A brief introduction to $\mathcal{THOTL}$*

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. In this paper we extend $\mathcal{HOTL}$ (Hypotheses and Observations Testing Logic) to provide a formal framework to test timed systems. The main idea underlying $\mathcal{HOTL}$ is to infer whether a set of *observations* (i.e., results of test applications) allows to claim that the IUT conforms to the specification *if* a specific set of hypotheses is assumed. In this paper we adapt $\mathcal{HOTL}$ to cope with the inclusion of time issues. In addition, we show the soundness and completeness of the new framework, that we call $\mathcal{THOTL}$, with respect to a general notion of timed conformance.

1 Introduction

The main goal of this paper is to extend $\mathcal{HOTL}$ [2] (*Hypotheses and Observations Testing Logic*) to deal with timed systems. The *correctness* of an implementation with respect to a given specification can be stated by using a notion of *conformance*: An implementation *conforms* to a specification if the former shows a behavior *similar* to that of the latter. In this line, we may use formal testing techniques to extract tests from the specification, each test representing a desirable behavior that the implementation under test (in the following IUT) must fulfill. In order to limit the (possibly infinite) time devoted to testing, testers add some reasonable assumptions about the structure of the IUT. However, a framework of hypotheses established in advance is very strict and limits the applicability of a specific testing methodology. The logical framework $\mathcal{HOTL}$ was introduced to cope with the rigidity of other frameworks. $\mathcal{HOTL}$ 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 first decision to define the new framework, that we call $\mathcal{THOTL}$, is to consider a formal language to represent timed systems. 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. Regarding the time domain, we decided to choose a simple approach but richer than singles values: Time intervals.

* 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. A longer version of this paper can be found in [1]

Once the language and a notion of timed conformance is fixed, we have to work on how to *adapt* $\mathcal{HOTL}$ to the new setting. First, we have 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. Next, we have to modify the existing rules.

The rest of the paper is organized as follows. In Section 2 we briefly review the main concepts underlying the definition of $\mathcal{HOTL}$. This section represents a (very small) summary of [2]. In Section 3 we introduce our extension of finite state machines to model timed systems and define two implementation relations. In Section 4, representing the bulk of the paper, we define the new logical framework. Finally, in Section 5 we present our conclusions and some directions for further research.

2 A short summary of $\mathcal{HOTL}$

The goal of $\mathcal{HOTL}$ is to decide whether a given finite set of observations, extracted by applying a test suite to an IUT, is *complete* in the case that the considered hypotheses hold. In other words, we assess whether obtaining these observations from the IUT implies that the IUT conforms to the specification *if* the hypotheses hold. Our logic assumes that specifications and implementations can be represented by using a classical formalism, finite state machines.

The behavior of the IUT observed during the application of tests is represented by means of *observations*, that is, a sequence of inputs and outputs denoting the test and the response produced by the IUT, respectively. Moreover, observations include *attributes* that allow to represent hypothesis concerning specific states of the IUT. Once the observations have been processed, the deduction *rules* of the logic will allow to infer whether we can claim that the IUT conforms to the specification.

Next, we will review the basic elements that are part of the original $\mathcal{HOTL}$: *Observations, predicates, and rules*. 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.

2.1 Observations

Observations follow the form $ob = (a_1, i_1/o_1, a_2, \ldots, a_n, i_n/o_n, a_{n+1}) \in \texttt{Obs}$, where ob is a unique identifier. This observation denotes that when the sequence of inputs $i_1, \ldots, i_n$ was proposed to the implementation, the sequence $o_1, \ldots, o_n$ was obtained as response. In addition, 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. For all $1 \leq j \leq n+1$ the attributes in 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. 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.

2.2 Predicates

A *model predicate* denotes our knowledge about the implementation. Models will be constructed according to the observations and hypotheses we consider. We denote model predicates by $\texttt{model}\,(m)$, where $m = (\mathcal{S}, \mathcal{T}, \mathcal{I}, \mathcal{A}, \mathcal{E}, \mathcal{D}, \mathcal{O})$ is a *model*. The meaning of the different components of the tuple is:

- $\mathcal{S}$ (*states*): The set of states that appear in the model.
- $\mathcal{T}$ (*transitions*): Set of transitions appearing in the graph of the model.
- $\mathcal{I}$ (*initial states*): Set of states that are initial in the model.
- $\mathcal{A}$ (*accounting*): The set of *accounting registers* of the model. An accounting register is a tuple $(s, i, outs, f, n)$ 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 labeled 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 $\top$.
- $\mathcal{E}$ (*equality relations*): Set of equalities relating states belonging to $\mathcal{S}$. Equalities have the form s `is` q, where $s \in \mathcal{S}$ is a state and $q \in \mathcal{Q}$ is a state identifier name.
- $\mathcal{D}$ (*deterministic states*): Set of states that are deterministic.
- $\mathcal{O}$ (*used observations*): Set of observations we have used so far for the construction of this model.

