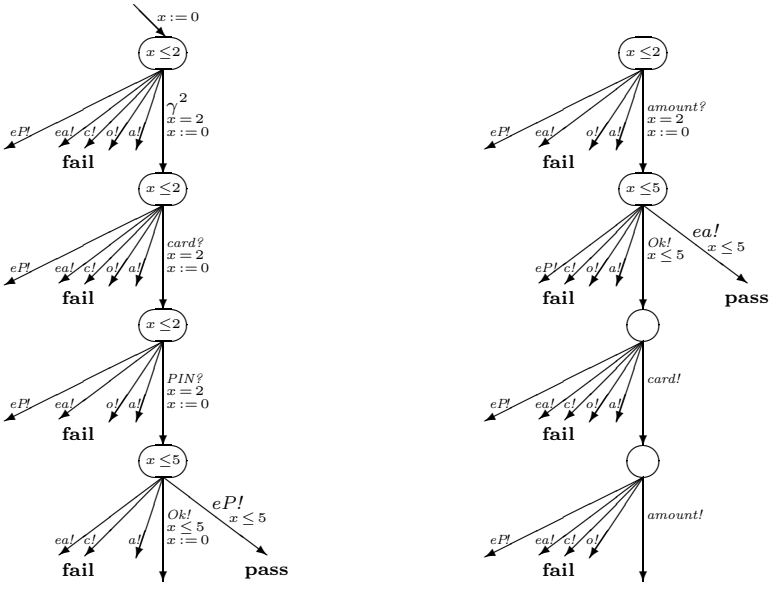


Fig. 4. A test case for the cash machine

$$imp \ \mathbf{mtioco}_{\mathcal{M}} \ spec \Rightarrow imp \ \mathbf{passes} \ \mathbf{t}.$$

The proof of this theorem as well as that of the following on completeness build on the notion of *saturation* of *nttraces*. Its definition can be found, together with the proofs, in [6].

Completeness. The test generation procedure is also exhaustive in the sense that for each non-conforming implementation a test case can be generated that detects the non-conformance.

Theorem 2. *Let spec be a specification in TIOTS*$(\mathcal{L}_I, \mathcal{L}_U)$*, then for all M-*quiescent *implementation imp in TMIOTS*$(\mathcal{L}_I, \mathcal{L}_U)$ *with: imp* $\mathbf{mti\not{o}co}_{\mathcal{M}}$ *spec, there exists a test case* $\mathbf{t}$ *generated from spec by the above procedure such that:*

$$imp \ \mathbf{pa\not{s}ses} \ \mathbf{t}.$$

The exhaustiveness of our test generation procedure, similar to the one in [5], is less useful than the corresponding result in the untimed case. There, the repeated execution of the test generation algorithm in a fair, nondeterministic manner, will generate for every error a test exposing it in finite time. This is not feasible for the real-time case, as the number of potential test cases is uncountable because of the underlying continuous time domain. It is possible to recover such *limit-completeness* by considering suitable equivalent classes of errors (i.e. an implementation has either all or no errors of a given class), such that a repeated test generation procedure will automatically expose an error in every equivalence class. This is ongoing work.

4 Conclusions

To the best of our knowledge, we propose the first attempt to generate test cases from multi input-output real-time specifications. More specifically, our contributions are:

- We show how the concept of *multi input-output transition systems* can be applied to the modelling of real-time systems.
- We develop a new parameterized conformance relation using the enriched *real-time multi input-output transition systems*.
- The relevance of the model and its theory for test generation is illustrated by modification of a small but realistic example of a cash machine due to Heerink [8].

Related Work. Heerink's work in [8] is an extension of Tretmans' **ioco** theory [19]. Its testing theory is based on singular observers: only one output channel is observed at the time. In [15] a similar theory is presented with an alternative type of observers: all-observer, which can observe all the output channels simultaneously. Both approaches are concerned with untimed systems. In [5] a test generation framework for real-time systems with repetitive *quiescence* is presented, extending the Tretmans' **ioco** theory [19] for real-time systems. This framework is the basis for the approach taken in this paper.

