Extending Digital Clocks to Support Diagonal Constraints in Probabilistic Timed Automata

MELANIA VARTIC, University of Twente, The Netherlands

Model-checking is a method used for verifying the temporal behaviour of a system and whether a model achieves its desired properties. This approach is often fully automated and is applied, among other areas, within the context of timed automata. Timed automata are a formalism that captures properties related to quantitative time, facilitating the modelling of real-time systems where time is a significant aspect. Probabilistic timed automata are an extension of classical timed automata, with the capacity to model the uncertainty of events along with timing constraints. The digital clock method in probabilistic timed automata abstracts real-valued clock variables into integer representations, resulting in a finite-state model that preserves the timing properties required in formal verification. This abstraction is typically sound for diagonal-free clocks, where the constraints of a clock can be only in relation to a constant, not to another clock. We analyse the behaviour of a digital clock abstraction algorithm developed in the Modest Toolset when presented with diagonal constraints and modify it to accommodate them. We used extended semantics to explore more states around the diagonal and changed the reset function to stay within the bounds, studying the correctness of the algorithm and how it behaves when presented with more clock variables.

Additional Key Words and Phrases: Probabilistic timed automata, Digital clock, Markov decision process, Diagonal digital clock, Probabilistic model checking, Timed automata

1 INTRODUCTION

Model checking [15] is an approach originally developed to verify the qualitative properties of systems that was subsequently extended to also handle quantitative features, such as real-time constraints [3]. This extension of model-checking is often applied to timed automata.

A timed automaton is a labelled transition system extended with a finite set of real-valued clocks, each of which evolves constantly at a unit rate [28]. Although this formalism introduced by Alur and Dill [2], provides a robust solution to capture time constraints within system models, it assumes idealised behaviour, which is unfeasible in real-time systems.

An extension of this formalism with integrated discrete probabilistic transitions leads to the notion of probabilistic timed automata. Probabilistic timed automata are widely used models for cyberphysical systems that capture the timing constraints and uncertainty of real-time behaviour [23, 42]. This concept is explained in detail in the work of Kwiatkowska et al. [37]. The authors propose the model and prove that it is practical and expressive through a series of worked examples and applications, while altering algorithms designed for timed automata [5] to incorporate probabilistic transitions. In this context, a timed automaton has a finite set of real-valued clocks that are independent of each other and can be reset during execution. Probabilistic timed automata are used to model systems based on timing and uncertainty, examples include communication protocols [17, 20, 32, 39], systems in industries such as aviation [21], networked systems [22, 42], automated human activity recognition [43], or self-adaptive multi-agent systems [41].

Previous work shows that for timed properties, the validity of the integral model follows from the validity in the dense-time model [31]. Integral semantics refers to the clock variables that have positive integer values and is often known as the digital clock transformation. We thus adopt the digital clock method, namely clocks with integer values that only change in unit steps. Kwiatkowska et al. [36] proved, in 2006, that digital clocks are sufficient for the analysis of probabilistic and expected reachability against closed diagonal-free probabilistic timed automata. Digital clock abstraction is a method used to ensure that the Markov decision process obtained from the transformation of a probabilistic timed automaton has a finite state space.

Formal verification [15] is a technique used to assess whether a model satisfies its specification often by exhaustively exploring its state space, and has been widely applied to embedded systems [14, 18, 45] and industrial settings [16, 19, 40, 48]. The form of model-checking addressed by this study involves the use of the Modest Toolset. Modest is a behavioural modelling language introduced by Hartmanns [25] for stochastic timed systems. This research explores a possible implementation of the digital clock abstraction algorithm in the Modest Toolset, that transforms a probabilistic timed automaton with diagonal constraints into a Markov decision process (MDP). The resulting MDP model will be subjected to formal verification using the Modest Toolset *mcsta* [13] model checker.

Although there are numerous model-checking algorithms such as region graph construction [2], backward reachability [38], forward reachability [49], each are optimised for specific cases. Due to its increased performance compared to these methods [34], the digital clock method followed by a model-checking algorithm on the resulting Markov decision process is an emerging solution in model-checking. Currently, the digital clock algorithm does not allow *diagonal constraints* in the input automaton. To support transformation into an MDP, solutions include the removal of diagonal constraints in probabilistic timed automata or the rejection of these input automata. Modest features the *moconv* tool, which converts a probabilistic timed automaton into its digital clock Markov decision process [11]. The current implementation of digital clocks does not allow as input a probabilistic timed automaton with diagonal constraints, and the algorithm does not treat this case.

The relational comparison among clock variables represents a paramount improvement in the compactness of a probabilistic timed automaton. Diagonal constraints facilitate modelling and representation of timing dependencies of a system, enabling synchronisation between its tasks. Although there is no improvement in expressiveness with the use of diagonal constraints in timed automata [9],

TScIT 37, July 8, 2025, Enschede, The Netherlands

diagonalisation helps improve model conciseness [8], and along with the performance of the digital clock method, it can be a powerful addition to existing model checking algorithms.

2 RESEARCH OBJECTIVE AND REQUIREMENTS

To address the problem of diagonal constraints not being supported by the digital clock abstraction algorithm, the main objective of this research is to analyse the behaviour and limitations under these conditions of the existing Modest-based implementation and modify it to support diagonalisation.

Specifically, we first focus on documenting and applying modifications to the Modest-based implementation of the digital clock conversion to support diagonal constraints, a requirement referred to in this work as RQM1. Similarly, another objective is studying the behaviour of an algorithm extending the digital clocks method based on a small example (RQM2). Finally, this study should address how the extended algorithm scales as the number of clocks increases by observing its behaviour and limitations (RQM3).

3 RELATED WORK

Various researchers analyse probabilistic timed automata, developing model-checking algorithms and tools. Bouyer et al. [9] proposes an approach to model-checking classical timed automata that takes into account the presence of diagonal constraints. The authors explored and proved the correctness of an algorithm that identifies and selectively removes only diagonal constraints that cause errors. Its primary limitations are the focus on timed automata without addressing probabilistic behaviour and its adoption of an approach based on diagonal constraint removal, rather than maintaining those constraints and adapting the checking procedure to support them. Their approach does not use digital clock abstraction, which is a focus point in our research, and thus it is not applicable in our scope.

Extensive research has been conducted on model checking in timed automata. Norman et al. [42] provides an overview of the algorithms for probabilistic timed automata, such as region graph construction, boundary region graph, digital clock method, and backward reachability. The region graph method is based on the construction of timed automata [2], therefore, it is not addressed in this study, due to our focus on the digital clock method. Norman et al. [42] defines also the boundary region graph as a finite Markov decision process equipped with a reward structure; however, this is not applicable in the context of this study, as considerations regarding cost and reward lie beyond its scope. Both are used primarily to establish the decidability and complexity of model checking, rather than for practical implementations. The digital clocks method is defined by the authors as a restriction of continuous-time semantics; however, in their study they assume no diagonal constraints; as a result, it cannot be applied to our study. Norman et al. [42] explain then the backward reachability algorithm as an analysis used for automata with both an infinite and a finite number of states. The limitation of this approach is that it is less performant than the digital clock method and is beyond our scope. Lastly, they explain abstraction refinement using stochastic games as building an abstraction of an infinite-state Markov decision process by components of two-player stochastic games. This is not applicable in this research, as it falls outside the scope of digital clocks.

In their research, Hartmanns and Hermanns [26], propose Modest as a modelling and description language for stochastic time systems. The approach introduces the first version of the tool *mcsta* that translates Modest-based specifications into a format compatible with the PRISM model checker, delivering significant improvements in terms of dynamic parallelism and exception handling. Although this applies to probabilistic timed automata in general, it shows an older version of the *mcsta* tool. We must use a newer version, as we check the Markov decision process resulting after applying digital clocks to the probabilistic timed automaton.

In 2000, Kwiatkowska et al. [38] proposed the backward reachability algorithm, designed for dense semantics. Their approach consists in each state being represented by a location and clock zone, thus replacing the need to track the value of each clock with monitoring the zones determined by the clock constraints. Zones are computed and represented using difference-bound matrices with an entry for each pair x, y recording the tightest bound on $x - y$. Although this is related to the problem by covering dense semantics and supporting diagonal constraints, this approach is significantly slower than the digital clock method [29], and therefore it is not applicable in our scope.

