Clock Reduction in Timed Automata while Preserving Design Parameters

Beyazit Yalcinkaya
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
beyazit.yalcinkaya@metu.edu.tr

Ebru Aydin Gol
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
ebrugol@metu.edu.tr

***Abstract*—Timed automata (TA) are widely used to model and verify real-time systems. In a TA, the real valued variables, called clocks, measure the time passed between events. The verification of TA is exponential in the number of clocks. That constitutes a bottleneck for its application in large systems. To address this issue, we propose a novel clock reduction method. We aim at reducing the number of clocks by developing a position (location and transition) based mapping for clocks. Motivated by that the locations and transitions of the automaton reflect the modeled systems physical properties and design parameters; the proposed method changes the clock constraints based on their positions to reduce the total number of clocks. To guarantee correctness, we prove that the resulting automaton is timed bisimilar to the original one. Finally, we present empirical results for the solution, which show that the proposed method significantly reduces the clock count without changing design parameters of the system.**

***Index Terms*—timed automata, clock reduction, bisimulation**

I. Introduction

Cyber-physical systems are everywhere from autonomous vehicles to medical devices to smart buildings and their importance is increasing as they are used in various fields. Designing reliable and safe real-time systems is among the most important goals in cyber-physical systems. Formal models are required for design and verification processes. *Timed automata* (TA) [1]–[3], a modeling formalism for real-time systems, lie at the core of the cyber-physical systems theory. From railroad crossing systems [4], [5] to cardiac pacemakers [6] to scheduling of real-time systems [7]–[9], TA are used to model and analyze complex and critical systems. Algorithms for verifying TA against formal specifications such as temporal logics exist and implemented in various tools including UPPAAL [10], a well-known and widely-used tool.

Reducing the number of clocks used in TA has been an attractive and challenging problem since its first introduction [1]. In [11], it has been shown that no algorithm can decide both minimality of the number of clocks and for the non-minimal case finding an equivalent TA with fewer clocks. Moreover, the problem of determining whether there exists an equivalent TA with fewer clocks is also shown to be undecidable [12]. An effective approach to the clock reduction problem is based on construction of a timed bisimilar TA with fewer number of clocks [13]–[15], as the bisimulation relation implies language equivalence. There are two main reasons why the clock reduction problem has been an important task. Firstly, most of the decidable problems for TA, including verification, is exponential in the number of clocks [1], [2]. Thus, to reduce computational complexity, it is essential to represent a TA with fewer number of clocks. Secondly, as these systems are used in practical areas with limited resources, e.g. embedded systems, and each clock corresponds to a physical resource in practice, it is necessary to reduce the clock count to improve efficiency.

Although, finding a TA with fewer number of clocks is crucial for computational performance of design and verification methods and runtime efficiency, in addition to the semantics of the automaton, it is also important to preserve *design parameters* of the system, i.e., locations, transitions, and constants appearing in clock constraints. Each of these concepts represents some important feature of the actual system, i.e., (i) a location may represent a particular state of the system, e.g. in the dim-light example in [16], off, dim, and bright locations represent the actual state of the light, (ii) a transition may perform some actions that is specifically important for the operation of the whole system, e.g. in a *network of timed automata* [10] synchronization channels on transitions are used for communication between different TA, and (iii) constants appearing in clock constraints of the model may reflect some properties of the modeled system, e.g. in scheduling of real-time systems, constants appearing in clock constraints are specifications of the modeled system such as period, deadline, execution time, suspension time etc. Therefore, the listed design parameters are particularly important for the modeled system. Moreover, not only are TA models used for verification, but also they are used for modeling purposes; in that case, design parameters are crucial for sustainability and understandability of models.

We argue that both reducing the number of clocks and preserving design parameters are important with their own practical concerns. Yet, a general solution for the clock reduction problem in TA while preserving the design parameters has not been introduced. In this paper, we propose an algorithm for this problem based on modifying clocks and clock constraints.

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 798482.

The main contribution of this paper is as follows: *Given any TA, we generate a TA that is timed bisimilar to the original one, has fewer (or equal) number of clocks as the original one, and preserves all of the design parameters.* Towards this goal, we make several side contributions: (i) we introduce new concepts for analyzing relations between clocks in TA, (ii) we formalize the *splitting* method for clocks, which was first addressed as a problem by Saeedloei et al. [15], (iii) we present an empirical evaluation to demonstrate algorithm's performance, and (iv) we present proofs for technical results.

The paper is outlined as follows. Sec. II gives basic definitions for TA, derived definitions for the solution, and the problem formulation. Sec. III includes the proposed methods and correctness results. Sec. IV presents results of an empirical evaluation. Sec. V compares proposed algorithm with related work. Finally, Sec. VI concludes the paper.

II. Preliminaries

A. Timed Automata

Now, we give some basic definitions of theory of timed automata [2], [17], [18]. For a set of clocks C, $\Phi(C)$ is a set of clock constraints over C. A *clock constraint* $\phi^d \in \Phi(C)$ is defined inductively in the disjunctive normal form as follows:

$$\begin{aligned} &\phi^d ::= \phi^c \mid \phi^c \vee \phi^d \\ &\phi^c ::= True \mid x < c \mid x \leq c \mid x \geq c \mid x > c \mid \phi^c \wedge \phi^c \end{aligned} \tag{1}$$

where $x \in C$, $c \in \mathbb{Q}$, and $True$ denotes the logical true.

