$\mathcal{THOTL}$: A timed extension of $\mathcal{HOTL}$*

Mercedes G. Merayo, Manuel Núñez and Ismael Rodríguez

Dept. Sistemas Informáticos y Computación
Universidad Complutense de Madrid, 28040 Madrid, Spain
e-mail: mgmerayo@fdi.ucm.es,mn@sip.ucm.es,isrodrig@sip.ucm.es

Abstract. $\mathcal{THOTL}$ represents a conservative extension of $\mathcal{HOTL}$ (Hypotheses and Observations Testing Logic) to deal with systems where time plays a fundamental role. We adapt some of the $\mathcal{HOTL}$ rules to cope with the new framework. In addition, we introduce several specific hypotheses and rules to appropriately express time assumptions. We provide a correctness result of $\mathcal{THOTL}$ with respect to a general notion of timed conformance.

1 Introduction

In order to determine the *correctness* of an implementation with respect to a given specification we can use a notion of *conformance*: An implementation *conforms* to a specification if the former shows a behavior *similar* to that of the latter. In order to check this property we may use formal testing techniques to extract tests from the specification. Each test represents a desirable behavior that the implementation under test (in the following IUT) must fulfill. In fact, the existence of such a formal framework facilitates the automation of the testing process. In order to limit the (possibly infinite) time devoted to testing, testers add some reasonable assumptions about the structure of the IUT. For example, the tester can assume that the implementation can be represented by means of a deterministic finite state machine, that it has at most n states, etc. In this line, a wide range of testing methodologies have been proposed which, for a specific set of initial hypotheses, guarantee that a test suite extracted from the specification is correct and complete to check the conformance of the IUT (e.g. [4,15,19,6]). However, a framework of hypotheses established in advance is very strict and limits the applicability of a specific testing methodology. For example, it could be desirable that, in a concrete environment, the tester make complex assumptions such as "*non-deterministic states of the implementation cannot show outputs that the machine did not show once the state has been tested 100 times.*" In a different scenario the tester could not believe this assumption but think that "*if she observes two sequences of length 200 and all their inputs and outputs coincide then they actually traverse the same IUT states.*" Let us remark that these are *hypotheses* that the tester is assuming. Thus, she might be wrong and reach a wrong conclusion. However, this is similar to the case when the tester assumes that the implementation is deterministic or that it has at most n states and, in reality, this is not the case.

The logical framework $\mathcal{HOTL}$ [20,21] (*Hypotheses and Observations Testing Logic*) was introduced to cope with the rigidity of other testing frameworks. $\mathcal{HOTL}$

* This research was partially supported by the Spanish MEC project WEST/FAST TIN2006-15578-C02-01 and the Marie Curie project MRTN-CT-2003-505121/TAROT.

aims to assess whether a given set of observations implies the correctness of the IUT under the assumption of a given set of hypotheses. The methodology consists of two phases. The first phase consists in the classical application of tests to the IUT. By using any of the available methods in the literature, a test suite will be derived from the specification. If the application of this test suite finds an unexpected result then the testing process stops: The IUT is not conforming. However, if such a wrong behavior is not detected then the tester cannot be sure that the IUT is correct. In this case, the second phase begins, that is, the tester applies $\mathcal{HOTL}$ to infer whether passing these tests *implies* that the IUT is correct if a given set of hypotheses is assumed. If it does then the IUT is assumed to be correct; otherwise, the tester may be interested in either applying more tests or in assuming more hypotheses (in the latter case, on the cost of *feasibility*) and then applying the logic again until the correctness of the IUT is effectively granted.

$\mathcal{HOTL}$ provides two types of hypotheses: Concerning specific parts (states) of the IUT and concerning the whole IUT. In order to unambiguously denote the states regarded by the former, they will be attached to the corresponding observations that reached these states. For example, if the IUT was showing the sequence of outputs $o_1, o_2, \ldots, o_n$ as answer to the sequence of inputs $i_1, i_2, \ldots, i_n$, the tester may think that the state reached after performing i_1/o_1 is deterministic or that the state reached after performing the sequence $i_1/o_1, i_2/o_2$ is the same as the one reached after performing the whole sequence $i_1/o_1, i_2/o_2, \ldots, i_n/o_n$. In addition to using hypotheses associated to observations, the tester can also consider global hypotheses that concern the whole IUT. These are assumptions such as the ones that we mentioned before: Assuming that the IUT is deterministic, that it has at most n states, that it has a unique initial state, etc. In order to denote the assumption of this kind of hypotheses, specific logic predicates will be used.

Let us remark that even though we work with rules and properties, $\mathcal{HOTL}$, and its timed extension presented in this paper, is not related to *model checking* [8] since we do not *check* the validity of properties: We assume that they hold and we infer results about the conformity of the IUT by using this assumption. In the same way, this work is not related to some recent work on passive testing where the validity of a set of properties (expressed by means of *invariants*) is checked by passively observing the execution of the system (e.g. [3,14]).

The main goal of this paper is to extend $\mathcal{HOTL}$ to deal with timed systems. Even though there exist several proposals to test timed systems (for example, [22,11,7,10,9,12,5,13,2,17]), these proposals suffer from the same *rigidity* previously commented. The first decision to define the new framework, that we call $\mathcal{THOTL}$, is to consider a formal language to represent timed systems. The natural candidate would be to consider timed automata [1]. However, since $\mathcal{HOTL}$ is oriented to deal with a language with a strict alternation between inputs and outputs, we decided to consider a timed extension of finite state machines in order to reuse, as much as possible, the definition of the predicates and rules. Intuitively, transitions in finite state machines indicate that if the machine is in a state s and receives an input i then it will produce and output o and it will change its state to s'. An appropriate notation for such a transition could be $s \xrightarrow{i/o} s'$. If we consider a timed extension of finite state machines, transitions as $s \xrightarrow{i/o}_d s'$ indicate that the time between receiving the input i and returning the output o is given by d, where d belongs to a certain time domain. Usually, d is considered to be a *simple* value (e.g. a non-negative real

number). However, we would like to use a more sophisticated time domain. Our first choice was to consider a *stochastic* version of finite state machines [18] where a transition $s \xrightarrow{i/o}_{\xi} s'$ indicates that the time elapsed between i and o is equal to t with probability $F_\xi(t)$, where F_ξ is the probability distribution function associated with the random variable ξ. However, this choice strongly complicated the definition of some rules and the notion of observation. In fact, the added complication was such that it would deviate the attention from the main goal of the paper: Introduce time in $\mathcal{HOTL}$. Thus, we decided to choose a simpler approach but richer than singles values: Time intervals. Thus, a transition $s \xrightarrow{i/o}_{[d_1,d_2]} s'$ indicates that the time elapsed from the moment i is offered until the moment o is produced is at most d_2 time units and at least d_1 time units. Actually, time intervals represent a simplification of random variables where the different *weight* given to each possible time value is not quantified.

Once the language and a notion of timed conformance were fixed, we had to work on how to *adapt* $\mathcal{HOTL}$ to the new setting. Initially, we thought that this task would be straightforward, consisting in modifying some of the rules so that time values were appropriately added and dealt with. However, once we started to work in the proposal, we realized that to develop the new framework would be much more involved than a simple rewriting of the non-timed framework. First, we had to adapt the notion of *observation* to take into account not only assumptions about the possible time interval governing transitions but also to record the observed time values: Since we consider time intervals, different observations of the very same transition can produce different time values. Next, we had to modify the existing rules. The addition of time complicated the rules linked to the *accounting* of observations. Finally, we introduced new hypotheses to express specific temporal constraints.

This paper represents an extension of [16] where we presented a preliminary timed extension of $\mathcal{HOTL}$. In particular, [16] concentrated on adapting existing $\mathcal{HOTL}$ hypotheses and rules to cope with a timed model. However, new hypotheses and rules to deal with specific time issues, what strongly complicates the theoretical development, were not included in [16].