4 BACKGROUND

In this section, we lay the formal foundations necessary to define digital clocks in probabilistic timed automata. In this paper, we use $\mathbb{R}_{\geq 0}$ to denote the set of non-negative real numbers, and $\mathbb{N}$ the set of positive integers.

Markov decision processes are an extension of stochastic sequential decision processes in which the cost and transition functions depend only on the current state of the system and the action [44]. This is the base model for the model-checking algorithms.

Definition 4.1 (Markov decision process). [33] A *Markov decision process* M is a tuple

$$M = (S, s, Steps, L)$$

where:

- S is a finite set of states,
- $s \in S$ is the initial state,
- *Steps* is a function that assigns to each state $s \in S$ a finite, non-empty set $Steps(s)$ of probability distributions on S, and
- L is the labelling function

Probabilistic timed automata [4] are an extension of timed automata [1], associating with each location a set of actions, where each action defines a probability distribution over the possible successor locations. Such automata are conventionally interpreted within a temporal-semantics framework, whereby the evolution of clock values is governed by a time domain. The dense-time semantics consists of clock values within $\mathbb{R}_{\geq 0}$, whereas the integral-time semantics imply clock values within $\mathbb{N}$. Digital clocks [36] are the practical implementation of integral semantics.

In this research, we assume for the probabilistic timed automaton a finite set of clocks $\mathcal{X}$. A function $v : \mathcal{X} \to \mathbb{R}_{\geq 0}$ is known as *clock*

valuation and the set of all clock valuations is denoted $\mathbb{R}^{\mathcal{X}}_{\geq 0}$. For any $v \in \mathbb{R}^{\mathcal{X}}_{\geq 0}$, $t \in \mathbb{R}_{\geq 0}$ and $X \subseteq \mathcal{X}$ the clock valuation which increments all clock values in v by t is denoted $(v + t)(x) = v(x) + t$ [46]. Similarly, $v[X := 0]$ represents the clock valuation [42] that resets all clocks in X to 0. The set of clock constraints on $\mathcal{X}$, often referred to as $CC(\mathcal{X})$ is defined by the syntax $\chi := true|false|x \geq c|x \leq d|x + c \leq y + d|\chi \wedge \chi$ where $x, y \in \mathcal{X}$ and $c, d \in \mathbb{N}$ [30]. Previous work [36, 42] includes $\neg\chi$ in the syntax; however, this paper is based on the assumption of *closed constraints*, that is, comparisons must use $\geq, =, or \leq$. A clock value v satisfies a clock constraint χ, $v \models \chi$, if χ is true when substituting all occurrences of x with $v(x)$ satisfy the constraint.

Definition 4.2 (Probabilistic Timed Automata). [37] A *probabilistic timed automaton* (PTA) is a tuple

$$(L, \bar{\ell}, \mathcal{X}, \Sigma, \text{inv}, \text{prob})$$

where:

- L: finite set of *locations*, with $\bar{\ell} \in L$ as the *initial location*;
- $\mathcal{X}$: finite set of *clocks*;
- Σ: finite set of *events*, including a subset $\Sigma_u \subseteq \Sigma$ of *urgent* ones;
- inv: $L \to CC(\mathcal{X})$: assigns a clock constraint (*invariant*) to each location;
- prob $\subseteq L \times CC(\mathcal{X}) \times \Sigma \times \text{Dist}(2^{\mathcal{X}} \times L)$: the *probabilistic edge relation.*

Each *state* of a PTA is a pair $(l, v) \in L \times \mathbb{R}^{\mathcal{X}}_{\geq 0}$ such that $v \models inv(l')$. In any state (l, v), a certain amount of time $t \in \mathbb{R}_{\geq 0}$ elapses or an action $a \in \Sigma$ is performed. For time to elapse, t must be chosen in such a way that $inv(l)$ remains satisfied after the time has passed. The resulting state is $(l, v+t)$. Similarly, an action a can be performed if it is enabled, namely $enab(l, a)$ is satisfied by v. Upon selecting an enabled action a, a successor location and a set of variables to be reset are sampled according to the distribution $prob(l, a)$. In this research, we denote the elements of the form $(X, l') \in 2^{\mathcal{X}} \times L$ supporting $prob(l, a)$ as edges, and their set $edge(l, a)$ [42].

Definition 4.3 (Path). A *path* is an infinite sequence $\pi = (l_0, v_0) \xrightarrow{a_0} (l_1, v_1) \xrightarrow{a_1} \cdots$, where each $(l_i, v_i) \in L \times \mathbb{T}^{\mathcal{X}}$ is a valid state, and each transition $(l_i, v_i) \xrightarrow{a_i} (l_{i+1}, v_{i+1})$ is valid under the automaton's semantics.

Furthermore, we use the integral semantics of probabilistic timed automata with diagonal constraints, that is, the time domain is $\mathbb{N}$ rather than $\mathbb{R}_{\geq 0}$. Thus, for any $x \in \mathcal{X}$ there exists c_x a maximal constant such that there exists $\chi \in CC(\mathcal{X})$ of the form $x \sim c_x$, where $\sim \in \{\leq, =, \geq\}$. In this paper, the digital clock method is applied to verify the temporal properties of the form $\mathbb{P}_{\bowtie p}[\cdot]$. We focus mainly on extremal reachability properties, specifically $\mathbb{P}_{\max} =?[\Diamond L]$ and $\mathbb{P}_{\min} =?[\Diamond L]$. $\mathbb{P}_{\max}$ denotes the maximum (best case) probability of reaching a given location, while $\mathbb{P}_{\min}$ represents its worst case value [35]. The actual reachability value lies in the interval defined by these two bounds. The digital clock valuation $v \in \mathbb{N}^{\mathcal{X}}$ is the mapping of the clocks with their values. We denote the time progression as the valuation increment function, namely $v \oplus 1$ [42].

$$(v \oplus 1)(x) = \min(v(x) + 1, c_x + 1) \tag{1}$$

This paper uses the standard definition of digital clock semantics found in the work of Norman et al. [42].

Definition 4.4 (Digital Clock Semantics). Let $\mathcal{X}$ be a set of clocks. The semantics of the digital clock abstraction are given by:

$$(S, Steps)$$

where:

- $S \subseteq L \times \mathbb{N}^{\mathcal{X}}$ is the set of symbolic states such that:

$$S = \left\{(l, v) \in L \times \mathbb{N}^{\mathcal{X}} \,\middle|\, v \models inv(l) \wedge \exists x \in \mathcal{X},\ v(x) \leq c_x + 1\right\}$$

- $Steps$: $(L \times \mathbb{N}^{\mathcal{X}}) \times (Act \cup \{1\}) \to \text{Dist}(S)$ is the transition function defined as follows:
  - *Time step:* if $a = 1$, and $v \oplus 1 \models inv(l)$, then
  $$Steps((l, v), 1) = \mu(l, v \oplus 1);$$
  - *Action step:* if $a \in Act$ and $v \models enab(l, a)$, then for all $(l', v') \in S$,
  $$Steps((l, v), a)(l', v') = \sum_{\substack{X \subseteq \mathcal{X} \\ v' = v[X:=0]}} prob(l, a)(X, l').$$

5 METHOD

5.1 Overview

In this research, we propose an extension of the digital clock abstraction algorithm that allows diagonal constraints. The main goal is to handle these types of constraints in a way that the resulting Markov decision process after transforming the probabilistic timed automaton has a finite state space while preserving the original properties of the automaton. In our context, by properties we refer to the reachability probabilities $\mathbb{P}_{\max}$ and $\mathbb{P}_{\min}$.

The digital clock abstraction algorithm begins by reducing the real clocks into integer variables. The algorithm builds a new discrete-time automaton by systematically applying all enabled actions at each location as transitions. Moreover, it unrolls the time steps, replacing the clock variables with their values in the resulting Markov decision process. The standard digital clocks will explore the state space until each clock variable reaches a value that exceeds the maximum constant appearing in any constraint involving comparisons with the given clock variable. The resulting Markov decision process after applying the standard digital clock abstraction is the region of Figure 2 highlighted in light pink.

