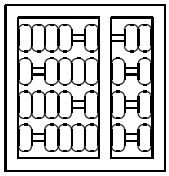

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

A New Timed Discretization Method for Automatic Test Generation for Timed Systems

A. Bonifácio *A. Moura*

Technical Report - IC-09-31 - Relatório Técnico

September - 2009 - Setembro

A New Discretization Method for Automatic Test Generation for Timed Systems

Adilson Luiz Bonifácio* Arnaldo Vieira Moura†

Abstract

Devising formal techniques and methods that can automatically generate test case suites for timed systems has remained a challenge. In this work we use Timed Input/Output Automata (TIOA) as a formal specification model for timed systems. We propose and prove the correctness of a new and more general discretization method that can be used to obtain grid automata corresponding to specification TIOA, using almost any granularity of interest. We also show how test purposes, for modeling specific system properties, can be used together with the specification TIOA in order to generate grid automata that captures the behavior of both the specification and the test purpose. From such grid automata one can, then, algorithmically extract test suites that can be used to verify whether given implementations conform to the specification and reflect the desired properties.

1 Introduction

In order to verify the correctness of computational systems, automatic test case generation methods have been intensively investigated. One of the most important and promising among such techniques is model-based testing [21, 6, 17, 20]. Although mathematical models of system requirements and formally specified system functionalities allow for some automation in the process of efficient generation of test case suites, devising techniques and methods that properly deal with critical and real-time systems has remained a challenge.

In this work we deal with timed system specifications and, in particular, with Timed Input/Output Automata (TIOA) [14, 7] which are a variant of the classical timed automata model [2, 1, 9, 5]. TIOA models can be used to specify timed systems and their executions, allowing continuous time evolution also to be represented in the models. In order to obtain a discrete representation for such models we use the notion of grid automata [8, 9], and show how they can be automatically obtained from the original timed models.

Discretization has already been proposed in connection with timed automata, in the form of clock regions [2, 16]. Here, however, we propose a different and more general notion of discretization, using adjusted values and limiting boundaries. This new proposal allows

*Computing Institute, University of Campinas, adilson@ic.unicamp.br, Supported by CNPq grant 141978/2008-2

†Computing Institute, University of Campinas, arnaldo@ic.unicamp.br, Supported by CNPq grant 472504/2007-0, and FAPESP grant

for a more relaxed way of choosing the granularities of interest. In fact, depending on the accuracy of the physical system being modeled, a corresponding grid automaton can be obtained for any desired granularity. In the next sections we expose the relationship between a specification TIOA and the corresponding grid automaton that results when the new discretization method is applied to the former. We also present proofs of correctness showing that the TIOA homomorphically simulates the grid automaton, and vice-versa. This forms the basis that will allow for the automation of test case generation methods that use the grid automata as a basis.

In order to test implementations against given specifications, we use the notion of a test purpose [19, 13]. A test purpose is a particular kind of TIOA that can be used to model specific properties of the system under test. Given a specification TIOA and a test purpose, their joint behavior is captured by computing the synchronous product between both models. Once the product is obtained, we can apply the discretization method to it, thus obtaining a grid automaton. Test sequences can then be automatically extracted from the resulting grid automaton. Having the test case suite, one can use conformance testing methods to verify whether implementations conform to the desired behaviors modelled by the original TIOA specification and test purpose [21].

This work is organized as follows. In Section 2, we define the TIOA model and some other important concepts. In Section 3, we present the new discretization method. We start with some basic concepts concerning bounding values and limiting functions in Subsection 3.1. Then, the grid construction and its properties are presented in Subsection 3.2, establishing the correctness of the homomorphic relationship between the TIOA and its corresponding grid automaton. Section 4 briefly discusses the process of generating timed test suites using the notions of test purpose and synchronous product. In Section 5 we discuss some related works. Finally, some concluding remarks appear in Section 6.

2 The TIOA model

In this section we define the Timed I/O Automata (TIOA) model. But first, we need the notions of timed words, clock variables and clock conditions.

2.1 Timed words

Time instants and time delays will be taken from the set of non-negative rationals[1], $\mathbb{Q}_{\geq}$.

Definition 1 *(Timed word) Let Σ be an alphabet. A* timed word *over Σ is a finite sequence $\langle \sigma_1, \sigma_2, \cdots, \sigma_n \rangle$, where $n \geq 0$ and $\sigma_i \in \Sigma \cup \mathbb{Q}_{\geq}$, for all i, $1 \leq i \leq n$.* ∎

For example, let $\Sigma = \{a, b, c, d\}$. Then $\langle 1, b, 2.3, d, d, 5.1, a \rangle$ is a timed word over Σ, and so is $\langle c, 5, a, 4 \rangle$. The intended interpretation is that a timed word like $\langle 1, c, 2, a \rangle$ can be seen as a symbol c arriving after one time unit and a symbol a arriving at $1 + 2 = 3$ time units, counting from the start. A time word like $\langle 1, c, a \rangle$ can be interpreted as $\langle 1, c, 0, a \rangle$, saying that symbols c and a arrive at the same time, but with c preceding a. Also, syntactically,

[1] $\mathbb{Q}$ is the set of rationals, $\mathbb{Q}_{\geq}$ is the set of non-negative rationals and $\mathbb{Q}_{>}$ is the set of positive rationals.

a timed word like $\langle 1, c, 2, 5, a\rangle$ is not the same as the timed word $\langle 1, c, 7, a\rangle$, although they convey the same intended information.

When there is no risk of confusion, we may also write $\sigma_1, \sigma_2, \ldots, \sigma_n$, or even $\sigma_1\sigma_2 \ldots \sigma_n$, in place of $\langle \sigma_1, \sigma_2, \cdots, \sigma_n\rangle$. The empty timed word will be denoted by ε. The set of all timed words over Σ will be denoted by Ψ_Σ, or simply Ψ if there is no room for confusion. The concatenation of timed words follows the standard definition for concatenation of finite sequences. That is, if $\psi_1 = \langle \sigma_1, \ldots, \sigma_n\rangle \in \Psi_\Sigma$ and $\psi_2 = \langle \sigma_{n+1}, \ldots, \sigma_{n+m}\rangle \in \Psi_\Sigma$, with $n, m \geq 0$, then $\psi_1 \cdot \psi_2 = \langle \sigma_1 \ldots \sigma_n, \sigma_{n+1}, \ldots, \sigma_{n+m}\rangle$. We may also write $\psi_1\psi_2$ instead of $\psi_1 \cdot \psi_2$. As an example, let $\psi_1 = \langle 1, a, 2, 5, b, c, 3\rangle$ and $\psi_2 = \langle 4, a, d, 2\rangle$. Then, we have $\psi_1 \cdot \psi_2 = \langle 1, a, 2, 5, b, c, 3, 4, a, d, 2\rangle$.

2.2 Clocks

We will also need a set of clock variables, or clocks for short, denoted by C. The set of all clock 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 clock variable and $\tau \in \mathbb{Q}_\geq$ is a time instant. We will take the usual liberties when writing clock 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 $(\tau_1 \leq c) \wedge (c \leq \tau_2)$. A clock interpretation over C is a partial function from C into $\mathbb{Q}_\geq$. A total clock interpretation over C is a clock interpretation over C whose domain[2] is C. The set of all clock interpretations over C will be denoted by $[C \rightharpoonup \mathbb{Q}_\geq]$, and $[C \to \mathbb{Q}_\geq]$ will denote the set of all total clock interpretations over C. Clearly, $[C \rightharpoonup \mathbb{Q}_\geq] \subseteq [C \to \mathbb{Q}_\geq]$. When the intended set C is clear from the context, we may write clock interpretation, or simply interpretation, instead of clock interpretation over C.

Let $\delta \in \Phi_C$ and let $\nu \in [C \rightharpoonup \mathbb{Q}_\geq]$ be such that all clock variables occurring in δ are in $\mathrm{dom}(\nu)$. Then we say that ν satisfies δ, denoted by $\nu \vDash \delta$, if δ evaluates to true when every clock c is replaced by $\nu(c)$ in δ and the value of the resulting propositional logic sentence is computed in the usual manner. When $\mathrm{dom}(\nu) = \emptyset$ we have that $\Phi_C = \{\texttt{true}\}$, and so $\nu \vDash \texttt{true}$ always holds, for all clock interpretation $\nu \in [C \rightharpoonup \mathbb{Q}_\geq]$.

Let $\nu \in [C \rightharpoonup \mathbb{Q}_\geq]$ be an interpretation and let $\tau \in \mathbb{Q}_\geq$ be a time delay. The interpretation $\nu + \tau$ is defined as

$$(\nu + \tau)(c) = \begin{cases} \nu(c) + \tau & \text{if } c \in \mathrm{dom}(\nu) \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Let $\mu \in [C \rightharpoonup \mathbb{Q}_\geq]$. We define the interpretation $\nu \oplus \mu$ thus

$$(\nu \oplus \mu)(c) = \begin{cases} \mu(c) & \text{if } c \in \mathrm{dom}(\mu) \\ \nu(c) & \text{if } c \in (\mathrm{dom}(\nu) - \mathrm{dom}(\mu)) \\ \text{undefined} & \text{if } c \notin (\mathrm{dom}(\nu) \cup \mathrm{dom}(\mu)), \end{cases}$$

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

for all $c \in C$. That is, $\nu \oplus \mu$ assigns clock values first according to μ and, baring that, then it assigns values according to ν, if at all possible. Note that when ν or μ is total, then so is $\nu \oplus \mu$.

2.3 Timed Input Output Automata

The Timed I/O Automata (TIOA) model is based on the BTDA (*Bounded Time Domain Automaton*) model of Gawlick et al [14]. A BTDA is composed by states, action symbols, clock variables, state invariants and transitions. The system progresses by a sequence of continuous time evolutions interrupted by discrete transitions [1, 4, 18]. During a continuous time evolution, the state does not change and its invariant must stay verified at all instants. A discrete transition is specified by a source and a target state, an action symbol, a transition guard and a (partial) clock reset function. We now make these concepts precise.

Definition 2 *(BTDA) A BTDA is a tuple* $(S, s_0, \Sigma, C, \nu_0, Inv, T)$*, where* S *is a finite set of states,* $s_0 \in S$ *is the initial state,* Σ *is the set of action symbols,* C *is a set of clocks,* $\nu_0 \in [C \to \mathbb{Q}_{\geq}]$ *is the initial clock interpretation, where* $\nu_0(c) = 0$ *for all* $c \in C$*,* $Inv : S \to \Phi_C$ *maps states to state invariants, and* T *is the set of transitions, where* $T \subseteq (S \times \Sigma \times \Phi_C \times [C \rightharpoonup \mathbb{Q}_{\geq}] \times S)$*.* ∎

The intended meaning for a transition $(s, z, \delta, \theta, r)$ is that the machine can move to state r from state s over the symbol z provided that the guard δ is 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. These notions will be made precise shortly. As an example, Figure 1 shows a BTDA with one clock variable, c, and two action symbols, $\Sigma = \{on, off\}$. The set of states is $S = \{q_0, q_1\}$, with the initial state q_0 being marked by an incoming arrow with no source node. The invariants are indicated right next to the corresponding state; for example, $Inv(q_0)$ is $c \leq K$, where we take $K > 5$. The remaining arrows indicate transitions. Next to each arrow we can see the corresponding action symbol, the logical expression is the transition guard and the attribution like notation indicates the clock resetting partial map. For example, the transition from q_0 to q_1 would write as $(q_0, on, c \leq K, \theta, q_1)$, where $\theta(c) = 0$.

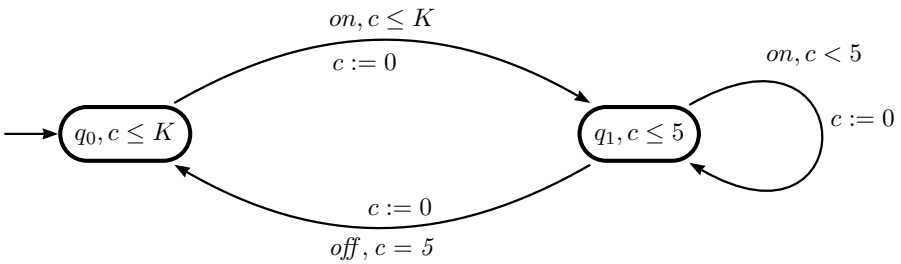

Figure 1: A BTDA model of a simple switch.

From now on, B will always denote a BTDA $B = (S, s_0, \Sigma, C, Inv, T)$, and we let any decorations carry over uniformly to the components of B, *e.g.*, $B' = (S', s_0', \Sigma', C', Inv', T')$.