The rest of the paper is organized as follows. In Section 2 we introduce our extension of finite state machines to model timed systems and define two implementation relations. In Section 3, we define the basics of the new logical framework. In Section 4, we introduce new hypotheses and provide a correctness result. Finally, in Section 5 we present our conclusions and some directions for further research.

2 A Timed extension of FSMs

In this section we present our timed extension of the classical finite state machine model. The main difference with respect to usual FSMs consists in the addition of *time* to indicate the lapse between offering an input and receiving an output. As we already indicated in the introduction, time intervals will be used to express time constraints associated with the performance of actions. First we need to introduce notation, related to time intervals and multisets, that we will use during the rest of the paper.

Definition 1. We say that $\hat{a} = [a_1, a_2]$ is a *time interval* if $a_1 \in \mathbb{R}_+$, $a_2 \in \mathbb{R}_+ \cup \{\infty\}$, and $a_1 \leq a_2$. We assume that for all $r \in \mathbb{R}_+$ we have $r < \infty$ and $r + \infty = \infty$. We

consider that $\mathcal{I}_{\mathbb{R}^+}$ denotes the set of time intervals. Let $\hat{a} = [a_1, a_2]$ and $\hat{b} = [b_1, b_2]$ be time intervals. We write $\hat{a} \subseteq \hat{b}$ if we have both $b_1 \leq a_1$ and $a_2 \leq b_2$. In addition, $\hat{a} + \hat{b}$ denotes the interval $[a_1 + b_1, a_2 + b_2]$ and $\pi_i(\hat{a})$, for $i \in \{1, 2\}$, denotes the value a_i.

We will use the delimiters $\{\!\!\{$ and $\}\!\!\}$ to denote multisets. We denote by $\wp(\mathbb{R}^+)$ the multisets of elements belonging to $\mathbb{R}^+$. Given $H \in \wp(\mathbb{R}^+)$, for all $r \in \mathbb{R}^+$ we have that $H(r)$ denotes the *multiplicity* of r in H. Given two multisets $H_1, H_2 \in \wp(\mathbb{R}^+)$, $H_1 \uplus H_2$ denotes the *union* of H_1 and H_2, and it is formally defined as $(H_1 \uplus H_2)(r) = H_1(r) + H_2(r)$ for all $r \in \mathbb{R}^+$. □

Let us note that in the case of $[t_1, \infty]$ and $[0, \infty]$ we are abusing the notation since these intervals are in fact half-closed intervals, that is, they represent the intervals $[t_1, \infty)$ and $[0, \infty)$, respectively.

Definition 2. A *Timed Finite State Machine*, in the following `TFSM`, is a tuple $M = (\mathcal{S}, \texttt{inputs}, \texttt{outputs}, \mathcal{I}, \mathcal{T})$ where $\mathcal{S}$ is a finite set of states, `inputs` is the set of input actions, `outputs` is the set of output actions, $\mathcal{T}$ is the set of transitions, and $\mathcal{I}$ is the set of initial states.

A transition belonging to $\mathcal{T}$ is a tuple $(s, s', i, o, \hat{d})$ where $s, s' \in \mathcal{S}$ are the initial and final states of the transition, respectively, $i \in \texttt{inputs}$ and $o \in \texttt{outputs}$ are the input and output actions, respectively, and $\hat{d} \in \mathcal{I}_{\mathbb{R}^+}$ denotes the possible time values the transition needs to be completed. We usually denote transitions by $s \xrightarrow{i/o}_{\hat{d}} s'$.

We say that $(s, s', (i_1/o_1, \ldots, i_r/o_r), \hat{d})$ is a *timed trace*, or simply *trace*, of M if there exist $(s, s_1, i_1, o_1, \hat{d_1}), \ldots, (s_{r-1}, s', i_r, o_r, \hat{d_r}) \in \mathcal{T}$, such that $\hat{d} = \sum \hat{d_i}$.

We say that $((i_1/o_1, \ldots, i_r/o_r), \hat{d})$ is a *timed evolution* of M if there exists $s_{in} \in \mathcal{I}$ such that $(s_{in}, s', (i_1/o_1, \ldots, i_r/o_r), \hat{d})$ is a trace of M. We denote by $\texttt{TEvol}(M)$ the set of timed evolutions of M. In addition, we say that $(i_1/o_1, \ldots, i_r/o_r)$ is a *non-timed evolution*, or simply *evolution*, of M and we denote by $\texttt{NTEvol}(M)$ the set of non-timed evolutions of M.

Let us consider $s, s' \in \mathcal{S}$. We say that the state s' is *reachable* from s, denoted by $\texttt{isReachable}(M, s, s')$, if either there exist $u, i, o, \hat{d}$ such that $s \xrightarrow{i/o}_{\hat{d}} u \in \mathcal{T}$ and $\texttt{isReachable}(M, u, s')$ holds, or $s = s'$. The set $\texttt{reachableStates}(M, s)$ contains all $s' \in \mathcal{S}$ such that $\texttt{isReachable}(M, s, s')$.

Let $s \in \mathcal{S}$ and $i \in \texttt{inputs}$. The set $\texttt{outs}(M, s, i)$ contains the outputs that can be produced in s in response to i, that is, $\{o \mid \exists\, s', \hat{d} : s \xrightarrow{i/o}_{\hat{d}} s' \in \mathcal{T}\}$.

Finally, we say that $s \in \mathcal{S}$ is *deterministic*, denoted by $\texttt{isDet}(M, s)$, if there do not exist $s \xrightarrow{i/o'}_{\hat{d}} s', s \xrightarrow{i/o''}_{\hat{d'}} s'' \in \mathcal{T}$ such that $o' \neq o''$ or $s' \neq s''$. □

Intuitively, a transition $(s, s', i, o, \hat{d})$ indicates that if the machine is in state s and receives the input i then, after a time belonging to the interval $\hat{d}$, the machine emits the output o and moves to s'. Traces are sequences of transitions. The time associated with the trace is computed by adding the intervals associated with each of the transitions conforming the trace. We allow machines to be non-deterministic. We assume that both implementations and specifications can be represented by appropriate `TFSM`s and we consider that IUTs are *input-enabled*. During the rest of the paper we will assume that a generic specification is given by $spec = (\mathcal{S}_{spec}, \texttt{inputs}_{spec}, \texttt{outputs}_{spec}, \mathcal{I}_{spec}, \mathcal{T}_{spec})$.

Next we introduce our first timed implementation relation. In addition to the *untimed* conformance of the implementation, we require some time conditions to hold. Intuitively, an IUT is conforming if it does not *invent* behaviors for those traces that can be executed by the specification and time values are as expected.

Definition 3. Let I and S be TFSMs. We say that I *conforms to* S, denoted by I $\mathtt{conf}$ S, if for all $\rho_1 = (i_1/o_1, \ldots, i_{n-1}/o_{n-1}, i_n/o_n) \in \mathtt{NTEvol}(S)$, with $n \geq 1$, we have $\rho_2 = (i_1/o_1, \ldots, i_{n-1}/o_{n-1}, i_n/o'_n) \in \mathtt{NTEvol}(I)$ implies $\rho_2 \in \mathtt{NTEvol}(S)$. We say that I *conforms in time* to S, denoted by I $\mathtt{conf}_{int}$ S, if I $\mathtt{conf}$ S and for all $e \in \mathtt{NTEvol}(I) \cap \mathtt{NTEvol}(S)$ and $\hat{d} \in \mathcal{I}_{\mathbb{R}_+}$, we have that $(e, \hat{d}) \in \mathtt{TEvol}(I)$ implies $(e, \hat{d}) \in \mathtt{TEvol}(S)$. □

Even though this is a very reasonable notion of conformance, a black-box testing framework disallows us to check whether the corresponding time intervals coincide. The problem is that we cannot compare in a direct way timed requirements of the *real* implementation with those established in the specification. In fact, we can *see* the time interval defining a given transition in the specification, but we cannot do the same with the corresponding transition of the implementation, since we do not have access to it. Thus, we have to give a more *realistic* implementation relation based on a finite set of observations. We will present an implementation relation being less *accurate* but *checkable*. Specifically, we will check that the observed time values (from the implementation) belong to the time interval indicated in the specification.