We propose an extended algorithm that also begins by discretising the real-valued clocks. Instead of capping each clock variable x at c_x where c_x is the maximum constant compared to x in the list of constraints, as formalised in Definition 1, we suggest continuing to increment clocks until all have reached their maximum value. This means that a clock can exceed its maximum value if any other clocks have not reached theirs. A key aspect is that we are not interested in exploring invalid states; therefore, we need to ensure that we are close to the accepted region of the diagonal constraint. We achieve this by checking whether the states are in a wider region bounded by lines parallel to the original diagonal constraints and shifted according to the maximum value of each clock as presented in Figure 3. Figure 2 highlights the additional states we explore with our proposed algorithm in darker pink.

Once a diagonal constraint is not satisfied, the difference between clocks along with the progression of time will be the same, thus the algorithm can terminate. In each location, the extended digital clock algorithm performs a *tick action* according to the increment function or an action according to the probability distribution. A *tick action* can only be performed if the value after applying the increment function satisfies the invariant of the target location. Such invariants are called strong; however, Modest uses weak invariants [24]. This means that Modest allows us to enter a location in which its invariant is immediately false, but time cannot progress there.

5.2 Changes in semantics

The increment of the digital clock method in the valuation function $v \oplus_d 1$ of x is no longer capped at $c_x + 1$ where c_x is the maximum constant compared to x, but it is extended to increment as long as there exists at least one other clock that did not reach its maximum and the values are kept in the zone that satisfies the diagonal bounds we set as in Definition 2.

$$(v \oplus_d 1)(x) = \begin{cases} v(x)+1, & \text{if } \exists\, x \in \mathcal{X} \text{ such that } v(x) < c_x + 1 \text{ and} \\ & \quad \forall\, a, b \in \mathcal{X},\ v(a) - v(b) \le c_a + 1 \\ v(x), & \text{otherwise} \end{cases} \tag{2}$$

Similarly to the standard algorithm, $v[X := 0]_d$ represents the clock valuation that resets all clocks from X to 0. For each clock $y \notin X$, its value is updated to the minimum of its current value or its associated maximum constant incremented by 1, so that the resulting pair (x, y) is between our defined bounds after reset.

$$v[X := 0]_d(a) = \begin{cases} 0 & \text{if } a \in X, \\ \min(v(a), c_a + 1) & \text{otherwise.} \end{cases} \tag{3}$$

The maximum value of the clock incremented by 1 typically corresponds to the last value of x where the point is positioned in the diagonal bound condition where the first member is x. This change is necessary to ensure that when clock variables values are reset, they do not fall outside of the accepted region.

The semantics of digital clocks with diagonal constraints are defined similarly to the standard semantics for the digital clock method.

Definition 5.3 (Digital Clock Semantics). Let $\mathcal{X}$ be a set of clocks. The semantics of the digital clock abstraction are given by:

$$(S,\ \mathit{Steps})$$

where:

- $S \subseteq L \times \mathbb{N}^{\mathcal{X}}$ is the set of symbolic states such that: $S = \{(l, v) \mid v \models \mathit{inv}(l) \wedge (\exists x \in \mathcal{X}, v(x) \le c_x + 1 \wedge \forall a, b \in \mathcal{X}, v(a) - v(b) \le c_a)\}$;
- $\mathit{Steps}\colon (L \times \mathbb{N}^{\mathcal{X}}) \times (\mathit{Act} \cup \{1\}) \to \mathrm{Dist}(S)$ is the transition function defined as follows:
  - *Time step:* if $a = 1$, and $v \oplus_d 1 \models \mathit{inv}(l)$, then

$$\mathit{Steps}((l, v), 1) = \mu(l, v \oplus_d 1);$$

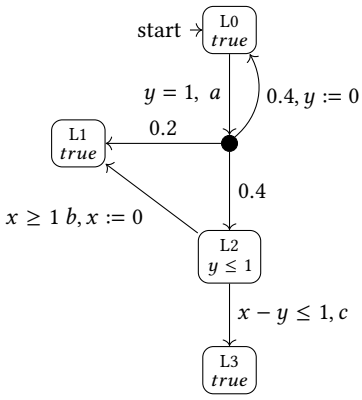

Fig. 1. Example Probabilistic Timed Automaton

  - *Action step:* if $a \in \mathit{Act}$ and $v \models \mathit{enab}(l, a)$, then for all $(l', v') \in S$,

$$\mathit{Steps}((l, v), a)(l', v') = \sum_{\substack{X \subseteq \mathcal{X} \\ v' = v[X:=0]_d}} \mathit{prob}(l, a)(X, l').$$

5.4 Case Study

To evaluate the accuracy of the extended digital clock abstraction, we constructed a representative probabilistic timed automaton that includes a diagonal constraint.

We consider the PTA from Figure 1. $\mathcal{A} = (L, \bar{\ell}, \mathcal{X}, \Sigma, \mathit{inv}, \mathit{prob})$ where: $L = \{L0, L1, L2, L3\}$ is the set of locations with initial location $\bar{\ell} = L0$; $\mathcal{X} = \{x, y\}$ is the set of clocks; $\Sigma = \{a, b, c\}$ is the set of actions; $\mathit{inv}(L2) = y \le 1$, and $\mathit{inv}(\ell) = \text{true}$ for all $\ell \in L \setminus \{L2\}$. From $L0$, if $y = 1$, the action a resets y and loops to $L0$ with probability 0.4; with probability 0.4, the system makes a probabilistic transition to an intermediate node that leads to $L1$ with probability 0.2 and to $L2$ with probability 0.4. From $L2$, the action b is enabled if $x \ge 1$, resets x and moves to $L1$; action c is enabled if $x - y \le 1$ and leads to $L3$ without resetting any clocks. This automaton is relevant in testing our extended digital clocks method, as it has valid states that the standard semantics would not explore.

After applying the algorithm, the resulting Markov decision process is represented in Figure 2. The algorithm starts by discretising all clocks and computes the maximum constant of each of them. For each location, it attempts to apply both after actions whose guards are satisfied and tick actions. To perform a tick action, it checks the conditions stated in Definition 2. In this definition, we apply our values to the increment function.

$$(v \oplus_d 1)(x) = \begin{cases} x+1, & \text{if } (x < 2 \vee y < 2 \\ & \quad \wedge\, x - y \le 2 \\ & \quad \wedge\, y - x \le 2 \\ x, & \text{otherwise} \end{cases} \tag{4}$$

To better illustrate the difference between the standard digital clock algorithm and the extension proposed by this research, we refer to Figure 3 which depicts all the possible states that can be

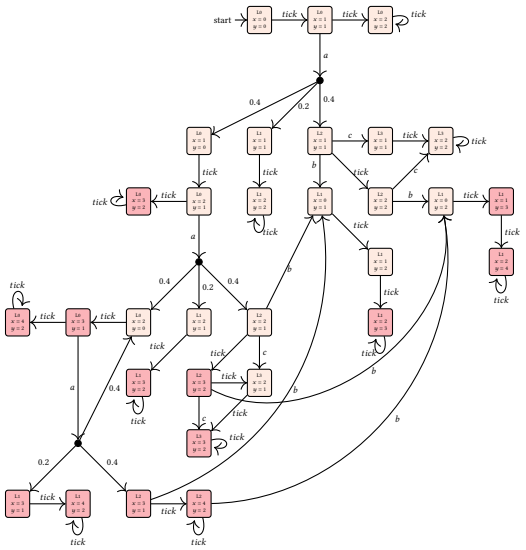

Fig. 2. MDP after digital clock abstraction on our example PTA

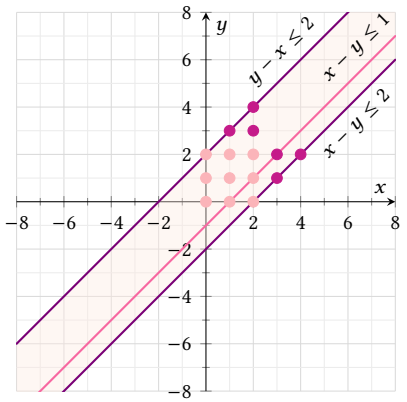

Fig. 3. A graph of the explored states with the extended semantics compared to standard digital clock algorithm

explored in our example automaton. Although the graph does not depict the exact states explored during the construction of the Markov decision process in Figure 2, it shows the full set of states that could potentially be reached. Consider that the automaton also contains a diagonal constraint $x - y \leq 1$. Figure 3 shows the main diagonal constraint $x - y \leq 1$, and the diagonal limits of the state exploration $y - x \leq 2$ and $x - y \leq 2$. In the state exploration, we want to explore the states between two diagonals parallel to the diagonal constraint. This graph illustrates the complete set of states reachable under the standard digital clock abstraction highlighted in light pink along with the additional states that we can explore using our extended algorithm in darker pink. If we keep the standard semantics, valid states like $(3, 2)$ would not be explored, thus affecting the computation of the automaton's properties. To ensure complete exploration of valid states during model checking, we extend the state space by drawing diagonal constraints between clock pairs relative to the maximum of the first clock. This captures a larger finite region while preserving the properties of the original probabilistic timed automaton. Furthermore, an important aspect that we changed in the initial semantics is the reset clock function. Let us take the same point $(3, 2)$, if we attempt to reset y to 0 using standard semantics, we end up in an invalid state $(3, 0)$. Therefore, we changed the reset to change the other clocks to their maximum value incremented by 1 if they are larger, so that after applying it, we are still within the limits in state $(2, 0)$ and we can continue exploring states in the accepted region.

