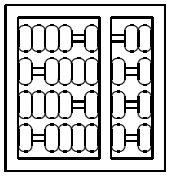

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

A more Flexible Discretization for Complex Timed Systems

A. L. Bonifacio *A. V. Moura*

Technical Report - IC-12-09 - Relatório Técnico

March - 2012 - Março

Attaining a more Flexible and Manageable Discretization for Complex Timed Systems

Adilson Luiz Bonifacio* Arnaldo Vieira Moura†

Abstract

Complex systems have been widely investigated using formal techniques for testing and verifying critical aspects of their reactive behavior. Such behavior is usually captured by the notion of context transformations that interact with the environment. Other aspects require timed models to describe the systems' continuous evolution in time. In this work, we propose a timed context formalism able to model the combination of time evolution and context transformations. Further, we derive a more flexible and manageable strategy to discretize such models and prove correctness of the discretized model against the original one. The flexibility to find suitable granularities both to time evolution as well as to values of context variables opens the possibility for constructing more compact grid automata. Based on this discretizing approach we present a testing framework to automatically generate test suites from the resulting automata. The test suites can then be used to verify properties of candidate implementations with the aid of test purposes.

1 Introduction

The process of verifying and testing complex computational systems has been intensively investigated. Several techniques have been proposed to model such systems, including their reactive and real-time behaviors. Interactions with the environment are usually captured by the notion of data flow transformations. Continuous time evolution aspects require a time discretization and the use of timed formal models.

Here we treat complex systems that obey both time requirements and also satisfy data flow transformations, with both these aspects being treated within the same formal model.Often, Timed Input/Output Automata (TIOA) [24, 4, 21], a variant of the classical timed automata [2, 3, 10], have been used to express continuous time evolution. In order to also capture data flow effects, we extend the TIOA model by introducing context variables [39, 29, 32]. The new formalism, Timed Input/Output Context Automata (TIOCA), allows for both continuous time evolution [14, 16] and data flow transformations [39, 32, 23].

Discretizing such complex models, however, incur in the well-known state explosion problem [38, 11, 9, 8]. Thus, devising techniques and methods that properly deal with

*Department of Computing, University of Londrina, bonifacio@uel.br,
†Computing Institute, University of Campinas, arnaldo@ic.unicamp.br,

reactive and real-time systems remains a challenge. Here, we propose a new discretization mechanism using the classical notion of grid automata [16, 15], but now also including data flow transformations. The discretization technique is inspired by more recent ideas [4, 5], which deviates from the traditional approach based on clock regions [3, 28] when treating timing aspects. Notably, these techniques allow for a much wider range of granularity choices and, furthermore, make precise the notion of simulation boundaries. It is then possible to precisely establish the relationship between the original system behavior and the corresponding grid automaton. More specifically, we show that a TIOCA homomorphically [1, 13] simulates its corresponding grid automaton, and vice-versa.

Several formalisms and methods have been developed to automatically construct test suites for pure timed systems [16, 26, 33] or for systems that display context transformations [32]. But these approaches [6, 30] do not capture both aspects in a proper way, specially regarding the continuous evolution of time. An approach to properly deal with these aspects in the same model has remained elusive. Using the new approach, once we are able to properly discretize the TIOCA behavior, we can use established model-based testing strategies [12, 31, 35, 36] in order to automatically generate test suites. For that, we use the notion of test purpose models [20, 34] when specifying which system properties will be subjected to testing. The joint behavior of a specification and a test purpose is then given by their classical synchronous product. After applying the new discretization approach to the product automaton we can automatically extract test suites. Such test suites, when applied to implementations, result in runs from which conformance verdicts can be obtained [36, 25]. This strategy is able to treat models that simultaneously deal with context variables and input parameters, as well as with timing aspects. Theoretical results indicate that the new discretization approach leads to more compact test suites.

Our focus, throughout this work, is to propose precise formal models and establish the correctness of the discretization process, so as to lay the foundations for new implementations and field tests.

This paper is organized as follows. In Section 2 we survey some related works. In Section 3 we present the TIOCA model. Section 4 discusses the process of test case generation and extraction, and also shows how to obtain test verdicts, in a general scenario. In Section 5 we describe the new discretization technique for timed context models. Finally, some concluding remarks and future directions appear in Section 6.

2 Related work

In the literature, one finds either techniques that deal only with time, or methods applicable to systems with context variables and no time constraints. Some alternatives deal with these aspects in the same model, but still present some shortcomings. In this section we describe some works that are more closely related to the ideas presented in this paper.

Springintveld et al. [33] propose a discretization method based on the number of clock regions [3]. Their proposal requires that a grid automaton represents all clock regions defined by a finite set of clock interpretations in the corresponding TIOA. However, their discretization imposes a relationship between the number of clock variables present in the

model and the granularities that must be chosen in order to obtain the corresponding grid automaton: granularities must be around $\frac{1}{n}$, where n is the number of clocks. A potential problem that can arise here is the large number of test sequences generated after constructing the grid automaton. As noted by those authors, the number of test sequences may be astronomically large, with time delays occurring in these sequences being microscopically small. We deviate from this restriction in the sense that we can use arbitrary granularities to discretize TIOCA models. Since there is a trade off between the chosen granularities and the number of test sequences obtained from the discrete model, our approach opens the possibility for obtaining smaller grids and, thus, shorter test suites. Further, we treat context transformations and time requirements in the same formalism.

En-Nouaary and Dssouli [16] discuss a test case generation method based on timed automata specifications and test purposes which are also combined by a synchronous product. Their discretization technique is based on classical clock regions [3], thus also imposing a relationship between the number of clock variables present in the models and the granularities that must be chosen in order to obtain the grid automaton. Moreover, instead of constructing a grid automaton directly, the notion of a region graph is first used to represent a possibly infinite transition system. Then, by a process of sampling, a grid is derived and from the latter test sequences can be extracted. Further, context variables are not allowed.

Bonifacio and Moura [5] propose a new method to discretize timed systems. That approach deviates from the classical notion of clock regions, giving rise to a more general algorithm that allows for an ample range of granularity values that can be chosen in the discretization process. They proved that the resulting grid automaton is capable of precisely simulating the original model, and conversely. In the present paper we extended that discretization technique in order to obtain grid automata for the more expressive TIOCA models, which can deal with data flow and timing constraints.

Fouchal [19]proposes another test generation method based on timed automata and test purposes. In that work, region graphs are again used and are likewise sampled in order to obtain grids. Test purposes are also used, following similar approaches adopted in other works [16]. Since region graphs are also sampled in order to obtain grid automata, no guarantees are postulated about the algorithms presented therein, which are used in an exhaustive test generation process. It is difficult to realize how precise are the timed test sequences that are obtained, specially when the original TIOA models are combined with test purposes.

In another work, Fouchal et al. [18] present a test strategy similar to the one discussed here. But, whereas our work deals with dense time in order to capture timed properties, there the notion of timed elements are used to imitate continuous time evolution, thus offering no guarantees of accuracy of the test suites. These models, also, do not deal with context variables, and so context transformations and values returned to the environment are not considered.

Merayo et al. [30] extend the EFSM model [32] in order to capture both time and context transformation aspects. Their extended EFSMs are similar to the timed extended FSMs presented by Bonifacio et. al. [6]. But the models are purely syntactic and there is no semantic considerations that could represent real-time evolution.

Similar approaches have appeared [14, 15, 17], but none of them deals with context

transformations. All of them use clock regions [3] to obtain grids from which test sequences can be extracted. But even a moderate number of clock variables in typical models often leads to huge grids, due to the exponential number of clock regions and the need to enforce the relationship between the number of clocks and the granularities used. In contrast, our approach allows for a more ample range of choices of granularity values for both time and context values, thus leading to more controllable grids and to more manageable test suites.

Petrenko et al. [32] offers a different approach. Their aim is to verify whether a test sequence yields the same output when applied to suspicious configurations. Suspicious configurations are obtained with the aid of expertise from system testers. Further, no treatment of continuous time evolution is provided. In our work, we can precisely model both faulty and desired behaviors and, in contrast, we can also handle timed systems with context variables.

Jeannet et al. [23] use the ioSTS formalism to specify models. Then, an enumeration of data values is used in order to avoid the state space explosion problem. But, since a discretization method is not used, it is problematic to capture continuous time evolution in an appropriate manner in these models, thus making it difficult to test timed properties. Moreover, the method can incur high costs in the process of calculating constraints using approximations when searching for test sequences. Further, the formal model and the proposed method do not deal simultaneously with timed requirements and context transformations.

Briones and Brinksma [7] propose an approach using a conformance relation based on the *ioco* framework. However, there is no mechanism to discretize the continuous behavior. As a consequence, time evolution is not captured in an appropriate manner for precisely testing timed properties. A different approach is undertaken by Khoumsi [25] using another conformance relation, called *tioco*. The latter approach, however, does not consider context transformations.

Other works [22] are based on a variant of the timed automata model, called TIOTS. There, a different mechanism for deriving test sequences is proposed based on a model-checking framework. An interesting strategy of trace inclusion is also used for conformance specifications. However, again, TIOTS do not have context variables and so do not allow for context transformations neither for the exchange of values with the environment.

Krichen and Tripakis [26] propose a formal framework for testing classical timed automata models. They discuss the problem of testing time faults for both analog-clock and digital-clock systems. Digital-clocks are modeled by a sequence of *ticks* which can be understood as a granularity for the purpose of discretization. A periodic sampling test is then formed by a sequence of actions and ticks. Time intervals between ticks, where input actions can occur, are modeled by analog-clocks. However, the semantic of continuous time evolution in analog-clocks, as described in another work [37], uses again the classical notion of region graphs, or clock regions [3, 28]. The approach, thus, also enforces a relationship between the number of clocks and certain bi-similar states. Concerning the continuous time evolution aspect only, their discretization method can be seen as a particular case of the method described here, if we choose granularity values similar to the tick values. But, in contrast, our approach offers a much more flexible choice for granularities that can be used in the discretization process, thus leading to more controllable grid automata and to

potentially more manageable test suites. Furthermore, our work includes context transformations and detailed proofs showing that the TIOCA homomorphically [1, 13] simulates its corresponding grid automaton, and conversely. Such guarantees are not presented by Krichen and Tripakis [26] in a detailed way.

Finally, Krichen [27] also presents a black-box conformance testing for real-time systems. In that work, a symbolic reachability graph is used in an attempt to reduce the number of states and edges in the original reachability graph, thus possibly reducing the number of test sequences that are generated. The same *ioco* framework [7] is applied here, thus incurring in the same limitations for capturing continuous behavior. Also, the work does not offer formal guarantees about the relationship between the behavior of the grid automata and the original timed models.

3 Timed I/O Context Automata

In this section we introduce the notions of context variables and of input parameters [32] in order to extend the classical timed automaton model. Context variables play the role of common variables in ordinary programming. They can enter into expressions and can be assigned to. Input parameters will be used to read in values passed down by the environment. Similarly, values computed by the system can be passed back to the environment. With this new formal model we will be able to capture both aspects related to continuous time evolution and to data flow transformations.

We start with some preliminary concepts that will be used to model time evolution and data flow. Next an example informally illustrates the devised TIOCA model. The formal model appears in the last subsection.

3.1 Conditions and interpretations

Conditions and interpretations over variables are required to model the continuous evolution of time and the data flow transformations. Such behaviors will be described by functions, conditions and interpretations with the set of non-negative rational numbers, $\mathbb{Q}_{\geq}$, as domain. Given $t \in \mathbb{Q}_{\geq}$, we will denote the integral and fractional parts of t by $\lfloor t \rfloor$ and $\lceil t \rceil$, respectively. Hence, $t = \lfloor t \rfloor + \lceil t \rceil$.

The following simple facts will be useful later.

Fact 1 *If $x, y, z \in \mathbb{Q}_{\geq}$ and k is a positive integer, then*

1. *If $x \geq y$ then $\lfloor x \rfloor \geq \lfloor y \rfloor$.*
2. *If $x = y + k$ then $\lceil x \rceil = \lceil y \rceil$ and $\lfloor x \rfloor = \lfloor y \rfloor + k$.*
3. *If $x = y + z$ then $\lceil x \rceil = \left\lceil \lceil y \rceil + \lceil z \rceil \right\rceil$ and $\lfloor x \rfloor = \lfloor y \rfloor + \lfloor z \rfloor + \left\lfloor \lceil y \rceil + \lceil z \rceil \right\rfloor$.*
4. *If $\lceil x \rceil = \lceil y \rceil$ then $\lceil x + z \rceil = \lceil y + z \rceil$.*

Proof We start with item 1.

Assume $\lfloor x \rfloor < \lfloor y \rfloor$. Then $\lfloor x \rfloor \leq \lfloor y \rfloor - 1$. So $x < \lfloor x \rfloor + 1 \leq \lfloor y \rfloor - 1 + 1 = \lfloor y \rfloor \leq k$. Hence, $x < y$, contradicting the hypothesis.

We now prove item 2.

We have that $x = y + k = (k + \lfloor y \rfloor) + \lceil y \rceil$. Since $k + \lfloor y \rfloor$ is an integer and $0 \leq \lceil y \rceil < 1$, we have $\lfloor x \rfloor = k + \lfloor y \rfloor$ and $\lceil x \rceil = \lceil y \rceil$.

Now we follow with item 3.

We have $x = (\lfloor y \rfloor + \lfloor z \rfloor) + \lceil y \rceil + \lceil z \rceil = (\lfloor y \rfloor + \lfloor z \rfloor + \lfloor \lceil y \rceil + \lceil z \rceil \rfloor) + \lceil \lceil y \rceil + \lceil z \rceil \rceil$. But $\lfloor y \rfloor + \lfloor z \rfloor + \lfloor \lceil y \rceil + \lceil z \rceil \rfloor$ is an integer and $0 \leq \lceil \lceil y \rceil + \lceil z \rceil \rceil < 1$. Then $\lfloor x \rfloor = \lfloor y \rfloor + \lfloor z \rfloor + \lfloor \lceil y \rceil + \lceil z \rceil \rfloor$ and $\lceil x \rceil = \lceil \lceil y \rceil + \lceil z \rceil \rceil$.

Lastly we prove item 4.

We have $\lceil x + z \rceil = \lceil \lceil x \rceil + \lceil z \rceil \rceil$, using item (3). Now, using the hypothesis we get $\lceil \lceil x \rceil + \lceil z \rceil \rceil = \lceil \lceil y \rceil + \lceil z \rceil \rceil$. But $\lceil \lceil y \rceil + \lceil z \rceil \rceil = \lceil y + z \rceil$, using item (3) again. ∎

Let C be a set of symbols, also called variables. An interpretation over C is a partial function from C into $\mathbb{Q}_{\geq}$. A total interpretation over C is an interpretation over C whose domain[1] is C. The set of all interpretations and total interpretations over C, respectively, will be denoted by $[C \curvearrowright \mathbb{Q}_{\geq}]$ and $[C \to \mathbb{Q}_{\geq}]$.

The set of all variable conditions is defined next.

Definition 2 *Let C be a set of variables. The set of all variable conditions, Φ_C, is comprised by all expressions δ that can be finitely generated using the rules*

$$\delta := \texttt{true} \mid c \leq \tau \mid \tau \leq c \mid \neg\delta \mid \delta_1 \wedge \delta_2,$$

where c is a variable and $\tau \in \mathbb{Q}_{\geq}$.

We will take the usual liberties when writing variable conditions, *e.g.*, we may write $c \geq \tau$ for $\tau \leq c$, or $c < \tau$ instead of $\neg(\tau \leq c)$, or $\tau_1 \leq c \leq \tau_2$ for $(\tau_1 \leq c) \wedge (c \leq \tau_2)$.

Satisfiability is treated in the usual way.

Definition 3 *Let $\delta \in \Phi_C$ and $\nu \in [C \curvearrowright \mathbb{Q}_{\geq}]$ where $dom(\nu)$ contains all variables occurring in δ. Then ν satisfies δ, denoted $\nu \vDash \delta$, if δ is true when every variable c is replaced by $\nu(c)$ in δ and the resulting propositional logic sentence is evaluated in the usual manner.*

We now define how to displace an interpretation.

Definition 4 *Let $\nu \in [C \curvearrowright \mathbb{Q}_{\geq}]$ and $\tau \in \mathbb{Q}_{\geq}$. Then $(\nu + \tau)(c) = \nu(c) + \tau$ if $c \in dom(\nu)$, or undefined otherwise, for all $c \in C$.*

That is, $\nu + \tau$ just adds τ to all values in $\mathrm{dom}(\nu)$. Another important operation is interpretation overruling.

Definition 5 *Let ν, $\mu \in [C \curvearrowright \mathbb{Q}_{\geq}]$. We define $(\nu \oplus \mu)(c)$ as $\mu(c)$ if $c \in dom(\mu)$, as $\nu(c)$ if $c \in (dom(\nu) - dom(\mu))$, or undefined otherwise, for all $c \in C$.*

[1] $\mathrm{dom}(f)$ will denote the domain of a function f.

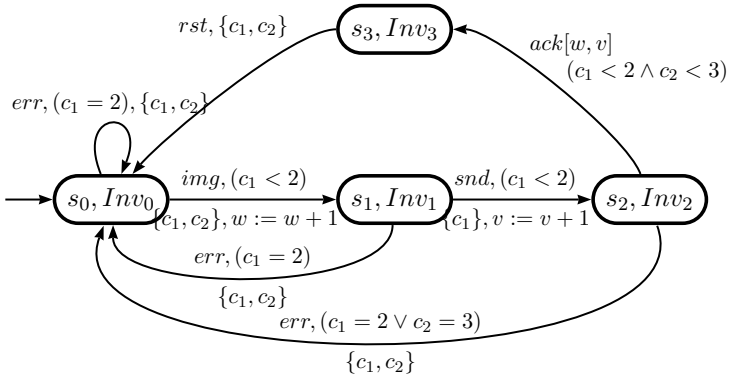

Figure 1: A TIOCA model for a multimedia protocol.

That is, $\nu \oplus \mu$ assigns values first according to μ and, baring that, then it assigns values according to ν, if at all possible.

Both operations of displacement and interpretation overruling will be required to capture the time progress and the zeroing over clock variables, and also to change the data information over context variables.

3.2 A multimedia protocol

A timed automaton that captures the behavior of a simple multimedia system is presented by En-Nouaary [15]. The protocol operates by receiving image frames followed by their respective sound tracks, the latter being supposed to arrive within two time units after the preceding image frame. When the input sound track does not follow the image frame within two time units, the system times out, issues an error message and goes back to the initial state. After the arrival of a sound track, the receiver must send an acknowledgment in no more than three time units after the reception of the image frame and no more than two time units after receiving the sound track. If the acknowledgment is not sent in time, an error occurs and the system resets itself to the initial state. After the acknowledgment is sent, a reset message reinitializes the system, so that it can await for the next image frame.

We model the multimedia protocol as depicted in Figure 1, now also allowing context transformations. There are four states, namely s_0, s_1, s_2 and s_3. The set of clocks is $C = \{c_1, c_2\}$, and the state invariants are $Inv_0 = (c_1 \leq 2)$, $Inv_1 = (c_1 \leq 2)$, $Inv_2 = (c_1 \leq 2 \wedge c_2 \leq 3)$, and $Inv_3 = (c_1 \leq 2 \wedge c_2 \leq 3)$. The set of inputs is $X = \{img, snd, rst\}$, and the set of outputs is $Y = \{err, ack\}$. The intended meaning for the input symbols img and snd is the arrival of an image frame and of a sound track, respectively. Symbol rst signals the arrival of a reset message. Symbol err signals an error, and the symbol ack sends an acknowledgment signal back to the environment.