Definition 4. Let I be a TFSM. We say that $((i_1/o_1, \ldots, i_n/o_n), t)$ is an *observed timed execution* of I, or simply *timed execution*, if the observation of I shows that the sequence $(i_1/o_1, \ldots, i_n/o_n)$ is performed in time t.

Let $\Phi = \{e_1, \ldots, e_m\}$ be a set of input/output sequences and let us consider a multiset of timed executions $H = \{\!\{(e'_1, t_1), \ldots, (e'_n, t_n)\}\!\}$. We say that $\mathtt{Sampling}_{(H,\Phi)} : \Phi \longrightarrow \wp(\mathbb{R}^+)$ is a *sampling application* of H for Φ if for all $e \in \Phi$ we have $\mathtt{Sampling}_{(H,\Phi)}(e) = \{\!\{t \mid (e,t) \in H\}\!\}$. □

Timed executions are input/output sequences together with the time that it took to perform the sequence. In a certain sense, timed executions can be seen as *instances* of the evolutions that the implementation can perform. Regarding the definition of sampling applications, we just associate with each evolution the multiset of observed execution time values.

Definition 5. Let I and S be two TFSMs, H be a multiset of timed executions of I, and $\Phi = \{e \mid \exists\, t : (e,t) \in H\} \cap \mathtt{NTEvol}(S)$. For all non-timed evolution $e \in \Phi$ we define the *sample interval* of e in H as

$$\widehat{S}_{(H,e)} = [\mathtt{min}(\mathtt{Sampling}_{(H,\Phi)}(e)), \mathtt{max}(\mathtt{Sampling}_{(H,\Phi)}(e))]$$

We say that I $H-$*timely conforms* to S, denoted by I $\mathtt{conf}^H_{int}$ S, if I $\mathtt{conf}$ S and for all $e \in \Phi$ and $\hat{d} \in \mathcal{I}_{\mathbb{R}_+}$ we have that $(e, \hat{d}) \in \mathtt{TEvol}(S)$ implies $\widehat{S}_{(H,e)} \subseteq \hat{d}$. □

3 $\mathcal{THOTL}$: An extension of $\mathcal{HOTL}$ with time

In this section we present the new formalism $\mathcal{THOTL}$. This framework represents an extension and adaption of $\mathcal{HOTL}$ to cope with systems where time plays a fundamental role. While some of the rules remain the same (the rules dealing with the

internal *functional* structure of the implementation), $\mathcal{THOTL}$ constitutes a complete *new* formalism. Next, we briefly describe the main contributions with respect to $\mathcal{HOTL}$. First, we have to redefine most components of the logic to consider temporal aspects. Observations will include the time values that the IUT takes to emit an output since an input is received. Additionally, the model will be extended to take into account the different time values appearing in the observations for each input/output outgoing from a state. We will add new hypotheses to allow the tester to represent assumptions about temporal behaviors concerning both specific states and the whole IUT. Finally, we will modify the deduction rules as well as include new rules to add the new hypotheses to the models. During the rest of the paper `Obs` denotes the multiset of observations collected during the preliminary interaction with the IUT while `Hyp` denotes the set of *hypotheses* the tester has assumed. In this latter set, we will not consider the hypotheses that are implicitly introduced by means of observations.

3.1 Temporal Observations

In our framework we consider that temporal observations follow the format $ob = (a_1, i_1/o_1/t_1, a_2, \ldots, a_n, i_n/o_n/t_n, a_{n+1}) \in \texttt{Obs}$. This expression denotes that when the sequence of inputs $i_1, \ldots, i_n$ was proposed from an initial state of the implementation, the sequence $o_1, \ldots, o_n$ was obtained as response in $t_1, \ldots, t_n$ time units, respectively. Moreover, for all $1 \leq j \leq n+1$, a_j represents a set of *special attributes* concerning the state of the implementation reached after performing $i_1/o_1, \ldots, i_{j-1}/o_{j-1}$ in *this* observation. Attributes denote our assumptions about this state. The attributes belonging to the set a_j are of the form $\texttt{imp}(q)$ or `det`, where $\texttt{imp}(q)$ denotes that the implementation state reached after $i_1/o_1, \ldots, i_{j-1}/o_{j-1}$ is associated to a *state identifier name* q and `det` denotes that the implementation state reached after $i_1/o_1, \ldots, i_{j-1}/o_{j-1}$ in this observation is deterministic. State identifier names are used to match equal states: If two states are associated with the same state identifier name then they represent the *same* state of the implementation. The set of all state identifier names will be denoted by $\mathcal{Q}$. In addition, attributes belonging to a_{n+1} can also be of the form $\texttt{spec}(s)$ denoting that the implementation state reached after $i_1/o_1, \ldots, i_n/o_n$ is such that the subgraph that can be reached from it is isomorphic to the subgraph that can be reached from the state s of the specification. Thus, the behavior of the implementation from that point on is known and there is no need to check its correctness. In addition to the previous attributes, already defined in $\mathcal{HOTL}$, temporal observations may include a new type of attribute that represents our assumption about the time interval in which a transition can be performed. For all $1 < j \leq n$, the attributes in the set a_j can be also of the form $\texttt{int}(\hat{d})$, with $\hat{d} \in \mathcal{I}_{\mathbb{R}_+}$. Such an attribute denotes that the time that the implementation takes from the state reached after performing $i_1/o_1, \ldots, i_{j-1}/o_{j-1}$, to emit the output o_j after it received the input i_j belongs to the interval $\hat{d}$. We assume that this attribute cannot appear in the set a_1, since the implementation is in an initial state, and no actions have taken place yet.

3.2 Model Predicates

Temporal observations will allow to create *model predicates* that denote our knowledge about the implementation. A model predicate is denoted by $\texttt{model}\,(m)$, where

$m = (\mathcal{S}, \mathcal{T}, \mathcal{I}, \mathcal{A}, \mathcal{E}, \mathcal{D}, \mathcal{O})$. As in $\mathcal{HOTL}$, $\mathcal{S}$ is the set of states appearing in the model, $\mathcal{T}$ is the set of transitions appearing in the graph of the model, $\mathcal{E}$ is the set of equalities relating states belonging to $\mathcal{S}$, $\mathcal{D}$ is the set of deterministic states, and $\mathcal{O}$ is the set of observations we have used so far for the construction of this model. In addition, $\mathcal{I}$ is the set of states that are initial in the model. Two additional symbols may appear in $\mathcal{I}$. The first special symbol, α, denotes that any state in $\mathcal{S}$ could eventually be initial. The second symbol, β, denotes that not only states belonging to $\mathcal{S}$ but also other states not explicitly represented in $\mathcal{S}$ could be initial. Finally, $\mathcal{A}$ is the set of *accounting registers* of the model. An accounting register is a tuple $(s, i, outs, f, \delta, n)$ denoting that in state $s \in \mathcal{S}$ the input i has been offered n times and we have obtained the outputs belonging to the set $outs$. Besides, for each transition departing from state s and labelled with input i, the function $f : \mathcal{T} \longrightarrow \mathbb{N}$ returns the number of times the transition has been observed. If, due to the hypotheses that we consider, we infer that the number of times we observed an input is high enough to believe that the implementation cannot react to that input either with an output that was not produced before or leading to a state that was not taken before, then the value n is set to $\top$. In this case, we say that the behavior at state s for the input i is *closed*. The only change we need to introduce with respect to $\mathcal{HOTL}$ is that each register includes a function $\delta : \mathcal{T} \longrightarrow \mathcal{I}_{\mathbb{R}_+} \times \wp(\mathbb{R}^+)$ that computes for each transition departing from state s with input i and output $o \in outs$ the time interval, according to our knowledge up to now, in which the transition could be performed and the set of time values the implementation took to perform the transition. If no assumptions about the interval are made by means of temporal observations, it will be set to $[0, \infty]$. In the case of transitions not fulfilling the conditions about s, i, and $outs$, an arbitrary value is returned. Let us remark that as new hypotheses and temporal observations are included in the model, the intervals will be reduced.