Definition 1 (Timed Automata): A *timed automaton* is a tuple $\mathscr{A} = (L, l_0, C, I, T)$, where

- L is a finite set of locations,
- $l_0 \in L$ is an initial location,
- C is a finite set of clocks,
- I is a mapping from L to $\Phi(C)$, and
- $T \subseteq L \times L \times 2^C \times \Phi(C)$ is a set of transitions. $t = (l_s, l_t, \lambda, \phi) \in T$ is a transition from location l_s to location l_t. λ is a set of clocks that are reset to zero on t and ϕ is a clock constraint tested for enabling t.

Since our solution does not include any operation on the particular language of the automaton, for the sake simplicity, we omit an alphabet in our TA definition. Moreover, we use a trivial extension of the traditional TA definition for the clock constraints by allowing disjunctions. A transition with a disjunction can be easily converted into the traditional definition by defining transitions with the same source, target, and set of clocks to be reset for each conjunction. Similarly a location with a disjunction in its invariant can be mapped to the classical definition by creating a new location for each conjunction with identical incoming and outgoing transitions.

A *time sequence* $\tau = \tau_1\tau_2\tau_3\ldots$ is an infinite sequence of time values $\tau_i \in \mathbb{R}_{\geq 0}$, satisfying the following:

1) *Monotonicity:* $\tau_i < \tau_{i+1}$ for all $i \geq 1$.
2) *Progress:* For every $t \in \mathbb{R}$, there is some $i \geq 1$ such that $\tau_i > t$.

A *clock interpretation* ν for a set of clocks C is a mapping from C to $\mathbb{R}_{\geq 0}$, that is, it assigns a nonnegative real value to each clock in C. ν satisfies a clock constraint if and only if that constraint evaluates to true when the values given by ν are used. For a clock interpretation ν and for some $\delta \in \mathbb{R}_{\geq 0}$, $\nu + \delta$ increments clock interpretation of every clock by δ, i.e., maps $\nu(x)$ to $\nu(x)+\delta$ for each $x \in C$. For $D \subseteq C$, $\nu[D := 0]$ assigns 0 to each $y \in D$ and agrees with ν for each $x \in C \setminus D$.

Definition 2 (Timed Transition System): A *timed transition system* for a timed automaton $\mathscr{A} = (L, l_0, C, I, T)$ is a transition system $\mathcal{T}(\mathscr{A}) = (Q, q_0, \rightarrow)$, where

- $Q = \{(l, \nu)$ is the set of states, where $l \in L$, $\nu \subseteq \mathbb{R}_{\geq 0}^{|C|}$ is a clock interpretation over C, and $\nu \models I(l)\}$,
- $q_0 = (l_0, \nu_0) \in Q$ where $\nu_0(x) = 0$ for each $x \in C$, is the initial state, and
- $\rightarrow \subseteq (Q \times \mathbb{R}_{\geq 0} \times Q) \cup (Q \times Q)$ is a relation consisting of moves. There are two kinds of moves, listed as follows.
  1) Due to time elapse. For $\delta \in \mathbb{R}_{\geq 0}$, $(l, \nu) \in Q$, and $(l, \nu+\delta) \in Q$, $(l, \nu) \xrightarrow{\delta} (l, \nu+\delta)$ if for all $\delta' \in [0, \delta]$, $\nu + \delta'$ evaluates $I(l)$ to true.
  2) Due to a transition. For $t = (l, l', \lambda, \phi) \in T$, $(l, \nu) \in Q$, and $(l', \nu[\lambda := 0]) \in Q$, $(l, \nu) \rightarrow (l', \nu[\lambda := 0])$ if ν evaluates both ϕ and $I(l')$ to true.

For the sake of simplicity of the presentation of the results, we assume that for any TA, there is a transition called *initiation transition*, i.e., $t_0 = (\tilde{l}, l_0, C, True)$ where $\tilde{l} \notin L$ is just an hypothetical location that does not exist, l_0 is the initial location, C is the set of clocks, and $t_0 \in T$. This transition guarantees that, initially, all clocks are zero.

A *run* ρ of a timed automaton $\mathscr{A} = (L, l_0, C, I, T)$ is an infinite sequence defined over the associated timed transition system $\mathcal{T}(\mathscr{A}) = (Q, q_0, \rightarrow)$ as follows:

$$\rho : q_0 \xrightarrow{\tau_1} q_0 \rightarrow q_1 \xrightarrow{\tau_2 - \tau_1} \ldots \xrightarrow{\tau_i - \tau_{i-1}} q_{i-1} \rightarrow q_i \xrightarrow{\tau_{i+1} - \tau_i} \ldots \tag{2}$$

where $\tau = \tau_1\tau_2\ldots$ is a time sequence and $q_i \in Q$ for each i.

