Testing multi input-output real-time systems
(Extended Version)

Laura Brandán Briones and Ed Brinksma

Faculty of Computer Science, University of Twente,
P.O.Box 217, 7500AE Enschede,
The Netherlands. Fax - (+31 53)-489-3247
{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 $\text{tioco}_M$ theory, an enriched theory for timed testing with repetitive quiescence, and allowing the partition of input sets and output sets, we introduce the $\text{mtioco}_M$ relation. A test derivation procedure which is nondeterministic and parameterized is further developed, and shown to be sound and complete wrt $\text{mtioco}_M$.

Keywords: testing, real-time, quiescent, channels.

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 [NH84], further elaborated in [Nic87, Hen88]. 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 [Bri87, Bri88]. Tretmans [Tre96] studied test generation for I/O transition systems. Building on this work, Heerink [Hee98] 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 [BB04], 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
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 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 BB04 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. BB04,Lan90,Tre96).

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 [Hee98] 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 $\text{tioco}_M$, by a vector of bounds $\mathcal{M} = \langle M_1, \ldots, M_n \rangle$ for the different output channels. In $\text{tioco}_M$ the global bound $M$ is a parameter which inform

Fig. 1. Relation between test generation approaches.
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 $\text{mtioco}_M$ conformance relation. We develop a test derivation procedure for $\text{mtioco}_M$, which is shown to be sound and complete. Therefore, our work can be seen as a real-time extension of Heerink’s $\text{mioco}$ 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 $\text{mtioco}_M$. 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 $\text{mtioco}_M$ relation. Finally, Section 4 presents the conclusions of the paper.

## 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 $\text{tmior}$ relation. Later on, the timed transition relation is extended with quiescence and refusals and a parameterized relation is defined: $\text{mtiorf}$ relation. Finally, the concept of observed outputs set is introduced and the $\text{mtioco}_M$ 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 $\tau$. 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 $\langle S, s_0, L_{\tau T}, \rightarrow \rangle$, where

- $S$ is a non-empty set of states. With $s_0 \in S$ as the initial state.
- $L_{\tau T} \equiv L \cup \{\tau\} \cup T$ are the actions $L$ including the internal action $\tau$ and time-passage actions. Where $\tau \notin L$ and $T \equiv \{d \mid d \in \mathbb{R}_{\geq 0}\}$ with $L \cup \{\tau\} \cup T = \emptyset$.
- $\rightarrow \subseteq (S \times L_{\tau T} \times S)$ is the timed transition relation with the following consistency constraints: $\forall \ d, d_1, d_2 \in T; \forall \ s, s', s'' \in S$
  - **Time Determinism** whenever $s \xrightarrow{d} s'$ and $s \xrightarrow{d} s''$ then $s' = s''$
  - **Time Additivity** ($\exists s' : s \xrightarrow{d_1} s', s'' \xrightarrow{d_2} s''$) 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 \( LT (L_T \triangleq L \cup T) \) represent the observable actions of a system, i.e. labelled actions and passage of time. The \( \tau \) 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:

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

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 [SVD01].

Example 1. Our example is an adapted version of the cash machine in [Hee98]. 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. If the PIN or the amount of money are too 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 \( \langle S, s_0, L_T, \rightarrow \rangle \), with \( S = \{q_0, \ldots, q_{11}\} \), \( s_0 = q_0 \), \( L_I = \{\text{card}?, \text{PIN}?, \text{amount}?) \) and \( L_U = \{\text{card}!, \text{amount}!, \text{Ok}!, \text{Err}-P!, \text{Err}-a!\} \).

\[
\begin{align*}
&T \quad \text{card!} \quad x > 5 \quad x \leq 0 \\
&P \quad \text{PIN?} \quad x \leq 5 \quad x \geq 0 \\
&\text{Err-P!} \quad x \geq 5 \quad x \leq 5 \\
&\text{Err-a!} \quad x \leq 5 \quad x > 5 \\
&\text{Ok!} \quad x \leq 5 \quad x \geq 0 \\
&\text{amount!} \quad x \leq 5 \quad x \geq 0 \\
&\text{card?} \quad x = 0 \\
&\text{amount?} \quad x = 0 \\
&\text{Ok}! \quad x = 0 \\
&\text{Err-P!} \quad x = 0 \\
&\text{Err-a!} \quad x = 0 \\
&\text{card!} \quad x = 0 \\
&\text{PIN?} \quad x = 0 \\
&\text{Err-P!} \quad x = 0 \\
&\text{Err-a!} \quad x = 0 \\
&\text{Ok!} \quad x = 0 \\
&\text{amount!} \quad x = 0 \\
&\text{card!} \quad x = 0 \\
&\text{PIN?} \quad x = 0 \\
&\text{Err-P!} \quad x = 0 \\
&\text{Err-a!} \quad x = 0 \\
&\text{Ok!} \quad x = 0 \\
&\text{amount!} \quad x = 0 \\
&\text{card!} \quad x = 0 \\
&\text{PIN?} \quad x = 0 \\
&\text{Err-P!} \quad x = 0 \\
&\text{Err-a!} \quad x = 0 \\
&\text{Ok!} \quad x = 0 \\
&\text{amount!} \quad x = 0 \\
&\text{card!} \quad x = 0 \\
&\text{PIN?} \quad x = 0 \\
&\text{Err-P!} \quad x = 0 \\
&\text{Err-a!} \quad x = 0 \\
&\text{Ok!} \quad x = 0 \\
&\text{amount!} \quad x = 0 \\
\end{align*}
\]