3.3 Basic $\mathcal{THOTL}$ Rules

First, we will include a new rule denoting how a model can be constructed from a temporal observation.

$$(tobser)\frac{ob = (a_1, i_1/o_1/t_1, a_2, \ldots, a_n, i_n/o_n/t_n, a_{n+1}) \in \texttt{Obs} \ \wedge \ s_1, \ldots, s_{n+1} \text{ fresh states}}{\texttt{model}\left(\begin{array}{c} \{s_1, \ldots, s_{n+1}\} \cup \mathcal{S}', \\ \{s_1 \xrightarrow{i_1/o_1} s_2, \ldots, s_n \xrightarrow{i_n/o_n} s_{n+1}\} \cup \mathcal{T}', \{s_1, \beta\}, \\ \{(s_j, i_j, \{o_j\}, f_{s_j}\delta_{s_j}, 1) \mid 1 \leq j \leq n\} \cup \mathcal{A}', \\ \{s_j \ \texttt{is} \ q_j | 1 \leq j \leq n+1 \ \wedge \ \texttt{imp}(q_j) \in a_j\}, \\ \{s_j \mid 1 \leq j \leq n+1 \ \wedge \ \texttt{det} \in a_j\} \cup \mathcal{D}', \{ob\} \end{array}\right)}$$

where $f_{s_j}(tr)$ is equal to 1 if $tr = s_j \xrightarrow{i_j/o_j} s_{j+1}$ and equal to 0 otherwise; and

$$\delta_{s_j}(tr) = \begin{cases} (\hat{d}, \{\!|t_j|\!\}) & \texttt{if } tr = s_j \xrightarrow{i_j/o_j} s_{j+1} \ \wedge \ \texttt{int}(\hat{d}) \in a_{j+1} \\ ([0, \infty], \{\!|t_j|\!\}) & \texttt{if } tr = s_j \xrightarrow{i_j/o_j} s_{j+1} \ \wedge \ \texttt{int}(\hat{d}) \notin a_{j+1} \\ ([0, \infty], \emptyset) & \text{otherwise} \end{cases}$$

The sets of states, transitions, accounting registers, and deterministic states will be extended with some extra elements, taken from the specification, if the tester

assumes that the last state of the observation is isomorphic to a state of the specification (i.e., $\texttt{spec}(s)$, for some $s \in \mathcal{S}_{spec}$). The new states and transitions $\mathcal{S}'$ and $\mathcal{T}'$, respectively, will copy the structure existing among the states that can be reached from s in the specification. The new accounting, $\mathcal{A}'$, will denote that the knowledge concerning the new states is *closed* for all inputs, that is, the only transitions departing from these states are those we copy from the specification and no other transitions will be added in the future. Additionally, accounting registers will reflect the time intervals associated to the transitions that are copied from the specification. Finally, those model states that correspond to deterministic specification states will be included in the set $\mathcal{D}'$ of deterministic states of the model. The formal definition of $\mathcal{S}'$, $\mathcal{T}'$, $\mathcal{A}'$, and $\mathcal{D}'$ follows. If there does not exist s' such that $\texttt{spec}(s') \in a_{n+1}$ then $(\mathcal{S}', \mathcal{T}', \mathcal{A}', \mathcal{D}') = (\emptyset, \emptyset, \emptyset, \emptyset)$. Otherwise, that is, if $\texttt{spec}(s) \in a_{n+1}$ for some $s \in \mathcal{S}_{spec}$, let us consider the following set of states:

$$U = \{u_j \mid u_j \text{ is a fresh state } \wedge\ 1 \leq j < |\texttt{reachableStates}(spec, s)|\}$$

and a bijective function $g : \texttt{reachableStates}(spec, s) \longrightarrow U \cup \{s_{n+1}\}$ such that $g(s) = s_{n+1}$. Then, $(\mathcal{S}', \mathcal{T}', \mathcal{A}', \mathcal{D}')$ is equal to

$$\left(\begin{array}{c} U, \\ \{g(s') \xrightarrow{i/o} g(s'') \mid \exists\ \hat{d} : s' \xrightarrow[\hat{d}]{i/o} s'' \in \mathcal{T}_{spec} \ \wedge\ \texttt{isReachable}(spec, s, s')\}, \\ \left\{ \left(\begin{array}{c} u, i, \\ \texttt{outs}(spec, g^{-1}(u), i), \\ f^i_u, \delta^i_u, \top \end{array} \right) \middle| \begin{array}{l} u \in U \cup \{s_{n+1}\} \ \wedge\ i \in \texttt{inputs}_{spec} \wedge \\ \exists u' \in U, o \in \texttt{outputs}_{spec} :\ u \xrightarrow{i/o} u' \in \mathcal{T}' \end{array} \right\}, \\ \{g(s') | \texttt{isReachable}(spec, s, s') \ \wedge\ \texttt{isDet}(spec, s')\} \end{array} \right)$$

where $f^i_u(tr)$ is equal to 1 if there exist o', u' such that $tr = u \xrightarrow{i/o'} u' \in \mathcal{T}'$ and equal to 0 otherwise; and

$$\delta^i_u(tr) = \begin{cases} (\hat{d}, \{\!|a| \ a \in \hat{d}|\!\}) & \texttt{if } \exists\ o', u' : tr = u \xrightarrow{i/o'} u' \in \mathcal{T}' \wedge \\ & \quad g^{-1}(u) \xrightarrow[\hat{d}]{i/o'} g^{-1}(u') \in \mathcal{T}_{spec} \\ ([0, \infty], \emptyset) & \text{otherwise} \end{cases}$$

We can join different models, created from different observations, into a single model by means of the *(fusion)* rule. The components of the new model are the union of the components of each model.

$$(fusion)\frac{\begin{array}{c} \texttt{model}\,(\mathcal{S}_1, \mathcal{T}_1, \mathcal{I}_1, \mathcal{A}_1, \mathcal{E}_1, \mathcal{D}_1, \mathcal{O}_1) \ \wedge \\ \texttt{model}\,(\mathcal{S}_2, \mathcal{T}_2, \mathcal{I}_2, \mathcal{A}_2, \mathcal{E}_2, \mathcal{D}_2, \mathcal{O}_2) \ \wedge\ \mathcal{O}_1 \cap \mathcal{O}_2 = \emptyset \end{array}}{\texttt{model}\,\big(\mathcal{S}_1 \cup \mathcal{S}_2, \mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{I}_1 \cup \mathcal{I}_2, \mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{E}_1 \cup \mathcal{E}_2, \mathcal{D}_1 \cup \mathcal{D}_2, \mathcal{O}_1 \cup \mathcal{O}_2\big)}$$

The iterative application of this rule allows us to join different models created from different temporal observations into a single model.