Definition 3 (Timed Bisimulation): Let $\mathcal{T}(\mathscr{A}) = (Q, q_0, \rightarrow)$ and $\mathcal{T}(\mathscr{A}') = (Q', q_0', \rightarrow')$ be two timed transition systems defined over timed automata $\mathscr{A}$ and $\mathscr{A}'$. A *timed bisimulation* is an equivalence relation $\approx_{\mathcal{T}}$ defined over $Q \times Q'$ such that whenever $q_1 \approx_{\mathcal{T}} q_1'$,

- ***P1*** if $q_1 \xrightarrow{\delta} q_2$ for $\delta \in \mathbb{R}_{\geq 0}$ (or $q_1 \rightarrow q_2$), then there exists q_2' with $q_1' \xrightarrow{\delta}{}' q_2'$ (or $q_1' \rightarrow' q_2'$) and $q_2 \approx_{\mathcal{T}} q_2'$, and
- ***P2*** if $q_1' \xrightarrow{\delta}{}' q_2'$ for $\delta \in \mathbb{R}_{\geq 0}$ (or $q_1' \rightarrow' q_2'$) , then there exists q_2 with $q_1 \xrightarrow{\delta} q_2$ (or $q_1 \rightarrow q_2$) and $q_2 \approx_{\mathcal{T}} q_2'$.

$\mathcal{T}(\mathscr{A})$ and $\mathcal{T}(\mathscr{A}')$ are timed bisimilar if there is a bisimulation relation $\approx_{\mathcal{T}} \subseteq Q \times Q'$ and $(q_0, q_0') \in \approx_{\mathcal{T}}$. Two timed automata $\mathscr{A}$ and $\mathscr{A}'$ are *timed bisimilar* if the corresponding timed transition systems $\mathcal{T}(\mathscr{A})$ and $\mathcal{T}(\mathscr{A}')$ are timed bisimilar. It is denoted by $\mathscr{A} \approx \mathscr{A}'$.

In the rest of the paper, it is assumed that in a TA all locations are reachable. Notice that it does not affect the generality of the proposed method, since it has been shown that the reachability problem for TA is decidable [16], [17] and an algorithm has been given for the reachability analysis [16]. Thus, for any given timed automaton, the proposed method can be applied after reachability analysis.

B. Derived Definitions

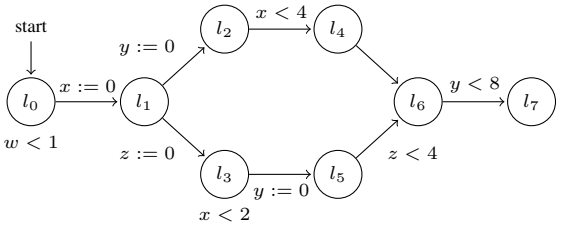
In the following, a set of definitions over a timed automaton $\mathscr{A} = (L, l_0, C, I, T)$ and $\Delta = \{\leq, \geq, <, >\}$ is given. These are defined to simplify the notation in the proposed methods.

- $source : T \rightarrow L$. It gives the source of the given transition. For $t = (l_s, l_t, \lambda, \phi) \in T$, $source(t) = l_s$.
- $target : T \rightarrow L$. It gives the target of the given transition. For $t = (l_s, l_t, \lambda, \phi) \in T$, $target(t) = l_t$.
- $clocks : \Phi(C) \rightarrow 2^C$. It gives the set of all clocks tested in a given constraint. For $\phi \in \Phi(C)$,

$$clocks(\phi) = \begin{cases} \{\} \text{ if } \phi = True \\ \{x\} \text{ if } \phi = x \sim c \\ clocks(\phi_1) \cup clocks(\phi_2) \\ \quad \text{if } \phi = \phi_1 \vee \phi_2 \text{ or } \phi = \phi_1 \wedge \phi_2 \end{cases}$$

- $resets : C \rightarrow 2^L$. It gives the set of locations in which a given clock is reset in an incoming transition. For $x \in C$, $resets(x) = \{l_t \mid (l_s, l_t, \lambda, \phi) \in T \text{ and } x \in \lambda\}$.
- $controls : C \rightarrow 2^L$. It gives the set of all locations on which given clock is tested as an invariant or it is tested on an outgoing transition. $controls(x) = \{l \mid x \in clocks(\phi) \text{ for } (l_s, l_t, \lambda, \phi) \in out(l) \text{ or } x \in clocks(I(l))\}$, where $out(l) = \{(l_s, l_t, \lambda, \phi) \in T \mid l_s = l\}$ is the set of outgoing transitions from l.

Fig. 1. A timed automaton $\mathscr{A} = (L, l_0, C, I, T)$ where $L = \{l_0, l_1, l_2, l_3, l_4, l_5, l_6, l_7\}$, l_0 is the initial location, $C = \{x, y, z, w\}$, $I(l_0) = w < 1$, $I(l_3) = x < 2$, $I(l_i) = True$ for $l_i \in L \setminus \{l_0, l_3\}$, and $T = \{t_0 = (\tilde{l}, l_0, C, True), t_1 = (l_0, l_1, \lambda_1, \phi_1), t_2 = (l_1, l_2, \lambda_2, \phi_2), t_3 = (l_1, l_3, \lambda_3, \phi_3), t_4 = (l_2, l_4, \lambda_4, \phi_4), t_5 = (l_3, l_5, \lambda_5, \phi_5), t_6 = (l_4, l_6, \lambda_6, \phi_6), t_7 = (l_5, l_6, \lambda_7, \phi_7)\}, t_8 = (l_6, l_7, \lambda_8, \phi_8)\}$.