Of the wealth of literature on test generation for real-time systems we mention the related work that can be found in [11,14], but these authors consider neither *quiescence* nor multiple channels. A related approach involving symbolic data can be found in [7].

Future Work. We are continuing our work along three lines. First, we are studying the *limit-completeness* over our approach as explained above. Second, we are working on a more detailed comparison of the present approach and the $\mathbf{tioco}_M$ theory [5]. Finally, we are working on the implementation of the multiple input-output theory as an extension of the TorX tool [1,2].

References

1. A. Belinfante, J. Feenstra, R. deVries, 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, *Int. Workshop on Testing of Communicating Systems 12*, pages 179–196. Kluwer, 1999.
2. H. Bohnenkamp and A. Belinfante. Timed testing with torx. In J. Fitzgerald, I.J. Hayes, and A. Tarlecki, editors, *FM 2005: Formal Methods*, pages 173–188. Springer, 2005.
3. E. Brinksma. On the existence of canonical testers. In *Memorandum INF-87-5*. University of Twente, Enschede, The Netherlands, 1987.
4. E. Brinksma. A theory for the derivation of tests. In *Protocol Specification, Testing, and Verification VIII, North-Holland*, page 6374. S. Aggarwal and K. Sabnani, 1988.
5. L. Brandán Briones and E. Brinksma. A test generation framework for quiescent real-time systems. In *Formal Approaches to Software Testing: 4th International Workshop, FATES*, volume 3395/2005. Springer-Verlag GmbH. Extended Version `http://fmt.cs.utwente.nl/research/testing/files/BBB04.ps.gz`, 2004.
6. L. Brandán Briones and E. Brinksma. Testing real-time multi input-output systems. Extended Version. Number TR-CTIT-05-40. `http://fmt.cs.utwente.nl/re-search/testing/files/BBB05.ps.gz`, 2005.
7. L. Frantzen, J. Tretmans, and T.A.C. Willemse. Test generation based on symbolic specifications. In *FATES 2004*. Springer-Verlag, 2005.
8. L. Heerink. Ins and outs in refusal testing. In *PhD thesis*, 1998.
9. M. Hennessy. Algebraic theory of processes. In *Foundations of Computing. Series. The MIT Press*, 1988.
10. ISO8807. *A formal description technique based on the temporal ordering of observational behaviour*. Int. Organization for Standardization, 1989.
11. M. Krichen and S. Tripakis. Black-box conformance testing for real-time systems. In *SPIN 2004*, pages 109–126. Springer-Verlag Heidelberg, 2004.
12. R. Langerak. A testing theory for lotos using deadlock detection. In *Proceedings of the IFIP WG 6.1 Ninth int. Symp. on Protocol Spec., Testing, and Verification*, pages 87–98. IFIP, 1990.
13. K. Larsen, M. Mikucionis, and B. Nielsen. Real-time system testing on-the-fly. In K Sere, M Walden, and A Karlsson, editors, *The 15th Nordic Workshop on Programming Theory (NWPT)*, Åbo Akademi University, Turku, Finland, oct 2003. Extended abstract.
14. K. Larsen, M. Mikucionis, and B. Nielsen. Online testing of real-time system using uppaal. In *Formal Approaches to Software Testing*, Linz, Austria, 2004.

15. Z. Li, J. Wu, and X. Yin. Testing multi input/output transition system with all-observer. In *TestCom*, pages 95–111, 2004.
16. R. De Nicola. Extensional equivalences for transition systems. In *Acta Informatica*, page 24:211237, 1987.
17. R. De Nicola and M.C.B. Hennessy. Testing equivalences for processes. In *Theoretical Computer Science*, page 34:83133, 1984.
18. J. Springintveld, F. Vaandrager, and P. D'Argenio. Testing timed automata. *Theoretical Computer Science*, 254(1–2):225–257, 2001.
19. J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. In *Software-Concepts and Tools, 17(3)*, pages 103–120. Also: Technical Report N0. 96-26, Center for Telematics and Information Technology, University of Twente, The Netherlands, 1996.