At this point, the inclusion of those hypotheses that are not covered by observations will begin. During this new phase, we will usually need several models to represent all the `FSMs` that are compatible with a set of observations and hypotheses. Some of the rules use the `modelElim` function. If we discover that a state of

the model coincides with another one, we will eliminate one of the states and will allocate all of its constraints to the other one. The `modelElim` function modifies the components that define the model, in particular the accounting set. A similar function appeared in the original formulation of $\mathcal{HOTL}$. However, due to the inclusion of time issues, this function must be adapted to deal with the new considerations. The `modelElim` function is constructed in two steps. First, we define how to modify the *accounting*. The $\texttt{countElim}(\mathcal{A}, s_1, s_2)$ function shows how an accounting $\mathcal{A}$ is updated when a state s_2 is modified because it is equal to another state s_1. Basically, we move all the accounting information from s_2 to s_1. The new accounting set is constructed by joining two sets. The first set denotes the accounting for all states different from s_1 and s_2. A register $(s, i, outs, f, \delta, n) \in \mathcal{A}$, with $s \neq s_1$ and $s \neq s_2$, will change only if there exists a registered transition from s to s_2 with input i, that is, if $f(s \xrightarrow{i/o} s_2) > 0$ for some $o \in outs$. In this case, the information provided by f must denote the replacement of s_2 by s_1: We set $f(s \xrightarrow{i/o} s_2) = 0$ and we increase the value of $f(s \xrightarrow{i/o} s_1)$. Additionally, we need to update the temporal information provided by the δ function. Finally, all information concerning s_2 is removed. Let us note that this operation may lead to inconsistent models from a temporal point of view. It can happen that for the state s with input i the pairs $\delta(s \xrightarrow{i/o} s_1) = (\hat{d_1}, H_1)$ and $\delta(s \xrightarrow{i/o} s_2) = (\hat{d_2}, H_2)$ are incompatible, either because $\hat{d_1} \cap \hat{d_2} = \emptyset$ or because $H_1 \uplus H_2 \not\subseteq \hat{d_1} \cap \hat{d_2}$, where $\uplus$ denotes the union of multisets introduced in Definition 1.

Definition 6. Let $p = (\hat{d_1}, H_1)$, $q = (\hat{d_1}, H_1) \in \mathcal{I}_{\mathbb{R}_+} \times \wp(\mathbb{R}^+)$. We denote by $p + q$ the pair defined as

$$p + q = \begin{cases} (\hat{d_1} \cap \hat{d_2}, H_1 \uplus H_2) & \texttt{if } \hat{d_1} \cap \hat{d_2} \neq \emptyset \ \wedge \ H_1 \uplus H_2 \subseteq \hat{d_1} \cap \hat{d_2} \\ \texttt{error} & \text{otherwise} \end{cases}$$

Let $\mathcal{A}$ be a set of *accounting registers* and s_1, s_2 be states. Then, we have

$$\texttt{countElim}(\mathcal{A}, s_1, s_2) =$$

$$\left\{ (s, i, outs, f', \delta', n) \;\middle|\; \begin{array}{l} s \notin \{s_1, s_2\} \wedge (s, i, outs, f, n) \in \mathcal{A} \wedge \\ f'(s \xrightarrow{i/o} s') = \begin{cases} f(s \xrightarrow{i/o} s') & \texttt{if } s' \neq s_1, s_2 \\ f(s \xrightarrow{i/o} s_1) + f(s \xrightarrow{i/o} s_2) & \texttt{if } s' = s_1 \\ 0 & \texttt{if } s' = s_2 \end{cases} \\ \wedge \\ \delta'(s \xrightarrow{i/o} s') = \begin{cases} \delta(s \xrightarrow{i/o} s') & \texttt{if } s' \neq s_1, s_2 \\ \delta(s \xrightarrow{i/o} s_1) + \delta(s \xrightarrow{i/o} s_2) & \texttt{if } s' = s_1 \\ ([0, \infty], \emptyset) & \texttt{if } s' = s_2 \end{cases} \end{array} \right\}$$

$$\bigcup$$

$$\left\{ \begin{pmatrix} s_1, i, \\ outs_1 \cup outs_2, \\ f', \delta', n \end{pmatrix} \;\middle|\; \begin{array}{l} \exists\, p, q, g, h, \delta_1, \delta_2 : \\ \quad ((s_1, i, outs_1, g, \delta_1, p) \in \mathcal{A} \ \vee \ (s_2, i, outs_2, h, \delta_2, q) \in \mathcal{A}) \wedge \\ n = \sum\{m \mid (s, i, outs, f, \delta, m) \in \mathcal{A}, s \in \{s_1, s_2\}\} \wedge \\ f'(tr) = \sum\{f(tr) | (s, i, outs, f, \delta, m) \in \mathcal{A}, s \in \{s_1, s_2\}\} \wedge \\ \delta'(tr) = \sum\{\delta(tr) | (s, i, outs, f, \delta, m) \in \mathcal{A}, s \in \{s_1, s_2\}\} \end{array} \right\}$$

We assume that for all $n \in \mathbb{N}$ we have $n + \top = \top$. □

The previous function is auxiliary to define how to eliminate a state s when we discover that this state is equal to another state s'. In the following we will denote by $[x/y]$ the renaming of any occurrence of x by y. As before, we can reach an *inconsistent* result. For example, if the behavior of the state s with input i is *closed*, that is, $(s, i, outs, f, \top) \in \mathcal{A}$, and s' has an outgoing transition labelled with i/o, being $o \notin outs$, then the model resulting by joining s and s' would be inconsistent because it would not preserve the closed behavior of s. In this case, an empty set of models will be returned (see case (a) of the following definition). The same happens if there exists $(s, i, outs, f, \delta, m) \in \texttt{countElim}(\mathcal{A}, s_1, s_2)$ such that for some $tr \in \mathcal{T}[s_2/s_1]$ we have $\delta(tr) = \texttt{error}$ (see case (b)). Otherwise, the new model is obtained by *substituting* all occurrences of the state to be eliminated by the state that will stay (see case (c)). In the next definition we use the following property: For all index $j \in \{1, 2\}$, the expression $3 - j$ always denotes the other number of the set.

Definition 7. Let $m = (\mathcal{S}, \mathcal{T}, \mathcal{I}, \mathcal{A}, \mathcal{E}, \mathcal{D}, \mathcal{O})$ be a model and $s_1, s_2 \in \mathcal{S}$. We define the predicate $\texttt{modelElim}(m, s_1, s_2)$ as $\texttt{models}\,(\mathcal{M})$, where $\mathcal{M}$ is constructed as follows:

(a) If there exist $i, outs, f, o, s$, and $j \in \{1, 2\}$ such that $s_{3-j} \xrightarrow{i/o} s \in \mathcal{T}$, $(s_j, i, outs, f, \top) \in \mathcal{A}$, and $o \notin outs$, then $\mathcal{M} = \emptyset$.

(b) If there exists $(s, i, outs, f, \delta, m) \in \texttt{countElim}(\mathcal{A}, s_1, s_2)$ such that for some $tr \in \mathcal{T}[s_2/s_1]$ we have $\delta(tr) = \texttt{error}$, then $\mathcal{M} = \emptyset$.

(c) Otherwise, $\mathcal{M} = \left\{ \begin{pmatrix} \mathcal{S}\backslash\{s_2\}, \mathcal{T}[s_2/s_1], \mathcal{I}[s_2/s_1], \\ \texttt{countElim}(\mathcal{A}, s_1, s_2), \mathcal{E}[s_2/s_1], \\ \mathcal{D}[s_2/s_1], \mathcal{O} \end{pmatrix} \right\}$

The previous function can be generalized to operate over *sets* of states as follows. Let $S \subseteq \mathcal{S}$ be a set of states. We have

$$\texttt{modelElim}(m, s, S) = \begin{cases} \{m\} & \texttt{if } S = \emptyset \\ \bigcup_{j=1}^{k} \texttt{modelElim}(m'_j, s, \{s_2, \ldots, s_n\}) & \texttt{if } S = \{s_1, \ldots, s_n\} \end{cases}$$

where $\{m'_1, \ldots, m'_k\} = \texttt{modelElim}(m, s, s_1)$. □

The rest of the rules belonging to $\mathcal{HOTL}$ do not vary in their formulation. It is only necessary to consider that those rules using `countElim` have to consider the temporal version of this function (that is, temporal issues are *transparent* in the formulation of those rules).