Fig. 2. A cash machine, a modified version of [Hee98]
To explicitly encode the inability for a state to perform any action in a set \( A \) or any internal action \( \tau \), we extend the \textit{timed transition relation} with self-loop transitions: \( s \xrightarrow{A} s \), in case \( A \) is a refusal of \( s \):

\[
s \xrightarrow{A} s' \iff \forall \mu \in (A \cup \{\tau\}) : s \xrightarrow{\mu} s = s'.
\]

We use the well-known notation: \( s \xrightarrow{\sigma} \) to denote that there exists a reachable state \( q \) from \( p \) by performing \( \sigma \) 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 \( p \xrightarrow{\sigma} \) instead of \( s_0 \xrightarrow{\sigma} \).

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

- \( \text{f-traces}(p) = \{ \sigma \in (P(L) \cup L_T)^* \mid p \xrightarrow{\sigma} \} \)
- \( \text{t-traces}(p) = \text{f-traces}(p) \cap L_T^* \)
- \( \text{init}(p) = \{ \mu \in L_T \mid \exists p' : p \xrightarrow{\mu} p' \} \)
- \( \text{der}(p) = \{ p' \mid \exists \sigma \in L_T^* : p \xrightarrow{\sigma} p' \} \)
- \( P\text{ after} \sigma = \{ p' \mid \exists p \in P : p \xrightarrow{\sigma} p' \} \)

\( p \) is deterministic if and only if \( \forall \sigma \in L_T^* : |P\text{ after} \sigma| \leq 1 \)

where \( P(L) \) denotes the power set of \( L \).

As expected, a \textit{timed trace} (ttrace) is a standard trace extended with time. A \textit{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 \textit{init} is the set of all possible actions from a given state, the \textit{der} is the set of all reachable states from a given state, and the \textit{after} is the set of all states reachable after a given ttrace. We call a system \textit{deterministic} if for all ttrace’s it has at most one reachable state.

**Example 2.** In Figure 2 we can observe that the \( \text{init}(q_0) = \{ \text{card?} \} \), the \( \text{der}(q_0) \) is the set of all states \( \{q_1, \ldots, q_{11}\} \) and \( \{q_{11}, q_8\} \text{ after Ok!} = \{q_8\} \). 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 \textit{TIOTS}(\( L_I, L_U \)) is a timed input-output transition system \( \text{TIOTS}(L_I, L_U) \) where the set of inputs and outputs are partitioned into channels \( L_I = \{L_1^I, \ldots, L_n^I\} \) and \( L_U = \{L_1^U, \ldots, L_m^U\} \).

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

**Definition 3.** Let \( p \) and \( q \) be \( \text{TIOTS}(L_I, L_U) \), then

\[
q \sqsubseteq_{\text{tmior}} p \iff \text{f-traces}(q) \cap (L_T \cup L_I \cup L_U)^* \subseteq \text{f-traces}(p).
\]
2.2 An extended model and relation

The \texttt{tmior} relation, from Section 2.1 induced us to define an extension of \textit{TIO}TS where the input and output sets are subdivides in channels. Then, a \textit{timed multi input-output transition system} (\textit{TMIOTS}(\mathcal{L}_I, \mathcal{L}_U)) is a \textit{TIO}TS(\mathcal{L}_I, \mathcal{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 \( \mathcal{L}_I = \{L^1_I, \cdots, L^n_I\} \) and \( \mathcal{L}_U = \{L^1_U, \cdots, L^m_U\} \) a \textit{Timed Multi Input-Output Transition System} \( p \) \textit{TMIOTS}(\mathcal{L}_I, \mathcal{L}_U) \textit{is a TIO}TS with \( L_I = \bigcup_{1 \leq i \leq n} L^i_I \) and \( L_U = \bigcup_{1 \leq j \leq m} L^j_U \), where

\[
\forall s \in \text{der}(p) : (\forall \mu \in L^i_I : s \not\xrightarrow{\mu}) \lor (\forall \mu \in L^j_I : s \xrightarrow{\mu})
\]