In order to specify the movements of a BTDA, we need the notion of a configuration. A configuration of a BTDA B is a pair (s, ν), where $s \in S$ is a state and $\nu \in [C \to \mathbb{Q}_{\geq}]$ is a total clock interpretation over C. The set of all configurations of B is denoted by Γ_B, or simply by Γ if no confusion can arise. The initial configuration is (s_0, ν_0), where $\nu_0(c) = 0$ for all $c \in C$. We require that $\nu_0 \vDash Inv(s_0)$. In order to ease the notation, when the set of clocks is small we indicate a configuration simply by listing the corresponding state followed by a list of the clock values, in some pre-arranged order. For example, from Figure 1 we can see that (q_0, ν) is a configuration, where $\nu(c) = 1.5$. We may also write $(q_0, 1.5)$ to indicate this same configuration.

We are now ready to specify the transitions of a BTDA.

Definition 3 *(BTDA semantics) Let B be a BTDA and let $\gamma_i = (s_i, \nu_i) \in \Gamma_B$, $i = 1, 2$, be two configurations of B. We define:*

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

2. *Let $x \in \Sigma$ be an action symbol. Then there exists a* discrete movement *from γ_1 to γ_2 over x, denoted by $\gamma_1 \overset{x}{\to} \gamma_2$, if and only if there exists a transition $(s_1, x, \delta, \theta, s_2) \in T$ such that: (i) $\nu_1 \vDash \delta$; (ii) $\nu_2 = \nu_1 \oplus \theta$; and (iii) $\nu_2 \vDash Inv(s_2)$.* ∎

In a discrete movement the state changes while the clock interpretation remains fixed, except for a subset of the clocks that are reset as indicated by the partial function θ. Moreover, we also require that the corresponding transition guard is enabled and that the new clock interpretation satisfies the invariant of the target state. By way of contrast, in a continuous movement the state does not change and the clock interpretation absorbs the delay τ. Note that the state invariant must remain satisfied during the whole period of the continuous movement. Clearly, we always have $\gamma \underset{0}{\to} \gamma$, for all configurations γ.

As an example, from Figure 1 we have: $(q_0, 0) \overset{on}{\to} (q_1, 0)$, $(q_1, 0) \underset{5}{\to} (q_1, 5)$, $(q_1, 5) \overset{off}{\to} (q_0, 0)$, and $(q_0, 0) \underset{1}{\to} (q_0, 1)$.

Note that when $\alpha \underset{\tau}{\to} \beta$, with $\alpha, \beta \in \Gamma_B$, then for all η, $0 \leq \eta \leq \tau$, there is a $\beta' \in \Gamma_B$ such that $\alpha \underset{\eta}{\to} \beta'$. Furthermore, we may also have infinite sequence of continuous movements,

$$\alpha \underset{\tau_1}{\to} \beta_1 \underset{\tau_2}{\to} \beta_2 \underset{\tau_3}{\to} \beta_3 \cdots,$$

where $\beta_i \in \Gamma_B$ and, *e.g.*, $\tau_i = \tau / 2^i$, for all $i \geq 1$.

Starting from a configuration, a timed word induces a movement in a BTDA.

Definition 4 *(BTDA movement) Let B be a BTDA. The* movement relation *of B, $\vdash_B$, is a binary relation over $\Psi_\Sigma \times \Gamma_B$ given by $(\psi_1, \gamma_1) \vdash_B (\psi_2, \gamma_2)$ if and only if $\psi_1 = \langle \sigma, \sigma_1, \ldots, \sigma_n \rangle$, $\psi_2 = \langle \sigma_1, \ldots, \sigma_n \rangle$, with $n \geq 0$, and either (i) $\sigma \in \mathbb{Q}_{\geq}$ and $\gamma_1 \underset{\sigma}{\to} \gamma_2$; or (ii) $\sigma \in \Sigma$ and $\gamma_1 \overset{\sigma}{\to} \gamma_2$.* ∎

The k-th power of $\underset{B}{\vdash}$ will be indicated by $\overset{k}{\underset{B}{\vdash}}$, $k \geq 0$, and its reflexive transitive closure by $\overset{\star}{\underset{B}{\vdash}}$. As usual, we may drop decorations when no confusion can arise.

As an example, consider the timed word $\psi = \langle K/2, on, 3, on, 5, off, K/3\rangle$ and the BTDA in Figure 1. Then, we have

$$\begin{aligned}\big(\psi, (q_0, 0)\big) &\vdash \big(\langle on, 3, on, 5, off, K/3\rangle, (q_0, K/2)\big) \vdash \big(\langle 3, on, 5, off, K/3\rangle, (q_1, 0)\big)\\ &\vdash \big(\langle on, 5, off, K/3\rangle, (q_1, 3)\big) \vdash \big(\langle 5, off, K/3\rangle, (q_1, 0)\big)\\ &\vdash \big(\langle off, K/3\rangle, (q_1, 5)\big) \vdash \big(\langle K/3\rangle, (q_0, 0)\big) \vdash \big(\varepsilon, (q_0, K/3)\big).\end{aligned}$$

Further, when $(\psi, \gamma) \vdash (\varepsilon, \rho)$ we also write $\gamma \overset{\psi}{\Vdash} \rho$, or $\gamma \Vdash \rho$ when the particular timed word is not relevant.

When $\gamma \overset{\psi}{\Vdash} \rho$ we say that ψ is a *run starting* at γ and *ending* at ρ. If, moreover, γ is the initial configuration of the BTDA, then ψ is an *execution ending* at ρ. A configuration γ is *reachable* if there is an execution ending at γ. Likewise, a state s is *reachable* if there is a reachable configuration (s, ν), for some interpretation ν. Clearly, as can be seen from Figure 1, we can write $(q_0, 0) \overset{\langle on, 5\rangle}{\Vdash} (q_1, 5)$, thus configuration $(q_1, 5)$ and state q_1 are reachable. Also, it is easy to see that configuration $(q_1, 6)$ is not reachable.

Timed I/O Automata [7] extend the BTDA model by partitioning the set of actions into input and output actions. Input actions are considered stimulus from the environment while output actions will be generated by the system.

Definition 5 *(TIOA) A* Timed Input/Output Automaton *is a triple $M = (B, X, Y)$ where:*

1. *$B = (S, s_0, \Sigma, C, \nu_0, Inv, T)$ is a BTDA, the subjacent BTDA of M, and*
2. *$\{X, Y\}$ partitions Σ into a set X of input actions and a set Y of output actions.* ∎

By letting $X = \{on\}$ and $Y = \{off\}$ in Figure 1, we partition the actions of the BTDA, thereby obtaining a TIOA. From now on, M will always denote the TIOA $M = (B, X, Y)$, and any decorations will carry over uniformly to the components of M. The set of configurations of M, Γ_M, as well as the notion of movements of M are taken directly from the corresponding notions for the subjacent BTDA B, as in Definitions 2 and 3.

3 TIOA Discretization

The number of test sequences for any TIOA is infinite. Clearly, it is not practical to consider the set of all such sequences when putting a system to the test. An alternative is to use some sort of discretization of the original TIOA. Such discretizations lead to the notion of a grid automaton [8, 9]. In this section we make these notions precise.

3.1 Bounded values and functions

In what follows, 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, $t = \lfloor t \rfloor + \lceil t \rceil$ always holds. For further reference, we note the following simple facts that will be useful later.

Fact 6 *Let $x, y \in \mathbb{Q}_{\geq}$. If $x \geq y$ then $\lfloor x \rfloor \geq \lfloor y \rfloor$.*

Proof 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. ∎

Fact 7 *Let $x, y \in \mathbb{Q}_{\geq}$ and let k a positive integer. If $x = y + k$ then $\lceil x \rceil = \lceil y \rceil$ and $\lfloor x \rfloor = \lfloor y \rfloor + k$.*

Proof 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 $\lceil x \rceil = \lceil y \rceil$. ∎

Fact 8 *Let $x, y, z \in \mathbb{Q}_{\geq}$. If $x = y + z$ then $\lceil x \rceil = \lceil \lceil y \rceil + \lceil z \rceil \rceil$ and $\lfloor x \rfloor = \lfloor y \rfloor + \lfloor z \rfloor + \lfloor \lceil y \rceil + \lceil z \rceil \rfloor$.*

Proof 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 $\lceil x \rceil = \lceil \lceil y \rceil + \lceil z \rceil \rceil$. ∎

Fact 9 *Let $x, y, z \in \mathbb{Q}_{\geq}$. If $\lceil x \rceil = \lceil y \rceil$ then $\lceil x + z \rceil = \lceil y + z \rceil$.*

Proof We have $\lceil x + z \rceil = \lceil \lceil x \rceil + \lceil z \rceil \rceil$, using Fact 8. 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 Fact 8 again. ∎

Next, we turn to the notion of bounded interpretations.

Definition 10 *(L bounds) Let $L \in \mathbb{Q}_{\geq}$ be a bound. Let $x \in \mathbb{Q}_{\geq}$ be a value, let A be a set and let $\alpha \in [A \to \mathbb{Q}_{\geq}]$ be a function. We define*

1. *The L-bounded x value, denoted x_L, is given by*

$$x_L = \begin{cases} x & \text{if } x \leq L \\ \lfloor L \rfloor + \lceil x \rceil & \text{otherwise.} \end{cases}$$ ∎

2. *The L-bounded α function, denoted α_L, is obtained by letting $\alpha_L(a) = (\alpha(a))_L$, for all $a \in A$.*

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

Fact 11 *Let $L \in \mathbb{Q}_{\geq}$ be a bound and let $x \in \mathbb{Q}_{\geq}$ be a value. Then $\lceil x_L \rceil = \lceil x \rceil$ and $x_L \leq x$.*

Proof The equality is immediate from Definition 10. 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 6, we get $\lfloor x \rfloor \geq \lfloor L \rfloor$. Then, $x_L \leq \lfloor x \rfloor + \lceil x \rceil = x$, and we are done. ∎

Fact 12 *Let $L \in \mathbb{Q}_{\geq}$ be a bound and let $x \in \mathbb{Q}_{\geq}$ be a value. If $x < \lfloor L \rfloor + 1$ then $x_L = x$.*

Proof 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 6. 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. ∎

Fact 13 *Let $L \in \mathbb{Q}_{\geq}$ be a bound and let $x \in \mathbb{Q}_{\geq}$ be a value. Then $x_L < \lfloor L \rfloor + 1$.*

Proof If $x < \lfloor L \rfloor + 1$ then $x_L = x$, using Fact 12. 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 $\lceil x \rceil < 1$. ∎

Fact 14 *Let $L \in \mathbb{Q}_{\geq}$ be a bound and let $x \in \mathbb{Q}_{\geq}$ be a value. Then $(x_L)_L = x_L$.*

Proof From Fact 13, we know that $x_L < \lfloor L \rfloor + 1$. Now, using Fact 12 we get $(x_L)_L = x_L$. ∎

The next three propositions state 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 15 *(L sum 1) Let $L \in \mathbb{Q}_{\geq}$ be a bound and let x, $y \in \mathbb{Q}_{\geq}$. Then $(x + y)_L = (x + y_L)_L$.*

Proof There are two cases.

Case 1: $y < \lfloor L \rfloor + 1$. Then, by Fact 12, $y = y_L$ 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 12, $(x + y_L)_L = x + y_L = x + \lfloor L \rfloor + \lceil y \rceil$. 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 8 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 7, $\lceil x + y_L \rceil = \lceil x + \lfloor L \rfloor + \lceil y \rceil \rceil = \lceil x + \lceil y \rceil \rceil$. And, by Fact 7 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. ∎

Proposition 16 *(L sum 2) Let $L \in \mathbb{Q}_{\geq}$ be a bound and let x, $y \in \mathbb{Q}_{\geq}$. Then $(x + y)_L = (x_L + y_L)_L$.*

Proof From Proposition 15 we get $(x + y)_L = (x + y_L)_L$. From Proposition 15 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 14, $((x_L + y_L)_L)_L = (x_L + y_L)_L$. Then, $(x + y)_L = (x_L + y_L)_L$. ∎

Proposition 17 *(L sum 3) Let $L \in \mathbb{Q}_{\geq}$ be a bound and let x, $y \in \mathbb{Q}_{\geq}$. Let $x' = x$ or $x' = x_L$ and let $y' = y$ or $y' = y_L$. Then $(x + y)_L = (x' + y')_L$.*

Proof If $x' = x$ and $y' = y$ we are done. If $x' = x$ and $y' = y_L$ use Proposition 15. Similarly, if $x' = x_L$ and $y' = y$. Finally, if $x' = x_L$ and $y' = y_L$, use Proposition 16. ∎

We can now extend these results for larger sums.