Depending on the form of m, a $\texttt{model}\,(m)$ predicate may denote some additional information about m. Models can be labeled by some *tags* to denote special characteristics. A different predicate, `allModelsCorrect`, represents a set of correct models. This predicate is the *goal* of the logic: If it holds then all the IUTs that could produce the observations in `Obs` and meet all the requirements in `Hyp` conform to the specification.

In general, several models can be constructed from a set of observations and hypotheses. Hence, our logic will deal with *sets* of models. If $\mathcal{M}$ is a set of models then the $\texttt{models}\,(\mathcal{M})$ predicate denotes that, according to the observations and hypotheses considered, $\mathcal{M}$ contains all the models that are valid candidates to properly describe the implementation.

2.3 The $\mathcal{HOTL}$ rules

Once all the observations have been considered a second phase, to add the hypotheses, starts. All rules of the second phase will include the requirement $\mathcal{O} = \texttt{Obs}$.

We present a rule to construct a model from a simple observation. Given a predicate denoting that an observation was collected, the rule deduces some details about the behavior of the implementation.

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

where $f_{s_j}(tr) = 1$ if $tr = s_j \xrightarrow{i_j/o_j} s_{j+1}$ and $f_{s_j}(tr) = 0$ otherwise. Intuitively, the sets $\mathcal{S}'$, $\mathcal{T}'$, $\mathcal{A}'$, and $\mathcal{D}'$, denote the additions due to the possible occurrence of an attribute of the form $\texttt{spec}(s)$. The formal definition of these sets can be found in [2].

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

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

By iteratively applying these two rules, we will eventually obtain a model where $\mathcal{O}$ includes all the observations belonging to the set $\texttt{Obs}$. At this point, the inclusion of those hypotheses that are not covered by observations will begin. Our logic considers several possible hypotheses about the IUT. The complete repertory of hypotheses appearing in $\mathcal{HOTL}$ appears in [2].

Since a model is a (probably incomplete) representation of the IUT, in order to check whether a model conforms to the specification, two aspects must be taken into account. First, only the conformance of consistent models will be considered. Second, we will check the conformance of a consistent model 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 will conform to the specification only if the unspecified parts of the model are not relevant for the correctness of the IUT it represents. $\mathcal{HOTL}$ provides us with an additional rule that allows to deduce the correctness of a model if the model is consistent and the worst instance of the model conforms to the specification.

3 Timed FSMs and timed implementations relations

In this section we present our timed extension of the classical finite state machine model. We also introduce an *implementation relation* to formally define what is a *correct* implementation. Time intervals will be used to express time constraints associated with the performance of actions. 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 $\mathbb{R}^+$. □

Let us note that in the case of $[t_1, \infty]$ we are abusing the notation since this interval is in fact a half-closed interval, that is, it represents the interval $[t_1, \infty)$.

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.

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

As usual, we assume that both implementations and specifications can be represented by appropriate TFSMs.

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 present the basic conformance relation that we consider in our framework. Intuitively, an IUT is conforming if it does not *invent* behaviors for those traces that can be executed by the specification.

Definition 3. Let S and I 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 $\rho_2 \in \mathtt{NTEvol}(S)$. □

In addition to the non-timed conformance of the implementation, we require some time conditions to hold. 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. Regarding sampling applications, we just associate with each evolution the multiset of observed execution time values.

Definition 5. Let I and S be 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 $e \in \Phi$ 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 $e \in \Phi$ we have that for all time interval $\hat{d} \in \mathcal{I}_{\mathbb{R}_+}$ we have that if $(e, \hat{d}) \in \mathtt{TEvol}(S)$ then $\widehat{S}_{(H,e)} \subseteq \hat{d}$ holds. □

4 Timed extension of $\mathcal{HOTL}$: $\mathcal{THOTL}$

In this section we show how $\mathcal{HOTL}$ has to be adapted and extended to cope with time issues. While some of the rules dealing with the internal *functional* structure of the implementation remain the same, the inclusion of time strongly complicates the framework, constituting $\mathcal{THOTL}$ almost a complete *new* formalism. First, we need 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. Finally, we will modify the deduction rules.