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

**Example 3.** It is possible to see the cash machine of Figure 2 as a \textit{TMIOTS} \( \langle \mathcal{L}_I, \mathcal{L}_U \rangle \langle S, s_0, L_T, \neg \rangle \), where \( S = \{q_0, \cdots, q_{11}\} \), \( s_0 = q_0 \), \( L_I = \{L^1_I, L^2_I\} \) and \( \mathcal{L}_U = \{L^1_U, L^2_U, L^3_U\} \) with \( L^1_I = \{\text{card}\}, L^2_I = \{\text{PIN}?, \text{amount}?!\} \) and \( L^1_U = \{\text{card}!\}, L^2_U = \{\text{amount}!\}, L^3_U = \{\text{Ok}!, \text{Err}!-\text{P}!, \text{Err}!-\text{a}\} \). 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 \textit{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 \textit{TIO}TS(\mathcal{L}_I, \mathcal{L}_U), implying \textit{TIO}TS 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 \( L^i_I \) must imply that the remaining channels stay silent for at least the period of time \( L^i_I \) 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 [BB04], 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 \textit{TIO}TS(\mathcal{L}_I, \mathcal{L}_U) with \( s \) an state of \( p \) is called \( L^i_U \)-quiescent, denoted \( \delta^i(s) \), if and only if
$\forall \mu \in L'_U : \forall d \in \mathbb{R}^{\geq 0} : s \not\xrightarrow{\mu(d)}$

where $s \not\xrightarrow{d}$ is used as the syntactic sugar for $\exists s' : s \xrightarrow{d} s' \not\xrightarrow{\mu},$ and $s \not\xrightarrow{\mu(d)}$ 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 $\tau$ transitions. In the timed case this is not possible. For example, take the definition above with a single arrow, and the following system, with $o! \in L'_U$: $s \not\xrightarrow{o!} s' \xrightarrow{\tau}, s'' \xrightarrow{d'} s''' \xrightarrow{o}$. 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 $L'_U$-quiescence, we extend the timed transition relation to include self-loop transitions for refusals and quiescence. Therefore, $s \not\xrightarrow{\tau} s$ if and only if $s$ refuses $L'_U$, and $s \xrightarrow{d}$ if and only if $s$ is $L'_U$-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$(p) \cap (L_T \cup L_I \cup \mathcal{L}_U)^{*} = ttraces(\Delta(p))$. Therefore, using this notation we can re-write the tmior relation as:

$$q \sqsubseteq_{tmior} p \text{ if and only if } ttraces(\Delta(q)) \subseteq ttraces(\Delta(p)).$$

**Example 4.** Figure 3 illustrates the cash machine with the extended timed transition relation with refusals and quiescence($\gamma^i$) 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.

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 $\mathcal{M} = (M_1, \cdots, M_m) : \forall 1 \leq j \leq m : M_j \in \mathbb{R}^{\geq 0},$ then

- $s$ is $M_j$-quiescent if and only if $\forall s' \in (s \text{ after } M_j) : s' \in L'_U$-quiescent
- $p$ is $M_j$-quiescent if and only if $\forall s \in S : M_j$-quiescent$(s)$
- $p$ is $\mathcal{M}$-quiescent if and only if $\forall 1 \leq j \leq m : M_j$-quiescent$(q)$. 

7
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 $p \in \text{TOTS}(L_I, L_U)$ is $M_j$-quiescent with $S$ as its states and $s \in S$, then:

$$\delta^j(s) \triangleq \forall \mu \in L_U^j : \forall d \in \mathbb{R}^\geq 0 : d \leq M_j : s \not\Rightarrow.$$ 

Proof.

\[
\Rightarrow \delta^j(s) \\
\Rightarrow \{\text{Definition 5}\} \\
\forall \mu \in L_U^j : \forall d \in \mathbb{R}^\geq 0 : s \not\Rightarrow \\
\Rightarrow \{\text{Direct}\} \\
\forall \mu \in L_U^j : \forall d \in \mathbb{R}^\geq 0 : d \leq M_j : s \not\Rightarrow
\]

Fig. 3. A cash machine, a modified version of [Hee98]
M_j-quiescent(p) \Rightarrow \{\text{Definition 6}\}
M_j-quiescent(s) \Rightarrow \{\text{Definition 6}\}
\forall s', \in (s \text{ after } M_j) : s' \in L^j_U - \text{quiescent}
\Rightarrow \{\text{using } \forall \mu \in L^j_U : \forall d \in \mathbb{R}^{\geq 0} : d \leq M_j : s \not\Rightarrow}\}
δ^j(s)

Corollary 1. Let \( p \in \text{TIOTS}(\mathcal{L}_I, \mathcal{L}_U) \) be \( M \)-quiescent with \( S \) as its states, \( s \in S \), and \( M = \langle M_1, \cdots, M_m \rangle \), then
\[
\forall j = 1, \cdots, m : (δ^j(s) \Leftrightarrow (\forall \mu \in L^j_U : \forall d \in \mathbb{R}^{\geq 0} : d \leq M_j : s \not\Rightarrow)).
\]

Proof.
\( M \)-quiescent(p)
\[\Leftrightarrow \{\text{Definition 5}\}\]
\[\forall j : j = 1, \cdots, m : M_j\text{-quiescent}(p)\]
\[\Leftrightarrow \{\text{Lemma 1}\}\]
\[\forall \mu \in L^j_U : \forall d \in \mathbb{R}^{\geq 0} : d \leq M_j : s \not\Rightarrow\]

Example 5. Considering the cash machine in Figure 3 for \( M = \langle M_1, M_2, M_3 \rangle \) 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 \( M \)-quiescence for a system, we define the \textbf{ntiorf} relation, parameterized by \( M \). In the traces considered in \textbf{ntiorf} a \( \delta^j \) can only occur after \( M_j \) timed units.