4 New $\mathcal{THOTL}$ hypotheses

In this section we will extend the repertory of hypotheses to include assumptions about temporal constraints of the transitions. The tester may assume that "the IUT cannot spend more (less) than t time units for producing the output o after

it receives the input i" or "the pair i/o never takes more (less) than t time units." This kind of hypotheses affects the whole IUT, so they will be included in the set `Hyp`. Besides, we need to define specific rules to apply the different hypotheses to our models and reflect how they are affected by their application.

Some of the new rules use the `updTime` function. Its role is to update the accounting set in the model to reflect the constrains established by the considered temporal hypotheses. The new accounting set is constructed by joining two sets. The first set denotes the accounting for all registers of states that either do not belong to S or have an input that does not belong to I. These registers will never change. The second set denotes the new registers. A register will change only if there exists a registered transition from $s \in S$ with input $i \in I$ such that $o \in O \cap outs$, that is, if $f(s \xrightarrow{i/o} s') > 0$ for some s'. In this case, the information provided by δ must denote the temporal constraint. One more time, we can reach an inconsistency if $\delta(s \xrightarrow{i/o} s') = (\hat{d}, H)$ and the temporal restriction imposed by the hypothesis, given by a time interval $\hat{d}'$, is incompatible due to either $\hat{d} \cap \hat{d}' = \emptyset$ or $H \not\subseteq \hat{d} \cap \hat{d}'$. In this case, the function δ will return `error`. Then, the resulting model would be inconsistent because it would not preserve the timed behavior for some transition. In this case, an empty set of models will be returned by the function.

First, we introduce a function to update the temporal functions of the accounting registers.

Definition 8. Let $\mathcal{A}$ be a set of *accounting registers*, $S \subseteq \mathcal{S}$, $I \subseteq \texttt{inputs}_{spec}$, $O \subseteq \texttt{outputs}_{spec}$, and $\hat{d}' \in \mathcal{I}_{\mathbb{R}_+}$. We define $\texttt{accTime}(\mathcal{A}, S, I, O, \hat{d}')$ as the set of accounting registers

$$\left\{(s', i', outs, f, \delta, n) \;\middle|\; (s' \notin S \vee i' \notin I \;\vee outs \cap O = \emptyset) \;\wedge\; (s', i', outs, f, \delta, n) \in \mathcal{A}\right\}$$

$$\bigcup$$

$$\left\{ \begin{pmatrix} s, i, outs, \\ f, \delta', n \end{pmatrix} \;\middle|\; \begin{array}{l} \exists\, f, outs, \delta, n : \\ \quad (s, i, outs, f, \delta, n) \in \mathcal{A} \;\wedge\; s \in S \wedge i \in I \;\wedge\; outs \cap O \neq \emptyset \;\wedge \\ \quad \delta'(tr) = \begin{cases} \delta(s' \xrightarrow{i'/o'} s'') & \texttt{if } s' \notin S \vee i' \notin I \;\vee o' \notin outs \cap O \\ (\hat{d} \cap \hat{d}', H) & \texttt{if } \delta(s \xrightarrow{i/o} s') = (\hat{d}, H) \wedge \\ & \quad \hat{d} \cap \hat{d}' \neq \emptyset \;\wedge\; H \subseteq \hat{d} \cap \hat{d}' \\ \texttt{error} & \text{otherwise} \end{cases} \end{array} \right\}$$

□

Let $m = (\mathcal{S}, \mathcal{T}, \mathcal{I}, \mathcal{A}, \mathcal{E}, \mathcal{D}, \mathcal{O})$ be a model. We define $\texttt{updTime}(m, S, I, O, \hat{d}')$ as the model

(a) $\emptyset$, if there exist $(s, i, outs, f, \delta, n) \in \texttt{accTime}(\mathcal{A}, S, I, O, \hat{d}')$ and $tr \in \mathcal{T}$ such that $\delta(tr) = \texttt{error}$;
(b) $(\mathcal{S}, \mathcal{T}, \mathcal{I}, \texttt{accTime}(\mathcal{A}, S, I, O, \hat{d}'), \mathcal{E}, \mathcal{D}, \mathcal{O})$, otherwise.

Next we introduce new hypotheses. The first two ones allow us to assume that the implementation never takes more (less) than t time units to produce the output o since it receives the input i.

$$(maxTime)\frac{\texttt{model}\,(\mathcal{S}, \mathcal{T}, \mathcal{I}, \mathcal{A}, \mathcal{E}, \mathcal{D}, \texttt{Obs}) \;\wedge\; \texttt{maxTime}(i, o, t) \in \texttt{Hyp}}{\texttt{models}\,(\texttt{updTime}(m, \mathcal{S}, \{i\}, \{o\}, [0, t]))}$$

$$(minTime)\frac{\texttt{model}\,(\mathcal{S},\mathcal{T},\mathcal{I},\mathcal{A},\mathcal{E},\mathcal{D},\texttt{Obs})\ \wedge\ \texttt{minTime}(i,o,t)\in\texttt{Hyp}}{\texttt{models}\,(\texttt{updTime}(m,\mathcal{S},\{i\},\{o\},[t,\infty]))}$$

The tester can assume that the performance of the pair i/o always consumes exactly t time units.

$$(oclockTime)\frac{\texttt{model}\,(\mathcal{S},\mathcal{T},\mathcal{I},\mathcal{A},\mathcal{E},\mathcal{D},\texttt{Obs})\ \wedge\ \texttt{oclockTime}(i,o,t)\in\texttt{Hyp}}{\texttt{models}\,(\texttt{updTime}(m,\mathcal{S},\{i\},\{o\},[t,t]))}$$

The logic $\mathcal{THOTL}$ allows to consider other temporal hypotheses about the IUT. For example, the predicate $\texttt{alwMax}(t)$ (resp. $\texttt{alwMin}(t)$) assumes that all pair i/o is performed in at most (resp. at least) t time units.

$$(alwaysMax)\frac{\texttt{model}\,(\mathcal{S},\mathcal{T},\mathcal{I},\mathcal{A},\mathcal{E},\mathcal{D},\texttt{Obs})\ \wedge\ \texttt{alwMax}(t)\in\texttt{Hyp}}{\texttt{models}\,(\texttt{updTime}(m,\mathcal{S},\mathcal{I},\mathcal{O},[0,t]))}$$

$$(alwaysMin)\frac{\texttt{model}\,(\mathcal{S},\mathcal{T},\mathcal{I},\mathcal{A},\mathcal{E},\mathcal{D},\texttt{Obs})\ \wedge\ \texttt{alwMin}(t)\in\texttt{Hyp}}{\texttt{models}\,(\texttt{updTime}(m,\mathcal{S},\mathcal{I},\mathcal{O},[t,\infty]))}$$

Another plausible assumption is that all actions that can be performed from a state s spend at least/most t time units.

$$(allOutMax)\frac{\texttt{model}\,(\mathcal{S},\mathcal{T},\mathcal{I},\mathcal{A},\mathcal{E},\mathcal{D},\texttt{Obs})\ \wedge\ \texttt{allOutMax}(s,t)\in\texttt{Hyp}}{\texttt{models}\,(\texttt{updTime}(m,\{s\},\mathcal{I},\mathcal{O},[0,t]))}$$

$$(allOutMin)\frac{\texttt{model}\,(\mathcal{S},\mathcal{T},\mathcal{I},\mathcal{A},\mathcal{E},\mathcal{D},\texttt{Obs})\ \wedge\ \texttt{allOutMin}(s,t)\in\texttt{Hyp}}{\texttt{models}\,(\texttt{updTime}(m,\{s\},\mathcal{I},\mathcal{O},[t,\infty]))}$$

The $\texttt{allTimes}(n)$ hypothesis allows to assume that if an input/output pair is produced n times at a given state then all the time values that the implementation may take at this state to perform this pair will belong to the interval delimited by the minimum and maximum observed time values.