Example 1: A timed automaton is given in Fig. 1, which will be used as a running example throughout the paper. Sample derived definitions for this automaton are:

- $source(t_1) = l_0$, $target(t_1) = l_1$, $clocks(\phi_4) = \{x\}$
- $resets(x) = \{l_0, l_1\}$, and $controls(x) = \{l_2, l_3\}$.

C. Problem Formulation

The verification problem for a TA has an exponential complexity on the number of clocks [1], [2]. Therefore, it is crucial to use minimal number of clocks for computational efficiency. However, for the designer, locations, transitions, constraint constants, and their place of appearance in the automaton may represent some important features of the system that needs to be shown in the model. Consequently, it is important to preserve these distinguishing features while reducing the number of clocks. In the problem definition given below, constraints over the construction of a timed bisimilar TA $\mathscr{A}_m$ emphasize preservation of these crucial design parameters of the original TA while reducing the number of clocks.

Problem 1: Given a TA $\mathscr{A} = (L, l_0, C, I, T)$, construct a new TA $\mathscr{A}_m = (L, l_0, C_m, I_m, T_m)$ such that (i) the underlying graph of both are the same, (ii) set of constants appearing on the constraints of invariants and transitions are the same for both TA, (iii) $|C_m| \leq |C|$, and (iv) $\mathscr{A}_m \approx \mathscr{A}$.

We propose to solve Prob. 1 by analyzing reset and control positions (transitions and locations) of the clocks. The proposed solution, first, eliminates unnecessary resets and then reduces each clock to its *minimal form*. In particular, first, a new automaton $\mathscr{A}_s$ that is timed bisimilar to $\mathscr{A}$ is generated such that each clock is reset and controlled in at most one position in $\mathscr{A}_s$. Then, the relations between the clocks of $\mathscr{A}_s$ are analyzed and mapped to a graph coloring problem. The solution of the coloring problem shows the clocks that can be represented by the same clock, i.e. clocks painted with the same color can be represented with a single clock, which is used to define $\mathscr{A}_m$ that is timed bisimilar to $\mathscr{A}_s$ (hence to $\mathscr{A}$).

III. Timed Automata Clock Reduction

In this section, we present our solution for Prob. 1. First, we present the proposed clock analysis method that identifies the clocks that can be merged to reduce the total number of clocks used in the TA. We prove that the TA obtained after the clock merge operation is timed bisimilar to the original one. Then, we introduce the *splitting* method over the clocks of TA and we prove that splitting clocks of a TA does not change the semantics, i.e., the obtained TA after splitting is timed bisimilar to the original one. Finally, we present the main algorithm that combines these methods, and prove that the resulting TA is a solution for Prob. 1.

A. Clock Analysis and Dependency Computation

In this section, we present the proposed dependency analysis methods for the clocks of a TA $\mathscr{A} = (L, l_0, C, I, T)$. The aim is to find the clocks that can be merged without changing the semantics. The clock merge operation, simply, takes two clocks $x, y \in C$, replaces each occurrence of y with x, and generates a TA with the set of clocks $C \setminus \{y\}$.

Definition 4 (Merge): The automaton obtained from $\mathscr{A} = (L, l_0, C, I, T)$ by merging clocks $x, y \in C$ is defined as $\mathscr{A}^{x \leftarrow y} = (L, l_0, C \setminus \{y\}, I', T')$, where I' and T' are obtained by replacing each occurrence of y with x from I and T, respectively, by the mappings $\mu_\Lambda : 2^C \times C \times C \rightarrow 2^C$ and $\mu_\Phi : \Phi(C) \times C \times C \rightarrow \Phi(C)$ which are defined as follows for $\bar{\lambda} \in 2^C$, $\bar{\phi} \in \Phi(C)$, and $x, y \in C$:

$$\mu_\Lambda(\bar{\lambda}, x, y) = \begin{cases} \bar{\lambda} \text{ if } y \notin \bar{\lambda} \\ \bar{\lambda} \setminus \{y\} \cup \{x\} \text{ if } y \in \bar{\lambda} \end{cases} \tag{3}$$

$$\mu_\Phi(\bar{\phi}, x, y) = \begin{cases} \bar{\phi} \text{ if } y \notin clocks(\bar{\phi}) \\ (x \sim c \wedge \mu_\Phi(\phi^c, x, y)) \vee \mu_\Phi(\phi^d, x, y) \\ \qquad \text{if } \bar{\phi} = (y \sim c \wedge \phi^c) \vee \phi^d \end{cases} \tag{4}$$

Using μ_Λ and μ_Φ, I' and T' are found as follows:

- For each $l \in L$, $I'(l) = \mu_\Phi(I(l), x, y)$.
- $T' = \bigcup_{(l_s,l_t,\bar{\lambda},\bar{\phi})\in T}\{(l_s, l_t, \mu_\Lambda(\bar{\lambda}, x, y), \mu_\Phi(\bar{\phi}, x, y))\}$.