Definition 7. Let \( p \) be a \( \text{TIOTS}(\mathcal{L}_I, \mathcal{L}_U) \) and \( q \) be a \( M \)-quiescent \( \text{TMIOTS} \) \( \langle \mathcal{L}_I, \mathcal{L}_U \rangle \), then
\[
q \sqsubseteq_{\text{nntiorf}M} p \text{ if and only if } \Delta_M(q) \subseteq \Delta_M(p)
\]
where for \( r \in \text{TIOTS}(\mathcal{L}_I, \mathcal{L}_U) \), with \( \epsilon \) as the empty word:
\[
\Delta_M(r) \triangleq \text{traces}(\Delta(r)) \cap \bigcup_j (((\{T \cup \{\epsilon\}\} \cdot (L \cup \gamma)) \cup M_j \cdot \delta^j)^*).
\]

2.3 The relation: \textbf{nntiorf}_\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 \textbf{nntiorf}.
Similarly to the definition of \textit{nttraces} for \textit{tioco}_M theory in [BB04], we present the \textit{normalized ttraces} for TMIOTS.

\textbf{Definition 8.} Let \( p \) be \( M \)-quiescent and \( \sigma \) be a \textit{ttraces} in \( \Delta(p) \), then
- \( \sigma \) is a \textit{normalized ttrace} if and only if \( \sigma \in \bigcup_i \bigcup_j (T \cdot (L \cup \gamma_i \cup \delta_j))^* \)
- \( \text{nttraces}(p) = \{ \sigma \in \bigcup_i \bigcup_j (T \cdot (L \cup \gamma_i \cup \delta_j))^* \mid p \stackrel{\sigma}{\Rightarrow} \} \)
- for \( \text{nttraces} \), \( \sigma = d_0 \delta_1 \gamma_1 \delta_2 a! \) we also write \( \hat{\sigma} = \delta_1(d_0) \gamma_1(d_1) a!(d_2) \).

Moreover, the definition of \textit{nttraces} already assumes that TIOTS(\( \mathcal{L}_I, \mathcal{L}_U \)) systems have the \textit{timed transition relation} extended, implying
\[ \text{nttraces}(\Delta(r)) = \text{nttraces}(r). \]

An example of an \textit{nttrace} in the cash machine is:
\[ \text{card}(3)\text{PIN}(2)\text{Err}^{-}P!(5)\gamma_1(6)\text{card}!(0). \]

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 \textit{nttraces} for two systems is equal to the inclusion of \textit{ttraces} for the corresponding extended systems. This result is given in the following lemma.

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

\textit{Proof.}
\[ \Rightarrow \] Direct using Definition 8
\[ \Leftrightarrow \] \( \sigma \in \text{ttraces}(\Delta(p_1)) \), and \( \mu \in L^i_I \) or \( \mu = \gamma^i \) then
\[ \sigma \cdot \mu \in \text{ttraces}(p_1) \]
\[ \Rightarrow \{ \text{Definition 8} \} \]
\[ \hat{\sigma} \cdot \mu \in \text{nttraces}(\Delta(p_1)) \]
\[ \Rightarrow \{ \text{property of nttraces} \} \]
\[ \hat{\sigma} \cdot \mu \in \text{nttraces}(p_1) \]
\[ \Rightarrow \{ \text{hyp.} \} \]
\[ \hat{\sigma} \cdot \mu \in \text{nttraces}(p_2) \]
\[ \Rightarrow \{ \text{property of nttraces} \} \]
\[ \hat{\sigma} \cdot \mu \in \text{nttraces}(\Delta(p_2)) \]
\[ \Rightarrow \{ \text{Definition 8} \} \]
\[ \sigma \cdot \mu \in \text{nttraces}(\Delta(p_2)) \]
\[ \Rightarrow \{ \text{density} \} \]
\[ \sigma \in \text{ttraces}(\Delta(p_2)) \]
\[ \square \]

The \textit{observed output set} of a given set of states \( P \), denoted \( \text{obsOut}_M(P) \), is defined as the union of two sets: the set of output actions enriched with \textit{quiescent}, denoted \( \text{obsOut}^q_M \), and the set of \textit{refusals}, denoted \( \text{obsOut}^r_M \). Hence, \( \text{obsOut}^q_M \) is the set of outputs that could happen after a period of time plus the special symbol \( \delta_j(M_j) \) expressing \textit{quiescence} on output channel \( j \) in case a reachable state after \( M_j \) is \textit{quiescent} on channel \( j \). And, the set \( \text{obsOut}^r_M \) is the set of \textit{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:

$$
\text{obsOut}_M(P) = \bigcup_{p \in P} \text{obsOut}^o_M(p) \bigcup_{p \in P} \text{obsOut}^r_M(p)
$$

where:

$$
\text{obsOut}^o_M(p) = \{ \mu(d) \mid \mu \in \mathcal{L}_U \land p \xrightarrow{\mu(d)} \} \bigcup \bigcup_{\delta^j(M_j)} \{ \gamma^i(d) \mid \forall \mu \in \mathcal{L}_I : p \xrightarrow{\mu(d)} \} \n$$

$$
\text{obsOut}^r_M(p) = \bigcup_{\delta^i(M_j)} \{ \delta_j(M_j) \mid p \xrightarrow{\delta^i(M_j)} \}
$$