Proposition 18 *(Arbitrary L sum) Let $L \in \mathbb{Q}_{\geq}$ be a bound and let $x^i \in \mathbb{Q}_{\geq}$ be values, 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*

$$\big(\sum_{i=1}^{n} x^i\big)_L = \big(\sum_{i=1}^{n} y^i\big)_L.$$

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 14.

Now, assume the result holds for some $n \geq 1$. Using Proposition 17, we get

$$\big(\sum_{i=1}^{n+1} x^i\big)_L = \big((\sum_{i=1}^{n} x^i) + x^{n+1}\big)_L = \big((\sum_{i=1}^{n} x^i)_L + y^{n+1}\big)_L.$$

Using the induction hypothesis and proposition 17, we get

$$\big(\sum_{i=1}^{n+1} x^i\big)_L = \big((\sum_{i=1}^{n} y^i)_L + y^{n+1}\big)_L = \big((\sum_{i=1}^{n} y^i) + y^{n+1}\big)_L = \big(\sum_{i=1}^{n+1} y^i\big)_L,$$

as desired. ∎

Proposition 19 *(Bounding functions) Let $L \in \mathbb{Q}_{\geq}$ be a bound. Let $W = \{w_1, \ldots, w_n\}$ be a set, 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*

$$\Big[\sum_{i=1}^{n} k_i \alpha(w_i)\Big]_L = \Big[\sum_{i=1}^{n} k_i \alpha_L(w_i)\Big]_L.$$

Proof Using Proposition 18 we get $\Big[\sum_{i=1}^{n} k_i \alpha(w_i)\Big]_L = \Big[\sum_{i=1}^{n} \big(k_i \alpha(w_i)\big)_L\Big]_L$.

Now, $\Big[k_i \alpha(w_i)\Big]_L = \Big[\sum_{j=1}^{k_i} \alpha(w_i)\Big]_L$.

Hence, using Proposition 18 again, we obtain $\Big[k_i \alpha(w_i)\Big]_L = \Big[\sum_{j=1}^{k_i} (\alpha(w_i))_L\Big]_L = \Big[k_i \alpha_L(w_i)\Big]_L$.

Putting it together, we have $\Big[\sum_{i=1}^{n} k_i \alpha(w_i)\Big]_L = \Big[\sum_{i=1}^{n} \big(k_i \alpha_L(w_i)\big)_L\Big]_L$.

Finally, using Proposition 18 once more, we get $\Big[\sum_{i=1}^{n} k_i \alpha(w_i)\Big]_L = \Big[\sum_{i=1}^{n} k_i \alpha_L(w_i)\Big]_L$,

as desired. ∎

In order to obtain a finite number of clock interpretations in the grid automata, to be defined shortly, we impose an upper bound on clock values. This notion is similar to the idea of clock regions [2, 22].

Definition 20 *(Clock bounds) Let C be a set of clocks and let $L \in \mathbb{Q}_{\geq}$ be a time instant. For any $\nu \in [C \to \mathbb{Q}_{\geq}]$, the L-bounded clock interpretation $\nu_L \in [C \to \mathbb{Q}_{\geq}]$, is constructed as in Definition 10 (2).* ∎

We also refer to ν_L as the clock interpretation ν bounded by L. Note that we could have specified a different bound L_c for each clock c. In order to keep the notation uncluttered, however we will consider a single bound L for all clock variables. It should be a simple matter to generalize all results to the case when some clock bounds may be distinct.

The next two propositions express the result of L-bounding time delays and clock resets.

Proposition 21 *(Bounding time delay) Let C be a set of clocks and let $\nu \in [C \to \mathbb{Q}_{\geq}]$ be a clock interpretation, $\eta \in \mathbb{Q}_{\geq}$, and let L be a positive integer. Then $(\nu + \eta)_L = (\nu_L + \eta)_L$.*

Proof It suffices to show that $(\nu + \eta)_L(c) = (\nu_L + \eta)_L(c)$, for all $c \in C$. We know that, by definition, $(\nu + \eta)_L(c) = \big[(\nu + \eta)(c)\big]_L = \big[\nu(c) + \eta\big]_L$. Using Proposition 18, we now obtain $\big[\nu(c) + \eta\big]_L = \big[(\nu(c))_L + \eta\big]_L$. Using the definitions again, $\big[(\nu(c))_L + \eta\big]_L = \big[\nu_L(c) + \eta\big]_L = (\nu_L + \eta)_L(c)$, and the result follows. ∎

Proposition 22 *(Bounding clock reset) Let C be a set of clocks and let $\nu \in [C \to \mathbb{Q}_{\geq}]$ and $\theta \in [C \rightharpoonup \mathbb{Q}_{\geq}]$ be clock interpretations, and let L be a positive integer. Then $(\nu \oplus \theta)_L = (\nu_L \oplus \theta)_L$.*

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 14. If $c \in \mathrm{dom}(\theta)$, we have $(\nu \oplus \theta)_L(c) = \theta_L(c)$ and $(\nu_L \oplus \theta)_L(c) = \theta_L(c)$, completing the proof. ∎

In the next two lemmas we present some useful relationships between limited interpretations, interpretations and evaluating conditions.

Lemma 23 *(Bounded evolution) Let C be a set of clocks and let α^i, $\beta^i \in [C \to \mathbb{Q}_{\geq}]$, $i = 1, 2$, be clock interpretations. Assume that $\beta^i = \alpha^i$ or $\beta^i = (\alpha^i)_L$ for all $i \in \{1, 2\}$. Further, let $I \in \Phi_C$ be a clock condition and let L be a positive integer greater than all constants occurring in I. Then $\alpha^1 + \alpha^2 \vDash I$ iff $\beta^1 + \beta^2 \vDash I$.*

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

From Fact 6 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, assume $\beta^1(c) + \beta^2(c) \leq \tau$. 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. In any case, we contradict $\beta^1(c) + \beta^2(c) \leq L$.

Similarly, we cannot have $\alpha^2(c) > L$. Hence, $\alpha^i(c) \leq L$ and we get $\beta^i(c) = \alpha^i(c)$, for $i = 1, 2$. Then, since $\beta^1(c) + \beta^2(c) \leq \tau$, we also 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 12, 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:

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 $\beta^1 + \beta^2 \vDash I$.

I-2B: δ is $(\delta_1 \wedge \delta_2)$, that is, I is equivalent to $(\neg\delta_1) \vee (\neg\delta_2)$. Note that δ has n propositional connectives. Then 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 get $\alpha^1 + \alpha^2 \vDash \neg\delta_i$ 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. ∎

Lemma 24 *(Bounded reset) Let C be a set of clocks and let $\nu \in [C \to \mathbb{Q}_{\geq}]$ and $\theta \in [C \curvearrowright \mathbb{Q}_{\geq}]$ be clock interpretations and let $I \in \Phi_C$ be a clock condition. If $\nu \oplus \theta \vDash I$ then $\nu_L \oplus \theta \vDash I$, when L is a positive integer greater than all constants occurring in I.*

Proof 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 Lemma 23, 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, $\nu_L \oplus \theta \vDash I$.

Now, we proceed by induction on the number $n \geq 0$ of propositional 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 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 $\delta_1 \wedge \delta_2$.

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:

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 $\nu_L \oplus \theta \vDash I$.

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 $\nu \oplus \theta \vDash \neg\delta_2$. Note that δ has n propositional connectives. Then 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 $\nu_L \oplus \theta \vDash I$. ∎

3.2 Grid automata

Given a TIOA M, in order to deal with a finite set of useful interpretations we choose a time granularity and discretize the movements of M. The granularity has to be chosen in a such way that neither it is too small, rendering the system almost continuous, nor is it too large, making the approximations too coarse. Some works [8, 9, 16] suggest a possible discretization step of $1/(n+1)$, if $n = 1$, and $1/(n+2)$, with $n \geq 2$, where n is the number of clocks in M.

In this work we show how to use any granularity $g = 1/k$, k being a positive integer, and still obtain an adequate discretization. Once a boundary L is also chosen, the behavior of M should be reproducible in the time interval of interest, from 0 to L, using L-bounded interpretations.

We start with the notion of adjusted values.

Definition 25 *(Adjusted values) Let $g \in \mathbb{Q}_{\geq}$. Then*

1. *A value $\ell \in \mathbb{Q}_{\geq}$ is* g-adjusted *iff ℓ is an integer multiple of g.*
2. *Let C be a set of clocks. A clock condition $\delta \in \Phi_C$ is* g-adjusted *iff all constants occurring in δ are g-adjusted values.*
3. *Let C be a set of clocks. A clock interpretation $\nu \in [C \rightharpoonup \mathbb{Q}_{\geq}]$ is* g-adjusted *iff $\nu(c)$ is a g-adjusted value, for all $c \in dom(\nu)$.* ∎

When the value g can be inferred from the context, we may write adjusted instead of g-adjusted.

Two simple facts about adjusted values follow.

Proposition 26 *(Adjusted parts) Let $g = 1/k$, with k a positive integer, and let $\ell \in \mathbb{Q}_{\geq}$. Then ℓ is a g-adjusted value iff 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 27 *(Adjusted bounded) Let $g = 1/k$, with k a positive integer, and let $L \in \mathbb{Q}_{\geq}$ be a g-adjusted value. Also, let C be a set of clocks and let $\nu \in [C \rightharpoonup \mathbb{Q}_{\geq}]$ be a g-adjusted clock interpretation. Then ν_L is also a g-adjusted clock interpretation.*

Proof Let $c \in C$. From Definition 20, we get $\nu_L(c) = \nu(c)$ or $\nu_L(c) = \lfloor L \rfloor + \lceil \nu(c) \rceil$. From Proposition 26, 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. ∎

Another simple fact states that extending g-adjusted clock interpretations, or resetting a g-adjusted clock interpretation, always results in a new clock interpretation that is also g-adjusted.

Proposition 28 *(Adjusted interpretation) Let C be a set of clocks, let ν, $\eta \in [C \curvearrowright \mathbb{Q}_{\geq}]$ be two g-adjusted clock interpretations, and let ℓ be a g-adjusted value. Then $\nu + \ell$ and $\nu \oplus \eta$ are also g-adjusted clock interpretations.*

Proof 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 25, $\nu + \ell$ 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. ∎

Next, we show that any set of L-bounded clock interpretations is finite, provided that L is properly adjusted.

Lemma 29 *(Number of bounded clock interpretations) Let C be a set of clocks, and let $g = 1/k$ with k a positive integer. Let $R \subseteq [C \curvearrowright \mathbb{Q}_{\geq}]$ be a set of g-adjusted clock interpretations, and let $L \in \mathbb{Q}_{\geq}$ be a g-adjusted value. Consider the L-bounded set $R_L = \{\nu_L \,|\, \nu \in R\}$. 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 27, t is always g-adjusted. Moreover, $t < \lfloor L \rfloor + 1$, by Fact 13. 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. ∎

Had we used a distinct g-adjusted bound L_c for each clock c, the lemma would yield the bound $|R_L| \leq \prod_{c \in C} (k\lfloor L_c \rfloor + k)$.

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

Definition 30 *(Adjusted timed word) Let Σ be an alphabet. A timed word $\langle \sigma_1, \ldots, \sigma_n \rangle \in \Psi_\Sigma$ is g-adjusted if σ_i is a g-adjusted value whenever $\sigma_i \in \mathbb{Q}_{\geq}$, for all i, $1 \leq i \leq n$.* ∎

The set of all g-adjusted timed words over Σ will be denoted by $\Psi_{g,\Sigma}$. As before, we may drop subscripts if there is no reason for confusion.

A TIOA will be said adjusted if both its guards and invariants are also adjusted. The notion of an adjusted TIOA will be important in reducing the set of L-bounded interpretations to a finite size.

Definition 31 *(Adjusted TIOA) Let M be a TIOA. Then M is* g-adjusted *iff for all transitions $(s, z, \delta, \theta, r) \in T$ we have that δ is a g-adjusted clock condition and θ is a g-adjusted clock interpretation. Moreover, for all states $s \in S$, we require that $Inv(s)$ be a g-adjusted clock condition.* ∎

A run over a g-adjusted timed word implies the g-reachability of the terminal configuration.

Definition 32 *(TIOA g-reachability) Let M be a g-adjusted TIOA, $s \in S$ and $\nu \in [C \to \mathbb{Q}_{\geq}]$. We say that (s, ν) is g-reachable in M iff there is a g-adjusted timed word $\psi \in \Psi_{g,\Sigma}$, such that $(s_0, \nu_0) \Vdash_M^{\psi} (s, \nu)$.* ∎

The result obtained by the discretization of a TIOA M will be a labelled transition system, called a grid automaton [9, 8].