There are seven transitions, indicated by the arrows in the diagram. At each transition, we first indicate the corresponding action symbol. The action symbol may be followed by a list of guards, given within parentheses. When there are no guards, the list is simply

omitted. In the example, at the transition from s_1 to s_2 there is a clock guard $(c_1 < 2) \in \Phi_C$. At the transition from s_3 to s_0 there are no guards. Next, after the list of guards, the set of clocks to be reset is given between braces. In this case, resets move the corresponding clocks back to zero, with the other clocks remaining unchanged. If there are no clocks to reset, the clock reset list is omitted. In the example, at the transition from s_1 to s_2, only clock c_1 is reset. There are no clock resets in the transition from s_2 to s_3. The last item at each transition indicates the corresponding context variable transformation. In the example, the set of context variables is $V = \{w, v\}$. As can be seen from the diagram, at the transition from s_0 to s_1, the context variable transformation is given by the expression $w := w + 1$. Its intended meaning, besides the assignment, is that the other context variable, v, remains unchanged. When all context variable transformations are omitted, as in the transition from s_2 to s_0, then all variables remain unchanged when that transition is taken. Variables w and v record the number of image and sound track arrivals, respectively.

In this example, input parameters associated to img, snd and rst assume a value of zero. Therefore transitions over those input symbols do not receive any value from the environment. As a consequence, the context transformations occur without considering any external stimuli to the system. Thus, the context variables w and v are simple counters, which are changed based only on their proper accrued values. But note that, in other situations, we could consider a positive integer associated to the input parameter img, and so the context variable transformations could be given by an expression, *e.g.* $w := w + p_{img} + 1$, where p_{img} is the input value taken from the environment.

There are four transitions associated with the set of outputs $Y = \{err, ack\}$. The transitions associated with err send no values back to the environment. At the transition from s_2 to s_3, over ack, the notation $ack[w, v]$ indicates that the current values of the variables w and v are passed back to the environment.

The next subsection makes these ideas precise.

3.3 The extended TIOA model

Starting with the standard definition of a TIOA [24, 4], we first informally discuss the extended model. The formal definition is given in the sequel.

Let X and Y be the set of input and output symbols of a TIOA M, respectively. In the new model we need a finite set R of input parameters, or parameters for short. For each $x \in X$, let $R_x \subseteq R$ be the set of parameters associated with x. A valuation in $[R_x \to \mathbb{Q}_{\geq}]$ gives a value for each parameter in R_x. The set of R_x conditions is Φ_x. So, we may write $\rho \vDash \delta$, where $\rho \in [R_x \rightharpoonup \mathbb{Q}_{\geq}]$ and $\delta \in \Phi_x$ — recall Definitions 2 and 3. Returning to the example, we have $X = \{img, snd, rst\}$, $Y = \{err, ack\}$ and we could have $R = \{p_{img}\}$, for example.

We will also need a set V of context variables with valuations in $[V \rightharpoonup \mathbb{Q}_{\geq}]$ and $[V \to \mathbb{Q}_{\geq}]$. The set of context variable conditions Φ_V and the notion of satisfiability $\lambda \vDash \delta$, where $\lambda \in [V \rightharpoonup \mathbb{Q}_{\geq}]$ and $\delta \in \Phi_V$, is as defined in Section 3.1. In the example, the set of context variables is $V = \{w, v\}$.

When taking a discrete transition on an input symbol x, the model may redefine the context variable values according to an expression involving input parameter values associated

to x and the current values of the context variables. We, thus, associate with each discrete transition on an input symbol x a map that takes a pair (ρ, λ), where $\rho \in [R_x \to \mathbb{Q}_{\geq}]$ and $\lambda \in [V \to \mathbb{Q}_{\geq}]$, and returns another valuation in $[V \to \mathbb{Q}_{\geq}]$. We collect the set of all such mappings in

$$F_x = \Big\{ \kappa \,|\, \kappa : [R_x \to \mathbb{Q}_{\geq}] \times [V \to \mathbb{Q}_{\geq}] \to [V \to \mathbb{Q}_{\geq}] \Big\}.$$

In the example, assuming we had an input parameter p_{img} associated to img, we would get $R_{img} = \{p_{img}\}$. Given that $V = \{w, v\}$, we would obtain

$$F_{img} = \Big\{ \kappa \,|\, \kappa : [\{p_{img}\} \to \mathbb{Q}_{\geq}] \times [\{w, v\} \to \mathbb{Q}_{\geq}] \to [\{w, v\} \to \mathbb{Q}_{\geq}] \Big\}.$$

On the other hand, since there is no input parameter associated to rst, we get $R_{rst} = \emptyset$, and so $F_{rst} = \Big\{ \kappa \,|\, \kappa : \{\emptyset\} \times [\{w, v\} \to \mathbb{Q}_{\geq}] \to [\{w, v\} \to \mathbb{Q}_{\geq}] \Big\}$.

A discrete transition over an input action symbol x moves the machine from a state s to a state r, provided that certain guard conditions over clocks, input parameters and context variables are satisfied. Also, when the transition is taken the machine can reset some clock values and redefine some context variable values. The new clock values are constants specified directly in the transition. The new context variable values are computed from the input parameter values and from the current values of the context variables. We collect these conditions and say that a discrete transition over an input x is a member of

$$S \times \{x\} \times \Phi_C \times \Phi_x \times \Phi_V \times [C \curvearrowright \mathbb{Q}_{\geq}] \times F_x \times S.$$

As an illustration, take the transition from s_0 to s_1, over the input img in Figure 1. The clock condition is $(c_1 < 2) \in \Phi_C$. The input parameter and context variable guards are both `true`, and so are not listed. The set $\{c_1, c_2\}$ says that both clocks are reset to zero. This can be specified by $\nu \in [C \curvearrowright \mathbb{Q}_{\geq}]$ where $\nu(c_1) = \nu(c_2) = 0$. Finally, the new value of the context variable w is defined by the expression $w := w + 1$. An appropriate mapping $\kappa \in F_x$ for this transition has the form

$$\kappa : \big[\{p_{img}\} \to \mathbb{Q}_{\geq}\big] \times \big[\{w, v\} \to \mathbb{Q}_{\geq}\big] \to \big[\{w, v\} \to \mathbb{Q}_{\geq}\big],$$

where $\kappa(\rho, \lambda)(w) = \lambda(w) + 1$ and $\kappa(\rho, \lambda)(v) = \lambda(v)$.

We have an analogous situation with discrete transitions over output symbols. The only difference is that, now, some context variables may have their values returned back to the environment. In general, if the current context variable valuation is λ, the machine will output a partial valuation given by $\xi(\lambda)$, where ξ is a mapping specified in the transition. The set of all such mappings is collected in

$$H_y^P = \big\{ \xi \,|\, \xi \in [V \to \mathbb{Q}_{\geq}] \to [V \curvearrowright \mathbb{Q}_{\geq}] \big\},$$

where y is the output symbol of interest. In the example, the transition from s_2 to s_3 specifies the output symbol as $ack[w, v]$ indicating that the values of the context variables w and v will be returned unmodified to the environment. In this case, the output mapping ξ would be just the identity, that is, $\xi(\lambda) = \lambda$, for all $\lambda \in [V \to \mathbb{Q}_{\geq}]$. Also, when an output

transition is taken, the values of the context variables may be modified. We model this by a mapping χ that takes a total context variable valuation and returns another total context variable valuation. We collect all such mappings in

$$H_y^T = \big\{\chi \,|\, \chi \in [V \to \mathbb{Q}_{\geq}] \to [V \to \mathbb{Q}_{\geq}]\big\},$$

where y is the output symbol. In the example, no output transition modifies the values of context variables, and so $\chi(\lambda) = \lambda$, for all $\lambda \in [V \to \mathbb{Q}_{\geq}]$, at all output transitions.

Putting it all together, we say that on a discrete transition over an output action symbol y the machine moves from a state s to a state r, provided that certain clock and context variable conditions are satisfied. Upon taking the transition, clocks may be reset, context variable values may be returned and may have their values modified. The clock reset values are constants specified directly at the transition. The values to be returned and the new context variable values are specified by choosing specific transformation mappings. That is, a discrete transition over an output action y is a member of

$$S \times \{y\} \times \Phi_C \times \Phi_V \times [C \rightharpoonup \mathbb{Q}_{\geq}] \times H_y^P \times H_y^T \times S.$$

We can now define the extended TIOA.

Definition 6 *A Timed Input/Output Context Automaton, TIOCA for short, M is given by a tuple $(S, s_0, \Sigma, C, \nu_0, Inv, R, V, \lambda_0, T_X, T_Y)$, where S is the set of states, $s_0 \in S$ is the initial state, $\Sigma = X \cup Y$ is the alphabet with $X \cap Y = \emptyset$, where X is the set of input symbols and Y is the set of output symbols, C is the set of clocks, ν_0 is the initial clock valuation with $\nu_0(c) = 0$ for all $c \in C$, and $Inv \in [S \to \Phi_C]$ gives the invariant at each state. The set of parameters is R, the set of context variables is V and λ_0 is the initial context variable valuation with $\lambda_0(v) = 0$ for all $v \in V$. The set of transitions over input symbols satisfies*

$$T_X \subseteq \bigcup_{x \in X} (S \times \{x\} \times \Phi_C \times \Phi_x \times \Phi_V \times [C \rightharpoonup \mathbb{Q}_{\geq}] \times F_x \times S),$$

where $F_x = \{\kappa \,|\, \kappa : [R_x \to \mathbb{Q}_{\geq}] \times [V \to \mathbb{Q}_{\geq}] \to [V \to \mathbb{Q}_{\geq}]\}$, and $R_x \subseteq R$. Finally, the set of transitions over output symbols satisfies

$$T_Y \subseteq \bigcup_{y \in Y} (S \times \{y\} \times \Phi_C \times \Phi_V \times [C \rightharpoonup \mathbb{Q}_{\geq}] \times H_y^P \times H_y^T \times S),$$

with $H_y^P = \big[[V \to \mathbb{Q}_{\geq}] \to [V \rightharpoonup \mathbb{Q}_{\geq}]\big]$, and $H_y^T = \big[[V \to \mathbb{Q}_{\geq}] \to [V \to \mathbb{Q}_{\geq}]\big]$.

So, a transition in T_X is a tuple $(s, x, \delta_1, \delta_2, \delta_3, \theta, \kappa, r)$ specifying that the machine can move from state s to state r over the input symbol x provided that the guards δ_1, δ_2, and δ_3, over clock variables, input parameters and context variables, respectively, are all enabled. Further, upon moving to state r, the mapping $\theta \in [C \rightharpoonup \mathbb{Q}_{\geq}]$ indicates which clocks are reset and to which values. Finally, $\kappa \in F_x$ denotes the context variable update function, mapping input parameter valuations and context variable valuations into new context variable valuations. A transition in T_Y is a tuple $(s, y, \delta_1, \delta_2, \theta, \xi, \chi, r)$ specifying that the machine can move from state s to state r over the output action symbol y, provided

that the guards δ_1 and δ_2, over clock variables and context variables, respectively, are enabled. Upon moving to state r the mapping $\theta \in [C \rightharpoonup \mathbb{Q}_{\geq}]$ gives which clocks are reset and to which values. Lastly, $\xi \in H_y^P$ specifies values to be passed to the environment, and $\chi \in H_y^T$ gives new values for the context variables.

Let M be a TIOCA with alphabet $\Sigma = X \cup Y$. Inputs to M are pairs (x, ρ), where $x \in X$ and $\rho \in [R_x \to \mathbb{Q}_{\geq}]$, and the machine sends back pairs (y, ξ) with $y \in Y$ and $\xi \in [V \rightharpoonup \mathbb{Q}_{\geq}]$. Define $\Psi_X = \{(x, \rho) \,|\, x \in X,\, \rho \in [R_x \to \mathbb{Q}_{\geq}]\}$ and $\Psi_Y = \{(y, \rho) \,|\, y \in Y,\, \rho \in [V \rightharpoonup \mathbb{Q}_{\geq}]\}$. Together, let $\Psi_\Sigma = \Psi_X \cup \Psi_Y$.

When M is in a state s, and before the next discrete transition out of s occurs, time can elapse by any amount $\tau \in \mathbb{Q}_{\geq}$, provided that the invariant at s, $Inv(s)$, holds continuously. So, the movements of M are governed by sequences of input or output pairs and time lapses.

Definition 7 *Let M be a TIOCA with alphabet $\Sigma = X \cup Y$. A timed context word, or timed word for short, for M is a sequence $\psi = \langle \sigma_1, \ldots, \sigma_n \rangle$, where $n \geq 0$ and $\sigma_i \in \Psi_\Sigma \cup \mathbb{Q}_{\geq}$, $1 \leq i \leq n$.*

We denote the set of all timed words for M by Ψ_M. The empty timed word will be denoted by ε. The concatenation of timed words $\psi_1, \psi_2 \in \Psi_M$ is denoted by $\psi_1 \cdot \psi_2$, or just $\psi_1\psi_2$ for short, and follows the standard notion of word concatenation.

A configuration specifies a state and valuations for all clocks and context variables, that is, a configuration is a triple (s, ν, λ) where $s \in S$, $\nu \in [C \to \mathbb{Q}_{\geq}]$ and $\lambda \in [V \to \mathbb{Q}_{\geq}]$. The initial configuration is (s_0, ν_0, λ_0). Let Γ_M denote the set of configurations of a TIOCA M. We can now define model semantics.

Definition 8 *Let M be a TIOCA and let $\gamma_i = (s_i, \nu_i, \lambda_i) \in \Gamma_M$, $i = 1, 2$, be two configurations of M.*

1. *Let $\tau \in \mathbb{Q}_{\geq}$. There exists a* continuous move *from γ_1 to γ_2 over τ, denoted by $\gamma_1 \xrightarrow[\tau]{} \gamma_2$, if and only if: (i) $s_1 = s_2$; (ii) $\nu_2 = \nu_1 + \tau$; (iii) $\nu_1 + \eta \vDash Inv(s_1)$ for all η, $0 < \eta \leq \tau$; and (iv) $\lambda_2 = \lambda_1$. When τ is positive, we have a* non-trivial continuous move*.*

2. *Let $(x, \rho) \in \Psi_X$. There is a* discrete move *from γ_1 to γ_2 over (x, ρ) if and only if there is a transition $(s_1, x, \delta_1, \delta_2, \delta_3, \theta, \kappa, s_2) \in T_X$ such that: (i) $\nu_1 \vDash \delta_1$, $\rho \vDash \delta_2$ and $\lambda_1 \vDash \delta_3$; (ii) $\nu_2 = \nu_1 \oplus \theta$ and $\nu_2 \vDash Inv(s_2)$; and (iii) $\lambda_2 = \kappa(\rho, \lambda_1)$. We denote this move by $\gamma_1 \overset{(x,\rho)}{\to} \gamma_2$.*

3. *Let $(y, \mu) \in \Psi_Y$. There is a* discrete move *from γ_1 to γ_2 over (y, μ), if and only if there is a transition $(s_1, y, \delta_1, \delta_2, \theta, \xi, \chi, s_2) \in T_Y$ such that: (i) $\nu_1 \vDash \delta_1$ and $\lambda_1 \vDash \delta_2$; (ii) $\nu_2 = \nu_1 \oplus \theta$ and $\nu_2 \vDash Inv(s_2)$; and (iii) $\mu = \xi(\lambda_1)$ and $\lambda_2 = \chi(\lambda_1)$. We denote this move by $\gamma_1 \overset{(y,\mu)}{\to} \gamma_2$.*

Decorations over and under $\to$ may be dropped, when there is no room for confusion. The movements of a TIOCA can now be specified.

Definition 9 *Let M be a TIOCA. The* movement relation *of M, $\vdash_M$, is a binary relation over $\Psi_M \times \Gamma_M$: $(\langle\sigma\rangle \cdot \psi, \gamma_1) \vdash_M (\psi, \gamma_2)$, with $\sigma \in (\Psi_\Sigma \cup \mathbb{Q}_{\geq})$, if and only if either (i) $\sigma \in \mathbb{Q}_{\geq}$ and $\gamma_1 \xrightarrow[\sigma]{} \gamma_2$; or (ii) $\sigma \in \Psi_\Sigma$ with $\gamma_1 \xrightarrow{\sigma} \gamma_2$.*

If $(\psi, \gamma) \vdash_M (\varepsilon, \varphi)$ we also write $\gamma \Vdash_M^{\psi} \varphi$, or $\gamma \Vdash_M \varphi$ when ψ is not relevant. If $\gamma \Vdash_M^{\psi} \varphi$ we say that we have a *run* from γ to φ over ψ. A configuration φ is *reachable* if there is a run from the initial configuration to φ. A state s is *reachable* if there is a reachable configuration (s, ν, λ), for some clock valuation ν and some context variable valuation λ.

4 Testing timed context models

Testing activities based on formal models usually rely on techniques that lead to unmanageable test suites when applied in practical situations [33]. This scenario worsens when not only time requirements are relevant but also data flow transformations are involved. Furthermore, such methods typically generate a infinite number of test sequences for timed context models since these sequences are extracted from a infinite number of possible distinct runs.

Discretization techniques have been applied in order to keep the number of states of a timed context model under control. However these approaches are usually based on classical clock regions which leads to large grid automata. Moreover, their discretization often imposes a strict relationship between the number of clocks present in the model and the granularities that must be chosen in order to obtain the corresponding grid automata [16, 33]. We deviate from this notion of clock regions to overcome difficulties that arise in practical applications.

The next subsections present a framework to generate test cases based on a new discretization strategy for timed context models and also serves as motivations for the more technical sections to follow.

4.1 Generating test cases for TIOCA

We extend the strategy presented by Bonifacio and Moura [5] to TIOA models. We also use the notion of test purposes [23] when generating test cases, but now with timed models that also include context transformations. Test purposes can focus on smaller parts of complex specifications, thus avoiding generating test cases for the whole system. We also extend the classical synchronous product to capture the coordination between specifications of real-time context systems and test purposes that model important system properties.

Formally, a test purpose is an acyclic TIOCA with special fail and desired states.