In the remainder of this section, we present methods to find $x, y \in C$ such that $\mathscr{A} \approx \mathscr{A}^{x\leftarrow y}$.

Definition 5 (Path): For a TA $\mathscr{A} = (L, l_0, C, I, T)$, a path is defined as a finite sequence of locations $p = l_1 l_2 \dots l_n$ such that for each $i < n$, there exists a $t \in T$ such that $source(t) = l_i$ and $target(t) = l_{i+1}$. The set of all paths between two locations $l_s, l_e \in L$ is defined as

$$\mathcal{P}(l_s, l_e) = \{p = l_s \dots l_e \mid p \text{ is a path from } l_s \text{ to } l_e\} \quad (5)$$

In the analysis, we first eliminate resets that do not affect the execution of the TA. A reset of a clock x on a transition $(l_s, l_r, \lambda, \phi)$ such that $x \in \lambda$ is unnecessary if (6) or (7) holds.

$$\bigcup_{l_c \in controls(x)} \mathcal{P}(l_r, l_c) = \emptyset \quad (6)$$

$$x \in \bigcap_{\substack{l_c \in controls(x),\\ \mathcal{P}(l_r,l_c)\neq\emptyset}} \left(\bigcap_{l_1 \dots l_n \in \mathcal{P}(l_r,l_c)} \left(\bigcup_{\substack{i=1,\dots,n-1,\\ (l_i,l_{i+1},\lambda_i,\phi_i)\in T}} \lambda_i \right) \right) \quad (7)$$

Eqn. (6) indicates that no control location for clock x is reachable from l_r; hence, the value of x will not be controlled after l_r. Eqn. (7) indicates that clock x is reset on every path from l_r to a control location of x. In this case, after l_r, clock x will reset again before it's value is controlled. Consequently, in both cases, the reset of x on $(l_s, l_r, \lambda, \phi)$ is unnecessary. For each $x \in C$ and reset transition $(l_s, l_r, \lambda, \phi)$, if (6) or (7) holds, we remove the reset by defining a new transition set:

$$T' := (T \setminus \{(l_s, l_r, \lambda, \phi)\}) \cup \{(l_s, l_r, \lambda \setminus \{x\}, \phi)\} \quad (8)$$

Proposition 1: For TA $\mathscr{A} = (L, l_0, C, I, T)$, let clock $x \in C$ and transition $(l_s, l_r, \lambda, \phi) \in T$ satisfy (6) or (7). Then, automaton $\mathscr{A}' = (L, l_0, C, I, T')$, where T' is as defined in (8), is timed bisimilar to $\mathscr{A}$.

Proof: The proof follows from that the value of clock x is not read before it is reset again from location l_r. Hence, removing x from λ does not effect the execution of $\mathscr{A}$. ∎

Definition 6 (Scope): Scope of a clock $x \in C$ is the set of all paths p between locations l_s and l_e such that (i) $l_s \in resets(x)$, (ii) $l_e \in controls(x)$, and (iii) x is not reset on p:

$$\mathcal{S}(x) = \{l_1 \dots l_n \in \overline{\mathcal{P}} \mid l_i \notin resets(x) \text{ for } i = 2, \dots, n\} \quad (9)$$

$$\text{where } \overline{\mathcal{P}} = \{p \in \mathcal{P}(l, l') \mid l \in resets(x), l' \in controls(x)\}$$

The scope of a clock refers to the set of paths of a TA in which the value of the clock is needed and it should not be intervened. Consequently, the scope can be used to decide whether one clock can be written in the place of another, i.e., merged into another one. Note that unnecessary resets (satisfying (6) or (7)) do not affect the scope computation. However, it is required to remove them for the merge operation. The scopes and resets of the clocks induce an equivalence relation, called dependency relation $\mathcal{D} \subseteq C \times C$:

Definition 7 (Dependency): Clocks $x, y \in C$ are dependent $(x, y) \in \mathcal{D}$ if and only if x (or y) is reset in the scope of y (or x), i.e. (10) or (11) holds:

$$resets(x) \bigcap \left(\bigcup_{l_1 l_2 \dots l_n \in \mathcal{S}(y)} \{l_2, \dots, l_n\} \right) \neq \emptyset \quad (10)$$

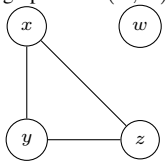
$$resets(y) \bigcap \left(\bigcup_{l_1 l_2 \dots l_n \in \mathcal{S}(x)} \{l_2, \dots, l_n\} \right) \neq \emptyset \quad (11)$$

If a clock is reset in the scope of another, then we say that these two clocks are dependent. Notice that the first location of the path is not checked in (10) (or in (11)), since clock y (or x) is reset on this location, and synchronous resets do not introduce dependency. Dependent clocks cannot be merged since in a path of the TA in which the value of one is needed, the other one is reset. Therefore, merging them would change the semantics of the automaton. To find the optimum set of clocks that can be merged, we analyze the graph induced by the dependency relation.

Definition 8 (Dependency Graph): The dependency graph is an undirected graph represented as $G = (C, \mathcal{D})$. Its vertices consists of clocks of the TA and there is an edge between two nodes if and only if corresponding clocks are dependent.