Definition 33 *(Grid Automaton) Let $g = 1/k$ with k a positive integer. Let M be a g-adjusted TIOA and let $L \in \mathbb{Q}_{\geq}$ be a g-adjusted value. Then the L, g-*grid automaton*, or simply L, g-*grid*, associated with M is the labelled transition system constructed by Algorithm 1.* ∎

Note that the input alphabet of $M_{L,g}$, Σ_G, is formed by the set Σ of all action symbols of M, together with the new symbol g. As usual, we may drop the qualifications L and g, when no confusion can arise.

Consider the TIOA depicted in Figure 2. Note that we have two clocks. Some previous works [8, 9, 16] suggest a granularity of at least $\frac{1}{4}$ when discretizing a TIOA with 2 clocks. Here, instead, we may choose coarser values. In fact, for this example we will choose the granularity $g = \frac{1}{2}$ and the bound $L = 6$. Further, let $K = 3$ in Figure 2. Under these assumptions, Figure 3 shows part of the grid automaton obtained from Algorithm 1, given Figure 2 as the input TIOA.

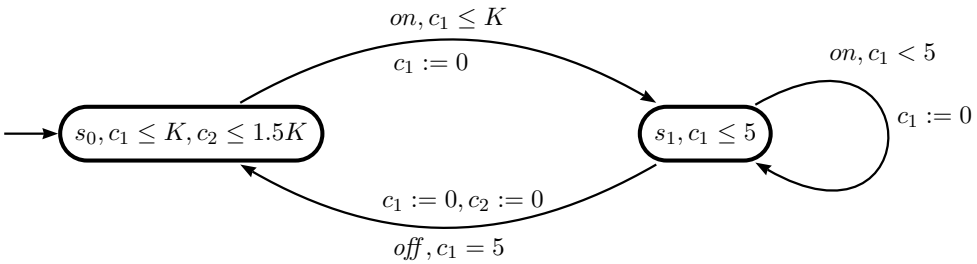

Figure 2: A TIOA model for a variation of the simple switch.

The next result proves that Algorithm 1 always terminates. Moreover, it also establishes a bound on the number of states in the resulting labelled transition system.

Lemma 34 *(Finite grid) Consider the procedure depicted as Algorithm 1. Then, it always halts with $|S_G| \leq |S| \times \big(k\lfloor L \rfloor + k\big)^{|C|}$.*

Proof Since L is g-adjusted and $g = 1/k$, Lemma 29 states that there are at most $\big(k\lfloor L \rfloor + k\big)^{|C|}$ L-bounded distinct clock interpretations, that is, $|[C \curvearrowright \mathbb{Q}_{\geq}]_L| \leq \big(k\lfloor L \rfloor + k\big)^{|C|}$. Then, there are at most $|S| \times \big(k\lfloor L \rfloor + k\big)^{|C|}$ configurations of the form (s, ν_L), where $s \in S$ and ν_L is an L-bounded clock interpretation of M.

1 **Input**: A value $g = 1/k$, with k a positive integer; a g-adjusted TIOA $M = (S, s, \Sigma, C, \nu, Inv, T)$, and a g-adjusted boundary $L \in \mathbb{Q}_{\geq}$.
2 **Output**: The L, g-grid $M_{L,g} = (S_G, s_G, \Sigma_G, T_G)$ associated with M.
3 **begin**
4 $T_G \leftarrow \emptyset$ // the set of transitions;
5 $RS \leftarrow s_G = (s, \nu)$ // where $\nu(c) = 0$ for all $c \in C$;
6 $HS \leftarrow \emptyset$ // the set of visited states;
7 **while** $RS \setminus HS \neq \emptyset$ **do**
8 **get** a state (s, ν) from $RS \setminus HS$ // choose a state;
9 **move** (s, ν) from RS to HS;
10 **foreach** $(s, z, \delta, \theta, r) \in T$ **do**
11 **if** $\nu \vDash \delta$ *and* $\nu \oplus \theta \vDash Inv(r)$ **then**
12 **let** $\eta = (\nu \oplus \theta)_L$;
13 **add** the transition $((s, \nu), z, (r, \eta))$ to T_G ;
14 **add** the state (r, η) to RS, if $(r, \eta) \notin HS$;
15 **end**
16 **end**
17 **if** $\nu + h \vDash Inv(s)$ *for all* $0 < h \leq g$ **then**
18 **let** $\eta = (\nu + g)_L$;
19 **add** the transition $((s, \nu), g, (s, \eta))$ to T_G;
20 **add** the state (s, η) to RS, if $(s, \eta) \notin HS$;
21 **end**
22 **end**
23 $S_G \leftarrow HS$;
24 $\Sigma_G \leftarrow \Sigma \cup \{g\}$;
25 **return**;
26 **end**

Algorithm 1: Grid algorithm.

Notice that ν_0, at line 5, is also g-adjusted and L-bounded, trivially. Then, using Proposition 28 and Proposition 27 we know that the interpretations constructed at lines 12 and 18 are also g-adjusted. Clearly, then whenever a configuration (p, η) is added to RS at lines 14 and 20, we have that η is a g-adjusted clock interpretation.

Note that at lines 14 and 20 a state (s, η) is added to RS only if it is not already in HS. Thus, a state $(s, \eta) \in S \times [C \curvearrowright \mathbb{Q}_{\geq}]_L$ enters RS at most once. Since the loop at line 7 moves one state (s, ν) from $RS \setminus HS$ into HS, we conclude that the loop at line 7 executes for at most $|S| \times (k\lfloor L \rfloor + k)^{|C|}$ times. Thus, clearly, the procedure halts and $|HS| \leq |S| \times |[C \curvearrowright \mathbb{Q}_{\geq}]_L|$. From line 23, we get $|S_G| \leq |S| \times \big(k\lfloor L \rfloor + k + 1\big)^{|C|}$. ∎

Note that, when we specify a possibly different bound L_c for each clock $c \in C$, the number of states in the grid automaton will be bounded by $k^{|C|} \times \prod_{c \in C} (\lfloor L_c \rfloor + k)$, which can be much

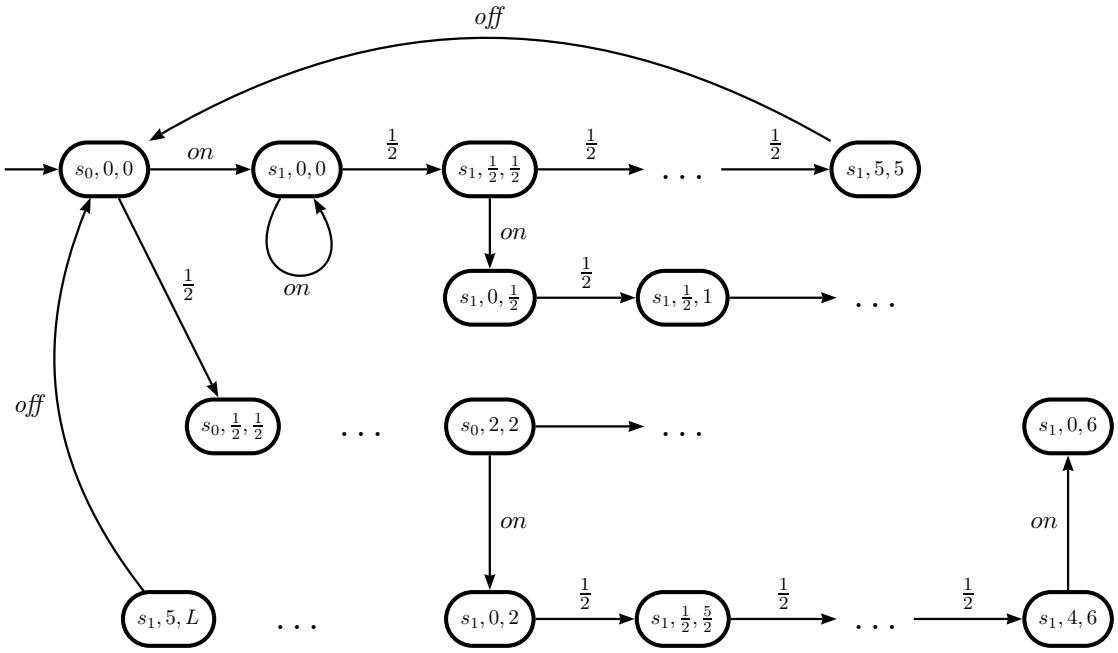

Figure 3: The partial grid automaton for Figure 2.

smaller then $k^{|C|} \times (\lfloor L \rfloor + k)^{|C|}$, if we take the safe value $L = \max_{c \in C} \{L_c\}$.

Next, we want to prove that the grid and the corresponding TIOA display compatible behaviors, provided that the common input timed word is g-adjusted. But, before, we need to define grid movements.

Definition 35 *(Grid movement) Let M_G be a grid automaton. The* movement relation *of M_G, $\vdash_G$, is a binary relation over $\Sigma_G^\star \times S_G$ given by $(\sigma \cdot \psi, s_1) \vdash_G (\psi, s_2)$ if and only there is a transition $(s_1, \sigma, s_2) \in T_G$, for all $\psi \in \Sigma_G^\star$ and $\sigma \in \Sigma_G$.* ∎

The k-th power of $\vdash_G$ will be indicated by $\overset{k}{\vdash}_G$, $k \geq 0$, and its reflexive transitive closure by $\overset{\star}{\vdash}_G$. When $(\psi, \gamma) \overset{\star}{\vdash}_G (\varepsilon, \rho)$ we also write $\gamma \overset{\psi}{\Vdash}_G \rho$, or $\gamma \Vdash_G \rho$ when the particular input grid word is not relevant.

The next technical result will be useful later.

Lemma 36 *(Grid run) Let M be a TIOA and let M_G be the corresponding grid. Let L be a positive integer greater than all constants occurring in M, and let $g = 1/k$ with k a positive integer. Also, take $\mu \in [C \to \mathbb{Q}_{\geq}]$ and $p \in S$ with $(p, \mu_L) \in S_G$. Finally, assume*

that $\mu + \eta \vDash Inv(p)$, $0 < \eta \leq ig$ *for some* $i \geq 0$. *Then* $(p, \omega_L) \in S_G$ *and* $(p, \mu_L) \overset{g^i}{\underset{G}{\Vdash}} (p, \omega_L)$, *with* $\omega = \mu + ig$.

Proof Define $\mu^j = \mu + jg$, for all j, $0 \leq j \leq i$. It suffices to prove that[3] $(p, \mu_L^j) \in S_G$ and $(p, \mu_L) \overset{g^j}{\underset{G}{\Vdash}} (p, \mu_L^j)$, for all j, $0 \leq j \leq i$. Note that $\mu_L^i = (\mu + ig)_L = \omega_L$.

We proceed by induction on $j \geq 0$.

Basis: when $j = 0$ we get $g^j = \varepsilon$. Then, trivially, $(p, \mu_L) \overset{g^j}{\underset{G}{\Vdash}} (p, \mu_L)$. We show $\mu_L = \mu_L^j$.

We have $\mu^j = \mu + 0g = \mu + 0 = \mu$ and so $\mu_L^j = \mu_L$, as desired.

Induction step: assume the result holds for some j, $0 \leq j < i$.

The induction hypothesis gives $(p, \mu_L) \overset{g^j}{\underset{G}{\Vdash}} (p, \mu_L^j)$ and $(p, \mu_L^j) \in S_G$. Note that $(p, \mu_L^j) \in S_G$ gives $(p, \mu_L^j) \in HS$ at line 23 of Algorithm 1. Also, pairs are moved from RS into HS one at a time at line 9. Hence, at some iteration, (p, μ_L^j) was chosen at line 8. We show that now line 17 applies.