5.5 Implementation

In this work, we extended the digital clock method and implemented it in the Modest Toolset. Since portability was a concern, we used the C# programming language to modify and integrate the algorithm extension into the Modest Toolset. The digital clock transformation is a syntax level conversion; thus, the *moconv* tool can transform the models specified in the Modest modelling language [6] or JANI format [12], into a Markov decision process (MDP). The resulting MDP is in JANI format and can be checked using the *mcsta* tool [27].

The representation of diagonal constraints is fully dependent on the modelling of clock constraints within the Modest Toolset and ultimately on the definition of Alur and Dill [2]. As a result, our implementation allows *diagonal constraints* of the form $x - y \leq c$ or $x - y \geq c$ where c is an arbitrary constant and x, y are clock variables and other equivalent relations. These constraints can be found as both location invariants and guards within the specified probabilistic timed automaton. We implemented the extended algorithm along three axes, namely in terms of modelling semantics, modifying the digital clock algorithm, and changes in other components of Modest Toolset to allow models with diagonal constraints to be checked.

5.5.1 Quantifier expressions. First, we formalised the extended semantics by implementing existential and universal quantifier expressions, therefore applying the modified semantics introduced in Definition 2. To understand the quantifier implementation, we must first examine how the Modest Toolset handles expressions. In Modest, the control flow is based on dynamically updating attributes on a shared abstract syntax tree to track active sub-expressions, enabling the structured execution and progression of processes [7]. Therefore, an expression in the Modest Toolset is an abstract syntax tree (AST) node where each inner expression is a subtree, each representing a mathematical, logical, or symbolic computation.

Quantifier expressions generalise Boolean conditions over a domain of variables. We implemented a quantifier expression that extends the expression as a base class. It contains methods for serialisation, deserialisation, AST traversing, variable collection, and string representation.

Building on the base quantifier expression, we implemented the universal quantifier that represents expressions of the form $\forall a \in D : \phi(a)$. Each universal quantification binds a variable to a specified domain, where the domain defines the set of values to be substituted for the quantified variable. It also includes a mapping from the variable to an expression. Therefore, we modelled the universal quantifier to allow for nested expressions, each expression of the

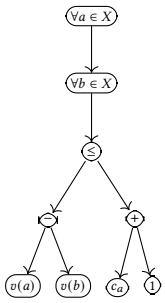

Fig. 4. Abstract Syntax Tree for expression $\forall\, a, b \in \mathcal{X},\ v(a) - v(b) \leq c_a + 1$

form $\forall a, b \in D : \phi(a, b)$. becomes a nested expression $\forall a \in D, \forall b \in D : \phi(a, b)$ where the second expression will be the predicate of the first. The abstract syntax tree of the expression introduced in our extended semantics $\forall a, b \in X : v(a) - v(b) \leq c_a + 1$. is illustrated in Figure 4.

When evaluating the expression, the solver traverses the universal quantifier expression AST and first substitutes the outermost variable, working its way to the innermost variable. While traversing, it transforms it into its most simple form, solving it if possible. In case of $\forall a, b \in X : v(a) - v(b) \leq c_a + 1$., it will consider $\forall a \in X, \forall b \in X : v(a) - v(b) \leq c_a + 1$., it will first substitute a with all values in its domain, and then recurse into the inner expression $\forall b \in (X \setminus \{a\}) : a - v(b) \leq c_a + 1$. and substitute b and perform Boolean conjunction on all sub-expressions. Similarly, it will recursively evaluate the inequality, then the difference $v(a) - v(b)$ and the addition $c_a + 1$, and ultimately the constant values $v(a), v(b), c_a, 1$. After evaluating the constant values, the algorithm will traverse back to the original expression, computing and substituting the values of the inner expressions. The universal expression is evaluated by applying the *Boolean and* operation to all its inner expressions. The logic and implementation of the existential quantifier are almost identical to the universal quantifier, with the primary distinction that its evaluation applies a *Boolean disjunction* to its inner expressions rather than a conjunction.

5.5.2 Digital Clocks Algorithm. The Modest Toolset's digital clock transformation is the underlying algorithm used by the *moconv* tool to transform a probabilistic timed automaton into its MDP representation in JANI format when given the *–digital-clocks* flag. The *moconv* tool does not apply valuations to the clock variables, it produces a symbolic expression-based representation of the resulting MDP in JANI format. The model checker applies concrete values to the clocks, in our case the *mcsta* tool as part of the state generation and analysis phase. In this section, we discuss the implementation and changes within digital clock abstraction.

We integrated the modelled expressions and the modified semantics into the existing implementation of the digital clock algorithm, we modified the current implementation of the reset function, and we made the parser of the clock constraints allow for open diagonal constraints. The extended digital clock abstraction relies, similarly to the standard algorithm, on the assumption of closed constraints; therefore, expressions of the form $x - y < c, x - y > c, x < c, x > c$ for clock variables x, y and an arbitrary constant c are not supported.

Implementing the new semantics implied changes in the increment function, that is, instead of the expression $\min(v(x) + 1, c_x + 1$, we injected the expression $\exists x \in X : v(x) < c_a + 1 \wedge \forall a, b \in X : v(a) - v(b) \leq c_a + 1$. For each clock, relevant information is encapsulated in a dedicated class storing its integer abstraction of the original real-valued clock, associated metadata such as the computed upper bound, and the increment assignment used to simulate time progression within the digital clock semantics. The increment function is already applied to all clocks; thus we only changed the function inside the constructor of the clock information class. We first create the existential quantifier $\exists x \in X : v(x) < c_x + 1$, then the universal one $\forall a, b \in X : v(a) - v(b) \leq c_a + 1$, then we apply a Boolean conjunction to them, and lastly we form a choice expression, which if the conjunction holds, it will increment the clock value; otherwise, it will keep the same one.

Another aspect we needed to modify to make the digital clock algorithm accommodate diagonal constraints is the reset function of the clocks. In its standard implementation, the function resets the current clock to 0; however, in our extended algorithm, we need all points (x, y) where x, y are clock variables to still be within the bounds of our diagonals to ensure that the expression $\forall a, b \in X : v(a) - v(b) \leq c_a + 1$ holds. We addressed this by checking for each reset whether the clock value pairs satisfy the diagonals defined by $\forall a, b \in X : v(a) - v(b) \leq c_a + 1$. For each reset of the clock, we retrieve the defined bound diagonals that contain the given clock, we choose the for the other variables within the constraints the minimum value between the current one and the maximum incremented by 1. This ensures that the additional states that we explore are valid and reachable.

5.5.3 Modest Toolset Integration. The last step to integrate the algorithm is to allow diagonal constraints in the probabilistic timed automaton given as input to the *moconv* tool and to ensure that when checking the model *mcsta* allows the clock valuations to go beyond their $c_x + 1$ maximum. We achieved this by removing checks that have been performed throughout the process for each tool separately, checks for diagonal-free automata, and for the maximum bound of values taken by the clock variables.

6 RESULTS

To evaluate the feasibility of our approach, as well as its scalability when introducing more clock variables, we performed two case studies. In this section, we demonstrate the extended digital clock transformation by presenting the results of selected case studies and comparing them against manually computed ones. We measure the difference between expected and actual probability and assess our algorithm. The benchmarks we analyse aim to measure the accuracy and closeness to the actual model of the model-checking in the Markov decision process generated by our algorithm.