Definition 10 *A TIOCA is acyclic if the subjacent graph — defined by its states as nodes and transitions as edges — is acyclic. A test purpose is an acyclic TIOCA with a set of fail states $F \subseteq S$ and a set of desired states $D \subseteq S$, with $F \cap D = \emptyset$.*

The set F of fail states represents undesired, or fail, properties of the system. The desired states in D model desired properties, *i.e.* properties the system should adhere to. Although,

Input: TIOCA M_1, M_2 with $C_1 \cap C_2 = \emptyset$, $V_1 \cap V_2 = \emptyset$.
Output: The synchronous product M_P.
begin
  $C_P \leftarrow C_1 \cup C_2$; $R_P \leftarrow R_1 \cup R_2$; $V_P \leftarrow V_1 \cup V_2$;
  $\Sigma_P \leftarrow \Sigma_1 \cup \Sigma_2$; $RS \leftarrow s_0^P = (s_0^1, s_0^2)$; $Inv_P(s_0^P) \leftarrow Inv_1(s_0^1) \wedge Inv_2(s_0^2)$; $T_P \leftarrow \emptyset$, $HS \leftarrow \emptyset$;
  while $RS \setminus HS \neq \emptyset$ **do**
    choose $s = (s_1, s_2)$ from $RS \setminus HS$;
    move s from RS to HS;
    foreach $a \in X$ **do**
      if $(s_i, a, \delta_i^1, \delta_i^2, \delta_i^3, \theta_i, \kappa_i, s_{i+2}) \in T_i^X$, $i = 1, 2$ **then**
        add (s_3, s_4) to RS;
        let $Inv_P(s_3, s_4) \leftarrow Inv_1(s_3) \wedge Inv_2(s_4)$;
        add $((s_1, s_2), a, \delta_1^1 \wedge \delta_2^1, \delta_1^2 \wedge \delta_2^2, \delta_1^3 \wedge \delta_2^3, \theta_1 \oplus \theta_2, \kappa_1 \oplus \kappa_2, (s_3, s_4))$ to T_P^X;
      end
      if $(s_i, a, \delta_i^1, \delta_i^2, \delta_i^3, \theta_i, \kappa_i, s_{i+2}) \in T_i^X$ *for some* $s_{i+2} \in S$, *and*
      $(s_j, a, \delta_j^1, \delta_j^2, \delta_j^3, \theta_j, \kappa_j, s_{j+2}) \notin T_j^X$ *for all* $s_{j+2} \in S$, *with* $i \neq j$, $i, j \in \{1, 2\}$ **then**
        if i=1 **then** (p, q)=(s_3, s_2)
        else (p, q)=(s_1, s_4);
        add (p, q) to RS;
        let $Inv_P(p, q) \leftarrow Inv_1(p) \wedge Inv_2(q)$;
        add $((s_1, s_2), a, \delta_i^1, \delta_i^2, \delta_i^3, \theta_i, \kappa_i, (p, q))$ to T_P^X;
      end
    end
    foreach $a \in Y$ **do**
      if $(s_i, a, \delta_i^1, \delta_i^2, \theta_i, \xi_i, \chi, s_{i+2}) \in T_i^Y$, $i = 1, 2$ **then**
        add (s_3, s_4) to RS;
        let $Inv_P(s_3, s_4) \leftarrow Inv_1(s_3) \wedge Inv_2(s_4)$;
        add $((s_1, s_2), a, \delta_1^1 \wedge \delta_2^1, \delta_1^2 \wedge \delta_2^2, \theta_1 \oplus \theta_2, \xi_1 \oplus \xi_2, \chi_1 \oplus \chi_2, (s_3, s_4))$ to T_P^Y;
      end
      if $(s_i, a, \delta_i^1, \delta_i^2, \theta_i, \xi_i, \chi_i, s_{i+2}) \in T_i^Y$ *for some* $s_{i+2} \in S$, *and*
      $(s_j, a, \delta_j^1, \delta_j^2, \theta_j, \xi_j, \chi_j, s_{j+2}) \notin T_j^Y$ *for all* $s_{j+2} \in S$, *with* $i \neq j$, $i, j \in \{1, 2\}$, **then**
        if $i = 1$ **then** $(p, q) = (s_3, s_2)$ **else** $(p, q) = (s_1, s_4)$;
        add (p, q) to RS;
        let $Inv_P(p, q) \leftarrow Inv_1(p) \wedge Inv_2(q)$;
        add $((s_1, s_2), a, \delta_i^1, \delta_i^2, \theta_i, \xi_i, \chi_i, (p, q))$ to T_P^Y;
      end
    end
  end
  $S_P \leftarrow HS$;
end

Algorithm 1: Synchronous product for TIOCA.

we require that $F \cap D = \emptyset$, in practice one usually has either $F = \emptyset$ and $D \neq \emptyset$, and we are testing whether the system satisfies *all* desired properties specified by the test purpose or, else, one has $D = \emptyset$ and $F \neq \emptyset$, and we are testing whether the system satisfies *any of* a set of failure properties.

In order to use test purposes to verify system properties, we capture the simultaneous behavior of the specification and the test purpose by taking the product [16, 6] of the specification and the test purpose. Algorithm 1 is an extension of the classical notion of synchronous product, now applied to TIOCA, which require both time and context transformations. Given a TIOCA and a test purpose, the algorithm first pairs the initial states of both TIOCA to get the initial state of the product. Then, product transitions are obtained by searching down for new states and transitions in the given models. A state (s_1, s_2) is a fail state in the product when s_2 is a fail state in the purpose model. Similarly for desired states.

```
1  TiocaTestGeneration(INPUT: state s of a TIOCA grid M_G; OUTPUT: Parameterized timed test
   sequence PTTS;)
2  begin
3  |  if s is a leaf then
4  |  |  Write PTTS;
5  |  else
6  |  |  foreach neighbor, r, of s reached over a transition on σ do
7  |  |  |  PTTS ← {σ} · PTTS;
8  |  |  |  TiocaTestGeneration(r,PTTS);
9  |  |  end
10 |  end
11 end
```

Algorithm 2: The traversal algorithm for TIOCA.

In order to keep the number of states under control, the grid automaton associated with the product is then obtained using the new discretization method. See Algorithm 3. Again, this approach treats both continuous time evolution and context transformations, as specified by the more expressive TIOCA models.

We emphasize that the user has the flexibility to choose granularities for both the context transformations and the continuous time evolution, in the form $g = 1/k$ and $h = 1/l$, respectively, where $k, l \geq 1$. Having ample flexibility when choosing granularities and boundary values is an important advantage. Further, the user can specify any time and context boundary values L and K, respectively, that satisfy the conditions of the new approach, expressed by Theorems 42 and 44, in Section 5. That section also details the discretization method itself.

4.2 Extracting test cases from grids

Once the grid is obtained, *grid sequences* are then extracted by a simple procedure that traverses it. The traversal operation starts at the initial state of the grid and searches down until it finds fail or desired states, depending on the nature of the test. Upon finding one such state, the corresponding grid sequence can be read from the path starting at the initial state. This amounts to a simple recursive traversal procedure. See Algorithm 2. The set of all test sequences is generated by a call in the form $TiocaTestGeneration(s_0, \epsilon)$, where s_0 is the initial state of the grid and ϵ is the empty string. Having the grid sequences, *timed context sequences* can then be derived using the $\widehat{h}$ morphism, described by Definition 43, in Subsection 5.2. The timed context sequences can, in turn, be used to test implementation candidates in order to identify desired or faulty behaviors, according to the test purpose model. More precisely, if $GS_{M,P}$ is the set of all grid sequences extracted from a TIOCA M and a test purpose P, we can then construct the set of timed context sequences $TCS_{M,P} = \{\widehat{h}(\psi) \,|\, \psi \in GS_{M,P}\}$. From the latter we can obtain *test cases* using projection operations.

Definition 11 *Let Σ be an alphabet and $\Upsilon \subseteq \Sigma$. Let $\psi = \langle \sigma_1, \ldots, \sigma_n \rangle$ be a timed word over Σ. The projection of ψ over Υ, $\psi{\downarrow_\Upsilon}$, is the timed word over Υ given by $\phi_1 \cdot \phi_2 \cdot \ldots \cdot \phi_n$, where $\phi_i = \langle \sigma_i \rangle$ if $\sigma_i \in \Psi_\Upsilon \cup \mathbb{Q}_{\geq}$, or ε otherwise.*

Thus, a projection extracts only the actions in the subset of interest, together with timing values. Projecting a timed word over the set of input symbols, X, yields a *test sequence*.

Definition 12 *A* test sequence *for a TIOCA M is a timed word ψ with $\psi = \psi{\downarrow_X}$.*

The desired set of test sequences is then given by $TS_{M,P} = \{\psi{\downarrow_X} \mid \psi \in TCS_{M,P}\}$.

Given a test sequence ψ and a configuration γ, we collect all words that project as ψ over X and have a run from γ.

Definition 13 *Let M be a TIOCA, $\psi \in \Psi_X$ and $\gamma \in \Gamma_M$. A* support *for ψ and γ is a timed word $\rho \in \Psi_M$ such that $\psi = \rho{\downarrow_X}$ and $\gamma \overset{\rho}{\Vdash} \mu$, for some $\mu \in \Gamma_M$.*

A support for an input timed word ψ and a configuration γ is, then, a timed word ρ that drives the TIOCA from γ to some configuration μ, and such that $\rho{\downarrow_X} = \rho$. The set of all supports for ψ and γ will be denoted by $\Lambda_{\psi,\gamma}$. When γ is the initial configuration we may write Λ_ψ.

Now we can define observables.

Definition 14 *Let M be a TIOCA and let $\psi \in \Psi_X$, $\gamma \in \Gamma_M$. A (*observable*)* behavior *of M from γ over ψ is a sequence $\rho{\downarrow_Y}$ where $\rho \in \Lambda_{\psi,\gamma}$. The set of observable behaviors of M from γ over ψ is $\mathcal{O}_{\psi,\gamma}$.*

So, a behavior is just the projection of adequate timed words over the set of output symbols. The conformance testing procedure is carried out by comparing corresponding observables obtained from the specification and the candidate implementation [36, 35].

4.3 Applying test cases to TIOCA

An implementation is investigated by applying stimuli to it and observing its responses [36]. Here, we extend again the TIOA framework [5], now dealing also with context transformations.

For the actual testing of an implementation, we submit a test sequence to it and collect its observable behavior. The test sequence and the observable behavior are composed, thus yielding a *test run*. Note that the occurrence of output actions are not under control of the observer, and so we cannot know in advance the exact instants when outputs will occur. When collecting a test run, the occurrence of an output splits the time lapse between two actions. More precisely, assume that the input test case is $\psi_1 \tau \psi_2$, $\tau \in \mathbb{Q}_{\geq}$, and that the prefix ψ_1 gave rise to a partial test run ψ_1'. If, after a time lapse $\tau_1 \leq \tau$ an output (y, μ) occurs, the run is extended to $\psi_1' \tau_1 (y, \mu)$ and the remaining test input sequence is $(\tau - \tau_1)\psi_2$.

After a test run is collected, we apply it to the corresponding grid automaton. The resulting test verdict depends on the nature of the test. When the test purpose specifies a faulty property, and applying a run to the grid leads to a faulty state, we can announce a *positive test verdict*, that is, a fault was confirmed over the implementation. If no run, over all test runs, leads to a faulty state then the test is deemed *inconclusive*. When the test purpose models a desired property, if all runs applied to the grid terminate at desired states, the test is deemed *positive*, otherwise it is *inconclusive*. When dealing with fault properties, avoiding the full grid construction can be a reasonable alternative, since we only need one hit. In this case, one can use the grid construction algorithm to submit runs to the grid on the fly, thus avoiding the need to keep the full grid description in memory.

We emphasize that grids are extracted from the product of the TIOCA model and the test purpose model. Theorems 42 and 44, in Section 5, provide the necessary formal foundation that guarantees that the grid behavior homomorphically imitates the synchronous TIOCA and purpose behaviors, and conversely.

5 TIOCA discretization

In this section we introduce a new discretization method for TIOCA models. Most other discretization methods lead to the notion of a grid automaton [16, 28, 33] that is obtained using the notions of clock regions and sampling graphs, and reflect only the evolution of time. Another approach [5] deviates from the classical notion of clock regions. Here we extend that approach to deal with context transformations together with time evolution.

In order to keep the number of states of a timed context model under control, bounded values and bounded functions will be used to limit the excursions of discretized TIOCA up to pre-defined boundaries. The discretization will also depend on the notion of adjusted values, defined in next subsection. After introducing the necessary concepts, we show how these can be used to discretize TIOCA models, at Subsection 5.2.

5.1 Bounded and adjusted discretization

We start with the notion of bounded values and bounded functions, as well as the notion of adjusted values.

5.1.1 Bounded values

First, we will need the notion of bounded interpretations.

Definition 15 *Let $L \in \mathbb{Q}_{\geq}$ be a bound. Let $x \in \mathbb{Q}_{\geq}$, let A be a set of variables and let $\alpha \in [A \curvearrowright \mathbb{Q}_{\geq}]$. Then,*

1. *The L-bounded x value, denoted x_L, is: (i) x, if $x \leq L$; or (ii) $\lfloor L \rfloor + \lceil x \rceil$, otherwise.*
2. *The L-bounded α function, denoted α_L, is given by $\alpha_L(a) = (\alpha(a))_L$, for $a \in A$.*

In practical applications the bound L can be chosen based on physical characteristics of the measuring equipment. Note that we could have specified a different bound L_a for each $a \in A$. By so doing, one might gain extra flexibility and, as will be seen, might result in models with a more manageable state-spaces. However, in order to keep the notation uncluttered we will consider a single bound L for all variables. It is a simple matter to generalize the following results to cover the case when some bounds are distinct.

Next, we list a few simple facts that will be used later.

Fact 16 *Let $L \in \mathbb{Q}_{\geq}$ be a bound and let $x \in \mathbb{Q}_{\geq}$ be a value. Then:*

1. *$\lceil x_L \rceil = \lceil x \rceil$ and $x_L \leq x$.*

2. *If* $x < \lfloor L \rfloor + 1$ *then* $x_L = x$.

3. $x_L < \lfloor L \rfloor + 1$.

4. $(x_L)_L = x_L$.

Proof We start with item 1.

The equality is immediate from Definition 15. Now, if $x \leq L$, then $x_L = x$ and so $x_L \leq x$. When $x > L$, then $x_L = \lfloor L \rfloor + \lceil x \rceil$. From Fact 1, we get $\lfloor x \rfloor \geq \lfloor L \rfloor$. Then, $x_L \leq \lfloor x \rfloor + \lceil x \rceil = x$, and we are done.

We now prove item 2.

If $x \leq L$ then we know that $x_L = x$. If $x > L$ then we have $\lfloor x \rfloor \geq \lfloor L \rfloor$, using Fact 1. From the hypothesis we get $\lfloor x \rfloor < \lfloor L \rfloor + 1$, and so $\lfloor x \rfloor \leq \lfloor L \rfloor$. Then, $\lfloor x \rfloor = \lfloor L \rfloor$ and we get $x_L = \lfloor L \rfloor + \lceil x \rceil = \lfloor x \rfloor + \lceil x \rceil = x$, as desired.

Now we follow with item 3.

If $x < \lfloor L \rfloor + 1$ then $x_L = x$, using item (2). So, $x_L < \lfloor L \rfloor + 1$. If $x \geq \lfloor L \rfloor + 1$ then $x > L$, and we get $x_L = \lfloor L \rfloor + \lceil x \rceil$. Then, $x_L < \lfloor L \rfloor + 1$, since $\lceil x \rceil < 1$.

Lastly we prove item 4.

From item (3), we know that $x_L < \lfloor L \rfloor + 1$. Now, using item (2) we get $(x_L)_L = x_L$.

∎

5.1.2 Bounded functions

The next proposition states that the L-bound of a sum of terms is the same as the L-bound of the sum of the L-bounds of the individual terms.

Proposition 17 *Let* $L \in \mathbb{Q}_{\geq}$ *be a bound and let* x, $y \in \mathbb{Q}_{\geq}$. *Then*

1. $(x + y)_L = (x + y_L)_L$.

2. $(x + y)_L = (x_L + y_L)_L$.

3. *If* $x' = x$ *or* $x' = x_L$, *and* $y' = y$ *or* $y' = y_L$, *then* $(x + y)_L = (x' + y')_L$.

Proof We start with item 1.

There are two cases.

Case 1: $y < \lfloor L \rfloor + 1$. Then, by Fact 16, $y = y_L$ and we are done.

Case 2: $y \geq \lfloor L \rfloor + 1$. Then, $y_L = \lfloor L \rfloor + \lceil y \rceil$.

Also, $x + y \geq \lfloor L \rfloor + 1$ and so $(x + y)_L = \lfloor L \rfloor + \lceil x + y \rceil$. There are two subcases.

Case 2A: $x + y_L < \lfloor L \rfloor + 1$. Then, by Fact 16, $(x + y_L)_L = x + y_L = x + \lfloor L \rfloor + \lceil y \rceil$. But $x + y_L < \lfloor L \rfloor + 1$ and so $x + \lfloor L \rfloor + \lceil y \rceil < \lfloor L \rfloor + 1$, and we have $x + \lceil y \rceil < 1$. Then $x = \lceil x \rceil$ and we get $\lceil x \rceil + \lceil y \rceil < 1$, and so $\lceil \lceil x \rceil + \lceil y \rceil \rceil = \lceil x \rceil + \lceil y \rceil$. Using Fact 1 we get $\lceil x + y \rceil = \lceil x \rceil + \lceil y \rceil = x + \lceil y \rceil$. This gives $(x + y)_L = (x + y_L)_L$, as desired.

CASE 2B: $x + y_L \geq \lfloor L \rfloor + 1$. Then $(x + y_L)_L = \lfloor L \rfloor + \lceil x + y_L \rceil$.
But, by Fact 1, $\lceil x + y_L \rceil = \lceil x + \lfloor L \rfloor + \lceil y \rceil \rceil = \lceil x + \lceil y \rceil \rceil$. And, by Fact 1 again, $\lceil x+y \rceil = \lceil x+\lfloor y \rfloor + \lceil y \rceil \rceil = \lceil x+\lceil y \rceil \rceil$. Then $(x+y_L)_L = \lfloor L \rfloor + \lceil x+y \rceil = (x+y)_L$, as desired.

Now we prove item 2.

From item (1) we get $(x + y)_L = (x + y_L)_L$. From item (1) again, we get $(x + y_L)_L = (y_L+x)_L = ((y_L+x_L)_L)_L = ((x_L+y_L)_L)_L$. Now, from Fact 16, $((x_L+y_L)_L)_L = (x_L+y_L)_L$. Then, $(x + y)_L = (x_L + y_L)_L$.

Finally we prove item 3.

If $x' = x$ and $y' = y$ we are done. If $x' = x$ and $y' = y_L$ use item (1). Similarly, if $x' = x_L$ and $y' = y$. Finally, if $x' = x_L$ and $y' = y_L$, use item (2). ∎

These results can be extended to larger sums, as stated in Propositions 18 and 19.

Proposition 18 *Let $L \in \mathbb{Q}_{\geq}$ and let $x^i \in \mathbb{Q}_{\geq}$, for $i = 1, \ldots, n$, with $n \geq 1$. Let $y^i = x^i$ or $y^i = (x^i)_L$, for $i = 1, \ldots, n$. Then $\left[\sum_{i=1}^n x^i\right]_L = \left[\sum_{i=1}^n y^i\right]_L$.*

Proof We denote $(x^i)_L$ by x^i_L and proceed inductively on n.

When $n = 1$, if $y^1 = x^1$ we are done, and when $y^1 = x^1_L$ we use Fact 16.