Let $0 < \eta \leq g$. We show that $\mu_L^j + \eta \vDash Inv(p)$. We have $0 < jg + \eta \leq (j+1)g \leq ig$. From the hypothesis we get $\mu + (jg + \eta) \vDash Inv(p)$, that is $(\mu + jg) + \eta \vDash Inv(p)$, and so $\mu^j + \eta \vDash Inv(p)$. Using Lemma 23 we get $\mu_L^j + \eta \vDash Inv(p)$, as desired. Thus, Algorithm 1, lines 18–20, will put $((p, \mu_L^j), g, (p, \rho_L))$ in T_G, with $\rho = \mu_L^j + g$. So, $(p, \mu_L^j) \overset{g}{\underset{G}{\Vdash}} (p, \rho_L)$. Also, by line 20, (p, ρ_L) will be added to RS if it is not already in HS. In any case, when the loop at line 7 terminates, we get (p, ρ_L) in HS and, by line 23, $(p, \rho_L) \in S_G$. Moreover, since $(p, \mu_L) \overset{g^j}{\underset{G}{\Vdash}} (p, \mu_L^j)$, we also get $(p, \mu_L) \overset{g^{j+1}}{\underset{G}{\Vdash}} (p, \rho_L)$. We extend the induction by showing that $\rho_L = \mu_L^{j+1}$. Since $\rho_L = ((\mu^j)_L + g)_L$, using Fact 21 we get $\rho_L = (\mu^j + g)_L = (\mu + jg + g)_L = (\mu + (j+1)g)_L = (\mu^{j+1})_L = \mu_L^{j+1}$, as desired. ∎

The next definitions map adjusted timed words into grid words, and vice-versa.

Definition 37 *(To grid word) Let* Σ *be an alphabet and let* $\psi \in \Psi_{g,\Sigma}$ *be a* g*-adjusted timed word. The mapping* $h : \Psi_{g,\Sigma} \to \Sigma_G^\star$ *is defined by letting* $h(\varepsilon) = \varepsilon$, *and for all* $\psi \in \Psi_{g,\Sigma}$ *and all* $\sigma \in \Sigma \cup \mathbb{Q}_{\geq}$:

$$h(\psi \cdot \langle \sigma \rangle) = \begin{cases} h(\psi) \cdot \sigma & \text{if } \sigma \in \Sigma \\ h(\psi) \cdot g^k & \text{where } \sigma = kg, \text{ for some } k \geq 0. \end{cases}$$ ∎

Definition 38 *(To adjusted timed word) Let* Σ *be an alphabet and let* $\psi \in \Sigma_G^\star$ *be a grid word. The function* $\widehat{h} : \Sigma_G^\star \to \Psi_{g,\Sigma}$ *is defined by letting* $\widehat{h}(\varepsilon) = \varepsilon$ *and, for all* $\varphi \in \Sigma_G^\star$ *and all* $\sigma \in \Sigma_G$, $\widehat{h}(\varphi \cdot \sigma) = \widehat{h}(\varphi) \cdot \langle \sigma \rangle$. ∎

[3] Here, μ_L^j denotes $(\mu^j)_L$.

Next we note that $\widehat{h}(\psi)$ is also g-adjusted.

Fact 39 *Let $\psi \in \Sigma_G^\star$. Then $\widehat{h}(\psi)$ is g-adjusted.*

Proof Immediate from Definition 38. ∎

The following lemma states that g-reachable configurations in a TIOA are states in the corresponding grid.

Lemma 40 *(Grid states) Let M_G be a grid corresponding to a TIOA M and let L be a positive integer greater than any constant occurring in M. If $(s,\nu) \in S \times [C \to \mathbb{Q}_{\geq}]$ is g-reachable in M then $(s,\nu_L) \in S_G$.*

Proof Using Definition 32, we know that $(s_0,\nu_0) \Vdash_M^{\psi} (s,\nu)$ for some g-adjusted timed word $\psi \in \Psi_{g,\Sigma}$. We proceed by induction on the length of ψ.

If $\psi = \varepsilon$ then $s_0 = s$ and $\nu_0 = \nu$. Algorithm 1, line 5, puts (s,ν) in RS. Now, the while loop at line 7, together with lines 8 and 9, will put (s_0,ν_0) in HS. Then, by line 23, we get $(s,\nu) \in S_G$.

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

From $(s_0,\nu_0) \Vdash_M^{\psi} (s,\nu)$ we obtain $(s_0,\nu_0) \Vdash_M^{\varphi} (r,\mu)$ and $(r,\mu) \Vdash_M^{\langle\sigma\rangle} (s,\nu)$ for some $r \in S$ and $\mu \in [C \to \mathbb{Q}_{\geq}]$. Since ψ is g-adjusted, we have that φ is g-adjusted. By the induction hypothesis, we get $(r,\mu_L) \in S_G$. Then, $(r,\mu_L) \in HS$ (line 23). Clearly, from line 6 of Algorithm 1, together with the while loop at line 7 and lines 8 and 9, we can conclude that (r,μ_L) will be in RS. So, at some point, (r,μ_L) will be chosen at line 8.

We have two cases:

Case 1: $\sigma \in \Sigma$.

> Since $(r,\mu) \Vdash_M^{\langle\sigma\rangle} (s,\nu)$ we must have in M a transition $(r,\sigma,\delta,\theta,s)$, for some $\delta \in \Phi_C$ and some $\theta \in [C \curvearrowright \mathbb{Q}_{\geq}]$, where $\mu \vDash \delta$ and $\mu \oplus \theta \vDash Inv(s)$, with $\nu = \mu \oplus \theta$.
>
> From Lemma 24, we obtain $\mu_L \oplus \theta \vDash Inv(s)$ and, from Lemma 23, we get $\mu_L \vDash \delta$. Since (r,μ_L) will be chosen at line 8, from lines 12–14, the state (s,η), with $\eta = (\mu_L \oplus \theta)_L$, will be put into RS, if it is not already in HS. In any case, by the loop at line 7, we will get (s,η) in HS, and so by line 23 we will have $(s,\eta) \in S_G$. But, using Proposition 22, we have $\eta = (\mu_L \oplus \theta)_L = (\mu \oplus \theta)_L$. Since $\nu = \mu \oplus \theta$, we obtain $\eta = \nu_L$ and so $(s,\nu_L) \in S_G$, as desired.

Case 2: $\sigma \in \mathbb{Q}_{\geq}$.

> Since ψ is g-adjusted, σ is also g-adjusted. Hence, $\sigma = kg$, for some $k \geq 0$. From $(r,\mu) \Vdash_M^{\langle kg\rangle} (s,\nu)$ we get $s = r$, $\nu = \mu + kg$ and $\mu + \eta \vDash Inv(r)$, for all $0 < \eta \leq kg$. Since we already have $(r,\mu_L) \in S_G$, Lemma 36 gives $(r,\nu_L) \in S_G$, completing the proof. ∎

Now we show that all grid states correspond to reachable configurations in the corresponding TIOA.

Lemma 41 *(Reachable grid states) Let M_G be the grid corresponding to a TIOA M. Let L be a positive integer greater than any constant occurring in M. If $(s,\nu) \in S_G$ then there is a $\rho \in [C \to \mathbb{Q}_{\geq}]$ and a g-adjusted timed word $\psi \in \Psi_{g,\Sigma}$ such that $(s_0,\nu_0) \overset{\psi}{\Vdash}_M (s,\rho)$ and $\rho_L = \nu$.*

Proof From line 23 of Algorithm 1, we know that S_G is the set *HS* when the loop at line 7 terminates. From line 6, *HS* starts empty and elements are added to it one at a time and only at lines 14 and 20. Hence, it suffices to show that the result holds for all pairs (s,ν) added to *RS* at these lines.

Let (r_i,μ_i), $i \geq 0$, be the elements added to *RS*, in order. Clearly, $(r_0,\mu_0) = (s_0,\nu_0)$ by line 5. Taking $\psi = \varepsilon$, the result is seen to hold for (r_0,μ_0). Note also that $\nu_0 = (\nu_0)_L$, since $\nu_0(c) = 0$, for all $c \in C$.

Assume the result holds for (r_j,μ_j), for all $0 \leq j < k$, for some $k \geq 1$. Consider (r_k,μ_k). Since $k \geq 1$, (r_k,μ_k) was added to *RS* at line 14 or at line 20. Hence, at that iteration, some (r_j,μ_j) with $j < k$ was chosen at line 8. The induction hypothesis gives some $\psi \in \Psi_{g,\Sigma}$ such that $(s_0,\nu_0) \overset{\psi}{\Vdash}_M (r_j,\rho)$ and $\rho_L = \mu_j$. There are two cases.

Case 1: At line 14. Then, from lines 10 and 11, we get a transition $(r_j,z,\delta,\theta,r_k)$ in T with $\mu_j \vDash \delta$ and $\mu_j \oplus \theta \vDash Inv(r_k)$. From lines 12 and 14, $\mu_k = (\mu_j \oplus \theta)_L$.

Since $\rho_L = \mu_j$, we get $\rho_L \vDash \delta$ and $\rho_L \oplus \theta \vDash Inv(r_k)$. From Lemma 23 we get $\rho \vDash \delta$ and from Lemma 23, with $\eta = 0$, we have $(\rho_L \oplus \theta)_L \vDash Inv(r_k)$. From Proposition 22 we may write $(\rho \oplus \theta)_L \vDash Inv(r_k)$ and so, from Lemma 23, with $\eta = 0$, we have $(\rho \oplus \theta) \vDash Inv(r_k)$.

Collecting, we have $(r_j,z,\delta,\theta,r_k)$ in T, $\rho \vDash \delta$ and $(\rho \oplus \theta) \vDash Inv(r_k)$. Then $(r_j,\rho) \overset{z}{\Vdash}_M (r_k,\rho \oplus \theta)$. Therefore, $(s_0,\nu_0) \overset{\psi\cdot\langle z\rangle}{\Vdash}_M (r_k,\rho \oplus \theta)$.

We complete this case by showing $(\rho \oplus \theta)_L = \mu_k$. From Proposition 22, $(\rho \oplus \theta)_L = (\rho_L \oplus \theta)_L$. Since $\rho_L = \mu_j$ and $\mu_k = (\mu_j \oplus \theta)_L$, we get $(\rho \oplus \theta)_L = (\mu_j \oplus \theta)_L = \mu_k$, as desired.

Case 2: At line 20. Then from line 17 we obtain $\mu_j + \eta \vDash Inv(r_j)$, $0 < \eta \leq g$. And from lines 18 and 20 we get $\mu_k = (\mu_j + g)_L$ and $r_k = r_j$, respectively. Since $\mu_j = \rho_L$ we get $\rho_L + \eta \vDash Inv(r_j)$ and so $\rho + \eta \vDash Inv(r_j)$ by Lemma 23, for all $0 < \eta \leq g$. Then $(r_j,\rho) \overset{\langle g\rangle}{\Vdash}_M (r_j,\rho + g)$ and, since $r_j = r_k$ we get $(r_j,\rho) \overset{\langle g\rangle}{\Vdash}_M (r_k,\rho + g)$. Therefore, $(s_0,\nu_0) \overset{\psi\cdot\langle g\rangle}{\Vdash}_M (r_k,\rho + g)$.

We complete this case by showing that $(\rho + g)_L = \mu_k$. Since $\mu_k = (\mu_j + g)_L$ and $\rho_L = \mu_j$, we get $\mu_k = (\rho_L + g)_L$. Using Proposition 21, we conclude that $\mu_k = (\rho + g)_L$, as desired.

The induction is extended and we are done. ∎

Next we show that a grid automaton can simulate the corresponding TIOA over a g-adjusted timed word.

Theorem 42 *(Grid simulates TIOA) Let M_G be the grid corresponding to a TIOA M. Let $s, r \in S$ and $\nu, \omega \in [C \to \mathbb{Q}_{\geq}]$ be such that (s, ν) is g-reachable in M with $g = 1/k$ and k a positive integer. Also let $\psi \in \Psi_{g,\Sigma}$ be a g-adjusted timed word, and $L \in \mathbb{Q}_{\geq}$ be a positive integer greater than all constants occurring in M. If $(s, \nu) \overset{\psi}{\Vdash}_M (r, \omega)$ then $(s, \nu_L), (r, \omega_L) \in S_G$ and $(s, \nu_L) \overset{h(\psi)}{\underset{G}{\Vdash}} (r, \omega_L)$.*

Proof Since (s, ν) is g-reachable in M, we get a g-adjusted timed word ψ' such that $(s_0, \nu_0) \overset{\psi'}{\Vdash}_M (s, \nu)$. Then $(s_0, \nu_0) \overset{\psi' \cdot \psi}{\Vdash}_M (r, \omega)$. Since $\psi' \cdot \psi$ is also g-adjusted, (r, ω) is also g-reachable in M. Thus, by Lemma 40, we get $(s, \nu_L), (r, \omega_L) \in S_G$. It remains to show that $(s, \nu_L) \overset{h(\psi)}{\underset{G}{\Vdash}} (r, \omega_L)$.

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

Basis: when $n = 0$, we get $\psi = \varepsilon$ and so $s = r$ and $\nu = \omega$. Thus $(s, \nu_L) = (r, \omega_L)$ and we get $(s, \nu_L) \overset{\varepsilon}{\underset{G}{\Vdash}} (r, \omega_L)$. Since $h(\psi) = \varepsilon$, the basis is complete.