We first analyse the results of the same automaton that we explained in the previous section illustrated in Figure 1 and its associated Markov decision process in Figure 2 resulted after applying digital clocks. When checking this model, we consider properties such as the reachability probabilities of the final locations $L1$ and $L3$. We computed the values in Table 1 by running the model-checking

Table 1. Model-Checking Results using *mcsta*

Property	Prob.	Bounds	Time (s)	Error	Iter.	VI Time (s)
P3Min	0	[0, 0]	0.0	0	1	0.0
P3Max	0.56	[0.56, 0.56]	0.0	0	2	0.0
P1Max	0.9360002	[0.9360002, 0.9360002]	0.0	0	2	0.0
P1Min	0	[0, 0]	0.0	0	2	0.0

Table 2. Comparison of Manually Computed and Tool-Based Reachability Probabilities

Property	Manually Computed Probability	Bounds computed with mcsta
P3Max	0.56	[0.56, 0.56]
P3Min	0	[0, 0]
P1Max	0.99	[0.936, 0.936]
P1Min	0	[0, 0]

mcsta tool on the JANI file resulted after performing the digital clock conversion using *moconv*.

The manual calculations supporting our results are presented in Table 2, and now we detail the methodology used to derive these values. In Table 2 we report reachability probabilities for specific target locations. The notation *PnMax* represents the maximum probability of reaching a location *Ln*, while *PnMin* denotes the corresponding minimum probability. For example, $P3Max = 0.56$ means that the best-case probability of reaching $L3$ is 0.56. The values computed by *mcsta* correspond to the same probabilities, providing lower and upper bounds for the reachability probabilities inferred by checking the model.

We begin by analysing the reachability probability of location $L1$. The worst-case scenario for reaching $L1$ is when this location is never reached. Examples of such a path are $\pi_1 = (l_0, l_2, l_3)$ or $\pi_1 = (l_0, l_0, l_2, l_3)$. By showing examples of such paths exist, we can compute $\mathbb{P}_{\min} = 0$. To calculate $\mathbb{P}_{\max}$ of reaching $L1$, we sum the probability of going from $L0$ to $L1$ which we denote $P1_{L1}$, with the probability of reaching $L1$ from $L0$ through $L2$ ($P2_{L1}$). We now compute the probability of going directly from $L0$ to $L1$, a path given by $\pi_n = (\underbrace{l_0, \ldots, l_0}_{n \text{ times}}, l_1)$. Because it is unknown how many times we will return to $L0$, this probability can be approximated using the formula for infinite geometric series [47].

$$\sum_{k=0}^{\infty} r^k = \frac{1}{1-r}, \quad \text{if } |r| < 1 \tag{5}$$

Thus, our $P1_{L1}$ can be calculated as follows.

$$P1_{L1} = 0.2 \cdot \sum_{k=0}^{\infty} 0.4^k = 0.2 \cdot \left(\frac{1}{1-0.4}\right) = 0.2 \cdot \left(\frac{1}{0.6}\right) = \frac{0.2}{0.6} = \frac{1}{3} \approx 0.33 \tag{6}$$

Now we calculate $P2_{L1}$. The paths are of the form $\pi_n = (\underbrace{l_0, \ldots, l_0}_{n \text{ times}}, l_2, l_1)$. Similarly to $P1_{L1}$, we can use the infinite geometric series formula to calculate $P2_{L1}$ as in Definition 7. We computed 1 the probability of being in $L2$ and transitioning to a location $L1$, dividing the number of cases in which there exists a path from $L2$ to $L1$ in Figure 2 by the number of times that location $L2$ is reached through that path. Now we calculate the probability of reaching $L1$ through $L2$ by multiplying the probability of reaching $L2$ from $L0$ the probability of reaching $L1$ once $L2$ is reached.

$$P2_{L1} = 1 \cdot 0.4 \sum_{k=0}^{\infty} 0.4^k = 0.4 \cdot \left(\frac{1}{1-0.4}\right) = 0.4 \cdot \left(\frac{1}{0.6}\right) = \frac{0.4}{0.6} = \frac{4}{6} \approx 0.66 \tag{7}$$

We computed the overall probability of reaching $L1$ to be approximately $\mathbb{P}_{\max} = P_{L1} = P1_{L1} + P2_{L1} \approx 0.99$. Ultimately, we calculate the probabilities of the reachability of $L3$. Similarly to $L1$, there are paths that do not reach $L3$ at all. Such examples include $\pi_1 = (l_0, l_2, l_1)$ or $\pi_1 = (l_0, l_1)$, and taking this into account the worst-case probability $\mathbb{P}_{\min} = 0$. Finally, we compute the best-case probability of reaching $L3$. To calculate $\mathbb{P}_{\max}$, we need to take into account when the diagonal constraint is satisfied in $L2$. From $L0$, with the probability of 0.4, there will be a reset of y and a return to $L0$. This increases the difference between x, y. To ensure that the diagonal constraint $x - y \leq 1$ is met, only two resets can be performed before reaching location $L2$ with valuations $(1, 1)$, $(2, 1)$. Performing a third reset would result in the valuation $(3, 1)$ for which the constraint does not hold.

$$P3Max = 0.4 + 0.4 \cdot 0.4 = 0.56 \tag{8}$$

We observe that the results we obtain by performing model checking on the Markov decision process resulting after the digital clock abstraction algorithm are similar to the results we expected. This suggests that in this particular case, the extension of the digital clock algorithm preserved the probability properties of reachability of the original probabilistic timed automaton; however, the difference between reachability of $L1$ is higher than expected. In the case of location $L3$ reachability, our findings are more significant because the bounds are tighter than of $L1$, we have the exact result we expected. We see a difference between the manually calculated probability of reaching $L1$, and the output result of the *mcsta* tool. This can perhaps be explained by the approximation we did in the infinite geometric series, we do not know how many times y will be reset, and our algorithm continues as long as there exists at least one clock which has not reached its maximum value, which in our context would be y. Thus, an approximation must be made in order to calculate the probability of reaching $L1$ directly from $L3$. However, we expected a reachability probability of $L1$ closer to 1 than we got as a result, because all paths through the automaton can reach $L1$ in the best-case scenario. This discrepancy of 0.06 may be caused by our extension algorithm or the way it integrates with the *mcsta* tool.

6.1 Probabilistic timed automaton with more than two clocks

This case study shows a limitation of our algorithm in terms of scalability. For simplicity, consider an example automaton with three clock variables x, y, z where the maximum values of the clocks are $c_x = 3, c_y = 3, c_z = 2$, thus we will have the increment function illustrated in Definition 10. Let the current location have invariants $x - y \leq 1 \wedge \leq y - z \leq 1$. Consider that we reach the point $(2, 2, 2)$ that satisfies the invariant and increment condition, which makes it

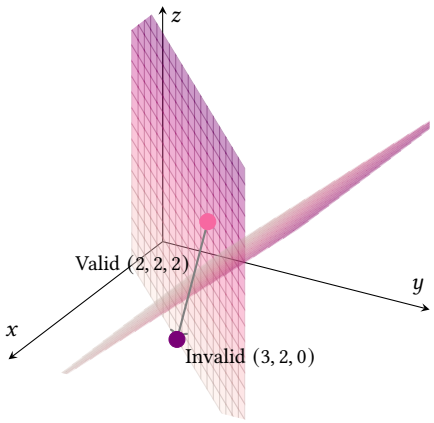

Fig. 5. Constrain Violation After Reset in PTA with Three Clocks

a valid state in the automaton. If we attempt to reset z and go back to this location, it yields $(3, 2, 0)$ where x is chosen to satisfy $z - x \leq 3$; however, the location invariant $y - z \leq 1$ is no longer satisfied. Therefore, our reset operation disrupted a diagonal involving two non-reset clocks; thus the diagonal constraints are still not supported for more than two clocks. The path is defined as follows.

$$(0,0,0) \xrightarrow{\oplus 1} (1,1,1) \xrightarrow{\oplus 1} (2,2,2) \xrightarrow{\text{reset } y} (3,2,0) \tag{9}$$

$$(v \oplus 1)(x) = \begin{cases} v(x)+1, & \text{if } x < 4 \wedge y < 4 \wedge z < 3 \\ & x - y \leq 4 \wedge x - z \leq 4 \wedge \\ & y - x \leq 4 \wedge y - z \leq 4 \wedge \\ & z - x \leq 3 \wedge z - y \leq 3 \\ v(x), & \text{otherwise} \end{cases} \tag{10}$$

This limitation of the extended digital clock abstraction algorithm can be visualised in Figure 5. The point $(2, 2, 2)$ is at the intersection of the accepted regions of the planes determined by the invariants $x - y \leq 1 \wedge \leq y - z \leq 1$. When resetting the clock z, the value of x also changed to maintain the values within the bounds; however, the invariant is no longer satisfied. This breaks the formal semantics of the model and leads to inconsistent analysis results.