Dependency graph of a given TA gives us a visual and more clear understanding of the dependency relations between clocks of the TA and it enables us to use some well-defined methods from the graph theory to reduce the number of clocks.

Fig. 2. Dependency graph $G = (C, \mathcal{D})$ of the TA in Fig. 1.



Example 2: For the TA $\mathscr{A} = (L, l_0, C, I, T)$ given in Fig. 1:

- $\mathcal{P}(l_0, l_7) = \{l_0 l_1 l_2 l_4 l_6 l_7, l_0 l_1 l_3 l_5 l_6 l_7\}$
- $\mathcal{S}(y) = \{l_2 l_4 l_6, l_5 l_6\}$
- $\mathcal{D} = \{(x, y), (x, z), (y, z)\}$
- $G = (C, \mathcal{D})$ is given in Fig. 2.

The TA obtained by merging independent clocks is timed bisimilar to the original one:

Theorem 1: Let $x, y \in C$, $(x, y) \notin \mathcal{D}$ and $\mathscr{A}^{x\leftarrow y} = (L, l_0, C \setminus \{y\}, I', T')$ be the automaton obtained from $\mathscr{A} = (L, l_0, C, I, T)$ by merging x and y according to Def. 4., then $\mathscr{A}^{x\leftarrow y} \approx \mathscr{A}$.

Proof: Here, we present the main idea of the proof and a sketch of it, and refer the interested reader to Appendix for a complete version of the proof.

Let $\mathcal{T}(\mathscr{A}) = (Q, q_0, \rightarrow)$ and $\mathcal{T}(\mathscr{A}^{x\leftarrow y}) = (Q', q_0', \rightarrow')$ be the two timed transition systems defined over TA $\mathscr{A}$ and $\mathscr{A}^{x\leftarrow y}$ as in Def. 2, respectively. We claim that $R \subseteq Q \times Q'$ is a bisimulation relation:

$$R = \{((l, \nu), (l, \nu')) \mid \nu(z) = \nu'(z) \forall z \in C \setminus \{x, y\} \text{ and } \nu'(x) = M(l, (\nu(x), \nu(y)))\}, \quad (12)$$

where $M : L \times (a,b) \to \{a,b\}$. The mapping is computed according to $scope(y)$ on $\mathscr{A}$. First, define the set of locations $L_z \subseteq L$ that appear on the scope of a clock $z \in C$ as:

$$L_z = \bigcup_{p=l_1 \ldots l_n \in \mathcal{S}(z)} \{l_1, \ldots, l_n\} \tag{13}$$

Then the mapping is computed according to L_y (13) as follows:

$$M(l,(\nu(x),\nu(y))) = \begin{cases} \nu(y) \text{ if } l \in L_y \\ \nu(x) \text{ otherwise} \end{cases} \tag{14}$$

The mapping, M, defines the value of clock x on $\mathcal{T}(\mathscr{A}^{x \leftarrow y})$ as $\nu(x)$ or $\nu(y)$ based on location l for relation R. Note that if l appears on the scope of both clock x and clock y, i.e. $l \in L_x \cap L_y$, then they reset on the same transition before reaching l. As they have the same value, the one picked by M is irrelevant in terms of clock constraints. If they do not have the same value, and a location l appears on both of the scopes, then one of them is reset in the scope of another one, which implies that they are dependent (contradicts with the assumption). Therefore, for any $((l,\nu),(l,\nu')) \in R$ the following holds, which summarizes the preceding discussion.

$$\nu'(x) = \begin{cases} \nu(x) \text{ and } \nu(x) = \nu(y) \text{ if } l \in L_x \cap L_y \\ \nu(x) \text{ if } l \notin L_y \\ \nu(y) \text{ if } l \in L_y \setminus L_x \end{cases} \tag{15}$$

For independent clocks x and y, $\nu(x) = \nu(y)$ for each $l \in L_x \cap L_y$ and $(l,\nu) \in Q$. In other words, either $L_x \cap L_y$ is empty, or for each location that lies in the intersection, the clocks always have the same value. Therefore, it is *safe* to use one clock in the place of another, which is achieved via mapping M (14). Essentially, for any $\phi \in \Phi(C)$, and $(l,\nu) \in Q$ that is reachable from the initial state q_0, if $((l,\nu),(l,\nu')) \in R$, then

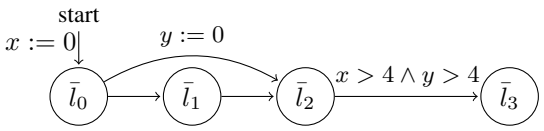
$$\textbf{MP: } \phi \text{ holds for } \nu \text{ if and only if } \mu_\Phi(\phi,x,y) \text{ holds for } \nu', \tag{16}$$

where μ_Φ is as defined in (4). In Appendix, we prove that the bisimulation property (***P1*** and ***P2*** in Def. 3) holds for R for both time elapse and location change transitions by using property **MP** (16). Therefore, R is a timed bisimulation relation. Observing that the initial states of $\mathcal{T}(\mathscr{A})$ and $\mathcal{T}(\mathscr{A}^{x \leftarrow y})$ are related in R, i.e. $(q_0, q_0') \in R$, $q_0 = (l_0, \nu_0), \nu_0(z) = 0, \forall z \in C, q_0' = (l_0, \nu_0'), \nu_0(z) = 0, \forall z \in C \setminus \{y\}$ and $((l_0,\nu_0),(l_0,\nu_0')) \in R$, completes the proof of $\mathscr{A}^{x \leftarrow y} \approx \mathscr{A}$. ∎

Example 3: For the TA given in the Fig. 1, the dependency graph is given in the Fig. 2. Merging two independent clocks x and w does not change the semantics of the TA.