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

Lemma 3. Let $p \in \text{TIOTS(}\mathcal{L}_I, \mathcal{L}_U)$ and $\sigma \in \text{nttraces}$, then

$$
\text{obsOut}_M(p \text{ after } \sigma) = \emptyset \text{ if and only if } \sigma \not\in \text{nttraces}(p).
$$

Proof.

$\left[\Rightarrow\right]$ Direct.

$\left[\Rightarrow\right]$ Suppose $L_j^i \in \mathcal{L}_U$ and $\sigma \in \text{nttraces}(p)$, then $\exists p' : p \xrightarrow{\sigma} p'$, then

1) $\exists \mu \in L_j^i : \exists d \in \mathbb{R}_{>0} : p' \xrightarrow{\mu(d)} \\
\mu(d) \in \text{obsOut}_M(p \text{ after } \sigma)$

or

2) $\forall \mu \in L_j^i : \forall d \in \mathbb{R}_{>0} : p' \xrightarrow[\neq]{\mu(d)} \\
\delta^j(M_j) \in \text{obsOut}_M(p \text{ after } \sigma)$

$\square$

We also prove that the parameterized $\text{mtiorf}$ relation is equal to checking the inclusion of observed output set for all $\text{nttraces}$ that only have $\delta^j$ after $M_j$ timed units.

Lemma 4. Let $p \in \text{TIOTS(}\mathcal{L}_I, \mathcal{L}_U)$ and $q$ be $\mathcal{M}$-quiescent $\text{TMIOTS(}\mathcal{L}_I, \mathcal{L}_U)$, then $q \subseteq^{\mathcal{M}} \text{mtiorf } p$ if and only if $\forall \sigma \in \Delta_M :$

$$
\text{obsOut}_M(q \text{ after } \sigma) \subseteq \text{obsOut}_M(p \text{ after } \sigma).
$$

Proof.

$\left[\Rightarrow\right]$ Let $\sigma \in \Delta_M$, then

1) if $\sigma \not\in \text{nttraces}(q)$, then $\text{obsOut}_M(q \text{ after } \sigma) = \emptyset$

2) if $\sigma \in \text{nttraces}(q)$, then

$\forall \mu(d) \in \text{obsOut}_M(q \text{ after } \sigma) \\
\Rightarrow \{\text{Definition 9}\} \\
\sigma \cdot \mu(d) \in \Delta_M(q) \\
\Rightarrow \{\text{hypothesis}\} \\
\sigma \cdot \mu(d) \in \Delta_M(p) \\
\Rightarrow \{\text{Definition 9}\} \\
\mu(d) \in \text{obsOut}_M(p \text{ after } \sigma)$
Let $\sigma \in \Delta_M(q) : \sigma \in \text{nttraces}(q) \cap \Delta_M$ then,

$$(\Delta(q) \text{ after } \sigma) \neq \emptyset$$

let $L_I^I \in L_I$, then

$$\exists a \in L_I^I : \mu = a(d) \land \exists q' : q' \in (\Delta(q) \text{ after } \sigma) \land q' \downarrow$$

$$\mu \in \text{obsOut}_M(q \text{ after } \sigma)$$

$$\Rightarrow \{\text{hypothesis}\}$$

$$\mu \in \text{obsOut}_M(p \text{ after } \sigma)$$

$$\Rightarrow$$

$$\text{obsOut}_M(p \text{ after } \sigma) \neq \emptyset$$

$$\Rightarrow \{\text{Lemma 3}\}$$

$$\sigma \in \text{nttraces}(p).$$

Finally, we are in position to define the $\text{mtioco}_{M}$ 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 $\text{TIOTS}(L_I, L_U)$ and $q$ an implementation in $\text{TMIOTS} (L_I, L_U)$: $q$ will be $\text{mtioco}_{M}$ to $p$ if and only if the observed output set of $q$, after every $\text{nttrace}$ of $p$ is a subset of the observed output set of $p$ after the same $\text{nttrace}$.

**Definition 10.** Let $p$ be a $\text{TIOTS}(L_I, L_U)$ and $q$ be $M$-quiescent $\text{TMIOTS} (L_I, L_U)$, then:

$$q \text{ mtioco}_{M} p \triangleq \forall \sigma \in \Delta_M(p) : \text{obsOut}_M(q \text{ after } \sigma) \subseteq \text{obsOut}_M(p \text{ after } \sigma).$$

The $\text{mtioco}_{M}$ 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 $\text{TMIOTS}(L_I, L_U)$.