4.1 Temporal observations

Temporal observation are an extension of the observations introduced in $\mathcal{HOTL}$. They follow the format $ob = (a_1, i_1/o_1/t_1, a_2, \ldots, a_n, i_n/o_n/t_n, a_{n+1}) \in \texttt{Obs}$. It 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.

In addition to the attributes presented in Section 2, temporal observations may include a new type of attribute. 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 once the implementation has performed $i_1/o_1, \ldots, i_{j-1}/o_{j-1}$, the time that it takes to emit the output o_j, after the input i_j is received, 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. Thus, at this stage, no actions have taken place yet.

4.2 New 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 $m = (\mathcal{S}, \mathcal{T}, \mathcal{I}, \mathcal{A}, \mathcal{E}, \mathcal{D}, \mathcal{O})$. The only change we need to introduce affects the accounting component $\mathcal{A}$. Now, each register will be a tuple $(s, i, outs, f, \delta, n)$ where the new function $\delta : \mathcal{T} \longrightarrow \mathcal{I}_{\mathbb{R}_+} \times \wp(\mathbb{R}^+)$ computes for each transition departing from state s with input i and output $o \in outs$ the time interval, according with our knowledge up to now, in which the transition could be performed. In addition, it also returns 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, the interval will be set to $[0, \infty]$. In the case of transitions not fulfilling the required conditions about s, i, and $outs$, an arbitrary value is returned.

4.3 Changing the existing rules

First, we will include a new rule to construct a model from a temporal observation. This rule plays a similar role to the original ($obser$) rule of $\mathcal{HOTL}$ that allows to build a model from a simple non-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. The sets $\mathcal{S}'$, $\mathcal{T}'$, $\mathcal{A}'$, and $\mathcal{D}'$ are formally defined in [1]

The iterative application of the previously introduced (*fusion*) rule (see Section 2.3) will allow 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 `TFSM`s that are compatible with a set of observations and hypotheses. Some of the rules use the `modelElim` function. If we find out 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. A formal definition of this function can be found in [1]. It is only necessary to consider that those rules using `modelElim` have to consider the temporal version of this function. The rest of the rules belonging to $\mathcal{HOTL}$ do not vary in their formulation.

In $\mathcal{HOTL}$ we have 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. Due to space limitations we do not include the details of this rule (the formal definition can be found in [2]).

Similar to $\mathcal{HOTL}$, as explained at the end of Section 2, 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 6. 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

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

□

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 introduce a correctness criterion. In the next definition, in order to uniquely denote observations, fresh names are assigned to them. Besides, let us note that all hypothesis predicates follow the form $h \in \texttt{Hyp}$ for some h belonging to Hyp.

Definition 7. 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 `TFSM`s that fulfill the requirements given by the predicate; given a predicate p, we denote this set by $\nu(p)$. Despite the differences, the construction is similar to that in [2] 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 `TFSM`s that can produce these observations and fulfill these hypotheses, that is, all the `TFSM`s that, according to our knowledge, can *define* the IUT. So, if our logic deduces that these `TFSM`s conform to the specification 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 $\nu(\bigwedge_{p \in C}) \neq \emptyset$.

5 Conclusions and future work

In this paper we have provided an 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. As we have already commented, the inclusion of time complicates not only the original framework, with a more involved definition of the *accounting* and the functions that modify it, but adds some new complexity with the addition of new rules. These new rules have not been included in this paper due to lack of space, but they can be found in [1]. Regarding future work, the first task is to show the validity of the method, that is, to formally prove the soundness and completeness of $\mathcal{THOTL}$. The second task is to construct a stochastic version of $\mathcal{HOTL}$. Taking the current paper as basis this task should be easy.

References

1. M.G. Merayo, M. Núñez, and I. Rodríguez. $\mathcal{THOTL}$: A timed extension of $\mathcal{HOTL}$, 2007. Available at `http://kimba.mat.ucm.es/testing/papers/thotl.pdf`.
2. I. Rodríguez, M.G. Merayo, and M. Núñez. $\mathcal{HOTL}$: Hypotheses and observations testing logic. *Journal of Logic and Algebraic Programming (in press)*, 2007. Available at `http://dx.doi.org/10.1016/j.jlap.2007.03.002`.