7 CONCLUSION

We presented an algorithm to extend the digital clock abstraction to allow and support diagonal constraints while preserving the properties of the original probabilistic timed automaton. We implemented it in the Modest Toolset as an underlying component of the *moconv* tool, which transforms a probabilistic timed automaton with diagonal constraints into a JANI file specification of a Markov decision process. This research aimed to provide a comprehensive explanation of the behaviour and changes necessary to extend the standard digital clock algorithm to support diagonal constraints, while preserving the initial properties of the probabilistic timed automaton and documenting the modifications applied *RQM1*.

To assess whether *RQM1* we documented the modifications applied to the initial implementation of the digital clock abstraction in the Modest Toolset and provided a case study to prove its correctness on some types of input probabilistic timed automata. We implemented and modelled the universal and existential quantifier expression to allow Modest to evaluate our extended semantics. Then we modelled the extended semantics using the universal and existential quantifier over the domain of clock variables, which will be substituted with values during model checking. Finally, we ensured that diagonal constraints of the form $x - y \leq c$ and $y - x \leq c$ are allowed by the tool *moconv* and can be modelled in the digital clock algorithm. The last significant change necessary was to modify the *mcsta* tool to allow the clock variables to exceed their maximum value. We applied these changes to the Modest Toolset source code and analysed its correctness in terms of properties. There appears to be no significant difference in the reachability probabilities given as a result of *mcsta* tool, and the expected manual results, which satisfies the *RQM1*.

We analyse the behaviour of our extended semantics in the case study and show the possibilities in terms of state-space exploration as formulated in *RQM2*. We proved that by not capping our clock variables at their own maximum value and allowing them to exceed them, we explore additional states that are still valid under the diagonal constraints. Therefore, we preserved the properties of the probabilistic timed automaton with a diagonal constraint, as suggested by the similarity of expected and observed results. We applied the extension of the digital clock algorithm to a dual sensor alarm system modelled as a probabilistic timed automaton with one diagonal constraint, thus achieving *RQM2*.

Lastly, a focus of our research was also the way our extended algorithm scales with a higher number of clock variables, a requirement specified in *RQM3*. Although the number of clocks does not seem to interfere with state exploration, not enough testing was done to ensure this. We discovered a limitation of the algorithm when provided with more than two variables; namely, when resetting the clocks based on our semantics there will be cases in which the changes in clocks will determine a violation in the location invariant. We illustrated this issue and hence, our extended algorithm currently does not support more than two clocks.

7.1 Future Work

A possible area of future research would be developing a formal mathematical proof to demonstrate the correctness of the extended digital clock abstraction. This research suggests the effectiveness of the extended algorithm on the probabilistic timed automaton analysed; however, it does not guarantee the correctness of the semantics. This can be achieved in future work with a formal proof. For future work, another field of improvement is the way the current algorithm deals with more than two clocks. As the number of clocks increases, the algorithm proposed in this research exhibits failure, mainly due to our approach of clock resets. This should be researched and analysed further and a solution should be found to ensure the scalability of our extended digital clock abstraction.

8 ACKNOWLEDGEMENTS

I thank Bram Kohlen for his guidance and assistance during this research. Additionally, my gratitude extends to the researchers who collaborated to implement the Modest Toolset and provided documentation for its source code.

REFERENCES

[1] Rajeev Alur. 1999. Timed Automata. In *Computer Aided Verification, Proceedings of the 11th International Conference on Computer-Aided Verification (CAV 1999)*, Nicolas Halbwachs and Doron A. Peled (Eds.). Lecture Notes in Computer Science, Vol. 1633. Springer Berlin Heidelberg, Berlin, Germany, 8–22. https://doi.org/10.1007/3-540-48683-6_3

[2] Rajeev Alur and David L. Dill. 1994. A Theory of Timed Automata. *Theoretical Computer Science* 126, 2 (1994), 183–235. https://doi.org/10.1016/0304-3975(94)90010-8

[3] Christel Baier, Luca de Alfaro, Vojtěch Forejt, and Marta Kwiatkowska. 2018. Model Checking Probabilistic Systems. In *Handbook of Model Checking*, Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem (Eds.). Lecture Notes in Computer Science, Vol. 10426. Springer International Publishing, Cham, Switzerland, 963–999. https://doi.org/10.1007/978-3-319-10575-8_28

[4] Danièle Beauquier. 2003. On probabilistic timed automata. *Theoretical Computer Science* 292, 1 (2003), 65–84. https://doi.org/10.1016/S0304-3975(01)00215-8

[5] Johan Bengtsson and Wang Yi. 2004. Timed Automata: Semantics, Algorithms and Tools. In *Lectures on Concurrency and Petri Nets*, Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg (Eds.). Lecture Notes in Computer Science, Vol. 3098. Springer Berlin Heidelberg, Berlin, Heidelberg, 87–124. https://doi.org/10.1007/978-3-540-27755-2_3

[6] Henrik Bohnenkamp, Pedro R. d'Argenio, Holger Hermanns, and Joost-Pieter Katoen. 2006. MoDeST: A Compositional Modeling Formalism for Hard and Softly Timed Systems. *IEEE Transactions on Software Engineering* 32, 10 (2006), 812–830. https://doi.org/10.1109/TSE.2006.104

[7] Henrik Bohnenkamp, Holger Hermanns, Joost-Pieter Katoen, and Ric Klaren. 2003. The Modest Modelling Tool and Its Implementation. In *Computer Performance Evaluation: Modelling Techniques and Tools, 13th International Conference (TOOLS 2003) (Lecture Notes in Computer Science, Vol. 2794)*, Peter Kemper and William H. Sanders (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 116–133. https://doi.org/10.1007/978-3-540-45232-4_8

[8] Patricia Bouyer and Fabrice Chevalier. 2005. On Conciseness of Extensions of Timed Automata. *Journal of Automata, Languages and Combinatorics* 10, 4 (2005), 393–405. https://doi.org/10.25596/jalc-2005-393

[9] Patricia Bouyer, François Laroussinie, and Pierre-Alain Reynier. 2005. Diagonal Constraints in Timed Automata: Forward Analysis of Timed Systems. In *Formal Modeling and Analysis of Timed Systems (FORMATS 2005) (Lecture Notes in Computer Science, Vol. 3829)*, Paul Pettersson and Wang Yi (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 112–126. https://doi.org/10.1007/11603009_10

[10] Pierre Brémaud. 2017. *Discrete Probability Models and Methods*. Probability Theory and Stochastic Modelling, Vol. 78. Springer International Publishing, Cham, Switzerland. xiv+559 pages. https://doi.org/10.1007/978-3-319-43476-6

[11] Carlos E. Budde, Pedro R. D'Argenio, Juan A. Fraire, Arnd Hartmanns, and Zhen Zhang. 2025. *Modest Models and Tools for Real Stochastic Timed Systems*. Lecture Notes in Computer Science, Vol. 15261. Springer, Cham, Switzerland, 115–142. https://doi.org/10.1007/978-3-031-75775-4_6

[12] Carlos E. Budde, Christian Dehnert, Ernst M. Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. 2017. JANI: Quantitative Model and Tool Interaction. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2017) (Lecture Notes in Computer Science, Vol. 10206)*, Axel Legay and Tiziana Margaria (Eds.). Springer Berlin Heidelberg, Berlin, Germany, 151–168. https://doi.org/10.1007/978-3-662-54580-5_9