In some exceptional cases, merging two dependent clocks does not change the semantics of the TA as in Ex. 4. Thm 1 states that it is safe to merge independent clocks. However, it cannot be concluded that merging only independent clocks gives the minimal number of clocks. Yet for the practical concerns analyzing only independent clocks for merging is sufficient (see Sec. IV). Note that it is required to analyze the satisfiability of the constraints to merge the clocks in Ex. 4.

Fig. 3. The TA $\bar{\mathscr{A}} = (\bar{L}, \bar{l}_0, \bar{C}_s, \bar{I}_s, \bar{T}_s)$.



Example 4: For the TA $\bar{\mathscr{A}}$ given in the Fig. 3, merging clocks x and y does not change the semantics of the TA even though these two clocks are dependent. Notice that, the constraint on the merged automaton is $x > 4 \wedge x > 4 \equiv x > 4$.

Corollary 1: Let $\mathcal{X} \subseteq C$ such that $(x,y) \notin \mathcal{D}$ for any $x, y \in \mathcal{X}$ with $x \neq y$, and let $\mathscr{A}^{x \leftarrow \mathcal{X}}$ be the automaton obtained by iteratively merging all clocks in $\mathcal{X}$. Then $\mathscr{A}^{x \leftarrow \mathcal{X}} \approx \mathscr{A}$.

Proof: Consider $x, y \in \mathcal{X}$. Since $(x,y) \notin \mathcal{D}$ and unnecessary resets are removed via (6) and (7), $controls(y)$ is not reachable from $resets(x)$ without passing through $resets(y)$ (and vice versa). As the scope of a clock is the set of all locations along the paths from the targets of the reset transitions to control locations without passing through other resets, the scope of x on $\mathscr{A}^{x \leftarrow y}$, denoted as $\mathcal{S}^{x \leftarrow y}(x)$ equals to the union of scopes x and y on $\mathscr{A}$:

$$\mathcal{S}^{x \leftarrow y}(x) = \mathcal{S}(x) \cup \mathcal{S}(y). \tag{17}$$

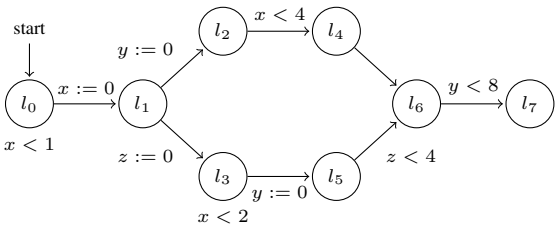
By (3), the resets of x on $\mathscr{A}^{x \leftarrow y}$, denoted as $resets^{x \leftarrow y}(x)$, equals to the union of resets of x and y on $\mathscr{A}$:

$$resets^{x \leftarrow y}(x) = resets(x) \cup resets(y). \tag{18}$$

Now consider $z \in \mathcal{X} \setminus \{x,y\}$. By (17) and (18), and the scope definition (9), clocks x and z are independent on $\mathscr{A}^{x \leftarrow y}$. Therefore, they can be merged. Iterative application of this argument completes the proof. ∎

In order to find the automaton with fewer clocks by applying Cor. 1 for the independent clocks, we need to find a set of set of clocks $\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ that is a partition of C (i.e., $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$, $C = \cup_{i=1} \mathcal{C}_i$) with minimal cardinality such that every pair of clocks $x, y \in \mathcal{C}_i$ are independent, $(x,y) \notin \mathcal{D}$, on $\mathscr{A}$ for each set $\mathcal{C}_i$. Then, we can merge each set to obtain $\mathscr{A}_m$ with $n \leq |C|$ clocks that is timed bisimilar to $\mathscr{A}$. Finding such a partition with minimal cardinality maps to the well-known graph coloring problem on the graph defined by the dependency relation [19]. In graph coloring problem, the goal is to assign a color to each vertex such that no neighboring vertices are assigned the same color. The smallest number of colors needed to color a graph is called its chromatic number. When the coloring problem is applied to the dependency graph, the clocks with the same color can be merged as in Cor. 1 to reduce the number of clocks.

Fig. 4. The TA obtained after merging independent clocks of $\mathscr{A}$ from Fig. 1.



Example 5: A partition of the set of clocks obtained by applying graph coloring algorithm to the dependency graph in Fig. 2 is $\{\{x, w\}, \{y\}, \{z\}\}$. The TA obtained after merging clocks in each set from this partition is given in Fig. 4.

Notice that, number of clocks of the TA in Fig. 4 can be further improved. In particular, the clock count can be reduced to two while satisfying the constraints in Prob. 1. In the original TA from Fig. 1, there are clocks reset or controlled in more than one position which may cause some pseudo-dependencies between clocks, i.e. it generates some dependency relations which can be avoided by assigning unique clocks for each reset-control pair. We formalize this intuition in the next section.