Now, assume the result holds for some $n \geq 1$. Using Proposition 17, we get $\left[\sum_{i=1}^{n+1} x^i\right]_L = \left[(\sum_{i=1}^n x^i) + x^{n+1}\right]_L = \left[(\sum_{i=1}^n x^i)_L + y^{n+1}\right]_L$. Using the induction hypothesis and Proposition 17, we get $\left[\sum_{i=1}^{n+1} x^i\right]_L = \left[(\sum_{i=1}^n y^i)_L + y^{n+1}\right]_L = \left[(\sum_{i=1}^n y^i) + y^{n+1}\right]_L = \left[\sum_{i=1}^{n+1} y^i\right]_L$. ∎

Proposition 19 *Let $L \in \mathbb{Q}_{\geq}$ be a bound and let $W = \{w_1, \ldots, w_n\}$ be a set of variables, with $n \geq 1$. Also let $k_i \geq 0$ be integer constants, $i = 1, \ldots, n$. Take $\alpha \in [W \to \mathbb{Q}_{\geq}]$. Then $\left[\sum_{i=1}^n k_i \alpha(w_i)\right]_L = \left[\sum_{i=1}^n k_i \alpha_L(w_i)\right]_L$.*

Proof Proposition 18 gives $\left[\sum_{i=1}^n k_i\alpha(w_i)\right]_L = \left[\sum_{i=1}^n \left(k_i\alpha(w_i)\right)_L\right]_L$. So, $\left[k_i\alpha(w_i)\right]_L = \left[\sum_{j=1}^{k_i} \alpha(w_i)\right]_L$. Again, from Proposition 18 we get $\left[k_i\alpha(w_i)\right]_L = \left[\sum_{j=1}^{k_i}(\alpha(w_i))_L\right]_L = \left[k_i\alpha_L(w_i)\right]_L$. Hence, $\left[\sum_{i=1}^n k_i\alpha(w_i)\right]_L = \left[\sum_{i=1}^n \left(k_i\alpha_L(w_i)\right)_L\right]_L$. Finally, using Proposition 18 again, we get $\left[\sum_{i=1}^n k_i\alpha(w_i)\right]_L = \left[\sum_{i=1}^n k_i\alpha_L(w_i)\right]_L$, as desired. ∎

Next, we treat displaced and overruled interpretations.

Proposition 20 *Let C be a set of variables, L a positive integer, $\tau \in \mathbb{Q}_{\geq}$ and $\nu, \theta \in [C \to \mathbb{Q}_{\geq}]$. Then $(\nu \oplus \theta)_L = (\nu_L \oplus \theta)_L$ and $(\nu + \tau)_L = (\nu_L + \tau)_L$.*

Proof It suffices to show that $(\nu \oplus \theta)_L(c) = (\nu_L \oplus \theta)_L(c)$, for all $c \in C$. If $c \notin \mathrm{dom}(\theta)$, we have $(\nu \oplus \theta)_L(c) = \nu_L(c)$ and $(\nu_L \oplus \theta)_L(c) = (\nu_L)_L(c) = \nu_L(c)$, using Fact 16. If $c \in \mathrm{dom}(\theta)$, we have $(\nu \oplus \theta)_L(c) = \theta_L(c)$ and $(\nu_L \oplus \theta)_L(c) = \theta_L(c)$. ∎

Function addition takes its usual meaning. The next lemma says that satisfiability of variable conditions is oblivious to L-bounding operands in function additions or overruling.

Lemma 21 *Let C be a set of variables, $I \in \Phi_C$ and L a positive integer greater than all constants occurring in I. Let $\alpha_1, \alpha_2 \in [C \to \mathbb{Q}_{\geq}]$ and $\nu, \theta \in [C \rightharpoonup \mathbb{Q}_{\geq}]$.*

1. *Then $\alpha_1 + \alpha_2 \vDash I$ iff $\beta_1 + \beta_2 \vDash I$, where $\beta_i = \alpha_i$ or $\beta_i = (\alpha_i)_L$, $i = 1, 2$.*
2. *If $\nu \oplus \theta \vDash I$ then $\nu_L \oplus \theta \vDash I$.*

Proof We start with item 1.

If I is `true` we are done. Assume now that I is not `true`.

From Fact 1 we know that $\beta_i(c) \leq \alpha_i(c)$, for all $c \in C$ and all $i \in \{1, 2\}$.

We treat first the four simple cases when I is $(c \leq \tau)$, $(c \geq \tau)$, $\neg(c \leq \tau)$ or $\neg(c \geq \tau)$.

Case 1: I is $(c \leq \tau)$, for some $c \in C$ and some $\tau \in \mathbb{Q}_{\geq}$.

If $\alpha_1 + \alpha_2 \vDash I$ then $(\alpha_1 + \alpha_2)(c) = \alpha_1(c) + \alpha_2(c) \leq \tau$. Then $\beta_1(c) + \beta_2(c) \leq \tau$ and so $\beta_1 + \beta_2 \vDash I$.

For the converse, let $\beta_1(c) + \beta_2(c) \leq \tau$. From the hypothesis, $\beta_1(c) + \beta_2(c) \leq L$. If $\alpha_1(c) > L$ then either (i) $\beta_1(c) = \alpha_1(c) > L$, or (ii) $\beta_1(c) = (\alpha_i(c))_L = \lfloor L \rfloor + \lceil \alpha_1(c) \rceil = L + \lceil \alpha_1(c) \rceil$, since L is an integer. Both cases contradict $\beta_1(c) + \beta_2(c) \leq L$.

Similarly, we cannot have $\alpha_2(c) > L$. Hence, $\alpha_i(c) \leq L$ gives $\beta_i(c) = \alpha_i(c)$, for $i = 1, 2$. Then, since $\beta_1(c) + \beta_2(c) \leq \tau$, we get $\alpha_1(c) + \alpha_2(c) \leq \tau$, and so $\alpha_1 + \alpha_2 \vDash I$.

Case 2: I is $(c \geq \tau)$, for some $c \in C$ and some $\tau \in \mathbb{Q}_{\geq}$.

First, let $\beta_1 + \beta_2 \vDash I$. Then $\beta_1(c) + \beta_2(c) \geq \tau$ and so $\alpha_1(c) + \alpha_2(c) \geq \tau$, since $\alpha_i(c) \geq \beta_i(c)$, $i = 1, 2$. Then, $\alpha_1 + \alpha_2 \vDash I$.

For the converse, assume $\alpha_1(c) + \alpha_2(c) \geq \tau$. If $\alpha_1(c) \geq L+1$, then $\beta_1(c) = \lfloor L \rfloor + \lceil \alpha_1(c) \rceil = L + \lceil \alpha_1(c) \rceil$, since L is an integer. Thus, $\beta_1(c) \geq \tau$, since $L > \tau$ from the hypothesis. Hence $\beta_1(c) + \beta_2(c) \geq \tau$ and so $\beta_1 + \beta_2 \vDash I$.

Similarly, when $\alpha_2(c) \geq L + 1$ the result also holds.

Now let $\alpha_i(c) < L + 1 = \lfloor L \rfloor + 1$ for $i = 1, 2$. From Fact 16, we get $\beta_i(c) = \alpha_i(c)$ and so $\beta_1(c) + \beta_2(c) \geq \tau$, and again $\beta_1 + \beta_2 \vDash I$.

Case 3: I is $\neg(c \leq \tau)$, for some $c \in C$ and some $\tau \in \mathbb{Q}_{\geq}$. Equivalently, we have $\alpha_1 + \alpha_2 \vDash (c > \tau)$. We proceed as in Case 2.

Case 4: I is $\neg(c \geq \tau)$, for some $c \in C$ and some $\tau \in \mathbb{Q}_{\geq}$. Equivalently, we have $\alpha_1 + \alpha_2 \vDash (c < \tau)$. We proceed as in Case 1.

Let $n \geq 0$ be the number of propositional connectives occurring in I. We proceed by induction on n.

Basis: $n = 0$. The result holds by Cases 1 and 2.

Induction step: assume the result holds for all I with at most n propositional connectives, where $n \geq 0$. Now, take some $I \in \Phi_C$ with $n + 1$ propositional connectives. We have two cases:

Case I-1: I is $\delta_1 \wedge \delta_2$.

Since δ_i has at most n propositional connectives, $i = 1, 2$, from the induction hypothesis we get $\alpha_1 + \alpha_2 \vDash \delta_i$ iff $\beta_1 + \beta_2 \vDash \delta_i$, for $i = 1, 2$. Then, clearly, $\alpha_1 + \alpha_2 \vDash I$ iff $\beta_1 + \beta_2 \vDash I$.

Case I-2: I is $\neg\delta$.

Then δ has $n \geq 0$ propositional connectives. When $n = 0$, δ is $(c \leq \tau)$ or $(c \geq \tau)$, and the result follows by Cases 3 and 4, respectively.

Assume now that $n \geq 1$. We have two sub-cases:

Case I-2A: δ is $\neg\delta_1$ and δ_1 has $n-1$ propositional connectives. Then $\neg\delta$ is equivalent to $\neg\neg\delta_1$, that is, I is equivalent to δ_1. We can use the induction hypothesis and conclude that $\alpha_1 + \alpha_2 \vDash I$ iff $\beta_1 + \beta_2 \vDash I$.

Case I-2B: δ is $(\delta_1 \wedge \delta_2)$. So, I is equivalent to $(\neg\delta_1) \vee (\neg\delta_2)$. Since δ has n propositional connectives, δ_i has at most $(n-1)$ propositional connectives, that is, $\neg\delta_i$ has at most n propositional connectives, for $i = 1, 2$. From the induction hypothesis, $\alpha_1 + \alpha_2 \vDash \neg\delta_i$ iff $\beta_1 + \beta_2 \vDash \neg\delta_i$. Thus, $\alpha_1 + \alpha_2 \vDash (\neg\delta_1) \vee (\neg\delta_2)$ iff $\alpha_1 + \alpha_2 \vDash \neg\delta_i$ for some $i \in \{1, 2\}$ iff $\beta_1 + \beta_2 \vDash \neg\delta_i$ for some $i \in \{1, 2\}$ iff $\beta_1 + \beta_2 \vDash (\neg\delta_1) \vee (\neg\delta_2)$, completing the proof.

We now prove item 2.

When I is `true` we are done. Now, assume that I is not `true`.

Now we treat the four cases when I is $(c \leq \tau)$, $(c \geq \tau)$, $\neg(c \leq \tau)$ or $\neg(c \geq \tau)$, where $c \in C$ and $\tau \in \mathbb{Q}_{\geq}$. If $c \notin \mathrm{dom}(\theta)$, we get $(\nu \oplus \theta)(c) = \nu(c)$ and so $\nu \vDash I$. From item 1, we get $\nu_L \vDash I$. Since $(\nu_L \oplus \theta)(c) = \nu_L(c)$ we conclude that $\nu_L \oplus \theta \vDash I$. If $c \in \mathrm{dom}(\theta)$, we get $(\nu \oplus \theta)(c) = \theta(c) = (\nu_L \oplus \theta)(c)$. Then, $\nu_L \oplus \theta \vDash I$.

We proceed by induction on the number $n \geq 0$ of connectives occurring in I.

Basis: $n = 0$. The result follows by the first two cases discussed above.

Induction step: assume the result holds for all I with at most $n \geq 0$ connectives.

Now, take some $I \in \Phi_C$ with $n+1$ propositional connectives. We have two cases:

Case I-1: I is $\delta_1 \wedge \delta_2$.

We have $\nu \oplus \theta \vDash \delta_i$, $i = 1, 2$. But δ_i has $n-1$ propositional connectives, and the induction hypothesis gives $\nu_L \oplus \theta \vDash \delta_i$, for $i = 1, 2$. Hence, $\nu_L \oplus \theta \vDash I$.

Case I-2: I is $\neg\delta$.

Then δ has $n \geq 0$ propositional connectives. When $n = 0$, δ is $(c \leq \tau)$ or $(c \geq \tau)$, and the result holds by third and forth cases discussed above.

Assume now that $n \geq 1$. We have two sub-cases:

Case I-2A: δ is $\neg\delta_1$ and δ_1 has $n-1$ propositional connectives. Then $\neg\delta$ is equivalent to $\neg\neg\delta_1$, that is, to δ_1. Then, from the original hypothesis, $\nu \oplus \theta \vDash \delta_1$. By the induction hypothesis, $\nu_L \oplus \theta \vDash \delta_1$, that is $\nu_L \oplus \theta \vDash \neg\neg\delta_1$. Then $\nu_L \oplus \theta \vDash I$.

Case I-2B: δ is $(\delta_1 \wedge \delta_2)$, that is I is equivalent to $(\neg\delta_1) \vee (\neg\delta_2)$. Hence, $\nu \oplus \theta \vDash \neg\delta_1$, or $\nu \oplus \theta \vDash \neg\delta_2$. Since δ has n propositional connectives, δ_i has at most $(n-1)$ propositional connectives, that is, $\neg\delta_i$ has at most n propositional connectives, for $i = 1, 2$. Using the induction hypothesis, we have $\nu_L \oplus \theta \vDash \neg\delta_1$ or $\nu_L \oplus \theta \vDash \neg\delta_2$. So, $\nu_L \oplus \theta \vDash \neg(\delta_1 \wedge \delta_2)$, that is $\nu_L \oplus \theta \vDash I$. ∎

5.1.3 Adjusted values

When discretizing a model, we will need to choose proper granularities. All values in the discretized model will be *adjusted*, or integer multiples of the granularity. The notion of adjusted values is defined next.

Definition 22 *Let $g \in \mathbb{Q}_{\geq}$ and C a set of variables. A value $x \in \mathbb{Q}_{\geq}$ is* g-adjusted *if it is an integer multiple of g. A condition $\delta \in \Phi_C$ is* g-adjusted *if all constants occurring in δ are g-adjusted. An interpretation $\nu \in [C \curvearrowright \mathbb{Q}_{\geq}]$ is* g-adjusted *if $\nu(c)$ is g-adjusted, for all $c \in dom(\nu)$.*

Two simple propositions about adjusted values follow.

Proposition 23 *Let $g = 1/k$ with $k \geq 1$, and let $\ell \in \mathbb{Q}_{\geq}$. Then ℓ is a g-adjusted value if both $\lfloor \ell \rfloor$ and $\lceil \ell \rceil$ are g-adjusted values.*

Proof Assume that $\lfloor \ell \rfloor = mg$ and $\lceil \ell \rceil = ng$, where m and n are non-negative integers. Then $\ell = (m+n)g$ and so ℓ is also a g-adjusted value. Conversely, assume that $\ell = mg$ for some non-negative integer m. Then $mg = (\lfloor \ell \rfloor g)/g + \lceil \ell \rceil$ and so $\lceil \ell \rceil = (m - \lfloor \ell \rfloor/g)g = (m - k\lfloor \ell \rfloor)g$. Since $(m - k\lfloor \ell \rfloor)$ is an integer, $\lceil \ell \rceil$ is also g-adjusted. Also, clearly, $\lfloor \ell \rfloor = \ell - \lceil \ell \rceil$ and so $\lfloor \ell \rfloor$ is g-adjusted since ℓ and $\lceil \ell \rceil$ are g-adjusted. ∎

Proposition 24 *Let $g = 1/k$ with $k \geq 1$. Let C be a set of variables. Let $L, \ell \in \mathbb{Q}_{\geq}$ and ν, $\eta \in [C \curvearrowright \mathbb{Q}_{\geq}]$ be g-adjusted. Then ν_L, $\nu + \ell$ and $\nu \oplus \eta$ are g-adjusted.*

Proof Let $c \in C$. From Definition 15, we get $\nu_L(c) = \nu(c)$ or $\nu_L(c) = \lfloor L \rfloor + \lceil \nu(c) \rceil$. From Proposition 23, we know that $\lfloor L \rfloor$ and $\lceil \nu(c) \rceil$ are g-adjusted. So, clearly, $\nu_L(c)$ is g-adjusted for all $c \in C$, and the result follows.

Assume that $\nu(c) \in \mathrm{dom}(\nu)$. Then, $(\nu + \ell)(c) = \nu(c) + \ell$, which is, clearly, a g-adjusted value. Thus, by Definition 22, $\nu + \ell$ is also g-adjusted.

Now, when $c \in \mathrm{dom}(\nu \oplus \eta)$ there are two cases. If $\eta(c) \in \mathrm{dom}(\eta)$, then $(\nu \oplus \eta)(c) = \eta(c)$ is g-adjusted. When $c \in (\mathrm{dom}(\nu) - \mathrm{dom}(\eta))$ then $(\nu \oplus \eta)(c) = \nu(c)$ is also g-adjusted. Clearly, $\nu \oplus \eta$ is a g-adjusted clock interpretation. ∎

Any set of L-bounded interpretations is finite, provided that L is properly adjusted.

Lemma 25 *Let C be a set of variables, and let $g = 1/k$ with k a positive integer. Let $R \subseteq [C \curvearrowright \mathbb{Q}_{\geq}]$ be a set of g-adjusted interpretations, and let $L \in \mathbb{Q}_{\geq}$ be g-adjusted. Let $R_L = \{\nu_L \mid \nu \in R\}$. Then $|R_L| \leq (k\lfloor L \rfloor + k)^{|C|}$.*

Proof Consider some $\nu_L \in R_L$ and some $c \in C$. Let $t = \nu_L(c)$. By Proposition 24, t is always g-adjusted. Moreover, $t < \lfloor L \rfloor + 1$, by Fact 16. But $\lfloor L \rfloor = ng$, for some non-negative integer n, and so $\lfloor L \rfloor + 1 = ng + (1/g)g = (n+k)g$. Hence, t can have at most $n+k$ distinct g-adjusted values (counting from zero).

We conclude that any clock $c \in C$ can be mapped to at most $n + k$ distinct g-adjusted values, by ν_L bounded interpretations. Therefore, there are at most $(n + k)^{|C|}$ distinct ν_L interpretations. Since $n = \lfloor L \rfloor / g = k\lfloor L \rfloor$, the result follows. ∎

If we were using a g-adjusted bound L_c for each variable c in C, the bound would be $|R_L| \leq \prod_{c \in C} (k\lfloor L_c \rfloor + k)$.

5.2 Discretizing TIOCA

In this section we present the new discretization technique for TIOCA models. We will use a clock boundary $L \in \mathbb{Q}_{\geq}$. Recall Definition 15. Note that we could have used a distinct L_c value as a boundary for each clock $c \in C$. Usually, by so doing, one gets smaller discrete models but, on the other hand, this overloads the notation. Instead, we shall use the same L value for all clock boundaries and note that it is easy to generalize all results to the case when distinct L_c values are used. For the same reason, we shall use a common $K \in \mathbb{Q}_{\geq}$ value to bound context variable values.

Before defining the grid, we will need that mappings in H_y^P and in H_y^T be of a special linear form that we now make precise. Recall Definition 6 and the discussion preceding it.

Definition 26 *Let V be a set of variables and ξ map $[V \to \mathbb{Q}_{\geq}]$ into $[V \curvearrowright \mathbb{Q}_{\geq}]$. Then ξ is* fully linear *if for each $\lambda \in [V \to \mathbb{Q}_{\geq}]$ and $v \in dom(\xi(\lambda))$ there is an integer constant b, and a set of integer constants $\{a_w \,|\, w \in V\}$ with $\xi(\lambda)(v) = \left[\sum_{w \in V} a_w \lambda(w)\right] + b$.*

Definition 27 *Let V be a set of variables and R a set of parameters. Let κ map $[R \to \mathbb{Q}_{\geq}] \times [V \to \mathbb{Q}_{\geq}]$ into $[V \curvearrowright \mathbb{Q}_{\geq}]$. Then κ is* fully bi-linear *if there are fully linear mappings ξ_1, from $[V \to \mathbb{Q}_{\geq}]$ into $[V \curvearrowright \mathbb{Q}_{\geq}]$, and ξ_2, from $[R \to \mathbb{Q}_{\geq}]$ into $[V \curvearrowright \mathbb{Q}_{\geq}]$, such that $\kappa(\rho, \lambda) = \xi_1(\lambda) + \xi_2(\rho)$, for all $\rho \in [R \to \mathbb{Q}_{\geq}]$ and all $\lambda \in [V \to \mathbb{Q}_{\geq}]$.*

We restrict mappings in $H_y^P \cup H_y^T$ to be fully linear and mappings $\kappa \in F_x$ to be fully bi-linear. Fully linear and fully bi-linear mappings keep the values of context variables and the continuous time evolution under control in the model. They also will allow for a homomorphic simulation of the original model by the corresponding grid, and conversely.