[13] Yuliya Butkova, Arnd Hartmanns, and Holger Hermanns. 2019. A Modest Approach to Modelling and Checking Markov Automata. In *Proceedings of the 16th International Conference on Quantitative Evaluation of Systems (QEST 2019) (Lecture Notes in Computer Science, Vol. 11785)*, David Parker and Verena Wolf (Eds.). Springer Cham, Cham, Switzerland, 52–69. https://doi.org/10.1007/978-3-030-30281-8_4

[14] Xi Chen, Harry Hsieh, Felice Balarin, and Yosinori Watanabe. 2003. Case Studies of Model Checking for Embedded System Designs. In *Proceedings of the 3rd International Conference on Application of Concurrency to System Design (ACSD)*, J. Lilius, F. Balarin, and R. J. Machado (Eds.). IEEE Computer Society, Los Alamitos, CA, USA, 20–28. https://doi.org/10.1109/CSD.2003.1207696

[15] Edmund M. Clarke. 1997. Model Checking. In *Proceedings of the Foundations of Software Technology and Theoretical Computer Science (FSTTCS 1997) (Lecture Notes in Computer Science, Vol. 1346)*, S. Ramesh and G. Sivakumar (Eds.). Springer, Berlin, Heidelberg, 54–56. https://doi.org/10.1007/BFb0058022

[16] Fady Copty, Limor Fix, Ranan Fraer, Enrico Giunchiglia, Gila Kamhi, Armando Tacchella, and Moshe Y. Vardi. 2001. Benefits of Bounded Model Checking in an Industrial Setting. In *Proceedings of the 13th International Conference on Computer-Aided Verification (CAV) (Lecture Notes in Computer Science, Vol. 2102)*, Gérard Berry, Hubert Comon, and Alain Finkel (Eds.). Springer Berlin Heidelberg, Paris, France, 436–453. https://doi.org/10.1007/3-540-44585-4_30

[17] Conrado Daws, Marta Kwiatkowska, and Gethin Norman. 2004. Automatic Verification of the IEEE 1394 Root-Contention Protocol with KRONOS and PRISM. *International Journal on Software Tools for Technology Transfer* 5, 2-3 (2004), 221–236. https://doi.org/10.1007/s10009-003-0118-5

[18] Stephen Edwards, Luciano Lavagno, Edward A. Lee, and Alberto Sangiovanni-Vincentelli. 1997. Design of embedded systems: formal models, validation, and synthesis. *Proc. IEEE* 85, 3 (1997), 366–390. https://doi.org/10.1109/5.558710

[19] Borja Fernández Adiego, Dániel Darvas, Enrique Blanco Viñuela, Jean-Charles Tournier, Simon Bliudze, Jan Olaf Blech, and Víctor Manuel González Suárez. 2015. Applying Model Checking to Industrial-Sized PLC Programs. *IEEE Transactions on Industrial Informatics* 11, 6 (2015), 1400–1410. https://doi.org/10.1109/TII.2015.2489184

[20] Matthias Fruth. 2006. Probabilistic Model Checking of Contention Resolution in the IEEE 802.15.4 LR-WPAN Protocol. In *International Symposium on Leveraging Applications of Formal Methods (ISoLA) (Lecture Notes in Computer Science, Vol. 4229)*. Springer Berlin Heidelberg, Berlin, Heidelberg, 290–297. https://doi.org/10.1007/978-3-319-06410-9_39

[21] Uwe Glässer, Sarah Rastkar, and Mona Vajihollahi. 2008. Modelling and Validation of Aviation Security. In *Intelligence and Security Informatics: Techniques and Applications*, Hsinchun Chen and Christopher C. Yang (Eds.). Studies in Computational Intelligence, Vol. 135. Springer Berlin Heidelberg, Berlin, Heidelberg, 337–355. https://doi.org/10.1007/978-3-540-69209-6_18

[22] Jürgen Greifeneder and Georg Frey. 2007. Probabilistic Timed Automata for Modelling Networked Automation Systems. *IFAC-PapersOnLine* 40, 6 (2007), 1–6. https://doi.org/10.1109/INDIN.2007.4384939

[23] Darion Haase and Joost-Pieter Katoen. 2024. Unknown Biases and Timing Constraints in Timed Automata. *arXiv preprint arXiv:2403.02210* (2024). https://doi.org/10.48550/arXiv.2403.02210

[24] Ernst M. Hahn, Arnd Hartmanns, Holger Hermanns, and Joost-Pieter Katoen. 2013. A compositional modelling and analysis framework for stochastic hybrid systems. *Formal Methods in System Design* 43, 2 (2013), 191–232. https://doi.org/10.1007/s10703-012-0167-z

[25] Arnd Hartmanns. 2010. Model Checking and Simulation for Stochastic Timed Systems. In *Proceedings of the 9th International Symposium on Formal Methods for Components and Objects (FMCO 2010) (Lecture Notes in Computer Science, Vol. 6957)*, Bernhard K. Aichernig, Frank S. de Boer, and Marcello M. Bonsangue (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 372–391. https://doi.org/10.1007/978-3-642-25271-6_20

[26] Arnd Hartmanns and Holger Hermanns. 2009. A Modest Approach to Checking Probabilistic Timed Automata. In *Proceedings of the 6th International Conference on Quantitative Evaluation of Systems (QEST 2009)*. IEEE Computer Society, Turin, Italy, 187–196. https://doi.org/10.1109/QEST.2009.41

[27] Arnd Hartmanns and Holger Hermanns. 2015. Explicit Model Checking of Very Large MDPs Using Partitioning and Secondary Storage. In *Proceedings of the 13th International Symposium on Automated Technology for Verification and Analysis (ATVA 2015) (Lecture Notes in Computer Science, Vol. 9364)*, Bernd Finkbeiner, Geguang Pu, and Lijun Zhang (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 131–147. https://doi.org/10.1007/978-3-319-24953-7_10

[28] Arnd Hartmanns, Joost-Pieter Katoen, Bram Kohlen, and Jip Spel. 2021. Tweaking the Odds in Probabilistic Timed Automata. In *Quantitative Evaluation of Systems (QEST) 2021 (Lecture Notes in Computer Science, Vol. 12846)*, Alessandro Abate and Andrea Marin (Eds.). Springer Cham, Cham, Switzerland, 39–58. https://doi.org/10.1007/978-3-030-85172-9_3

[29] Arnd Hartmanns and Bram Kohlen. 2022. Backwards Reachability for Probabilistic Timed Automata: A Replication Report. *arXiv preprint arXiv:2208.11928* (2022). https://doi.org/10.48550/arXiv.2208.11928

[30] Arnd Hartmanns, Sean Sedwards, and Pedro R. D'Argenio. 2017. Efficient Simulation-Based Verification of Probabilistic Timed Automata. In *Proceedings of the 2017 Winter Simulation Conference (WSC 2017) (Winter Simulation Conference Proceedings, Vol. 2017)*, W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page (Eds.). IEEE, Las Vegas, Nevada, USA, 1419–1430. https://doi.org/10.1109/WSC.2017.8247885

[31] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. 1992. What Good Are Digital Clocks?. In *Automata, Languages and Programming: 19th International Colloquium (ICALP 1992) (Lecture Notes in Computer Science, Vol. 623)*, Werner Kuich (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 545–558. https://doi.org/10.1007/3-540-55719-9_103

[32] Marcin Jurdziński, François Laroussinie, and Jeremy Sproston. 2008. Model Checking Probabilistic Timed Automata with One or Two Clocks. *Logical Methods in Computer Science* 4, 3:12 (2008), 1–28. https://doi.org/10.2168/LMCS-4(3:12)2008

[33] Marta Kwiatkowska. 2003. Model Checking for Probability and Time: From Theory to Practice. In *Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS 2003)*, Phokion G. Kolaitis (Ed.). IEEE Computer Society Press, Los Alamitos, CA, USA, 351–360. https://doi.org/10.1109/LICS.2003.1210075

[34] Marta Kwiatkowska. 2007. Quantitative Verification: Models, Techniques and Tools. In *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2007) (ESEC/FSE '07)*, Michal Young and Steve Clarke (Eds.).