Induction step: assume the result holds for all g-adjusted timed words of length at most n. Take $\psi = \varphi \cdot \langle \sigma \rangle$, where φ has length n, and $\sigma \in \Sigma \cup \mathbb{Q}_{\geq}$. Then, $(s, \nu) \overset{\psi}{\Vdash}_M (r, \omega)$ gives $(s, \nu) \overset{\varphi}{\Vdash}_M (p, \mu)$ and $(p, \mu) \overset{\langle \sigma \rangle}{\Vdash}_M (r, \omega)$.

By the induction hypothesis, $(p, \mu_L) \in S_G$ and $(s, \nu_L) \overset{h(\varphi)}{\underset{G}{\Vdash}} (p, \mu_L)$. Since, by definition, $h(\psi) = h(\varphi) \cdot h(\langle \sigma \rangle)$, it remains to show that $(p, \mu_L) \overset{h(\langle \sigma \rangle)}{\underset{G}{\Vdash}} (r, \omega_L)$.

There are two cases: when $\sigma \in \Sigma$ and when $\sigma \in \mathbb{Q}_{\geq}$.

Case 1: $\sigma \in \Sigma$.

Since $(p, \mu) \overset{\langle \sigma \rangle}{\Vdash}_M (r, \omega)$, we must have a transition $(p, \sigma, \delta, \theta, r)$ in M, with $\mu \vDash \delta$, $\omega = \mu \oplus \theta$, and $\omega \vDash Inv(r)$. Recall that $(p, \mu_L) \in S_G$. Hence, at some point, Algorithm 1 has chosen (p, μ_L) at line 8. From $\mu \vDash \delta$, Lemma 23 gives $\mu_L \vDash \delta$. From $\mu \oplus \theta \vDash Inv(r)$ and Lemma 24 we get $\mu_L \oplus \theta \vDash Inv(r)$. Then Algorithm 1, lines 12 and 13, adds $((p, \mu_L), \sigma, (r, \rho_L))$ to T_G, where $\rho = \mu_L \oplus \theta$. Hence, $(p, \mu_L) \overset{h(\langle \sigma \rangle)}{\underset{G}{\Vdash}} (r, \rho_L)$, since $h(\langle \sigma \rangle) = \sigma$.

We complete this case by showing that $\rho_L = \omega_L$. Since $\rho_L = (\mu_L \oplus \theta)_L$, Proposition 22 gives $\rho_L = (\mu \oplus \theta)_L$. Then $\rho_L = \omega_L$ since $\omega = \mu \oplus \theta$.

CASE 2: $\sigma \in \mathbb{Q}_{\geq}$.

Since $\psi = \varphi \cdot \langle \sigma \rangle$ is g-adjusted, we have that σ is also g-adjusted. Let $\sigma = ig$, for some $i \geq 0$. Moreover, since $(p, \mu) \overset{\langle ig \rangle}{\Vdash_M} (r, \omega)$, we conclude that $p = r$, $\omega = \mu + ig$ and $\mu + \eta \vDash Inv(r)$, for all $0 < \eta \leq ig$. Since we already have $(p, \mu_L) \in S_G$, Lemma 36 gives $(p, \rho_L) \in S_G$ and $(p, \mu_L) \overset{g^i}{\Vdash_G} (p, \rho_L)$, with $\rho = \mu + ig = \omega$. But $h(\langle \sigma \rangle) = g^i$ and $p = r$. So, we have $(r, \omega_L) \in S_G$ and $(p, \mu_L) \overset{h(\langle \sigma \rangle)}{\Vdash_G} (r, \omega_L)$, completing the proof. ∎

In order to illustrate Theorem 42, consider the TIOA shown in Figure 2 and the corresponding partial grid depicted in Figure 3. The chosen granularity is $g = \frac{1}{2}$. Recall that we chose $K = 3$ and $L = 6$. Take the g-adjusted timed word $\alpha = 2on4\, on5\, off$. The corresponding grid word is $h(\alpha) = \beta = (\frac{1}{2})^4\, on\, (\frac{1}{2})^8 on (\frac{1}{2})^{10}\ off$.

We represent a configuration (s, ν) of the TIOA by (s, t_1, t_2), where $t_1 = \nu(c_1)$ and $t_2 = \nu(c_2)$. Then, the start configuration is $(s_0, 0, 0)$. Computing over the input timed word α, from Figure 2, we get

$$(s_0,0,0) \underset{2}{\rightarrow} (s_0,2,2) \overset{on}{\rightarrow} (s_1,0,2) \underset{4}{\rightarrow} (s_1,4,6) \overset{on}{\rightarrow} (s_1,0,6) \underset{5}{\rightarrow} (s_1,5,11) \overset{off}{\rightarrow} (s_0,0,0).$$

Now we apply β to the grid, starting the run in the state $(s_0, 0, 0)$. From Figure 3, the grid moves as follows

$$(s_0,0,0) \overset{\frac{1}{2}}{\Vdash_G} (s_0,\tfrac{1}{2},\tfrac{1}{2}) \overset{\frac{1}{2}}{\Vdash_G} \cdots \overset{\frac{1}{2}}{\Vdash_G} (s_0,2,2) \overset{on}{\Vdash_G}$$

$$(s_1,0,2) \overset{\frac{1}{2}}{\Vdash_G} (s_1,\tfrac{1}{2},\tfrac{5}{2}) \overset{\frac{1}{2}}{\Vdash_G} \cdots \overset{\frac{1}{2}}{\Vdash_G} (s_1,4,6) \overset{on}{\Vdash_G} (s_1,0,6)$$

Now we follow the grid movements from Figure 4:

$$(s_1,0,6) \overset{\frac{1}{2}}{\Vdash_G} (s_1,\tfrac{1}{2},\tfrac{13}{2}) \overset{\frac{1}{2}}{\Vdash_G} (s_1,1,6) \overset{\frac{1}{2}}{\Vdash_G}$$

$$(s_1,\tfrac{3}{2},\tfrac{13}{2}) \overset{\frac{1}{2}}{\Vdash_G} (s_1,2,6) \overset{\frac{1}{2}}{\Vdash_G}$$

$$\vdots$$

$$(s_1,\tfrac{9}{2},\tfrac{13}{2}) \overset{\frac{1}{2}}{\Vdash_G} (s_1,5,6) \overset{off}{\Vdash_G} (s_0,0,0)$$

Note that when, at state s_1, clock c_2 reaches the $\lfloor L \rfloor + 1 = 6 + 1 = 7$ boundary, then the corresponding grid state goes from $(s_1, t, 6)$ to $(s_1, t + \frac{1}{2}, \frac{13}{2})$, and back to $(s_1, t + 1, 6)$, and then cycles between 6 and $\frac{13}{2}$ in the value of clock 2. When the value of the first clock reaches 5, then a discrete movement is forced and both clocks are reset to zero. See Figure 4.

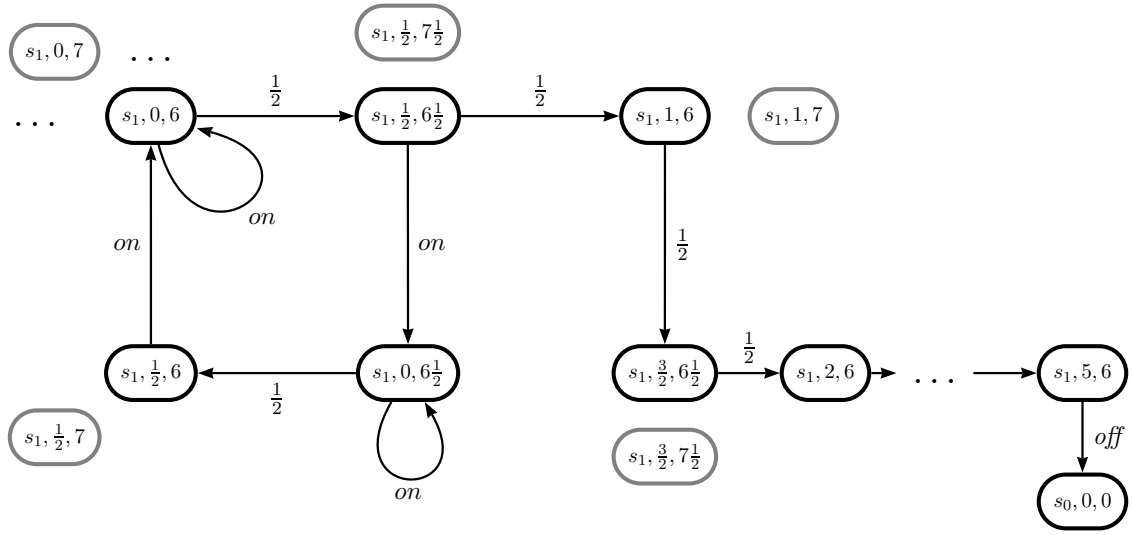

Figure 4: The partial grid L-bound for TIOA in Figure 2.

The next theorem shows that a TIOA imitates the corresponding grid automaton.

Theorem 43 *(TIOA imitates grid) Let M_G be the grid corresponding to a TIOA M and let $g = 1/k$ with k a positive integer. Let $(s, \nu), (r, \omega) \in S_G$. Also let $L \in \mathbb{Q}_{\geq}$ be a positive integer greater than all constants occurring in M. If $(s, \nu) \overset{\psi}{\underset{G}{\Vdash}} (r, \omega)$ for some $\psi \in \Sigma_G^\star$, then there are $\rho, \mu \in [C \to \mathbb{Q}_{\geq}]$ such that $(s, \rho) \overset{\widehat{h}(\psi)}{\Vdash_M} (r, \mu)$, with $\nu = \rho_L$ and $\omega = \mu_L$.*

Proof We proceed by induction on the length $n \geq 0$ of ψ.

Basis: when $n = 0$, we get $\psi = \varepsilon$, $s = r$ and $\nu = \omega$. So, $\widehat{h}(\psi) = \varepsilon$. Using Lemma 41, we get $\rho \in [C \to \mathbb{Q}_{\geq}]$ with $\rho_L = \nu$. Take $\mu = \rho$. Then $(s, \rho) \overset{\widehat{h}(\psi)}{\Vdash_M} (r, \mu)$. Moreover, $\rho_L = \nu$ and $\mu_L = \rho_L = \nu = \omega$, completing the basis.

Induction step: assume 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, $(s, \nu) \overset{\psi}{\underset{G}{\Vdash}} (r, \omega)$ gives $(s, \nu) \overset{\varphi}{\underset{G}{\Vdash}} (p, \lambda)$ and $(p, \lambda) \overset{\sigma}{\underset{G}{\Vdash}} (r, \omega)$. By the induction hypothesis we get $\widehat{\nu}, \widehat{\lambda} \in [C \to \mathbb{Q}_{\geq}]$, where $\widehat{\nu}_L = \nu$ and $\widehat{\lambda}_L = \lambda$, and such that $(s, \widehat{\nu}) \overset{\widehat{h}(\varphi)}{\Vdash_M} (p, \widehat{\lambda})$. Since $\widehat{h}(\psi) = \widehat{h}(\varphi) \cdot \widehat{h}(\sigma)$, it remains to show that $(p, \widehat{\lambda}) \overset{\widehat{h}(\sigma)}{\Vdash_M} (r, \widehat{\omega})$, for some $\widehat{\omega} \in [C \to \mathbb{Q}_{\geq}]$, where $\widehat{\omega}_L = \omega$.

There are two cases: when $\sigma \in \Sigma$ and when $\sigma = g$.

Case 1: $\sigma \in \Sigma$.

Since $(p,\lambda) \overset{\sigma}{\Vdash}_G (r,\omega)$, we must have a transition $((p,\lambda),\sigma,(r,\omega))$ in T_G. From Algorithm 1, lines 10 to 14, we have $(p,\sigma,\delta,\theta,r)$ in T, for some $\delta \in \Phi_C$, $\theta \in [C \curvearrowright \mathbb{Q}_{\geq}]$, and such that $\lambda \vDash \delta$, $\lambda \oplus \theta \vDash Inv(r)$ and $\omega = (\lambda \oplus \theta)_L$.