Next we present some facts that will support proofs of the relationship between a TIOCA and the corresponding grid automaton.

Fact 28 *Let V be a set of variables and let $K \geq 0$. Let ξ be a fully linear mapping from $[V \to \mathbb{Q}_{\geq}]$ into $[V \curvearrowright \mathbb{Q}_{\geq}]$. Then $\xi_K(\lambda) = \xi_K(\lambda_K)$, for all $\lambda \in [V \to \mathbb{Q}_{\geq}]$.*

Proof It suffices to show that $\xi_K(\lambda)(v) = \xi_K(\lambda_K)(v)$, for all v in the domain of $\xi(\lambda)$.

Definition 26 gives constants b and a_w, for all $w \in V$, such that $\xi_K(\lambda)(v) = (\xi(\lambda)(v))_K = \left[\sum_{w \in V} a_w \lambda(w) + b\right]_K = \left[\left[\sum_{w \in V} a_w \lambda(w)\right]_K + b\right]_K$, where we used Proposition 18. Now, using Proposition 19, we get $\left[\sum_{w \in V} a_w \lambda(w)\right]_K = \left[\sum_{w \in V} a_w \lambda_K(w)\right]_K$. Then using Proposition 18 again, we have $\xi_K(\lambda)(v) = \left[\left[\sum_{w \in V} a_w \lambda_K(w)\right]_K + b\right]_K = \left[\sum_{w \in V} a_w \lambda_K(w) + b\right]_K$ $= (\xi(\lambda_K)(v))_K = \xi_K(\lambda_K)(v)$.

∎

A similar result holds for bi-linear mappings.

Fact 29 *Let V be a set of variables, R a set of parameters and K a positive integer. Let κ be a fully bi-linear mapping from $[R \to \mathbb{Q}_{\geq}] \times [V \to \mathbb{Q}_{\geq}]$ into $[V \curvearrowright \mathbb{Q}_{\geq}]$. Then $\kappa_K(\rho, \lambda) = \kappa_K(\rho_K, \lambda_K)$, for all $\rho \in [R \to \mathbb{Q}_{\geq}]$ and all $\lambda \in [V \to \mathbb{Q}_{\geq}]$.*

Proof Definition 27 gives two mappings, namely, ξ from $[V \to \mathbb{Q}_{\geq}]$ into $[V \curvearrowright \mathbb{Q}_{\geq}]$, and χ from $[R \to \mathbb{Q}_{\geq}]$ into $[V \curvearrowright \mathbb{Q}_{\geq}]$, such that $\kappa(\rho, \lambda) = \xi(\lambda) + \chi(\rho)$.

Take $\rho \in [R \to \mathbb{Q}_{\geq}]$ and $\lambda \in [V \to \mathbb{Q}_{\geq}]$. It suffices to show that $\kappa_K(\rho, \lambda)(v) = \kappa_K(\rho_K, \lambda_K)(v)$, for all v in the domain of $\kappa_K(\rho, \lambda)$. Let v be in the domain of $\kappa(\rho, \lambda)$. Then $\kappa(\rho, \lambda)(v) = \xi(\lambda)(v) + \chi(\rho)(v)$. Using Proposition 18, we have $\kappa_K(\rho, \lambda)(v) = \big[\kappa(\rho, \lambda)(v)\big]_K = \big[\chi(\rho)(v) + \xi(\lambda)(v)\big]_K = \Big[\big[\chi(\rho)(v)\big]_K + \big[\xi(\lambda)(v)\big]_K\Big]_K$. Now, using Fact 28, we may write $\big[\xi(\lambda)(v)\big]_K = \xi_K(\lambda)(v) = \xi_K(\lambda_K)(v)$. By the same reasoning, $\big[\chi(\rho)(v)\big]_K = \chi_K(\rho)(v) = \chi_K(\rho_K)(v)$. Then, $\kappa_K(\rho, \lambda)(v) = \big[\chi_K(\rho_K)(v) + \xi_K(\lambda_K)(v)\big]_K$ and, using Proposition 18 again, $\kappa_K(\rho, \lambda)(v) = \big[\chi(\rho_K)(v) + \xi(\lambda_K)(v)\big]_K = \big[\kappa(\rho_K, \lambda_K)(v)\big]_K = \kappa_K(\rho_K, \lambda_K)(v)$. ∎

For the sake of shortening statements, we stipulate that the following assumption is in order from now on.

Assumption 30 *Let M be a TIOCA. Mappings in F_x are fully bi-linear mappings, for all $x \in X$, and mappings in $H_y^P \cup H_y^T$ are fully linear mappings, for all $y \in Y$.*

Next, we need grid units. Clock and context grid units will be in the form $g = 1/k$ and $h = 1/\ell$, with k and ℓ positive integers, respectively. Note that this formulation gives rise to a rich choice of possible grid units. Now, assume that λ is a h-adjusted context valuation. Then, because all $\chi \in H_y^P$ and all $\xi \in H_y^T$ are fully linear, it is easy to see that both $\chi(\lambda)$ and $\xi(\lambda)$ are also h-adjusted. Similarly, because each $\kappa \in F_x$ is fully bi-linear, then $\kappa(\rho, \lambda)$ is h-adjusted when λ and ρ are h-adjusted.

By a reasoning analogous to the proof of Lemma 25, we get that $(\ell\lfloor K \rfloor + \ell)^{|V|}$ is an upper bound on the size of any set of K-bounded context variable interpretations. Note that, had we used distinct h-adjusted bounds K_v for each context variable v, then $\prod_{v \in V} (\ell\lfloor K_v \rfloor + \ell)$ would be a proper upper bound. Similar results can be also derived for any set of input parameter valuations.

Now, we can say when a TIOCA is adjusted. We simply require that all constants occurring in the definition of the TIOCA be properly adjusted.

Definition 31 *A TIOCA M is* $[g, h]$-adjusted *iff for all $(s, x, \delta_1, \delta_2, \delta_3, \theta, \kappa, r) \in T_X$, and $(s, y, \delta_1, \delta_3, \theta, \xi, \chi, r) \in T_Y$ we have that δ_1, θ are g-adjusted, and δ_2, δ_3 are h-adjusted. Further, for all $s \in S$, $Inv(s)$ is g-adjusted.*

The next simple proposition states that applying context update functions over h-adjusted parameter valuations and h-adjusted context variable valuations, always result in h-adjusted context variable valuations.

Proposition 32 *If M is a $[g, h]$-adjusted TIOCA then $\xi(\lambda)$ and $\kappa(\rho, \lambda)$ are h-adjusted, for all $y \in Y$, $x \in X$, $\kappa \in F_x$, $\xi \in H_y^P \cup H_y^T$, $\lambda \in [V \to \mathbb{Q}_{\geq}]$ and $\rho \in [R_x \to \mathbb{Q}_{\geq}]$.*

Proof Consider the first statement. Take $\lambda \in [V \to \mathbb{Q}_{\geq}]$ and $v \in V$, such that v is in the domain of $\xi(\lambda)$. It suffices to show that $\xi(\lambda)(v)$ is h-adjusted. By Assumption 30, we know that ξ is fully linear and so, by Definition 26, we can write $\xi(\lambda)(v) = \big[\sum_{w \in V} a_w \lambda(w)\big] + b$,

where a_w and b are integer constants. Since λ is h-adjusted and a_w is an integer, we get that each term $a_w\lambda(w)$ is also h-adjusted. Then, clearly, the sum $\sum_{w\in V} a_w\lambda(w)$ is also h-adjusted. Finally, since b is an integer, and so is also h-adjusted, the final sum is h-adjusted too.

Now consider the second statement. Take $\rho \in [R_x \to \mathbb{Q}_{\geq}]$, for some $x \in X$, and $\lambda \in [V \to \mathbb{Q}_{\geq}]$. It suffices to show that $\kappa(\rho,\lambda)(v)$ is h-adjusted, for all v is in the domain of $\kappa(\rho,\lambda)$. Again, by Assumption 30 we know that κ is fully bi-linear. Then, from Definition 27 we get two fully linear mappings, ξ and χ, such that $\kappa(\rho,\lambda)(v) = \xi(\rho)(v) + \chi(\lambda)(v)$. By the reasoning just given above, we get that $\xi(\rho)(v)$ and $\chi(\lambda)(v)$ are both h-adjusted. Then, clearly, $\kappa(\rho,\lambda)(v)$ is also h-adjusted. ∎

A timed word is adjusted if all its time instants and all its parameter values are properly adjusted.

Definition 33 *A timed word $\langle\sigma_1,\sigma_2,\ldots,\sigma_n\rangle$ is $[g,h]$-adjusted iff σ_i is g-adjusted when $\sigma_i \in \mathbb{Q}_{\geq}$, and ρ is h-adjusted when $\sigma_i = (z,\rho) \in \Psi_\Sigma$, all $i \in \{1,\ldots,n\}$.*

The set of all $[g,h]$-adjusted timed words over Σ and R will be denoted by $\Psi_{[g,h]}$.

We will also use the notion of $[g,h]$-reachability.

Definition 34 *Let M be a $[g,h]$-adjusted TIOCA, $s \in S$, $\nu \in [C \to \mathbb{Q}_{\geq}]$, $\lambda \in [V \to \mathbb{Q}_{\geq}]$. Then (s,ν,λ) is $[g,h]$-reachable if $(s_0,\nu_0,\lambda_0) \Vdash_M^{\psi} (s,\nu,\lambda)$ for some $\psi \in \Psi_{[g,h]}$.*

Next we define the grid automaton corresponding to a TIOCA discretization.

Definition 35 *Let M be a $[g,h]$-adjusted TIOCA. The grid automaton associated with M is the labeled transition system constructed by Algorithm 3.*

Algorithm 3 traverses the original TIOCA and, for each of its transitions, it checks whether a (input or output) symbol labeled, or a delay labeled, equivalent transition should be added to the grid. Although it resembles similar constructions appearing in earlier works [14, 16, 15, 5, 33], its correctness is supported by the rigorous treatment of L,K-bounds and adjusted values, developed in this and in the previous subsections. Note that the input alphabet of the grid M_G is formed by the set of all action symbols of M associated to discretized parameters, together with the new symbol g. Again, the qualifications L, K, g or h may be dropped when no confusion can arise.

As an illustration, consider the TIOCA depicted in Figure 1. Note that there are two clocks that control the multimedia protocol. In other works a granularity of $\frac{1}{4}$, or even smaller, must be used when discretizing a timed model with two clocks. Here, instead, coarser values can be chosen for the TIOCA model. In fact, for this example a granularity of $g = \frac{1}{2}$ is chosen. Regarding context discretization, a granularity of 1 is chosen since the context variables simply count the numbers of image and sound frames. Also bounds $L = 3$ and $K = 3$ are chosen. Under these assumptions, Figure 2 shows part of the grid automaton obtained from Algorithm 3, given Figure 1 as the input TIOCA.

Algorithm 3 always terminates and there is a bound on the number of states in the resulting transition system.

```
 1  Input: L, K ∈ ℚ≥; k, ℓ positive integers; g = 1/k, h = 1/ℓ; [g,h]-adjusted TIOCA M.
 2  Output: Grid M_G = (S_G, s_G, Σ_G, T_G).
 3  let Ψ_z^K = {(z, ρ_K) | ρ ∈ [V → ℚ≥] }, z ∈ Σ;
 4  begin
 5      let s_G = (s, ν_0, λ_0); T_G ← ∅; HS ← ∅; RS ← s_G ;
 6      while RS \ HS ≠ ∅ do
 7          get a (s, ν, λ) from RS \ HS;
 8          move (s, ν, λ) to HS;
 9          foreach (s, x, δ_1, δ_2, δ_3, θ, κ, r) ∈ T_X do
10              if ν ⊨ δ_1 and λ ⊨ δ_3 and ν ⊕ θ ⊨ Inv(r) then
11                  let η = (ν ⊕ θ)_L;
12                  foreach (x, ρ) in Ψ_x^K do
13                      if ρ ⊨ δ_2 then
14                          let μ = κ_K(ρ, λ)
15                          add ((r, ν, λ), (x, ρ), (r, η, μ)) to T_G;
16                          add (r, η, μ) to RS, if (r, η, μ) ∉ HS;
17                      end
18                  end
19              end
20          end
21          foreach (s, y, δ_1, δ_2, θ, ξ, χ, r) ∈ T_Y do
22              if ν ⊨ δ_1 and λ ⊨ δ_2 and ν ⊕ θ ⊨ Inv(r) then
23                  let η = (ν ⊕ θ)_L; ρ = ξ_K(λ); μ = χ_K(λ);
24                  add ((s, ν, λ), (y, ρ), (r, η, μ) to T_G;
25                  add (r, η, μ) to RS, if (r, η, λ) ∉ HS;
26              end
27          end
28          if ν + h ⊨ Inv(s) for all 0 < h ≤ g then
29              let η = (ν + g)_L;
30              add ((s, ν, λ), g, (s, η, λ)) to T_G;
31              add e (s, η, λ) to RS, if (s, η, λ) ∉ HS;
32          end
33      end
34      S_G ← HS,   Σ_G ← {g} ∪ ⋃_{z∈Σ} Ψ_z^K;
35      return;
36  end
```

Algorithm 3: TIOCA grid algorithm.

Lemma 36 *Algorithm 3 halts with* $|S_G| \leq |S| \times \big(k\lfloor L\rfloor + k\big)^{|C|} \times \big(\ell\lfloor K\rfloor + \ell\big)^{|V|}$.

Proof First note that, by construction, Algorithm 3 only adds to RS elements in the form (r, η, μ), where s is a state, η is an L-bounded clock interpretation and μ is K-bounded context variable valuation. Moreover, by Proposition 24 we know that the interpretations constructed at lines 11, 23 and 29 named η are also g-adjusted. Also, by Proposition 32 the interpretation constructed at lines 14 and 23 named μ is also h-adjusted. Hence, by Lemma 25, the number of distinct g-adjusted clock interpretations is bounded by $\big(k\lfloor L\rfloor + k\big)^{|C|}$. By the same reasoning, the number of h-adjusted variable valuations is at most $\big(\ell\lfloor K\rfloor + \ell\big)^{|V|}$. Since the number of states in M is $|S|$, the result follows. ∎

Again, if we had different L_c and K_v values for clock and context variables c and v, respectively, the number of states in the grid would be bounded by $|S| \times \prod_{c\in C} (k\lfloor L_c\rfloor + k) \times \prod_{v\in V} (\ell\lfloor K_v\rfloor + \ell)$, which can be much smaller then $|S| \times \big(k\lfloor L\rfloor + k\big)^{|C|} \times \big(\ell\lfloor K\rfloor + \ell\big)^{|V|}$, if we

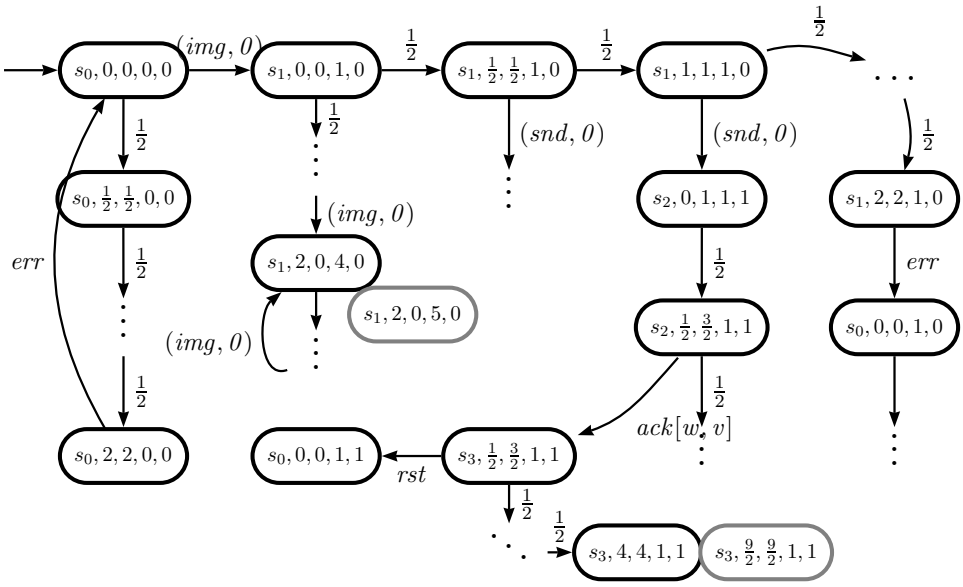

Figure 2: The partial grid automaton for Figure 1.

take the safe values $L = \max_{c \in C}\{L_c\}$ and $K = \max_{v \in V}\{L_v\}$.

Now, we want to argue that a TIOCA and its grid display compatible behaviors. That result leads to automatic methods for generating test suites for systems that exhibit both continuous time evolution as well as context transformations. A grid word in $\Sigma_G^\star$ will induce a movement in the grid in the usual way.

Definition 37 *Let M_G be the grid automaton corresponding to a TIOCA M. The* movement relation *of M_G, denoted by $\vdash_G$, is a binary relation over the set $\Sigma_G^\star \times S_G$ given by $(\langle\sigma\rangle\psi, s) \vdash_G (\psi, r)$ if and only if there is a transition (s, σ, r) in M_G.*

We may write $\gamma \Vdash_G^\psi \rho$ instead of $(\psi, \gamma) \vdash_G^\star (\varepsilon, \rho)$, or even $\gamma \Vdash_G \rho$ when ψ is not relevant.

Now, we expose the relationship between the TIOCA model and its corresponding grid. We start with a technical result that will be useful later on.

Lemma 38 *Let M be a TIOCA and let M_G be the corresponding grid. Let L, K be positive integers greater than all constants occurring in M, and let $g = 1/k, h = 1/\ell$ with k, ℓ positive integers. Take $\nu \in [C \to \mathbb{Q}_{\geq}]$, $\lambda \in [V \to \mathbb{Q}_{\geq}]$ and $p \in S$. Let $(p, \nu_L, \lambda_K) \in S_G$, and assume that $\nu + \eta \vDash Inv(p)$ for all $0 < \eta \leq ig$, where $i \geq 0$. Then $(p, \nu_L, \lambda_K) \Vdash_G^{g^i} (p, \omega_L, \lambda_K)$ and $(p, \omega_L, \lambda_K) \in S_G$, with $\omega = \nu + ig$.*

Proof Define $\nu^j = \nu + jg$, for all j, $0 \leq j \leq i$. It suffices to prove that[2] $(p, \nu_L^j, \lambda_K) \in S_G$ and $(p, \nu_L, \lambda_K) \Vdash_G^{g^j} (p, \nu_L^j, \lambda_K)$, for all j, $0 \leq j \leq i$, since $\nu_L^i = (\nu + ig)_L = \omega_L$.
We proceed by induction on $j \geq 0$.