### 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 $\text{mtioco}_{M}$ relation. Moreover, it is shown that this procedure is sound and complete.

A test case $t$ is a $\text{TIOTS}(L_I, L_U)$ 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 $\text{pass}$ and $\text{fail}$ without outgoing transitions. For any state different from $\text{pass}$ and $\text{fail}$ 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 $\tau$-transitions are not allowed. The class of test cases over $L_I$ and $L_U$ is denoted as $TTEST(L_I, L_U)$. A test suite $T$ is a
set of test cases: \( T \subseteq TTEST(\mathcal{L}_I, \mathcal{L}_U) \). 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 \( t \) be a test in \( TTEST(\mathcal{L}_I, \mathcal{L}_U) \) and \( \text{imp} \) be a \( \mathcal{M} \)-quiescent implementation in \( TMIOTS(\mathcal{L}_I, \mathcal{L}_U) \), then

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

\[
\begin{align*}
\text{imp} \xrightarrow{\tau} \text{imp}' & \quad \vdash t||\text{imp} \xrightarrow{\tau} t||\text{imp}' \\
\text{imp} \xrightarrow{\delta \gamma} t' & \quad \vdash t||\text{imp} \xrightarrow{\delta \gamma} t'||\text{imp}' \\
\text{imp} \xrightarrow{\mu t', \text{imp} \xrightarrow{\mu}} \text{imp}', \mu \in L_i & \quad \vdash t||\text{imp} \xrightarrow{\gamma \mu} t'||\text{imp}' \\
\text{imp} \xrightarrow{d} t', \text{imp} \xrightarrow{d} \text{imp}', d \in \mathbb{R}^\geq & \quad \vdash t||\text{imp} \xrightarrow{d} t'||\text{imp}'
\end{align*}
\]

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

\[
\begin{align*}
\text{test run of } t \text{ and } \text{imp} & \triangleq \exists \text{imp}': (t||\text{imp} \xrightarrow{\sigma} \text{pass}||\text{imp}' \text{ or } (t||\text{imp} \xrightarrow{\sigma} \text{fail}||\text{imp}')) \\
\text{imp passes } t & \triangleq \forall \sigma \in \Delta_M : \forall \text{imp}': t||\text{imp} \xrightarrow{\sigma} \text{pass}||\text{imp}' \\
\text{imp passes } T & \triangleq \forall t \in T : \text{imp passes } t \\
\text{imp fails } T & \triangleq \exists t \in T : \text{imp passes } t.
\end{align*}
\]

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 [ISO89]:

\[
B \triangleq a; B \mid B + B \mid \Sigma B
\]

where \( a \in L_{\gamma \delta} ; \left( L_{\gamma \delta} \triangleq L_{\gamma} \cup \{\gamma^i \mid 1 \leq i \leq n\} \cup \{\delta^j \mid 1 \leq j \leq m\} \right) \), \( B \) is a countable set of behaviour expressions, and axioms plus inference rules are:
\( a \in L \) \( \vdash a; B \xrightarrow{a} B' \)
\( a = d, d' < d \) \( \vdash d; B \xrightarrow{d'} d - d'; B \)
\( a = d \) \( \vdash B \xrightarrow{d} B' \)
\( B_1 \xrightarrow{\mu} B'_1, \mu \in L_{T\gamma\delta} \) \( \vdash B_1 + B_2 \xrightarrow{\mu} B'_1 \)
\( B_2 \xrightarrow{\mu} B'_2, \mu \in L_{T\gamma\delta} \) \( \vdash B_1 + B_2 \xrightarrow{\mu} B'_2 \)
\( B \xrightarrow{\mu} B', B \in B, \mu \in L_{T\gamma\delta} \) \( \vdash \Sigma B \xrightarrow{\mu} B' \)

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 T IOTS \((L_I, L_U)\). Similar to [Tre96, BB04] 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: \( o(d) \in \text{obsOut}_M(S) \) or \( \gamma^i(d) \in \text{obsOut}_M(S) \); 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 [KT04, LMN03] with sub partitioning of the input and output sets.

1. termination

\( t := \text{pass} \)

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

2. inputs

Choose \( k \in [0, \text{Max}\{M_1, \ldots, M_m\}] \) and \( \mu \in L_I \)

\( x := 0 \)

\[ \begin{align*}
  & \text{\( o_1(d_1) \in \text{obsOut}_M(S) \)} \\
  & \text{\( o_1(d_1) \notin \text{obsOut}_M(S) \)} \\
  & x = d_1 \\
  & x = d_n \\
  & x = k \\
  & \text{\( o_2(d_2) \)} \\
  & \text{\( o_2(d_2) \notin \text{obsOut}_M(S) \)} \\
  & x = d_1 \\
  & x = d_n \\
  & x = k \\
  & \text{\( o_3(d_3) \)} \\
  & \text{\( o_3(d_3) \notin \text{obsOut}_M(S) \)} \\
  & x = d_1 \\
  & x = d_n \\
  & x = k \\
  & \text{\( o_m(d_m) \)} \\
  & \text{\( o_m(d_m) \notin \text{obsOut}_M(S) \)} \\
  & x = d_1 \\
  & x = d_n \\
  & x = k \\
\end{align*} \]