$$(allTimes)\frac{\texttt{model}\,(\mathcal{S},\mathcal{T},\mathcal{I},\mathcal{A},\mathcal{E},\mathcal{D},\texttt{Obs})\ \wedge\ n\in\mathbb{N}\ \wedge\ \texttt{allTimes}(n)\in\texttt{Hyp}}{\texttt{models}\,(\{(\mathcal{S},\mathcal{T},\mathcal{I},\mathcal{A}',\mathcal{E},\mathcal{S},\texttt{Obs})\})}$$

where

$$\mathcal{A}' = \{(s,i,outs,f,\delta,n')|(s,i,outs,f,\delta,n')\in\mathcal{A}\ \wedge\ n'<n\}$$

$$\bigcup$$

$$\left\{(s,i,outs,f,\delta,n')\ \middle|\ \begin{array}{l}(s,i,outs,f,\delta,n')\in\mathcal{A}\ \wedge\ n'\geq n\ \wedge\\ \forall\ o\in outs:\sum\{f(tr)|tr=s\xrightarrow{i/o}s'\}<n\end{array}\right\}$$

$$\bigcup$$

$$\left\{(s,i,outs,f,\delta_s^i,n')\ \middle|\ \begin{array}{l}(s,i,outs,f,\delta,n')\in\mathcal{A}\ \wedge\ n'\geq n\ \wedge\\ \exists\ o\in outs:\sum\{f(tr)|tr=s\xrightarrow{i/o}s'\}\geq n\end{array}\right\}$$

and

$$\delta_s^i(s_1 \xrightarrow{i'/o'} s_2) = \begin{cases} \delta(s_1 \xrightarrow{i'/o'} s_2) & \texttt{if } s_1 \neq s \vee i' \neq i \vee \\ & \sum\{f(tr) | tr = s \xrightarrow{i/o'} s'\} < n) \\ (\hat{d}, \pi_2(\delta(s_1 \xrightarrow{i'/o'} s_2))) & \text{otherwise} \end{cases}$$

$$\hat{d} = \begin{bmatrix} \texttt{min}\{t | \exists\ s_1 \xrightarrow{i'/o'} s' : t \in \pi_2(\delta(s_1 \xrightarrow{i'/o'} s'))\}, \\ \texttt{max}\{t | \exists\ s_1 \xrightarrow{i'/o'} s' : t \in \pi_2(\delta(s_1 \xrightarrow{i'/o'} s'))\} \end{bmatrix}$$

The new accounting set $\mathcal{A}'$ is the union of three set of registers. The first set collects those registers that do not change because they correspond to states whose outgoing transitions have been observed less than n times. The second set gathers the registers associated to states whose outgoing transitions have been observed at least n times, but all input/output pairs attached to them have been observed less than n times. The third set introduces the assumption for those registers $(s, i, outs, f, \delta, n')$ that present more than n observations of some pair i/o in the transitions outgoing from s. In this case, we must update the temporal function. For all transition tr leaving the state s and labelled by i/o we change the associated interval. This interval, $\hat{d}$, has as lower (resp. upper) bound the minimum (resp. maximum) time values observed in all the transitions outgoing from s and labelled by i/o.

We have shown some rules that may lead to inconsistent models. In some of these cases, an empty set of models is produced, that is, the inconsistent model is eliminated. Before granting conformance, we need to be sure that at least one model belonging to the set is consistent. $\mathcal{HOTL}$ already provides us with a rule that labels a model as *consistent*. Let us note that the inconsistences created by a rule can be detected by the subsequent applications of rules. Thus, a model is free of inconsistencies if for any other rule either it is not applicable to the model or the application does not modify the model (that is, it deduces the same model). Due to space limitations we do not include the details of this rule (the formal definition can be found in [21]).

Similar to $\mathcal{HOTL}$, in order to check whether a model conforms to the specification we have to take into account that only the conformance of consistent models will be considered. In addition, given a consistent model, we will check its conformance with respect to the specification by considering the *worst* instance of the model, that is, if this instance conforms to the specification then any other instance extracted from the model does so. This worst instance is constructed as follows: For each state s and input i such that the behavior of s for i is not closed *and* either s is not deterministic or no transition with input i exists in the model, a new *malicious* transition is created. The new transition is labelled with a special output *error* that does not belong to $\texttt{outputs}_{spec}$. This transition leads to a new state $\perp$ having no outgoing transitions. Since the specification cannot produce the output *error*, this worst instance will conform to the specification only if the unspecified parts of the model are not relevant for the correctness of the IUT it represents.

Definition 9. Let $m = (\mathcal{S}, \mathcal{T}, \mathcal{I}, \mathcal{A}, \mathcal{E}, \mathcal{D}, \texttt{Obs})$ be a model. We define the *worst temporal instance* of the model m with respect to the considered specification *spec*, denoted by $\texttt{worstTempCase}(m)$, as the TFSM

$$\left(\begin{array}{l} \mathcal{S} \cup \{\bot\}, \texttt{inputs}_{spec}, \texttt{outputs}_{spec} \cup \{error\}, \\ \left\{ s \xrightarrow{i/o}_{\hat{d}} s' \;\middle|\; \begin{array}{l} s \xrightarrow{i/o} s' \in \mathcal{T} \wedge \\ \exists\, outs, f, \delta, n : (s, i, outs, f, \delta, n) \in \mathcal{A} \wedge \\ o \in outs \;\wedge\; \pi_1(\delta(s \xrightarrow{i/o} s')) = \hat{d} \end{array} \right\} \\ \\ \bigcup \\ \\ \left\{ s \xrightarrow{i/error}_{[o,\infty]} \bot \;\middle|\; \begin{array}{l} s \in \mathcal{S} \;\wedge\; i \in \texttt{inputs}_{spec} \wedge \\ \nexists\, outs : (s, i, outs, f, \delta, \top) \in \mathcal{A} \wedge \\ (s \notin \mathcal{D} \vee \;\; \nexists\, s', o : s \xrightarrow{i/o} s') \end{array} \right\}, \mathcal{I} \end{array}\right)$$

□

Thus, the rule for indicating the correctness of a model is

$$(correct)\frac{\begin{array}{c} m = (\mathcal{S}, \mathcal{T}, \mathcal{I}, \mathcal{A}, \mathcal{E}, \mathcal{D}, \texttt{Obs}) \;\wedge\; \texttt{consistent}(m) \wedge \\ H = \texttt{reduce}(\texttt{Obs}) \;\wedge\; \texttt{worstTempCase}(m)\; \texttt{conf}^{H}_{int}\; spec \end{array}}{\texttt{models}\,(\{\texttt{correct}(m)\})}$$

where

$$\texttt{reduce}(\texttt{Obs}) = \{(i_1/o_1/t_1, \ldots, i_n/o_n/t_n) | (a_1, i_1/o_1/t_1, \ldots, a_n, i_n/o_n/t_n, a_{n+1}) \in \texttt{Obs}\}$$

Now we can consider the conformance of a set of models. A set conforms to the specification if all the elements do so and the set contains at least one element. Note that an empty set of models denotes that all the models were inconsistent.

$$(allCorrect)\frac{\texttt{models}\,(\mathcal{M}) \;\wedge\; \mathcal{M} \neq \emptyset \;\wedge\; \mathcal{M} = \{\texttt{correct}(m_1), \ldots, \texttt{correct}(m_n)\}}{\texttt{allModelsCorrect}}$$

Now that we have presented the set of deduction rules, we introduce a correctness criterion. In the next definition, in order to uniquely denote observations, fresh names are assigned to them.

Definition 10. Let *spec* be a TFSM, Obs be a set of observations, and Hyp be a set of hypotheses. Let $A = \{ob = o \mid ob \text{ is a fresh name } \wedge\; o \in \texttt{Obs}\}$ and $B = \{h_1 \in \texttt{Hyp}, \ldots, h_n \in \texttt{Hyp}\}$, where $\texttt{Hyp} = \{h_1, \ldots, h_n\}$.