Since $\lambda = \widehat{\lambda}_L$, we get $\widehat{\lambda}_L \vDash \delta$. Using Lemma 23 we get $\widehat{\lambda} \vDash \delta$. Also $\widehat{\lambda}_L \oplus \theta \vDash Inv(r)$, since $\lambda \oplus \theta \vDash Inv(r)$. Now, from Lemma 23 (with $\eta = 0$) we get $(\widehat{\lambda}_L \oplus \theta)_L \vDash Inv(r)$. But $(\widehat{\lambda}_L \oplus \theta)_L = (\widehat{\lambda} \oplus \theta)_L$, using Proposition 22. Then $(\widehat{\lambda} \oplus \theta)_L \vDash Inv(r)$, and so from Lemma 23 (with $\eta = 0$) we get $\widehat{\lambda} \oplus \theta \vDash Inv(r)$.

Collecting, we have $\widehat{\lambda} \vDash \delta$, $\widehat{\lambda} \oplus \theta \vDash Inv(r)$ and $(p,\sigma,\delta,\theta,r) \in T$. Then $(p,\widehat{\lambda}) \overset{\widehat{h}(\sigma)}{\Vdash}_M (r, \widehat{\lambda} \oplus \theta)$, since $\widehat{h}(\sigma) = \sigma$. Let $\widehat{\omega} = \widehat{\lambda} \oplus \theta$. To complete this case, we need $\widehat{\omega}_L = \omega$. But $\widehat{\omega}_L = (\widehat{\lambda} \oplus \theta)_L = (\widehat{\lambda}_L \oplus \theta)_L$, by Proposition 22. Since $\widehat{\lambda}_L = \lambda$ we get $\widehat{\omega}_L = (\lambda \oplus \theta)_L = \omega$, as desired.

Case 2: $\sigma = g$.

Since $(p,\lambda) \overset{g}{\Vdash}_G (r,\omega)$, we must have a transition $((p,\lambda),g,(r,\omega))$ in T_G. From Algorithm 1, lines 17– 20, we have $p = r$, $\omega = (\lambda + g)_L$, and $\lambda + \eta \vDash Inv(p)$, for all $0 < \eta \leq g$.

Fix any η, $0 < \eta \leq g$. Since $\lambda = \widehat{\lambda}_L$, we get $\widehat{\lambda}_L + \eta \vDash Inv(p)$. Using Lemma 23 we get $(\widehat{\lambda}_L + \eta)_L \vDash Inv(p)$, and so $(\widehat{\lambda} + \eta)_L \vDash Inv(p)$, using Proposition 21. Hence $\widehat{\lambda} + \eta \vDash Inv(p)$, using Lemma 23 again. Therefore, $\widehat{\lambda} + \eta \vDash Inv(p)$, $0 < \eta \leq g$. Then $(p,\widehat{\lambda}) \overset{\langle g \rangle}{\Vdash}_M (p, \widehat{\lambda} + g)$. Since $p = r$, by letting $\widehat{\omega} = \widehat{\lambda} + g$ we get $(p,\widehat{\lambda}) \overset{\widehat{h}(\sigma)}{\Vdash}_M (r,\widehat{\omega})$, because $\widehat{h}(\sigma) = \widehat{h}(g) = \langle g \rangle$.

In order to complete this case, we need $\widehat{\omega}_L = \omega$. But $\widehat{\omega}_L = (\widehat{\lambda} + g)_L = (\widehat{\lambda}_L + g)_L$ by Proposition 21. Since $\widehat{\lambda}_L = \lambda$, we get $\widehat{\omega}_L = (\lambda + g)_L = \omega$, as desired.

This concludes the proof. ∎

For the TIOA depicted in Figure 2 a corresponding partial grid is shown in Figure 4. In that figure, the gray states represent possible TIOA configurations corresponding to the nearest grid state.

4 Generating test cases based on test purposes

In this section we use the notion of a test purpose [15] in order to generate test sequences for TIOA.

4.1 Test purposes

We start by noticing that several types of faults or desired properties, of practical interest, can be modeled by a specific family of TIOA.

Definition 44 *(Acyclic TIOA) A TIOA is acyclic iff its subjacent directed graph, defined by taking states as nodes and transitions as edges, is acyclic.* ∎

Test purposes are acyclic TIOA equipped with fail and desired states.

Definition 45 *(Test purposes) A test purpose is an acyclic TIOA with two special sets of states: a set $F \subseteq S$, of fail states, and a set $D \subseteq S$, of desired states, with $F \cap D = \emptyset$.* ∎

A test purpose focuses on specific parts of the system, with the aim of verifying whether the implementation meets certain properties. The fail states in F represent undesired, or fail, properties of the system. Later, a test purpose can be combined with the specification TIOA so that both are driven synchronously. Then, using the test purpose as a guide, one can extract certain input test sequences that drive the specification and the test purpose TIOA to fail states. Such a set of test sequences can then be applied to the implementation in order to determine if the implementation also reaches a fail condition. A similar reasoning applies to the desired states in D.

Consider the following example.

Example 46 *The acyclic TIOA depicted in Figure 5 is related to the specification TIOA shown in Figure 1. Its intention is to verify whether the implementation accepts the input*

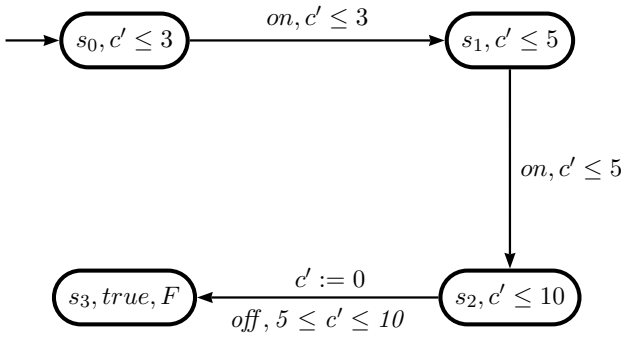

Figure 5: An example of test purpose for the switch system.

on *within three time units followed by another* on *symbol which arrives within five time units, counting from the start. Then, the implementation must respond with an* off *symbol within no more than ten time units from the start, and no less than five time units also from the start. In this case, the special sets are $F = \{s_3\}$ and $D = \emptyset$.* ∎

Using test purposes one can focus only on parts of the specification model, thus avoiding generating test cases for the whole system. Together with a test purpose, verifying an implementation requires that it satisfies both the specification model and the test purpose. This will be formalized by constructing the (synchronous) product [3, 9] of the specification

and the test purpose. Next, in order to control the number of states, we obtain the grid automaton associated with the product [9]. Then, we can traverse the resulting grid searching down for faulty or desired states, collecting along a set of input sequences. Test sequences are then derived from these input sequences. The test sequences can, then, be used to test implementation candidates in order to identify desired behaviors or faults, when those have been modeled by the test purpose.

4.2 The synchronous product

The intended meaning of a product of two TIOA is to force the synchronism between the participating TIOA, whenever possible. For that, states in the product are modeled as pairs of states from the participating TIOA. When at a state (s_1, s_2) is in the product and there are transitions out of s_1 and of s_2 on a same action symbol a, then these transitions must be taken in parallel. In the product, this is modeled by inserting a transition out of (s_1, s_2), over the same symbol a, and in such a way as to capture the effects of the individual transitions in the participating TIOA. When only one of the states, s_1 or s_2, has an outgoing transition on a symbol a, then the product reflects that action, while keeping the other state unchanged. Clearly, the initial state in the product is the pair of initial states in the participating TIOA. Finally, in order to avoid clock reset conflicts, we require that both sets of clocks in the participating TIOA to be disjoint.

The synchronous product is constructed algorithmically.

Definition 47 *(Synchronous product) Let M_1 and M_2 be two TIOA, with $C_1 \cap C_2 = \emptyset$. The synchronous product of M_1 and M_2 is the TIOA $M_1 \otimes M_2$ constructed by Algorithm 2.*

Algorithm 2 constructs the synchronous product for two TIOA by first pairing the initial state of both participating TIOA in order to create the initial state of the product TIOA. Then, the product transitions are constructed by an exhaustive search for new transitions and new states in the product.

We want a state (s_1, s_2) to be a fail state in the product when s_2 is a fail state in the purpose model. Similarly for the desired states.

Definition 48 *(Special states in the product) Let M_1 be a TIOA and M_2 be test purpose. A state (s_1, s_2) in the product is a desired or fail state iff s_2 is a desired or fail state, respectively, in M_2.*

Next, we present an example of a synchronous product.

Example 49 *Consider the specification given in Figure 1 and the test purpose given in Figure 5. The resulting product is shown in Figure 6. The invariant conditions are shown in Table 1. Note that (q_0, s_3, F) and (q_1, s_3, F) are the fail states in the product automaton. Also, note that some input words that are accepted by the specification may not be accepted by the product, since some timing requirements in the specification may not be satisfied simultaneously by the test purpose. For instance, taking $K = 5$, the timed word "$\frac{14}{3}$ on" induces a run in the specification but not in the synchronous product since the invariant at (q_0, s_0) requires $c' \leq 3$ and we have $14/3 > 3$.* ∎

Input: TIOA M_1 and M_2 with $C_1 \cap C_2 = \emptyset$.
Output: The synchronous product $M_P = M_1 \otimes M_2$.
begin
  $C_P \leftarrow C_1 \cup C_2$ // the set of clock variables;
  $X_P \leftarrow X_1 \cup X_2$, $Y_P \leftarrow Y_1 \cup Y_2$, $\Sigma_P \leftarrow \Sigma_1 \cup \Sigma_2$ // action symbols;
  $RS \leftarrow s_0^P = (s_0^1, s_0^2)$, $Inv_P(s_0^P) \leftarrow Inv_1(s_0^1) \wedge Inv_2(s_0^2)$ // initial state;
  $T_P \leftarrow \emptyset$, $HS \leftarrow \emptyset$ // transitions and visited states;
  while $RS \setminus HS \neq \emptyset$ **do**
    Choose $s = (s_1, s_2)$ from $RS \setminus HS$;
    Move s from RS to HS;
    foreach $a \in \Sigma$ **do**
      if $(s_i, a, \delta_i, \theta_i, s_{i+2}) \in T_i$ *for some* $s_{i+1} \in S$, $\delta_i \in \Phi_{C_i}$, $\theta_i \in [C_i \curvearrowright \mathbb{Q}_{\geq}]$, $i = 1, 2$ **then**
        let $(p, q) = (s_3, s_4)$, $\delta = \delta_1 \wedge \delta_2$, $\theta = \theta_1 \oplus \theta_2$, $I = Inv_1(s_3) \wedge Inv_2(s_4)$
      end
      if $(s_1, a, \delta_1, \theta_1, s_3) \in T_1$ *for some* $s_3 \in S$, $\delta_1 \in \Phi_{C_1}$, $\theta_1 \in [C_1 \curvearrowright \mathbb{Q}_{\geq}]$ *and* $(s_2, a, \delta_2, \theta_2, s_4) \notin T_2$ *for all* $s_4 \in S$, $\delta_2 \in \Phi_{C_2}$,$\theta_2 \in [C_2 \curvearrowright \mathbb{Q}_{\geq}]$ **then**
        let $(p, q) = (s_3, s_2)$, $\delta = \delta_1$, $\theta = \theta_1$, $I = Inv_1(s_3) \wedge Inv_2(s_2)$
      end
      if $(s_2, a, \delta_2, \theta_2, s_4) \in T_2$ *for some* $s_4 \in S$, $\delta_2 \in \Phi_{C_2}$, $\theta_2 \in [C_2 \curvearrowright \mathbb{Q}_{\geq}]$ *and* $(s_1, a, \delta_1, \theta_1, s_3) \notin T_1$ *for all* $s_3 \in S$, $\delta_1 \in \Phi_{C_1}$,$\theta_1 \in [C_1 \curvearrowright \mathbb{Q}_{\geq}]$ **then**
        let $(p, q) = (s_1, s_4)$, $\delta = \delta_2$, $\theta = \theta_2$, $I = Inv_1(s_1) \wedge Inv_2(s_4)$
      end
      Add (p, q) to RS, **let** $Inv_P(p, q) \leftarrow I$, **add** $((s_1, s_2), a, \delta, \theta, (p, q))$ to T_P;
    end
  end
  $S_P \leftarrow HS$;
end

Algorithm 2: Synchronous product.

4.3 Test case generation and the grid automaton

We can generate timed test cases using the notion of a grid automaton extracted from the product of the specification and the test purpose, when both the latter are given by their respective TIOAs.