Basis: when $j = 0$ we get $g^j = \varepsilon$. Then, trivially, $(p, \nu_L, \lambda_K) \Vdash_G^{g^j} (p, \nu_L, \lambda_K)$. Also $\omega = \nu + 0g = \nu$ and so $\omega_L = \nu_L$. Then, $(p, \omega_L, \lambda_K) = (p, \nu_L, \lambda_K)$. Thus, $(p, \omega_L, \lambda_K) \in S_G$ and $(p, \nu_L, \lambda_K) \Vdash_G^{g^j} (p, \omega_L, \lambda_K)$, completing the basis.
Induction step: assume the result holds for some j, $0 \leq j < i$.

The induction hypothesis gives $(p, \nu_L, \lambda_K) \Vdash_G^{g^j} (p, \nu_L^j, \lambda_K)$ and $(p, \nu_L^j, \lambda_K) \in S_G$. Note that $(p, \nu_L^j, \lambda_K) \in S_G$ gives $(p, \nu_L^j, \lambda_K) \in HS$ at line 34 of Algorithm 3. Also, pairs are moved from RS into HS one at a time at line 8. Hence, at some iteration, (p, ν_L^j, λ_K) was chosen at line 7. We show that $\nu_L^j + \eta \vDash Inv(p)$, for all $0 < \eta \leq g$, so that line 28 of Algorithm 3 applies.

Let $0 < \eta \leq g$. We need $\nu_L^j + \eta \vDash Inv(p)$. We have $0 < jg + \eta \leq (j+1)g \leq ig$. From the hypothesis we get $\nu + (jg + \eta) \vDash Inv(p)$, that is $(\nu + jg) + \eta \vDash Inv(p)$, and so $\nu^j + \eta \vDash Inv(p)$. Using Lemma 21 we get $\nu_L^j + \eta \vDash Inv(p)$, as desired.

Thus, Algorithm 3, lines 29 – 31, will put $((p, \nu_L^j, \lambda_K), g, (p, \rho, \lambda_K))$ in T_G, where $\rho = (\nu_L^j + g)_L$. Therefore, we get $(p, \nu_L^j, \lambda_K) \Vdash_G^{g} (p, \rho_L, \lambda_K)$. Also, by line 31, (p, ρ_L, λ_K) will be added to RS if it is not already in HS. In any case, when the loop at line 6 terminates, we get (p, ρ_L, λ_K) in HS and, by line 34, $(p, \rho_L, \lambda_K) \in S_G$. Moreover, since $(p, \nu_L, \lambda_K) \Vdash_G^{g^j} (p, \nu_L^j, \lambda_K)$, we also get $(p, \nu_L, \lambda_K) \Vdash_G^{g^{j+1}} (p, \rho_L, \lambda_K)$. We extend the induction by showing that $\rho_L = \nu_L^{j+1}$. Since $\rho_L = ((\nu^j)_L + g)_L$, Fact 20 gives $\rho_L = (\nu^j + g)_L = (\nu + jg + g)_L = (\nu + (j+1)g)_L = (\nu^{j+1})_L = \nu_L^{j+1}$. ∎

Now, we show that all $[g, h]$-reachable configurations are states in the grid.

Lemma 39 *M is a TIOCA and M_G its associated grid. Let L, K be positive integers greater than any constant occurring in M. If $(s, \nu, \lambda) \in S \times [C \to \mathbb{Q}_{\geq}] \times [V \to \mathbb{Q}_{\geq}]$ is $[g, h]$-reachable in M then $(s, \nu_L, \lambda_K) \in S_G$.*

Proof Definition 34 gives $(s_0, \nu_0, \lambda_0) \Vdash_M^{\psi} (s, \nu, \lambda)$ for some $[g, h]$-adjusted parameterized timed word $\psi \in \Psi_{[g,h]}$. We proceed by induction on the length of ψ.

If $\psi = \varepsilon$ then $s_0 = s$, $\nu_0 = \nu$, and $\lambda_0 = \lambda$. Algorithm 3, line 4, puts (s, ν, λ) in RS. Now, the while loop at line 6, together with lines 7 and 8, will put (s_0, ν_0, λ_0) in HS. Then, by line 34, we get $(s, \nu, \lambda) \in S_G$.

Assume the result holds for any $[g, h]$-adjusted timed word ψ of length at most n, $n \geq 0$. Take $\varphi \in \Psi_{[g,h]}$ and $\sigma \in \Sigma \cup \mathbb{Q}_{\geq}$ with $\psi = \varphi \cdot \langle \sigma \rangle$, where φ has length n.

[2] Here, ν_L^j denotes $(\nu^j)_L$.

From $(s_0,\nu_0,\lambda_0) \Vdash_M^{\psi} (s,\nu,\lambda)$ we obtain $(s_0,\nu_0,\lambda_0) \Vdash_M^{\varphi} (r,\mu,\omega)$ and $(r,\mu,\omega) \Vdash_M^{\langle\sigma\rangle} (s,\nu,\lambda)$ for some $r \in S$, $\mu \in [C \to \mathbb{Q}_{\geq}]$ and $\omega \in [V \to \mathbb{Q}_{\geq}]$. Since ψ is $[g,h]$-adjusted, we have that σ and φ are $[g,h]$-adjusted. By the induction hypothesis, we get $(r,\mu_L,\omega_K) \in S_G$. Then, $(r,\mu_L,\omega_K) \in HS$ (line 34). Clearly, from line 4 of Algorithm 3, together with the while loop at line 6 and lines 7 and 8, we can conclude that (r,μ_L,ω_K) will be in RS. So, at some point, (r,μ_L,ω_K) will be chosen at line 7.

We have three cases:

CASE 1: $\sigma = (x,\rho)$, for some $x \in X$ and some $\rho \in [R_x \to \mathbb{Q}_{\geq}]$.

Since $(r,\mu,\omega) \Vdash_M^{\langle\sigma\rangle} (s,\nu,\lambda)$ we must have in M a transition $(r,x,\delta_1,\delta_2,\delta_3,\theta,\kappa,s)$, for some $\delta_1 \in \Phi_C$, $\delta_2 \in \Phi_{R_x}$, $\delta_3 \in \Phi_V$, $\theta \in [C \curvearrowright \mathbb{Q}_{\geq}]$, and $\kappa \in F_x$, satisfying $\mu \vDash \delta_1$, $\rho \vDash \delta_2$, $\omega \vDash \delta_3$, $\lambda = \kappa(\rho,\omega)$, $\nu = \mu \oplus \theta$, and $\nu \vDash Inv(s)$.

From Lemma 21 we obtain $\mu_L \oplus \theta \vDash Inv(s)$ and $\mu_L \vDash \delta_1$, respectively. We also obtain $\omega_K \vDash \delta_3$ from Lemma 21. Since (r,μ_L,ω_K) will be chosen at line 7 of Algorithm 3, line 10 applies. Let $\alpha = (\mu_L \oplus \theta)_L$, as in line 11. Since $\rho \in [R_x \to \mathbb{Q}_{\geq}]$, we get $(x,\rho_K) \in I_x^K$, and so (x,ρ_K) will be chosen at line 12. Now, since $\rho \vDash \delta_2$, Lemma 21 gives $\rho_K \vDash \delta_2$, and we conclude that line 13 also applies. Let $\beta = \kappa_K(\rho_K,\omega_K)$, as in line 14. By line 16, the state (s,α,β) will be put into RS, if it is not already in HS. In any case, by the loop at line 6, we will get (s,α,β) in HS, and so, by line 34, we will have $(s,\alpha,\beta) \in S_G$. But, using Proposition 20, we have $\alpha = (\mu_L \oplus \theta)_L = (\mu \oplus \theta)_L$. Since $\nu = \mu \oplus \theta$, we obtain $\alpha = \nu_L$. Also, using Fact 29 we may write $\beta = \kappa_K(\rho_K,\omega_K) = \kappa_K(\rho,\omega)$. Since $\lambda = \kappa(\rho,\omega)$, we obtain $\beta = \lambda_K$, and so $(s,\nu_L,\lambda_K) \in S_G$, as desired.

CASE 2: $\sigma = (y,\rho)$, for some $y \in Y$ and some $\rho \in [V \to \mathbb{Q}_{\geq}]$.

Since $(r,\mu,\omega) \Vdash_M^{\langle\sigma\rangle} (s,\nu,\lambda)$ we must have a transition $(r,\sigma,\delta_1,\delta_2,\theta,\xi,\chi,s)$ in T_y, for some $\delta_1 \in \Phi_C$, $\delta_2 \in \Phi_V$, $\theta \in [C \curvearrowright \mathbb{Q}_{\geq}]$, $\xi \in H_y^P$ and $\chi \in H_y^T$, satisfying $\mu \vDash \delta_1$, $\omega \vDash \delta_2$, $\lambda = \chi(\omega)$, $\rho = \xi(\lambda)$, and $\nu \vDash Inv(s)$, where $\nu = \mu \oplus \theta$.

From Lemma 21 we obtain $\mu_L \oplus \theta \vDash Inv(s)$. And from Lemma 21 we get $\mu_L \vDash \delta_1$ and $\omega_K \vDash \delta_2$. This enables line 22. Since (r,μ_L,ω_K) will be chosen at line 7, we let $\alpha = (\mu_L \oplus \theta)_L$ as in line 23, and $\beta = \chi_K(\lambda_K)$ as in line 23. Then, line 25 inserts the state (s,α,β) into RS, if it is not already in HS. In any case, by the loop at line 6, we will get (s,α,β) in HS, and so by line 34 we will have $(s,\alpha,\beta) \in S_G$. But, using Proposition 20, we have $\alpha = (\mu_L \oplus \theta)_L = (\mu \oplus \theta)_L$ and, since $\nu = \mu \oplus \theta$, we obtain $\alpha = \nu_L$. Also, $\lambda_K = \chi_K(\omega) = \chi_K(\omega_K) = \beta$, using Fact 28. Then, $(s,\nu_L,\lambda_K) \in S_G$.

CASE 3: $\sigma \in \mathbb{Q}_{\geq}$.

Since σ is $[g,h]$-adjusted, we may write $\sigma = kg$, for some $k \geq 0$. Then, we obtain $(r,\mu,\omega) \Vdash_M^{\langle kg\rangle} (s,\nu,\lambda)$ where $s = r$, $\lambda = \omega$ and $\nu = \mu + kg$, with $\mu + \eta \vDash Inv(r)$, for all $0 < \eta \leq kg$. Since we already have $(r,\mu_L,\omega_K) \in S_G$, Lemma 38 gives $(r,\nu_L,\omega_K) \in S_G$. But $\lambda = \omega$, and so $(r,\nu_L,\lambda_K) \in S_G$, completing the proof. ∎

Conversely, grid states correspond to reachable configurations in the TIOCA model.

Lemma 40 *Let M_G be the grid corresponding to a TIOCA M. Let L,K be positive integers greater than any constant occurring in M. If $(s,\nu,\lambda) \in S_G$ then there are $\mu \in [C \to$*

$\mathbb{Q}_{\geq}]$, $\omega \in [V \to \mathbb{Q}_{\geq}]$ *and a* $[g,h]$*-adjusted parameterized timed word* $\psi \in \Psi_{[g,h]}$ *such that* $(s_0,\nu_0,\lambda_0) \Vdash_M^{\psi} (s,\mu,\omega)$*, with* $\mu_L = \nu$ *and* $\omega_K = \lambda$.

Proof From line 34 of Algorithm 3, we know that S_G is the set HS when the loop at line 6 terminates. From line 4, HS starts empty and elements are added to it one at a time and only at lines 16, 25, and 31. Hence, it suffices to show that the result holds for all triples (s,ν,λ) added to RS at these lines.

Let (r_i,μ_i,ω_i), $i \geq 0$, be the elements added to RS, in order. Clearly, $(r_0,\mu_0,\omega_0) = (s_0,\nu_0,\lambda_0)$ at line 4. Taking $\psi = \varepsilon$, the result is seen to hold for (r_0,μ_0,ω_0). Note also that $\nu_0 = (\nu_0)_L$, since $\nu_0(c) = 0$, for all $c \in C$, and also $\lambda_0 = (\lambda_0)_K$, since $\lambda_0(v) = 0$, for all $v \in V$.

Assume the result holds for (r_j,μ_j,ω_j), for all $0 \leq j < k$, for some $k \geq 1$. Consider (r_k,μ_k,ω_k). Since $k \geq 1$, (r_k,μ_k,ω_k) was added to RS at line 16, or at line 25, or at line 31. Hence, at that same iteration k some (r_j,μ_j,ω_j) with $j < k$ was chosen at line 7, with (r_k,μ_k,ω_k) subsequently added to RS also at iteration k. The induction hypothesis gives some $\psi \in \Psi_{[g,h]}$ such that $(s_0,\nu_0,\lambda_0) \Vdash_M^{\psi} (r_j,\mu,\omega)$ and $\mu_L = \mu_j$ and $\omega_K = \omega_j$.

There are three cases:

Case 1: (r_k,μ_k,ω_k) was added to RS at line 16. Then, from line 9 we obtain a transition $(r_j,x,\delta_1,\delta_2,\delta_3,\theta,\kappa,r_k)$ in T_X with $\mu_j \vDash \delta_1$, $\omega_j \vDash \delta_3$ and $\mu_j \oplus \theta \vDash Inv(r_k)$. From lines 11 and 16, we get $\mu_k = (\mu_j \oplus \theta)_L$. From lines 12 and 13 we get a $\rho \in [R_x \to \mathbb{Q}_{\geq}]$ with $(x,\rho) \in I_x$ and $\rho_K \vDash \delta_2$. Then, lines 14 and 16 give $\omega_k = \kappa_K(\rho_K,\omega_j)$.

Since $\mu_L = \mu_j$ and $\omega_K = \omega_j$ we get $\mu_L \vDash \delta_1$ and $\omega_K \vDash \delta_3$. Together with $\rho_K \vDash \delta_2$, Lemma 21 gives $\mu \vDash \delta_1$, $\rho \vDash \delta_2$ and $\omega \vDash \delta_3$. Moreover, from $\mu_L = \mu_j$ and $\mu_j \oplus \theta \vDash Inv(r_k)$ we get $\mu_L \oplus \theta \vDash Inv(r_k)$. From Lemma 21 we may write $(\mu_L \oplus \theta)_L \vDash Inv(r_k)$, and then using Proposition 20 we may write $(\mu \oplus \theta)_L \vDash Inv(r_k)$. Using Lemma 21 again, we have $\mu \oplus \theta \vDash Inv(r_k)$.

Collecting, we have $(r_j,x,\delta_1,\delta_2,\delta_3,\theta,\kappa,r_k)$ in T_X, $\mu \vDash \delta_1$, $\rho \vDash \delta_2$, $\omega \vDash \delta_3$ and $\mu \oplus \theta \vDash Inv(r_k)$. Then we may write $(r_j,\mu,\omega) \Vdash_M^{(x,\rho)} (r_k,\mu \oplus \theta,\kappa(\rho,\omega))$. Therefore, composing we get $(s_0,\nu_0,\lambda_0) \Vdash_M^{\psi\langle(x,\rho)\rangle} (r_k,\mu \oplus \theta,\kappa(\rho,\omega))$.

We complete this case by showing $(\mu \oplus \theta)_L = \mu_k$ and $(\kappa(\rho,\omega))_K = \omega_k$. From Proposition 20, $(\mu \oplus \theta)_L = (\mu_L \oplus \theta)_L$. Since $\mu_L = \mu_j$ and $\mu_k = (\mu_j \oplus \theta)_L$, we get $(\mu \oplus \theta)_L = (\mu_j \oplus \theta)_L = \mu_k$, as desired. From Fact 29 $(\kappa(\rho,\omega))_K = \kappa_K(\rho,\omega) = \kappa_K(\rho_K,\omega_K)$. Since $\omega_K = \omega_j$ and $\omega_k = \kappa_K(\rho_K,\omega_j)$, we get $(\kappa(\rho,\omega))_K = \omega_k$, as also desired.

Case 2: (r_k,μ_k,ω_k) was added to RS at line 25. Then, from lines 21 and 22, we get a transition $(r_j,y,\delta_1,\delta_2,\theta,\xi,\chi,r_k)$ in T_Y with $\mu_j \vDash \delta_1$, $\omega_j \vDash \delta_2$ and $\mu_j \oplus \theta \vDash Inv(r_k)$. From lines 23 and 25, $\mu_k = (\mu_j \oplus \theta)_L$, and $\omega_k = \chi_K(\omega_j)$.

Since $\mu_L = \mu_j$ and $\omega_K = \omega_j$, we get $\mu_L \vDash \delta_1$ and $\omega_K \vDash \delta_2$. Then, Lemma 21 yields $\mu \vDash \delta_1$ and $\omega \vDash \delta_2$. Also, from $\mu_L = \mu_j$ and $\mu_j \oplus \theta \vDash Inv(r_k)$, we get $\mu_L \oplus \theta \vDash Inv(r_k)$. From Lemma 21, we have $(\mu_L \oplus \theta)_L \vDash Inv(r_k)$. From Proposition 20 we may write $(\mu \oplus \theta)_L \vDash Inv(r_k)$ and so, from Lemma 21, we obtain $(\mu \oplus \theta) \vDash Inv(r_k)$.

Collecting, we have $(r_j,y,\delta_1,\delta_2,\theta,\xi,\chi,r_k)$ in T_Y, $\mu \vDash \delta_1$, $\omega \vDash \delta_2$, $(\mu \oplus \theta) \vDash Inv(r_k)$.

Then $(r_j, \mu, \omega) \overset{(y,\rho)}{\Vdash_M} (r_k, \mu \oplus \theta, \chi(\omega))$. Therefore, $(s_0, \nu_0, \lambda_0) \overset{\psi\langle(y,\rho)\rangle}{\Vdash_M} (r_k, \mu \oplus \theta, \chi(\omega))$.

We complete this case by showing $(\mu \oplus \theta)_L = \mu_k$ and $\chi_K(\omega) = \omega_k$. From Proposition 20, $(\mu \oplus \theta)_L = (\mu_L \oplus \theta)_L$. Since $\mu_L = \mu_j$ and $\mu_k = (\mu_j \oplus \theta)_L$, we get $(\mu \oplus \theta)_L = (\mu_j \oplus \theta)_L = \mu_k$, as desired. Also, since $\omega_K = \omega_j$ and $\omega_k = \chi_K(\omega_j)$, we get $\omega_k = \chi_K(\omega_K) = \chi_K(\omega)$, where we used Fact 28. The argument for this case is complete.

CASE 3: (r_k, μ_k, ω_k) was added to RS at line 31. Then from line 28 we obtain $\mu_j + \eta \vDash Inv(r_j)$, $0 < \eta \leq g$. Let $\mu_k = (\mu_j + g)_L$, as in line 29. Then, line 31 gives $\omega_k = \omega_j$ and $r_k = r_j$. Since $\mu_j = \mu_L$ we get $\mu_L + \eta \vDash Inv(r_j)$ and so $\mu + \eta \vDash Inv(r_j)$ by Lemma 21, for all $0 < \eta \leq g$. Also, since $\omega_j = \omega_K$ we get $\omega_K = \omega_k$. Then $(r_j, \mu, \omega) \overset{\langle g \rangle}{\Vdash_M} (r_j, \mu + g, \omega)$ and, since $r_j = r_k$ we get $(r_j, \mu, \omega) \overset{\langle g \rangle}{\Vdash_M} (r_k, \mu + g, \omega)$. Therefore, $(s_0, \nu_0, \lambda_0) \overset{\psi \cdot \langle g \rangle}{\Vdash_M} (r_k, \mu + g, \omega)$.