ACM Press, New York, NY, USA, 449–458. https://doi.org/10.1145/1287624.1287688
[35] Marta Kwiatkowska, Gethin Norman, and David Parker. 2018. Probabilistic Model Checking: Advances and Applications. In *Formal System Verification: State-of-the-Art and Future Trends*, Rolf Drechsler (Ed.). Lecture Notes in Computer Science, Vol. 10443. Springer International Publishing, Cham, Switzerland, 73–121. https://doi.org/10.1007/978-3-319-57685-5_3
[36] Marta Kwiatkowska, Gethin Norman, David Parker, and Jeremy Sproston. 2006. Performance Analysis of Probabilistic Timed Automata Using Digital Clocks. *Formal Methods in System Design* 29, 1 (2006), 33–78. https://doi.org/10.1007/s10703-006-0005-2
[37] Marta Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. 2002. Automatic Verification of Real-Time Systems with Discrete Probability Distributions. *Theoretical Computer Science* 282, 1 (2002), 101–150. https://doi.org/10.1016/S0304-3975(01)00046-9
[38] Marta Kwiatkowska, Gethin Norman, and Jeremy Sproston. 2000. *Symbolic Model Checking of Probabilistic Timed Automata Using Backwards Reachability.* Research Report CSR-00-1 CSR-00-1. School of Computer Science, University of Birmingham, Birmingham, United Kingdom. 1–26 pages. https://doi.org/10.48550/arXiv.2403.02210
[39] Marta Kwiatkowska, Gethin Norman, and Jeremy Sproston. 2003. Probabilistic Model Checking of Deadline Properties in the IEEE 1394 FireWire Root Contention Protocol. *Formal Aspects of Computing* 14, 3 (2003), 295–318. https://doi.org/10.1007/s001650300007
[40] Raluca Marinescu, Henrik Kaijser, Marius Mikučionis, Cristina Seceleanu, Henrik Lönn, and Alexandre David. 2015. Analyzing Industrial Architectural Models by Simulation and Model Checking. In *Formal Techniques for Safety-Critical Systems (FTSCS 2014) (Communications in Computer and Information Science, Vol. 476)*, Cyrille Artho and Peter Csaba Ölveczky (Eds.). Springer Cham, Luxembourg, 189–205. https://doi.org/10.1007/978-3-319-17581-2_13
[41] Yongan Mu, Wei Liu, Tao Lu, Juan Li, Sheng Gao, and Zihao Wang. 2023. Runtime Verification of Self-adaptive Multi-agent Systems Using Probabilistic Timed Automata. *Journal of Intelligent & Fuzzy Systems* 45, 6 (2023), 10305–10322. https://doi.org/10.3233/JIFS-232397
[42] Gethin Norman, David Parker, and Jeremy Sproston. 2013. Model Checking for Probabilistic Timed Automata. *Formal Methods in System Design* 43, 2 (2013), 164–190. https://doi.org/10.1007/s10703-012-0177-x
[43] Lucjan Pelc and Bogdan Kwolek. 2008. Activity Recognition Using Probabilistic Timed Automata. In *Pattern Recognition: Techniques, Technology and Application*, Patrick Yin (Ed.). I-Tech Education and Publishing, Vienna, Austria, 119–135.
[44] Martin L. Puterman. 1990. Markov decision processes. In *Handbooks in Operations Research and Management Science 2: Stochastic Models*, D. P. Heyman and M. J. Sobel (Eds.). Handbooks in Operations Research and Management Science, Vol. 2. North Holland, Elsevier, Amsterdam, Netherlands, 331–434. https://doi.org/10.1016/S0927-0507(05)80172-0
[45] Bastian Schlich and Stefan Kowalewski. 2009. Model checking C source code for embedded systems. *International Journal on Software Tools for Technology Transfer* 11, 3 (2009), 187–202. https://doi.org/10.1007/s10009-009-0106-5
[46] Jeremy Sproston. 2021. Probabilistic Timed Automata with Clock-Dependent Probabilities. *Fundamenta Informaticae* 178, 1–2 (2021), 101–138. https://doi.org/10.3233/FI-2021-2000
[47] James Stewart. 2015. *Calculus: Early Transcendentals* (8 ed.). Cengage Learning, Boston, MA, USA.
[48] Emília Villani, Rodrigo P. Pontes, Guilherme K. Coracini, and Ana M. Ambrósio. 2019. Integrating model checking and model-based testing for industrial software development. *Computers in Industry* 104, 23 (2019), 88–102. https://doi.org/10.1016/j.compind.2018.08.003
[49] Fuzhi Wang and Marta Kwiatkowska. 2005. An MTBDD-Based Implementation of Forward Reachability for Probabilistic Timed Automata. In *Automated Technology for Verification and Analysis (ATVA)*, Doron A. Peled and Yih-Kuen Tsay (Eds.). Lecture Notes in Computer Science, Vol. 3707. Springer Berlin Heidelberg, Berlin, Heidelberg, 385–399. https://doi.org/10.1007/11562948_29

A PRELIMINARIES

Definition A.1 (Discrete probability). [10] A *probability* on $(\Omega, \mathcal{F})$ where Ω is a set of events, is a mapping $\mathcal{P} : \mathcal{F} \to [0, 1]$ such that for an event A:

- $0 \leq P(A) \leq 1$,
- $P(\Omega) = 1$, *and*
- $P\left(\sum_{k=1}^{\infty} A_k\right) = \sum_{k=1}^{\infty} P(A_k)$

B DIGITAL CLOCK SEMANTICS IMPLEMENTATION

Listing 1. Building the universal difference-bound condition in the digital-clock conversion

```
public Expression CreateForallCondition(ILocation loc,
      ↪ Dictionary<VariableSymbol,Expression> maxValues,
      ↪ Dictionary<VariableSymbol,VariableSymbol> integerVars)
{
    // (1) Map each integer clock to its "bound = max+1"
    var boundMap = integerVars.ToDictionary(
        kv => kv.Value,
        kv => new Addition(
                maxValues[kv.Key],
                new NumericValue(1, loc),
                loc).Optimize());

    // (2) Collect the integer clocks, e.g. [x, y]
    var ints = boundMap.Keys.ToList();

    // (3) For every distinct pair (a,b):
    // (a \neq b) \Rightarrow ((a - b) \leq bound(a))
    var conjuncts = new List<Expression>();
    foreach (var aSym in ints) {
        var aRef = new VariableReference(aSym, loc);
        foreach (var bSym in ints) {
            if (aSym == bSym) continue;

            var bRef = new VariableReference(bSym, loc);
            var neq = new BooleanNot(new Equal(aRef, bRef, loc),
                  ↪ loc);
            var diffLe = new LessThanOrEqual(
                          new Subtraction(aRef, bRef, loc),
                          boundMap[aSym], loc);

            conjuncts.Add(Implication.Create(neq, diffLe, loc));
        }
    }

    // (4) Combine all implications with \land
    if (conjuncts.Count == 0)
        return new BooleanValue(true, loc);

    var result = conjuncts[0];
    for (int i = 1; i < conjuncts.Count; i++)
        result = BooleanAnd.Combine(result, conjuncts[i]);

    return result.Optimize();
}
```

Listing 2. Building the existential condition in the digital-clock conversion

```
    public Expression CreateExistsCondition(ILocation loc,
          ↪ Dictionary<VariableSymbol, Expression> maxValues,
          ↪ Dictionary<VariableSymbol, VariableSymbol> integerVars)
      {
        // One quantified symbolic variable that will range over the
              ↪ clocks
        var quantified = new VariableSymbol("v",
              ↪ IntegerType.Instance, false, false, loc);

        var maxInt = integerVars.ToDictionary(
        kv => kv.Value,
          kv => new Addition(maxValues[kv.Key], new NumericValue(1,
                ↪ loc), loc).Optimize()
);

        // The domain is just the list of variable references from
              ↪ the dictionary keys
        var domain = maxInt.Keys
          .Select(clock => (Expression)new VariableReference(clock,
                ↪ loc))
          .ToList();

        // Build the predicate: v < ClockMaxLookup(v)
        Expression body = new LessThan(
          new VariableReference(quantified, loc),
          new ClockLookup(quantified, maxInt, loc),
```

```
20          loc
21        );
22
23        return new ExistsExpression(
24          new List<VariableSymbol> { quantified },
25          domain,
26          _ => body, // all use the same predicate
27          loc
28        ).Optimize();
29      }
```

C AI STATEMENT

During the preparation of this work, I used ChatGPT to help me understand certain snippets of the Modest Toolset code or to provide me with alternative ways of expressing certain ideas or expressions in words. Lastly, I used Writefull as a spell-checker.