\( t_1 \) \( t_n \) \( t_\mu \) \( t_\gamma \)

fail fail fail
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 \( \delta^u \), then it is clear
that the test will never succeed to make the input or check for \( \gamma' \). This knowledge
could be used, once it is known that an arrow for \( \delta^u \) 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 \( \gamma' \) seems to mean that we should check that
for all \( \mu \) in \( L_1^j \) the impossibility to do \( \mu \) 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 6. 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!.
We use the term saturation to refer of saturations of $\delta$'s in ntraces. The following definition makes a relation between ntraces with $\delta$ in a particular place and a similar one, without $\delta$ in that place.

**Definition 12.** Let $p \in \text{TIO}TS \ (\mathcal{L}_I, \mathcal{L}_U)$ and $\sigma, \sigma'$ be normal form of $\Delta_M(p)$, then

- $\sigma \delta_M \sigma'$ if and only if  $\exists j : \exists \sigma_1, \sigma_2 : \exists \mu : \exists d \geq \max\{M_1, \cdots, M_m\}$:
  
  $\sigma = \sigma_1 \cdot \mu(d) \cdot \sigma_2$  
  $\sigma' = \sigma_1 \cdot \delta^j(M_j) \cdot \mu(d - M_j) \cdot \sigma_2$

- $\sigma \in \Delta_M(p)$ is $\delta_M$-saturated if and only if  $\forall \sigma' : \sigma \delta_M \sigma' : \sigma = \sigma'$.

A $\delta_M$-saturated ntrace is a ntrace that do not allowed an action came after max$\{M_1, \cdots, M_m\}$ without observe quiescence.

**Proposition 1.** If $\sigma$ is $\delta_M$-saturated, then for all $\mu(d) \neq \delta^j(M_j)$ occurring in $\sigma$ we have $d < \max\{M_1, \cdots, M_m\}$.

**Proof.** Directly from the definition of the $\delta_M$ relation. \hfill $\square$

**Soundness** The test generation procedure presented is sound with respect to mtioco$_M$ relation. This very important property is shown in the following theorem.

**Theorem 1.** Let spec be a specification in TIO$TS(\mathcal{L}_I, \mathcal{L}_U)$, then for all $\mathcal{M}$-quiescent implementations imp in TMIOTS$TS(\mathcal{L}_I, \mathcal{L}_U)$ and all test cases $t$ obtained from spec by the above procedure:
\[ \text{imp } \text{ntioco}_M \text{ spec } \Rightarrow \text{imp passes t}. \]

**Proof.** Let \( \text{imp} \) be \( M \)-quiescent with \( (\text{imp } \text{ntioco}_M \text{ spec}) \), then we will show that for all \( \sigma \in \Delta_M(\text{spec}) \) and all test cases \( t \) generated by the procedure from \( \text{spec} \):

\[
\text{if } (t \parallel \text{imp} \trans \triangledown t' \parallel \text{imp'}) \text{ then } (t' \neq \text{fail}).
\]

Without loss of generality we can assume that \( \sigma \) is \( \delta_M \)-saturated.

By induction on the length of \( \sigma \):

- If \( \sigma = \epsilon \) and \( t \parallel \text{imp} \trans t' \parallel \text{imp'} \)
  - if \( t \) was constructed using case 1 in the first step, then \( t \parallel \text{imp} \trans \text{pass} \parallel \text{imp'} \)
  - if \( t \) was constructed using cases 2 or 3 in the first step, then \( t = t' \neq \text{fail} \) and all derivations of \( \trans \) have the form:
    \[
    t \parallel \text{imp} \trans t \parallel \text{imp'} \]
- If \( \sigma = \sigma' \cdot a \) and \( (t \parallel \text{imp} \trans t'' \parallel \text{imp''} \trans t' \parallel \text{imp'}) \wedge a = \mu(d) \),
  - From case 2: \( \mu \in (L_1 \text{ or } \mu = \gamma') \wedge d < \text{Max}\{M_1, \cdots, M_m \} \), then because \( \text{imp } \text{ntioco}_M \text{ spec} \)
    - if \( \mu \in (L_1 \text{ or } \mu = \gamma') \) then \( \gamma'(d) \notin \text{obsOut}_M(\text{spec after } \sigma') \), then \( (t \parallel \text{imp} \trans t' \parallel \text{imp'}) \wedge t' \neq \text{fail} \)
    - if \( \mu = \gamma' \) then \( \gamma'(d) \notin \text{obsOut}_M(\text{spec after } \sigma') \), then \( (t \parallel \text{imp} \trans t' \parallel \text{imp'}) \wedge t' \neq \text{fail} \)
  - From case 3 for \( j: \mu \in L_U \wedge a \in \text{obsOut}_M(\text{imp after } \sigma') \), then because \( (\text{imp } \text{ntioco}_M \text{ spec}): \mu(d) \in \text{obsOut}_M(\text{spec after } \sigma') \), and thus \( (t \parallel \text{imp} \trans t' \parallel \text{imp'}) \wedge t' \neq \text{fail} \).