We complete this case by showing that $(\mu + g)_L = \mu_k$ and $\omega_K = \omega_k$. Since $\mu_k = (\mu_j + g)_L$ and $\mu_L = \mu_j$, we get $\mu_k = (\mu_L + g)_L$. Using Proposition 20, we conclude that $\mu_k = (\mu + g)_L$. Also, since we already have $\omega_K = \omega_j$ and $\omega_k = \omega_j$, we get $\omega_K = \omega_k$, as desired.

The induction is extended and we are done. ∎

Now we can verify that the grid homomorphically imitates the corresponding TIOCA. We give the morphism first. Recall Definition 33.

Definition 41 *Let M be a TIOCA. Define the basic mappings: (i) every g-adjusted time value $ig \in \mathbb{Q}_{\geq}$, with $i \geq 0$, is mapped to the grid word $g^i \in \Sigma_G^\star$; and (ii) every symbol $(z, \mu) \in \Psi_\Sigma$ is mapped to (z, μ_K). Define the morphism $f : \Psi_{[g,h]} \to \Sigma_G^\star$ by extending these basic mappings in the usual way.*

The next important result shows that the grid can imitate all TIOCA movements.

Theorem 42 *M is a TIOCA. Let $L, K \in \mathbb{Q}_{\geq}$ be greater than all constants occurring in M, and let M_G be the grid of M. Let $s, r \in S$, $\nu, \omega \in [C \to \mathbb{Q}_{\geq}]$ and $\lambda, \mu \in [V \to \mathbb{Q}_{\geq}]$ be such that (s, ν, λ) is $[g,h]$-reachable in M. If $(s, \nu, \lambda) \overset{\psi}{\Vdash_M} (r, \omega, \mu)$ then $(s, \nu_L, \lambda_K), (r, \omega_L, \mu_K) \in S_G$ and $(s, \nu_L, \lambda_K) \overset{f(\psi)}{\Vdash_G} (r, \omega_L, \mu_K)$.*

Proof Since (s, ν, λ) is $[g,h]$-reachable in M, we get a $[g,h]$-adjusted parameterized timed word ϕ such that $(s_0, \nu_0, \lambda_0) \overset{\phi}{\Vdash_M} (s, \nu, \lambda)$. Then $(s_0, \nu_0, \lambda_0) \overset{\phi \cdot \psi}{\Vdash_M} (r, \omega, \mu)$. Since $\phi \cdot \psi$ is also $[g,h]$-adjusted, (r, ω, μ) is also $[g,h]$-reachable in M. Thus, by Lemma 39, we get $(s, \nu_L, \lambda_K), (r, \omega_L, \mu_K) \in S_G$. It remains to show that $(s, \nu_L, \lambda_K) \overset{f(\psi)}{\Vdash_G} (r, \omega_L, \mu_K)$.

We proceed by induction on the length $n \geq 0$ of ψ, noting that $(s, \nu, \lambda) \overset{\psi}{\Vdash_M} (r, \omega, \mu)$.

BASIS: when $n = 0$, we get $\psi = \varepsilon$ and so $s = r$, $\nu = \omega$ and $\lambda = \mu$. Thus $(s, \nu_L, \lambda_K) = (r, \omega_L, \mu_K)$ and so $(s, \nu_L, \lambda_K) \overset{\varepsilon}{\Vdash_G} (r, \omega_L, \mu_K)$. Since $f(\psi) = \varepsilon$, the basis is complete.

INDUCTION STEP: For all $x \in X$, we will denote the set of all $[g,h]$-adjusted parameterized timed inputs by $I_x = \{(x, \rho) | \rho \in [R_x \to \mathbb{Q}_{\geq}] \text{is } [g,h]\text{-adjusted}\}$, and will denote by $I_x^K =$

$\{(x, \rho_K) | (x, \rho) \in I_x\}$ the set of all K-bounded such inputs in I_x. Similarly, we define I_y and I_y^K for all $y \in Y$.

Now, assume that the result holds for all $[g, h]$-adjusted parameterized timed words of length at most n. Take $\psi = \varphi \cdot \langle\sigma\rangle$, where φ has length n and $\langle\sigma\rangle$ has length one. Then, $(s, \nu, \lambda) \overset{\psi}{\Vdash_M} (r, \omega, \mu)$ gives $(s, \nu, \lambda) \overset{\varphi}{\Vdash_M} (p, \alpha, \beta)$ and $(p, \alpha, \beta) \overset{\langle\sigma\rangle}{\Vdash_M} (r, \omega, \mu)$, for some $\alpha \in [C \to \mathbb{Q}_{\geq}]$ and $\beta \in [V \to \mathbb{Q}_{\geq}]$. By the induction hypothesis, $(p, \alpha_L, \beta_K) \in S_G$ and $(s, \nu_L, \lambda_K) \overset{f(\varphi)}{\underset{G}{\Vdash}} (p, \alpha_L, \beta_K)$. Since, by definition, $f(\psi) = f(\varphi) \cdot f(\langle\sigma\rangle)$, it remains to show that $(p, \alpha_L, \beta_K) \overset{f(\langle\sigma\rangle)}{\underset{G}{\Vdash}} (r, \omega_L, \mu_K)$. Note that, since ψ is $[g, h]$-adjusted, then so is $\langle\sigma\rangle$. Then, we have three cases: when $\sigma \in I_x$ for some $x \in X$, when $\sigma \in I_y$ for some $y \in Y$, and when $\sigma \in \mathbb{Q}_{\geq}$.

Case 1: $\sigma = (x, \rho) \in I_x$ for some $x \in X$.

Since $(p, \alpha, \beta) \overset{(x,\rho)}{\Vdash_M} (r, \omega, \mu)$, we must have a transition $(p, x, \delta_1, \delta_2, \delta_3, \theta, \kappa, r)$ in T_X with $\alpha \vDash \delta_1$, $\rho \vDash \delta_2$, $\beta \vDash \delta_3$, $\mu = \kappa(\rho, \beta)$, $\omega = \alpha \oplus \theta$, and $\omega \vDash Inv(r)$.

Recall that $(p, \alpha_L, \beta_K) \in S_G$. Hence, at some point, Algorithm 3 has chosen (p, α_L, β_K) at line 7. Moreover, using Lemma 21, from $\alpha \vDash \delta_1$ and $\beta \vDash \delta_3$ we get $\alpha_L \vDash \delta_1$ and $\beta_K \vDash \delta_3$. From $\alpha \oplus \theta \vDash Inv(r)$ and Lemma 21 we get $\alpha_L \oplus \theta \vDash Inv(r)$. Then, line 10 at Algorithm 3 is enabled. Let $\alpha' = (\alpha_L \oplus \theta)_L$, as in line 11. Also, clearly, $(x, \rho_K) \in I_x^K$ and, from $\rho \vDash \delta_2$ and Lemma 21, we get $\rho_K \vDash \delta_2$, showing that line 13 is also enabled. Let $\beta' = \kappa_K(\rho_K, \beta_K)$, as in line 14. Then line 15 adds $((p, \alpha_L, \beta_K), (x, \rho_K), (r, \alpha', \beta'))$ to T_G. Now, since $\sigma = (x, \rho)$, we get $f(\langle\sigma\rangle) = (x, \rho_K)$. Then, $(p, \alpha_L, \beta_K) \overset{f(\langle\sigma\rangle)}{\underset{G}{\Vdash}} (r, \alpha', \beta')$.

We complete this case by showing that $\alpha' = \omega_L$ and that $\beta' = \mu_K$. Since $\alpha' = (\alpha_L \oplus \theta)_L$, Proposition 20 gives $\alpha' = (\alpha \oplus \theta)_L$. Then $\alpha' = \omega_L$ since $\omega = \alpha \oplus \theta$. Also, since $\beta' = \kappa_K(\rho_K, \beta_K)$, Fact 29 gives $\beta' = \kappa_K(\rho, \beta)$. Then $\beta' = \mu_K$ since $\mu = \kappa(\rho, \beta)$.

Case 2: $\sigma = (y, \rho) \in I_y$ for some $y \in Y$.

Since $(p, \alpha, \beta) \overset{(y,\rho)}{\Vdash_M} (r, \omega, \mu)$, we must have a transition $(p, y, \delta_1, \delta_2, \theta, \xi, \chi, r)$ in T_Y, with $\alpha \vDash \delta_1$, $\beta \vDash \delta_2$, $\mu = \chi(\beta)$, $\rho = \xi(\beta)$, $\omega = \alpha \oplus \theta$, and $\omega \vDash Inv(r)$.

Recall that $(p, \alpha_L, \beta_K) \in S_G$. Hence, at some point, Algorithm 3 has chosen (p, α_L, β_K) at line 7. From $\alpha \vDash \delta_1$ and $\beta \vDash \delta_2$, Lemma 21 gives $\alpha_L \vDash \delta_1$ and $\beta_K \vDash \delta_2$. From $\alpha \oplus \theta \vDash Inv(r)$ and Lemma 21 we get $\alpha_L \oplus \theta \vDash Inv(r)$. Then Algorithm 3, lines 23 and 24, adds $((p, \alpha_L, \beta_K), (y, \rho_K), (r, \alpha', \beta'))$ to T_G, where $\alpha' = (\alpha_L \oplus \theta)_L$ and $\beta' = \chi_K(\beta_K)$. Since $\sigma = (y, \rho)$, we get $f(\langle\sigma\rangle) = (y, \rho_K)$. Hence, $(p, \alpha_L, \beta_K) \overset{f(\langle\sigma\rangle)}{\underset{G}{\Vdash}} (r, \alpha', \beta')$.

We complete this case by showing that $\alpha' = \omega_L$ and that $\beta' = \mu_K$. Since $\alpha' = (\alpha_L \oplus \theta)_L$, Fact 20 gives $\alpha' = (\alpha \oplus \theta)_L$. Then $\alpha' = \omega_L$ since $\omega = \alpha \oplus \theta$. Since $\beta' = \chi_K(\beta_K)$, Fact 28 gives $\beta' = \chi_K(\beta)$. Since $\mu = \chi(\beta)$ we get $\beta' = \mu_K$.

Case 3: $\sigma \in \mathbb{Q}_{\geq}$.

Since σ is $[g, h]$-adjusted, let $\sigma = ig$, for some $i \geq 0$. Moreover, since $(p, \alpha, \beta) \overset{\langle ig\rangle}{\Vdash_M} (r, \omega, \mu)$, we have $p = r$, $\omega = \alpha + ig$, $\alpha + \eta \vDash Inv(r)$ for all $0 < \eta \leq ig$, and $\mu = \beta$. Hence, $\alpha + \eta \vDash Inv(p)$ for $0 < \eta \leq ig$. Since we already have $(p, \alpha_L, \beta_K) \in S_G$, Lemma 38

gives $(p, \alpha'_L, \beta_K) \in S_G$ and $(p, \alpha_L, \beta_K) \overset{g^i}{\Vdash}_G (p, \alpha'_L, \beta_K)$, with $\alpha' = \alpha + ig$. So, $\alpha' = \omega$ and, since we already have $\mu = \beta$, we get $(p, \alpha'_L, \beta_K) = (r, \omega_L, \mu_K)$. But $f(\langle\sigma\rangle) = g^i$, and so $(p, \alpha_L, \beta_K) \overset{f(\langle\sigma\rangle)}{\Vdash}_G (r, \omega_L, \mu_K)$, as desired.

The induction is extended, completing the proof. ∎

In order to illustrate Theorem 42, consider the TIOCA depicted at Figure 1. A configuration (s, ν, λ) will be written (s, t_1, t_2, v_1, v_2), where $t_1 = \nu(c_1)$, $t_2 = \nu(c_2)$, $v_1 = \lambda(v_1)$ and $v_2 = \lambda(v_2)$. The machine starts at configuration $(s_0, 0, 0, 0, 0)$, since the input parameter associated to *img* is valued at zero. Choose the granularities $g = \frac{1}{2}$ and $h = 1$, and let the boundary values be $K = 3$ and $L = 3$. Take the $[g, h]$-adjusted timed word $\alpha = 0(img, 0)\,1\,(snd, 0)\,\frac{1}{2}(ack[w, v])\,0\,rst$. There is no relevant value greater than the chosen granularity $h = 1$ to be discretized on the input parameter valuations. From Figure 1, computing over the timed word α and starting at $(s_0, 0, 0, 0, 0)$, reveals

$$(s_0, 0, 0, 0, 0) \overset{(img, 0)}{\to} (s_1, 0, 0, 1, 0) \underset{1}{\to} (s_1, 1, 1, 1, 0) \overset{(snd, 0)}{\to}$$
$$(s_2, 0, 1, 1, 1) \underset{1/2}{\to} (s_2, \tfrac{1}{2}, \tfrac{3}{2}, 1, 0) \overset{(ack[w, v])}{\to} (s_3, \tfrac{1}{2}, \tfrac{3}{2}, 1, 0) \overset{rst}{\to}$$
$$(s_0, 0, 0, 1, 0).$$

Then, the morphism f gives the grid word

$$f(\alpha) = \beta = (\frac{1}{2})^0\,(img,0)\,(\frac{1}{2})^2\,(snd,0)\,(\frac{1}{2})^1\,(ack[w,v])\,(\frac{1}{2})^0\,rst.$$

Apply β to the grid, starting at state $(s_0, 0, 0, 0, 0)$. From Figure 2, the grid moves as

$$(s_0, 0, 0, 0, 0) \overset{(img,0)}{\Vdash}_G (s_1, 0, 0, 1, 0) \overset{\frac{1}{2}}{\Vdash}_G (s_1, \tfrac{1}{2}, \tfrac{1}{2}, 1, 0)$$
$$\overset{\frac{1}{2}}{\Vdash}_G (s_1, 1, 1, 1, 0) \overset{(snd,0)}{\Vdash}_G (s_2, 0, 1, 1, 1) \overset{\frac{1}{2}}{\Vdash}_G (s_2, \tfrac{1}{2}, \tfrac{3}{2}, 1, 1)$$
$$\overset{(ack[w,v])}{\Vdash}_G (s_3, \tfrac{1}{2}, \tfrac{3}{2}, 1, 1) \overset{rst}{\Vdash}_G (s_0, 0, 0, 1, 1).$$

Also, note that at state s_3 both clocks can reach the $\lfloor L \rfloor + 1 = 3 + 1 = 4$ boundary value. Then the corresponding grid state goes from $(s_3, 4, 4, v_1, v_2)$ to $(s_3, \frac{9}{2}, \frac{9}{2}, v_1, v_2)$, and back to $(s_1, 4, 4, v_1, v_2)$. The clock values will then cycle between 4 and 9/2.

The next morphism is used to show the converse, that is, how the original TIOCA can mimic movements of the corresponding grid automaton.

Definition 43 *The morphism $\widehat{h} : \Sigma^\star_G \to \Psi_{[g,h]}$ is defined by the natural extension of the basic mappings: (i) the grid word $g \in \Sigma_G$ maps to the time value $g \in \mathbb{Q}_{\geq}$; (ii) the grid word $(x, \rho) \in \Sigma_G$, with $x \in X$ and $\rho \in [R_x \to \mathbb{Q}_{\geq}]$, maps to the input $(x, \rho) \in \Psi_X$; and (iii) the grid word $(y, \lambda) \in \Sigma_G$, with $y \in Y$ and $\lambda \in [V \to \mathbb{Q}_{\geq}]$, maps to the output $(y, \lambda) \in \Psi_Y$.*

We also have the converse, that is, a TIOCA homomorphically imitates the corresponding grid automaton.

Theorem 44 *M_G is the grid of a TIOCA M. Let $L, K \in \mathbb{Q}_{\geq}$ be greater than all constants occurring in M. Let $(s,\widehat{\omega},\widehat{\mu}),(r,\omega,\mu) \in S_G$ with $(s,\widehat{\omega},\widehat{\mu}) \overset{\psi}{\Vdash}_G (r,\omega,\mu)$, for some $\psi \in \Sigma_G^\star$. Then there are $\alpha,\widehat{\alpha} \in [C \to \mathbb{Q}_{\geq}]$, $\beta,\widehat{\beta} \in [V \to \mathbb{Q}_{\geq}]$ and $\widehat{\psi} \in \Psi_{[g,h]}$, such that $(s,\widehat{\alpha},\widehat{\beta}) \overset{\widehat{\psi}}{\vdash}_M (r,\alpha,\beta)$, with $\widehat{\omega} = \widehat{\alpha}_L$, $\omega = \alpha_L$, $\widehat{\mu} = \widehat{\beta}_K$, $\mu = \beta_K$ and $\widehat{h}(\psi) = \widehat{\psi}$.*

Proof We proceed by induction on the length $n \geq 0$ of ψ.
BASIS: when $n = 0$, we get $\psi = \varepsilon$, $s = r$, $\widehat{\omega} = \omega$ and $\widehat{\mu} = \mu$. Let $\widehat{\psi} = \varepsilon$. Since $(s,\widehat{\omega},\widehat{\mu}) \in S_G$, Lemma 40, gives $\widehat{\alpha} \in [C \to \mathbb{Q}_{\geq}]$, with $\widehat{\alpha}_L = \widehat{\omega}$, and $\widehat{\beta} \in [V \to \mathbb{Q}_{\geq}]$, with $\widehat{\beta}_K = \widehat{\mu}$. Take $\alpha = \widehat{\alpha}$ and $\beta = \widehat{\beta}$. Then $(s,\widehat{\alpha},\widehat{\beta}) \overset{\varepsilon}{\vdash}_M (r,\alpha,\beta)$, and so $(s,\widehat{\alpha},\widehat{\beta}) \overset{\widehat{\psi}}{\vdash}_M (r,\alpha,\beta)$, with $\widehat{h}(\psi) = \widehat{\psi}$. Moreover, $\omega = \widehat{\omega} = \widehat{\alpha}_L = \alpha_L$, and $\mu = \widehat{\mu} = \widehat{\beta}_K = \beta_K$, completing the basis.
INDUCTION STEP: For all $x \in X$, we will denote the set of all $[g,h]$-adjusted parameterized timed inputs by $I_x = \{(x,\rho) | \rho \in [R_x \to \mathbb{Q}_{\geq}] \text{is } [g,h]\text{-adjusted}\}$, and will denote by $I_x^K = \{(x,\rho_K) | (x,\rho) \in I_x\}$ the set of all K-bounded such inputs. Similarly, we define I_y and I_y^K for all $y \in Y$.