If the deduction rules allow to infer allModelsCorrect from the set of predicates $C = A \cup B$, then we say that C *logically conforms to spec* and we denote it by C logicConf *spec*. □

In order to prove the validity of our method, we have to relate the deductions obtained by using our logic with the notion of conformance introduced in Definition 5. The *semantics* of a predicate is described in terms of the set of TFSMs that fulfill the requirements given by the predicate; given a predicate p, we denote this set by $\nu(p)$. Due to space limitations we cannot include the definition of ν (despite the differences, the construction is similar to that in [21] for classical finite state machines). Let us consider that P is the conjunction of all the considered observation and hypothesis predicates. Intuitively, the set $\nu(P)$ denotes all the TFSMs that can produce these observations and fulfill these hypotheses, that is, all the TFSMs that,

according to our knowledge, can *define* the IUT. So, if our logic deduces that these TFSMs conform to the specification (i.e., allModelsCorrect is obtained) then the IUT actually conforms to the specification.

Theorem 1. Let *spec* be a TFSM, Obs be a set of observations, and Hyp be a set of hypotheses. Let $A = \{ob = o \mid ob \text{ is a fresh name } \wedge\ o \in \texttt{Obs}\} \neq \emptyset$ and $B = \{h_1 \in \texttt{Hyp}, \ldots, h_n \in \texttt{Hyp}\}$, where $\texttt{Hyp} = \{h_1, \ldots, h_n\}$. Let $C = A \cup B$ be a set of predicates and $H = \texttt{reduce}(\texttt{Obs})$. Then, C logicConf *spec* iff for all TFSM $M \in \nu(\bigwedge_{p \in C})$ we have $M\ \texttt{conf}^H_{int}$ *spec* and $\nu(\bigwedge_{p \in C}) \neq \emptyset$.

Corollary 1. Let IUT and *spec* be TFSMs, Obs be a set of observations, and Hyp be a set of hypotheses. Let $A = \{ob = o \mid ob \text{ is a fresh name } \wedge\ o \in \texttt{Obs}\} \neq \emptyset$ and $B = \{h_1 \in \texttt{Hyp}, \ldots, h_n \in \texttt{Hyp}\}$, where $\texttt{Hyp} = \{h_1, \ldots, h_n\}$. Let $C = A \cup B$ and $H = \texttt{reduce}(\texttt{Obs})$. If $IUT \in \nu(\bigwedge_{p \in C})$ then C logicConf *spec* implies $IUT\ \texttt{conf}^H_{int}$ *spec*. If there exists $M \in \nu(\bigwedge_{p \in C})$ such that $M\ \texttt{conf}^H_{int}$ *spec* does not hold then C logicConf *spec* does not hold.

5 Conclusions and Future Work

In this paper we have presented $\mathcal{THOTL}$: A timed extension of $\mathcal{HOTL}$ to deal with systems presenting temporal information. What started as a simple exercise, where only a couple of rules were going to be modified, became a much more difficult task. The inclusion of time complicated not only the original framework, with a more involved definition of the *accounting* and the functions that modify it, but added some new complexity with the inclusion of new rules. The first task for future work is to produce a longer version of this paper were all the issues that either could not be included or could not be explained with enough details, are considered. This includes to elaborate on the semantics of predicates. The second task is to construct, taking as basis the current paper and [18], a stochastic version of $\mathcal{HOTL}$.

References

1. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
2. S.S. Batth, E. Rodrigues Vieira, A. Cavalli, and M.Ü. Uyar. Specification of timed EFSM fault models in SDL. In *27th IFIP WG 6.1 Int. Conf. on Formal Techniques for Networked and Distributed Systems, FORTE'07, LNCS 4574*, pages 50–65. Springer, 2007.
3. E. Bayse, A. Cavalli, M. Núñez, and F. Zaïdi. A passive testing approach based on invariants: Application to the WAP. *Computer Networks*, 48(2):247–266, 2005.
4. B.S. Bosik and M.Ü. Uyar. Finite state machine based formal methods in protocol conformance testing. *Computer Networks & ISDN Systems*, 22:7–33, 1991.
5. L. Brandán Briones and E. Brinksma. Testing real-time multi input-output systems. In *7th Int. Conf. on Formal Engineering Methods, ICFEM'05, LNCS 3785*, pages 264–279. Springer, 2005.
6. E. Brinksma and J. Tretmans. Testing transition systems: An annotated bibliography. In *4th Summer School on Modeling and Verification of Parallel Processes, MOVEP'00, LNCS 2067*, pages 187–195. Springer, 2001.
7. R. Cardell-Oliver. Conformance tests for real-time systems with timed automata specifications. *Formal Aspects of Computing*, 12(5):350–371, 2000.

8. E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.
9. A. En-Nouaary and R. Dssouli. A guided method for testing timed input output automata. In *15th Int. Conf. on Testing Communicating Systems, TestCom'03, LNCS 2644*, pages 211–225. Springer, 2003.
10. M.A. Fecko, M.Ü. Uyar, A.Y. Duale, and P.D. Amer. A technique to generate feasible tests for communications systems with multiple timers. *IEEE/ACM Transactions on Networking*, 11(5):796–809, 2003.
11. T. Higashino, A. Nakata, K. Taniguchi, and A. Cavalli. Generating test cases for a timed I/O automaton model. In *12th Int. Workshop on Testing of Communicating Systems, IWTCS'99*, pages 197–214. Kluwer Academic Publishers, 1999.
12. G.-D. Huang and F. Wang. Automatic test case generation with region-related coverage annotations for real-time systems. In *3rd Int. Symposium on Automated Technology for Verification and Analysis, ATVA'05, LNCS 3707*, pages 144–158. Springer, 2005.
13. M. Krichen and S. Tripakis. An expressive and implementable formal framework for testing real-time systems. In *17th Int. Conf. on Testing of Communicating Systems, TestCom'05, LNCS 3502*, pages 209–225. Springer, 2005.
14. B.T. Ladani, B. Alcalde, and A.R. Cavalli. Passive testing - a constrained invariant checking approach. In *17th Int. Conf. on Testing of Communicating Systems, TestCom'05, LNCS 3502*, pages 9–22. Springer, 2005.
15. D. Lee and M. Yannakakis. Principles and methods of testing finite state machines: A survey. *Proceedings of the IEEE*, 84(8):1090–1123, 1996.
16. M.G. Merayo, M. Núñez, and I. Rodríguez. A brief introduction to $\mathcal{THOTL}$. In *5th Int. Symposium on Automated Technology for Verification and Analysis, ATVA'07, LNCS 4762*, pages 501–510. Springer, 2007.
17. M.G. Merayo, M. Núñez, and I. Rodríguez. Formal testing from timed finite state machines. *Computer Networks*, 52(2):432–460, 2008.
18. M. Núñez and I. Rodríguez. Towards testing stochastic timed systems. In *23rd IFIP WG 6.1 Int. Conf. on Formal Techniques for Networked and Distributed Systems, FORTE'03, LNCS 2767*, pages 335–350. Springer, 2003.
19. A. Petrenko. Fault model-driven test derivation from finite state models: Annotated bibliography. In *4th Summer School on Modeling and Verification of Parallel Processes, MOVEP'00, LNCS 2067*, pages 196–205. Springer, 2001.
20. I. Rodríguez, M.G. Merayo, and M. Núñez. A logic for assessing sets of heterogeneous testing hypotheses. In *18th Int. Conf. on Testing Communicating Systems, TestCom'06, LNCS 3964*, pages 39–54. Springer, 2006.
21. I. Rodríguez, M.G. Merayo, and M. Núñez. $\mathcal{HOTL}$: Hypotheses and observations testing logic. *Journal of Logic and Algebraic Programming*, 74(2):57–93, 2008.
22. J. Springintveld, F. Vaandrager, and P.R. D'Argenio. Testing timed automata. *Theoretical Computer Science*, 254(1-2):225–257, 2001. Previously appeared as Technical Report CTIT-97-17, University of Twente, 1997.