The grid automaton is obtained from the resulting product following the method presented in Subsection 3.2. Then, test sequences are extracted by traversing the grid. 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 test sequence is output. A recursive traversal procedure is depicted in Algorithm 3. Clearly, the set of all test sequences is generated by a call in the form $TestGeneration(s_0, \epsilon)$, where s_0 is the initial state of the grid obtained from the product automaton and ϵ is the empty string. Note that one can construct g-adjusted timed

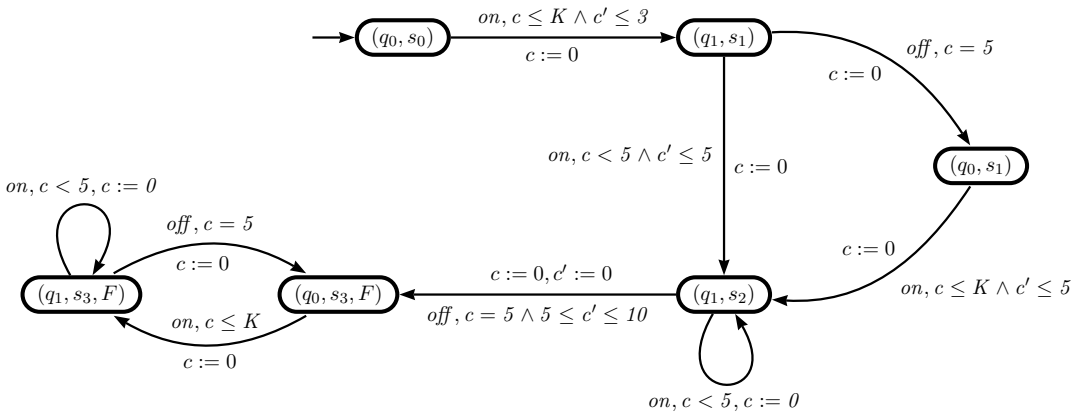

Figure 6: The product of a specification and a test purpose.

State	Invariant
(q_0, s_0)	$(c' \leq 3) \wedge (c \leq K)$
(q_1, s_1)	$(c' \leq 5) \wedge (c \leq 5)$
(q_1, s_2)	$(c' \leq 10) \wedge (c \leq 5)$
(q_0, s_3)	$(c \leq K)$
(q_0, s_1)	$(c' \leq 5) \wedge (c \leq K)$
(q_1, s_3)	$(c \leq 5)$

Table 1: Invariants for the product

words from all grid words extracted from the grid automaton, using the mappings given at Definition 37 and Definition 38. Note also that delay transitions in the grid represent the continuous evolution in the original specification, up to the chosen boundary.

When the test purpose models desired behaviors of a system, the verification process issues a "pass" verdict only when the implementation respects the specification and it satisfies the test purpose, for all sequences in the test suite. If, on the other hand, the testing if based on a purpose automaton with fail states, then the verification process issues a positive verdict only when the implementation satisfies the specification and also reaches a faulty state, for any of the sequence in the test suite.

Revisiting Example 49, with $K = 5$, we see that the sequence $\beta = \frac{8}{3} on \frac{7}{3} on 5 \, off$ is an execution for the product. If an implementation under test conforms to the specification and the test purpose, we should be able to observe the same behavior when β is applied in this implementation. In Figure 7 we present a partial grid automaton resulting from the synchronous product shown in Figure 6, and where we have chosen a granularity of $\frac{1}{3}$. Note that we have relabeled the states in the figure in order to keep the notation uncluttered. Next, we can traverse the grid and extract grid words, as previously indicated by Algorithm 3. For example, the sequence $\alpha = (\frac{1}{3})^2 on (\frac{1}{3})^4 \, on (\frac{1}{3})^{15} \, off$ can be so extracted. Using the reverse mapping of Definition 38 we obtain the product timed word $\frac{2}{3} on \frac{4}{3} on 5 \, off$.

```
1  TestGeneration(INPUT: state s of a grid M_G; OUTPUT: A timed test sequence
   TTS;)
2  begin
3  |  if s is a leaf then
4  |  |  Write TTS;
5  |  else
6  |  |  foreach neighbor, r, of s reached over a transition on z do
7  |  |  |  Concatenate z with TTS;
8  |  |  |  TestGeneration(r,TTS);
9  |  |  end
10 |  end
11 end
```

Algorithm 3: The traversal algorithm *TestGeneration*.

Some other sequences that would be extracted from the grid and then mapped to timed words are:

$$\frac{1}{3}\,on\,\frac{14}{3}\,on\,5\,off \qquad \frac{2}{3}\,on\,\frac{13}{3}\,on\,5\,off \qquad 1\,on\,4\,on\,5\,off$$

$$\frac{4}{3}\,on\,\frac{11}{3}\,on\,5\,off \qquad \frac{5}{3}\,on\,\frac{10}{3}\,on\,5\,off \qquad 2\,on\,3\,on\,5\,off$$

$$\frac{7}{3}\,on\,\frac{8}{3}\,on\,5\,off \qquad \frac{8}{3}\,on\,\frac{7}{3}\,on\,5\,off \qquad 3\,on\,2\,on\,5\,off\,.$$

Note that some sequences cannot be obtained by traversing the grid, although they might be executions over the specification. For example, sequences $on\,on\,4\,off$ and $\frac{2}{3}\,on\,\frac{13}{3}\,on\,\frac{17}{3}\,off$ cannot be extracted in the grid in Figure 6. The first one has a total time elapsed of less than five time unit after the last *on* and this is not allowed in the test purpose although it is a valid execution in the specification model. In the second one, the last *off* symbol occurs more than five time units after the previous *on* symbol, which is not allowed in the specification, even though it is allowed in the test purpose.

5 Related works

In this section we describe some related works.

In [9], En-Nouaary and Dssouli also discuss a timed model-based testing on TIOA specifications which uses test purposes and synchronous products. However, their discretization technique is based on the classical notion of clock regions, thus imposing a strict relationship between the number of clock variables present in the models and the granularity that must be chosen in order to obtain the corresponding discrete automaton. Instead of constructing a grid automaton directly, the notion of a region graph is first used to represent a possibly

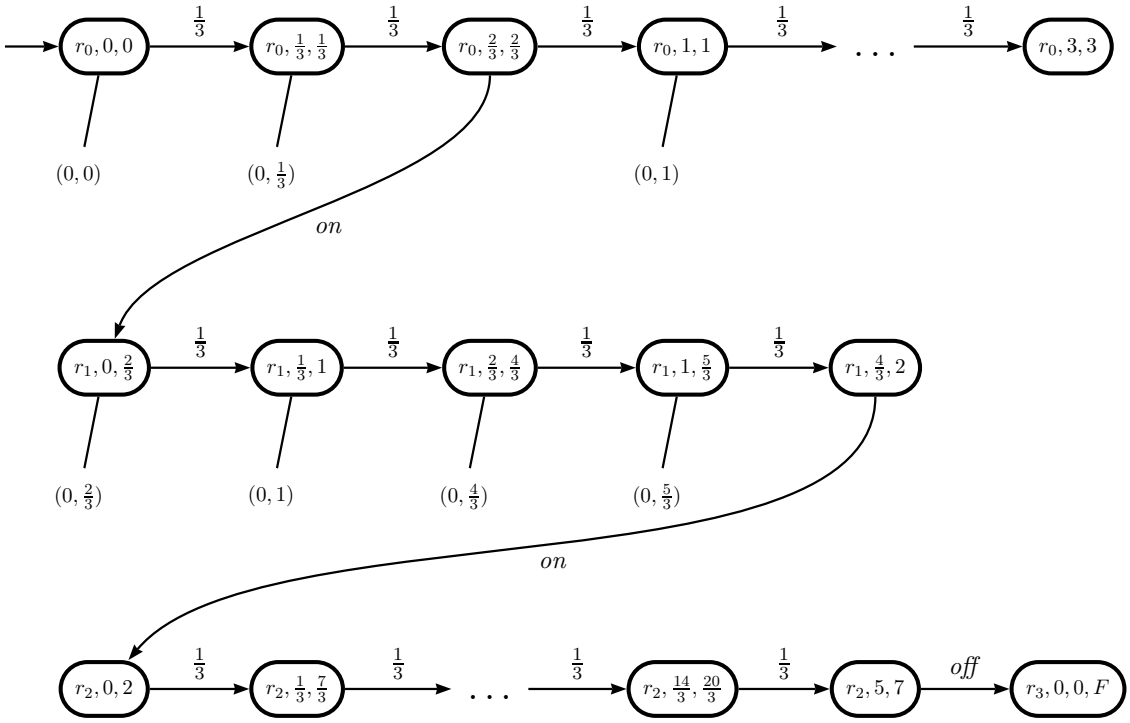

Figure 7: The partial grid automaton for Figure 6.

infinite transition system and, then, by a process of sampling, a grid automaton is derived. Moreover, it is not mentioned how one could apply the timed test cases that are generated.

Another test generation method based on TIOA is proposed in [12], and test purposes are also discussed like in [9]. In these works, following similar lines, region graphs are also sampled in order to obtain a grid automaton. But no guarantees are postulated about the algorithms presented therein, which are used in an exhaustive test generation process. Also, dense time has but a superficial treatment, making it difficult to see how to generate precise timed test cases when combining test purposes and TIOA models.

In [11], Fouchal and co-workers discuss a test execution strategy similar to the one discussed in this work. But, whereas our work deals with dense time in order to capture timed properties, in [11] the notion of timed elements are used to imitate continuous time evolution, thus offering no guarantees of accuracy about the extracted timed test sequences.

Another approach is taken in [15], now using enumeration of data values in order to avoid the state space explosion problem. In this work the ioSTS formalism is used as the formal specification model. Since a discretization method is not used to obtain a grid automaton, time evolution is not captured in appropriate manner for testing timed properties. Also, the method could incur in high cost when calculating constraints using approximations in order to find test sequences.

Many similar approaches [10, 8, 7] use the classical notion of clock regions in order to obtain grid automata. In these cases, the large number of clock variables present in models will often lead to huge grid automata, due to the exponential number of clock regions and the need to enforce the relationship between the number of clocks and the chosen granularity. By contrast, our approach allow for an ample range of choice for appropriate granularity values, thus leading to more controllable grid automata and to more manageable test suites.

6 Concluding Remarks

Many approaches have been proposed to automatically construct test suites for timed systems. The problem is specially challenging since timed systems exhibit continuous time evolution and, often, formal models techniques for dealing with such systems incur in the state explosion problem.

In this work we proposed a new way of discretizing timed system models. In the process, we noticed that the classical notion of clock regions were unnecessary, giving rise to a more general notion that allowed for an ample range of granularity values that could be chosen in the discretization process. We also demonstrated that the grid automaton thus obtained was capable of homomorphically simulating the original timed system, and vice-versa. This formed the basis for the development of automatic methods for generating test suites.

In order to test more specific system properties over the implementations, we made use of the notion of a timed test purpose model. Using the notion of a synchronous product between a timed system and a timed purpose model, the discretization algorithm is able to generate a corresponding grid automaton that reflects both the behavior of the original timed system, as well as the desired properties specified by the timed test purpose model. From this grid automaton, one can then automate the process of extracting test case sequences. Detailed proofs of correctness were provided for all properties of interest.

As suggestions for expanding this work, we cite the possible representation of data flow within the formal models, thus also capturing the idea of system context variables. With that notion in place, one should be able to deal both with control flow and time evolution, as well as with data flow issues, the latter being captured by context variables manipulations. Finally, the discretization ideas presented here could also be used for future investigations that could lead to automatic test case generation when context variables are also present in the formal models.

References

[1] R. Alur. Timed automata. In *CAV'99*, number 1633 in LNCS, 1999.

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

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

[4] A. L. Bonifácio, A. V. Moura, A. S. Simão, and J. C. Maldonado. Conformance Testing by Model Checking Timed Extended Finite State Machines. In *Brazilian Symposium on Formal Methods (SBMF'06)*, pages 43–58, Natal, September 2006.

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

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

[7] 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.

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

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

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

[11] 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.

[12] Hacène Fouchal. Conformance testing techniques for timed systems. In *SOFSEM*, pages 1–19, 2002.

[13] A. 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, 2005.

[14] R. Gawlick, R. Segala, J. F. Sogaard-Andersen, and N. A. Lynch. Liveness in timed and untimed systems. In *Automata, Languages and Programming*, pages 166–177, 1994.

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

[16] K. G. Larsen and W. Yi. Time abstracted bisimulation: Implicit specifications and decidability. *j-LECT-NOTES-COMP-SCI*, 802:160–176, 1994.

[17] Brian Nielsen and Arne Skou. Test generation for time critical systems: Tool and case study. *ecrts*, 00:0155, 2001.

[18] J. Springintveld, F. Vaandrager, and P. R. D'Argenio. Testing timed automata. *Theor. Comput. Sci.*, 254(1-2):225–257, 2001.

[19] 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. Springer, 1996.

[20] 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.

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

[22] Farn Wang. Efficient verification of timed automata with bdd-like data structures. *STTT*, 6(1):77–97, 2004.