Now, assume that the result holds for all grid words of length at most n. Take $\psi = \varphi{\cdot}\sigma \in \Sigma_G^\star$, where φ has length n and $\sigma \in \Sigma_G$. Then, $(s,\widehat{\omega},\widehat{\mu}) \overset{\psi}{\Vdash}_G (r,\omega,\mu)$ gives $(s,\widehat{\omega},\widehat{\mu}) \overset{\varphi}{\Vdash}_G (p,\omega',\mu')$ and $(p,\omega',\mu') \overset{\sigma}{\Vdash}_G (r,\omega,\mu)$, with $(p,\omega',\mu') \in S_G$. By the induction hypothesis we get some $\widehat{\alpha},\alpha' \in [C \to \mathbb{Q}_{\geq}]$, some $\widehat{\beta},\beta' \in [V \to \mathbb{Q}_{\geq}]$ and some $\varphi' \in \Psi_{[g,h]}$, with $(s,\widehat{\alpha},\widehat{\beta}) \overset{\varphi'}{\vdash}_M (p,\alpha',\beta')$, $\widehat{\alpha}_L = \widehat{\omega}$, $\alpha'_L = \omega'$, $\widehat{\beta}_K = \widehat{\mu}$, $\beta'_K = \mu'$ and $\widehat{h}(\varphi) = \varphi'$.

We extend the induction by showing that $(p,\alpha',\beta') \overset{\langle\sigma'\rangle}{\vdash}_M (r,\alpha,\beta)$, for some $\alpha \in [C \to \mathbb{Q}_{\geq}]$ some $\beta \in [V \to \mathbb{Q}_{\geq}]$ and some parameterized word σ', with $\alpha_L = \omega$, $\beta_K = \mu$ and $\widehat{h}(\sigma) = \langle\sigma'\rangle$. Note that then we will have $(p,\widehat{\alpha},\widehat{\beta}) \overset{\varphi'\langle\sigma'\rangle}{\vdash}_M (r,\alpha,\beta)$, with $\widehat{h}(\psi) = \widehat{h}(\varphi)\,\widehat{h}(\sigma) = \varphi'\langle\sigma'\rangle = \widehat{\psi}$, as desired.

Note that, since $\alpha'_L = \omega'$ and $\beta'_K = \mu'$, we also have $(p,\alpha'_L,\beta'_K) \overset{\sigma}{\Vdash}_G (r,\omega,\mu)$. There are three cases: when $\sigma \in I_x^K$ for some $x \in X$, when $\sigma \in I_y^K$ for some $y \in Y$, and when $\sigma = g$.
Case 1: $\sigma = (x,\rho) \in I_x^K$, for some $x \in X$.

Then $(p,\alpha'_L,\beta'_K) \overset{(x,\rho)}{\Vdash}_G (r,\omega,\mu)$, and we must have a transition $((p,\alpha'_L,\beta'_K),(x,\rho),(r,\omega,\mu))$ in T_G. From Algorithm 3, this can only happen if such a transition was added to T_G at line 15. Hence, from lines 9 to 16, we must have a transition $(p,x,\delta_1,\delta_2,\delta_3,\theta,\kappa,r)$ in T_X. Moreover, from line 10 we get $\alpha'_L \vDash \delta_1$, $\beta'_K \vDash \delta_3$ and $\alpha'_L \oplus \theta \vDash Inv(r)$. Also, from line 13 we get $\rho \vDash \delta_2$. Finally, from lines 11 and 14, we may write, respectively, $\omega = (\alpha'_L \oplus \theta)_L$ and $\mu = \kappa_K(\rho,\beta'_K)$.

Recall that we want $(p,\alpha',\beta') \overset{\langle\sigma'\rangle}{\vdash}_M (r,\alpha,\beta)$, with $\alpha_L = \omega$, $\beta_K = \mu$ and $\widehat{h}(\sigma) = \langle\sigma'\rangle$. Since $(x,\rho) \in I_x^K$ we know that $\rho = \rho'_K$, for some $\rho' \in [V \to \mathbb{Q}_{\geq}]$. Then, by Fact 16, we get $\rho_K = (\rho'_K)_K = \rho'_K = \rho$. Thus, $\widehat{h}(\sigma) = \widehat{h}((x,\rho)) = \widehat{h}((x,\rho_K)) = (x,\rho'_K) = (x,\rho') = \langle\sigma'\rangle$. It

remains to show that $(p, \alpha', \beta') \overset{(x,\rho)}{\Vdash_M} (r, \alpha, \beta)$, with $\alpha_L = \omega$ and $\beta_K = \mu$.

Since we already have $\alpha'_L \vDash \delta_1$ and $\beta'_K \vDash \delta_3$, Lemma 21 yields $\alpha' \vDash \delta_1$ and $\beta' \vDash \delta_3$. Also, from $\alpha'_L \oplus \theta \vDash Inv(r)$ and Lemma 21, we get $(\alpha'_L \oplus \theta)_L \vDash Inv(r)$, and using Proposition 20 we have $(\alpha' \oplus \theta)_L \vDash Inv(r)$, and so $\alpha' \oplus \theta \vDash Inv(r)$ by Lemma 21 again. Since we know that the transition $(p, x, \delta_1, \delta_2, \delta_3, \theta, \kappa, r)$ is in T_X, and that $\rho \vDash \delta_2$, we can write $(p, \alpha', \beta') \overset{\langle(x,\rho)\rangle}{\Vdash_M} (r, \alpha' \oplus \theta, \kappa(\rho, \beta'))$. Let $\alpha = \alpha' \oplus \theta$. Then $\alpha_L = (\alpha' \oplus \theta)_L$ and, using Proposition 20, we get $\alpha_L = (\alpha'_L \oplus \theta)_L = \omega$, as desired. Now let $\beta = \kappa(\rho, \beta')$. Then, $\beta_K = \kappa_K(\rho, \beta')$. By Fact 29 we get $\beta_K = \kappa_K(\rho_K, \beta'_K)$. Since we already have $\rho_K = \rho$, we get $\beta_K = \kappa_K(\rho, \beta'_K) = \mu$, concluding this case.

Case 2: $\sigma = (y, \rho) \in I_y^K$, for some $y \in Y$.

Then $(p, \alpha'_L, \beta'_K) \overset{(y,\rho)}{\Vdash_G} (r, \omega, \mu)$, and we must have a transition $((p, \alpha'_L, \beta'_K), (y, \rho), (r, \omega, \mu))$ in T_G. From Algorithm 3, this can only happen if such a transition was added to T_G at line 24. Hence, from lines 21 and 22, we must have a transition $(p, y, \delta_1, \delta_2, \theta, \xi, \chi, r)$ in T_Y, with $\alpha'_L \vDash \delta_1$, $\beta'_K \vDash \delta_2$ and $\alpha'_L \oplus \theta \vDash Inv(r)$. Further, from lines 23 to 25, we know that $\omega = (\alpha'_L \oplus \theta)_L$, $\mu = \chi_K(\beta'_K)$ and $\rho = \xi_K(\beta'_K)$.

Recall that we want $(p, \alpha', \beta') \overset{\langle\sigma'\rangle}{\Vdash_M} (r, \alpha, \beta)$, with $\alpha_L = \omega$, $\beta_K = \mu$ and $\widehat{h}(\sigma) = \langle\sigma'\rangle$.

Since we already have $\alpha'_L \vDash \delta_1$ and $\beta'_K \vDash \delta_2$, Lemma 21 yields $\alpha' \vDash \delta_1$ and $\beta' \vDash \delta_2$. Also, from $\alpha'_L \oplus \theta \vDash Inv(r)$ and Lemma 21, we get $(\alpha'_L \oplus \theta)_L \vDash Inv(r)$, and using Proposition 20 we have $(\alpha' \oplus \theta)_L \vDash Inv(r)$, and so $\alpha' \oplus \theta \vDash Inv(r)$ by Lemma 21 again. We know that the transition $(p, y, \delta_1, \delta_2, \theta, \xi, \chi, r)$ is in T_Y. We can then write $(p, \alpha', \beta') \overset{\langle(y,\rho')\rangle}{\Vdash_M} (r, \alpha, \beta)$, with $\alpha = \alpha' \oplus \theta$, $\beta = \chi(\beta')$ and $\rho' = \xi(\beta')$. It remains to show that $\widehat{h}(\sigma) = \langle\sigma'\rangle$, $\alpha_L = \omega$ and $\beta_K = \mu$.

Since $\rho' = \xi(\beta')$, using Fact 28, we obtain $\rho'_K = \xi_K(\beta') = \xi_K(\beta'_K) = \rho$. Then, $\widehat{h}(\sigma) = \widehat{h}((y, \rho)) = \langle(y, \rho'_K)\rangle = \langle\sigma'\rangle$. Next, $\alpha_L = (\alpha' \oplus \theta)_L = (\alpha'_L \oplus \theta)_L = \omega$, using Proposition 20. Also, $\beta_K = \chi_K(\beta') = \chi_K(\beta'_K)$, using Fact 28. We then get $\beta_K = \mu$, completing this case.

Case 3: $\sigma = g$.

Since $(p, \alpha'_L, \beta'_L) \overset{g}{\Vdash_G} (r, \omega, \mu)$, we must have a transition $((p, \alpha'_L, \beta'_K), g, (r, \omega, \mu))$ in T_G. From Algorithm 3, lines 28–31, we have $p = r$, $\omega = (\alpha'_L + g)_L$, $\alpha'_L + \eta \vDash Inv(p)$ for all $0 < \eta \leq g$, and $\mu = \beta'_K$.

We need $(p, \alpha'_l, \beta'_L) \overset{\langle\sigma'\rangle}{\Vdash_M} (r, \alpha, \beta)$, with $\alpha_L = \omega$, $\beta_K = \mu$ and $\widehat{h}(\sigma) = \langle\sigma'\rangle$.

We already have $p = r$. Fix any η, $0 < \eta \leq g$. From $\alpha'_L + \eta \vDash Inv(p)$ and Lemma 21 we get $(\alpha'_L + \eta)_L \vDash Inv(p)$. From Proposition 20 it follows that $(\alpha' + \eta)_L \vDash Inv(p)$, and from Lemma 21 again, $\alpha' + \eta \vDash Inv(p)$. We may conclude that $\alpha' + \eta \vDash Inv(p)$ for all $0 < \eta \leq g$. We can then write $(p, \alpha'_L, \beta'_L) \overset{\langle\sigma'\rangle}{\Vdash_M} (r, \alpha, \beta)$ where $\sigma' = g$, $\alpha = \alpha' \oplus \theta$ and $\beta = \beta'$.

Then, $\widehat{h}(\sigma) = \widehat{h}(g) = \langle g\rangle = \langle\sigma'\rangle$. Also, using Proposition 20, $\alpha_L = (\alpha' \oplus \theta)_L = (\alpha'_L \oplus \theta)_L = \omega$. Finally, $\beta_K = \beta'_K = \mu$, completing this case.

The induction is extended and the proof is complete. ∎

Theorems 42 and 44 form the basic support for the formal verification of timed context

systems and model-based testing strategies.

6 Concluding Remarks

Many methods and techniques have been proposed to verify and test critical and reactive systems, several of which are based on formal methods. Some of them deal with continuous time evolution, others allow some form of data flow analysis. But a single formalism for treating both these two issues simultaneously has been lacking.

The basic formal model used here is the Timed Input/Output Context Automaton (TIOCA), a generalization of the earlier Timed Input/Output Automaton. This work proposes and proves correct a new and more flexible method to discretize TIOCA models, thereby making more manageable the activities of verifying and testing real-time systems with data flow transformations. The discretization technique avoids the classical notion of clock regions, and allows for an ample range of granularity values that could be chosen in the discretization process for both continuous time and context variable values. The grid automaton thus obtained is capable of homomorphically simulating the original timed context system, and vice-versa. This leads to automatic methods for verifying and testing systems that exhibit both continuous time evolution as well as context transformations.

In order to model specific system properties, we use test purpose TIOCA, which are combined with the specification TIOCA using the notion of a synchronous product. The discretization algorithm is then applied to the resulting product to generate a grid automaton that reflects both the behavior of the original system, as well as the properties specified by the test purpose. By automating the extraction of test cases from the grid, the desired test suites are then constructed. We illustrate the discretization method by using it in a testing framework that automatically generates test suites for timed systems with context variables.

As for further studies along these lines we can suggest a formal development of the notion of conformance testing using the TIOCA framework. More efficient processes could be explored by constructing the grid automaton on the fly while extracting the test sequences. The grid algorithm could also be used implicitly when testing runs. As another suggestion, one might consider factoring out common sub-words from test cases in order to obtain more manageable test suites.

References

[1] I.T. Adamson. *Introduction to Field Theory.* Dover Books on Mathematics Series. Dover Publications, 2007.

[2] Rajeev Alur. Timed automata. In *CAV*, number 1633 in LNCS, pages 8–22, 1999.

[3] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[4] Adilson Luiz Bonifacio and Arnaldo Vieira Moura. A New Timed Discretization Method for Automatic Test Generation for Timed Systems. Technical Report IC-09-31, Institute of Computing, University of Campinas, September 2009. http://www.ic.unicamp.br/∼reltech/2009/09-31.pdf.

[5] Adilson Luiz Bonifacio and Arnaldo Vieira Moura. A new method for testing timed systems. *Softw. Test., Verif. Reliab.*, 2011. Article first published online.

[6] Adilson Luiz Bonifacio, Arnaldo Vieira Moura, Adenilso da Silva Simao, and Jose Carlos Maldonado. Towards deriving test sequences by model checking. *Electron. Notes Theor. Comput. Sci.*, 195:21–40, 2008.

[7] Laura Brandán Briones and Ed Brinksma. A test generation framework for *uiescent* real-time systems. In *FATES*, pages 64–78, 2004.

[8] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *5th Annual IEEE Symposium on Logic in Computer Science*, IEEE, pages 428–439, 1990.

[9] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–171, 1992.

[10] Rachel Cardell-oliver. Conformance testing of real-time systems with timed automata specifications. *Formal Aspects of Computing*, 12(5):350–371, 2000.

[11] E.M. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT Press, 1999.

[12] Steven J. Cunning and Jerzy W. Rozenblit. Automating test generation for discrete event oriented embedded systems. *J. Intell. Robotics Syst.*, 41(2-3):87–112, 2005.

[13] D.S. Dummit and R.M. Foote. *Abstract algebra*. Prentice Hall, 1999.

[14] A. En-Nouaary, R. Dssouli, F. Khendek, and A. Elqortobi. Timed test cases generation based on state characterization technique. In *RTSS '98: Proceedings of the IEEE Real-Time Systems Symposium*, page 220, Washington, DC, USA, 1998. IEEE Computer Society.

[15] Abdeslam En-Nouaary. A scalable method for testing real-time systems. *Software Quality Journal*, 16(1):3–22, 2008.

[16] Abdeslam En-Nouaary and Rachida Dssouli. A guided method for testing timed input output automata. In *TestCom*, pages 211–225, 2003.

[17] Abdeslam En-Nouaary, Rachida Dssouli, and Ferhat Khendek. Timed wp-method: Testing real-time systems. *IEEE Trans. Softw. Eng.*, 28(11):1023–1038, 2002.

[18] H. Fouchal, E. Petitjean, and S. Salva. Testing timed systems with timed purposes. In *RTCSA '00: Proceedings of the Seventh International Conference on Real-Time Systems and Applications*, page 166, Washington, DC, USA, 2000. IEEE Computer Society.

[19] Hacène Fouchal. Conformance testing techniques for timed systems. In *SOFSEM '02: Proceedings of the 29th Conference on Current Trends in Theory and Practice of Informatics*, pages 1–19, London, UK, 2002. Springer-Verlag.

[20] Angelo Gargantini. Conformance testing. In M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, editors, *Model-Based Testing of Reactive Systems: Advanced Lectures*, volume 3472 of *Lecture Notes in Computer Science*, pages 87–111. Springer-Verlag, 2004.

[21] Rainer Gawlick, Roberto Segala, Jørgen F. Søgaard-Andersen, and Nancy A. Lynch. Liveness in timed and untimed systems. In *ICALP '94: Proceedings of the 21st International Colloquium on Automata, Languages and Programming*, pages 166–177, London, UK, 1994. Springer-Verlag.

[22] Anders Hessel, Kim Guldstrand Larsen, Marius Mikucionis, Brian Nielsen, Paul Pettersson, and Arne Skou. Testing real-time systems using uppaal. In *Formal Methods and Testing*, pages 77–117, 2008.

[23] Bertrand Jeannet, Thierry Jéron, and Vlad Rusu. Model-based test selection for infinite-state reactive systems. In *FMCO*, pages 47–69, 2006.

[24] Dilsun Kirli Kaynar, Nancy A. Lynch, Roberto Segala, and Frits W. Vaandrager. *The Theory of Timed I/O Automata*. Synthesis Lectures on Computer Science. Morgan & Claypool Publishers, 2009.

[25] Ahmed Khoumsi. A supervisory control method for ensuring the conformance of real-time discrete event systems. *Discrete Event Dynamic Systems*, 15(4):397–431, 2005.

[26] M. Krichen and S. Tripakis. Black-box conformance testing for real-time systems. In *Model Checking Software: 11th International SPIN Workshop*, number 2989 in Lecture Notes in Computer Science, pages 109–126, Barcelona, Spain, April 2004.

[27] Moez Krichen and Stavros Tripakis. Conformance testing for real-time systems. *Form. Methods Syst. Des.*, 34(3):238–304, 2009.

[28] Kim Guldstrand Larsen and Wang Yi. Time-abstracted bisimulation: Implicit specifications and decidability. *Inf. Comput.*, 134(2):75–101, 1997.

[29] Gang Luo, G. von Bochmann, and A. Petrenko. Test selection based on communicating nondeterministic finite-state ma chines using a generalized wp-method. *IEEE Trans. Softw. Eng.*, 20(2):149–162, 1994.

[30] Mercedes Merayo, Manuel Nú nez, and Ismael Rodríguez. Extending efsms to specify and test timed systems with action duration s and time-outs. *IEEE Trans. Comput.*, 57(6):835–844, 2008.

[31] Brian Nielsen and Arne Skou. Test generation for time critical systems: Tool and case study. In *ECRTS*, pages 155–162, 2001.

[32] Alexandre Petrenko, Sergiy Boroday, and Roland Groz. Confirming configurations in efsm testing. *IEEE Trans. Softw. Eng.*, 30(1):29–42, 2004.

[33] Jan Springintveld, Frits Vaandrager, and Pedro R. D'Argenio. Testing timed automata. *Theor. Comput. Sci.*, 254(1-2):225–257, 2001.

[34] Jan Tretmans. Test generation with inputs, outputs, and quiescence. In Tiziana Margaria and Bernhard Steffen, editors, *Tools and Algorithms for Construction and Analysis of Systems, Second International Workshop, TACAS '96, Passau, Germany, March 27-29, 1996, Proceedings*, volume 1055 of *Lecture Notes in Computer Science*, pages 127–146, London, UK, 1996. Springer-Verlag.

[35] Jan Tretmans. Testing concurrent systems: A formal approach. In J.C.M Baeten and S. Mauw, editors, *CONCUR '99: Proceedings of the 10th International Conference on Co ncurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 46–65, London, UK, 1999. Springer-Verlag.

[36] Jan Tretmans. Model based testing with labelled transition systems. In *Formal Methods and Testing*, pages 1–38, 2008.

[37] Stavros Tripakis. Fault diagnosis for timed automata. In *FTRTFT '02: Proceedings of the 7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 205–224, London, UK, 2002. Springer-Verlag.

[38] Antti Valmari. The state explosion problem. In *Petri Nets*, pages 429–528, 1996.

[39] Chang-Jia Wang and Ming T. Liu. Generating test cases for efsm with given fault models. In *INFOCOM*, pages 774–781, 1993.