\( \Box \)

**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.

Before proving the theorem of completeness, we establish a useful property. For every specification \( \text{spec} \) in \( \text{TIOTS}(L_1, L_U) \) and every \( \sigma \) \( \delta_M \)-saturated \( \text{nttrace}(\text{spec}) \) such that exist a test \( t' \) for \( (\text{spec after } \sigma) \) then, also exist a test \( t \) such that from \( t \) doing \( \sigma \) it is possible to obtain \( t' \). This property is reflected in the next lemma.

**Lemma 5.** Let \( \text{spec} \in \text{TIOTS}(L_1, L_U), \sigma \in \Delta_M(\text{spec}) \) and \( \sigma \) is \( \delta_M \)-saturated, and \( t' \) a test case generated by the procedure for \( (\text{spec after } \sigma) \) then there exists a test case \( t \) generated from \( \text{spec} \) with \( t \trans t' \).

**Proof.** By induction on the length of \( \sigma \):

- \( |\sigma| = 0 \) then take \( t = t' \)
- Suppose \( t \) exists for all \( \sigma \) with length \( n \) and let \( \sigma = \sigma' \cdot a \) and \( a = \mu(d) \)
  - if \( (\mu \in L_1 \text{ or } \mu = \gamma') \) then using case 2 for the input \( \mu \):
    \[
    t \sigma' \parallel \text{imp''} \trans a \parallel t'
    \]

17
− if \( a \in \text{obsOut}_M(\text{spec after } \sigma) \), using case 3 for \( j (\mu \in L_U^j) \):
\[
  t \xrightarrow{\sigma'} t'' \xrightarrow{a} t' \]

\[ \square \]

**Theorem 2.** Let \( \text{spec} \) be a specification in \( \text{TIOTS}(L_I, L_U) \), then for all \( M \)-quiescent implementation \( \text{imp} \) in \( \text{TMIOTS}(L_I, L_U) \) with: \( \text{imp mti\text{co}}_M \text{spec} \), there exists a test case \( t \) generated from \( \text{spec} \) by the above procedure such that:

\[ \text{imp passes } t. \]

**Proof.** If \( (\text{imp mti\text{co}}_M \text{spec}) \) then there exists \( \sigma \in \Delta_M(\text{spec}) : \text{obsOut}_M(\text{imp after } \sigma) \not\subseteq \text{obsOut}_M(\text{spec after } \sigma) \).

Without loss of generality we can assume that \( \sigma \) is \( \delta_M \)-saturated.

Then, let be \( a = \mu(d) \) such that
\[ a \in \text{obsOut}_M(\text{imp after } \sigma) \setminus \text{obsOut}_M(\text{spec after } \sigma) \text{ and } (\text{imp } \sigma \xrightarrow{a} \text{imp'} \xrightarrow{a'} \text{imp''}). \]

- If \( \mu \in L_U^j \) then let \( t' \) be the result of applying case 3 for \( j \) of the procedure to \( (\Delta(\text{spec}) \text{ after } \sigma) \), and let \( t \) be the test case constructed out of \( t' \) and \( \sigma \) by lemma 5. Because \( a \not\in \text{obsOut}_M(\text{spec after } \sigma) \): \( (t\mid\text{imp } \sigma \xrightarrow{a} \text{fail}||\text{imp''}) \), \( \text{imp passes } t \).

- If \( \mu = \gamma^i \) then let \( t' \) be the result of applying case 2 for \( \gamma^i \) of the procedure to \( (\Delta(\text{spec}) \text{ after } \sigma) \), and let \( t \) be the test case constructed out of \( t' \) and \( \sigma \) by Lemma 5. Because \( a \not\in \text{obsOut}_M(\text{spec after } \sigma) \): \( (t\mid\text{imp } \sigma \xrightarrow{a} \text{fail}||\text{imp''}) \), \( \text{imp passes } t \).

\[ \square \]

The exhaustiveness of our test generation procedure, similar to the one in [BB04], 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 [Hee98].

**Related work.** Heerink’s work in [Hee98] is an extension of Tretmans’ ioco theory [Tre96]. Its testing theory is based on singular observers: only one output channel is observed at the time. In [LWY04] 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 [BB04] a test generation framework for real-time systems with repetitive quiescence is presented, extending the Tretmans’ ioco theory [Tre96] 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 [KT04,LMN04], but these authors consider neither quiescence nor multiple channels. A related approach involving symbolic data can be found in [FTW05].

**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 tiocoM theory [BB04]. Finally, we are working on the implementation of the multiple input-output theory as an extension of the TorX tool [BFd+99,BB05].
References