B. Splitting

The dependency analysis might fail to find the minimum number of clocks if a clock is reset or controlled in multiple places as shown in Ex. 5. In any TA, the clocks of the automaton are reset to zero in some transitions and they are controlled with some constraints on transitions and/or in locations. Thus, each clock can be observed as a set of resets and a set of controls. By this observation, in a TA, the *minimal form* of a clock is one reset and one corresponding control. We reduce each clock to its minimal form by *splitting*, i.e., we assign a new clock to each reset-control pair. We prove that the resulting TA is timed bisimilar to the original one.

Definition 9 (Clock split): Given a TA $\mathscr{A} = (L, l_0, C, I, T)$ and a clock $x \in C$, a TA $\mathscr{A}_{s,x} = (L, l_0, C_s, I_s, T_s)$ in minimal form for clock x is defined via clock $\pi_C : C \to 2^{C_s}$, reset $\pi_\Lambda : C \times T \to 2^{C_s}$, transition $\pi_T : C \times T \to T_s$ and constraint $\pi_\Phi : C \times \Phi(C) \to \Phi(C_s)$ transformations.

- $C_s = \pi_C(x) \cup C \setminus \{x\}$, where $\pi_C(x) = \{x_{i,j} \mid 1 \leq i \leq |resets(x)|, 1 \leq j \leq |controls(x)|\}$ is the set of clocks obtained by splitting x (i.e. a new clock for each reset-control pair).
- $I_s(l) = \pi_\Phi(x, I(l))$ where π_Φ generates a clock constraint over $\Phi(C_s)$ by iteratively mapping the constraints on x to $\pi_C(x)$ as follows:

$$\pi_\Phi(x, x \sim c \wedge \phi^c \vee \phi^d) = \begin{cases} ((x_{1,j} \sim c \vee \ldots \vee x_{|resets(x)|,j} \sim c) \wedge \phi^c) \vee \phi^d \\ \qquad \text{if } \sim \in \{<, \leq\} \\ (x_{1,j} \sim c \wedge \ldots \wedge x_{|resets(x)|,j} \sim c \wedge \phi^c) \vee \phi^d \\ \qquad \text{if } \sim \in \{>, \geq\} \end{cases} \tag{19}$$

  where $x \sim c$ is the j^{th} control of x. The transformation (19) is applied iteratively for each $x \sim c$ in ϕ until all constraints on x are mapped to constraints on $\pi_C(x)$.
- $\pi_\Lambda : C \times T \to 2^{C_s}$ transforms resets on x:

$$\pi_\Lambda(x,(l_s, l_t, \lambda, \phi)) = \lambda \setminus \{x\} \cup \begin{cases} \emptyset \text{ if } x \notin \lambda \\ \{x_{i,1}, \ldots, x_{i,|controls(x)|}\} \text{ if } x \in \lambda \end{cases} \tag{20}$$

  where i is the index of l_t in $resets(x)$.
- $T_s = \bigcup_{t \in T} \pi_T(x, t)$, where

$$\pi_T(x, (l_s, l_t, \lambda, \phi)) = (l_s, l_t, \pi_\Lambda(x, (l_s, l_t, \lambda, \phi)), \pi_\Phi(x, \phi)) \tag{21}$$

Thus, clock $x \in C$ is reduced to its minimal form by defining a new set of clocks C_s, a new set of transitions T_s, and a new mapping for invariants I_s, that is each clock $x_{i,j}$ in C_s has only one reset and only one constraint defined over it. Furthermore, the resulting TA in minimal form $\mathscr{A}_{s,x}$ is timed bisimilar to $\mathscr{A}$:

Theorem 2: The TA $\mathscr{A}_{s,x} = (L, l_0, C_s, I_s, T_s)$ obtained by splitting x on $\mathscr{A} = (L, l_0, C, I, T)$ as given in Def. 9 is timed bisimilar to $\mathscr{A}$, i.e., $\mathscr{A}_s \approx \mathscr{A}$.

Proof: Let $\mathcal{T}(\mathscr{A}) = (Q, q_0, \to)$ and $\mathcal{T}(\mathscr{A}_{s,x}) = (Q', q_0', \to')$ be the two timed transition systems defined over TA $\mathscr{A}$ and $\mathscr{A}_s$ as in Def. 2, respectively. Define mapping $M^C : \mathbb{R}_{\geq 0}^{|C_s|} \to \mathbb{R}_{\geq 0}^{|C|}$:

$$\nu(y) = \begin{cases} \min\{\nu'(x_{i,j}) \mid x_{i,j} \in \pi_C(x)\} \text{ if } y = x \text{ (for clock } x) \\ \nu'(y) \text{ otherwise} \end{cases} \tag{22}$$

We claim that R is a timed bisimulation relation:

$$R = \{(q = (l, M^C(\nu')), q' = (l, \nu')) \mid (l, \nu') \in Q' \text{ and } q' \text{ is reachable from } q_0'\} \tag{23}$$

First, observe that the splitting operation guarantees that $\nu'(x_{i,a}) = \nu'(x_{i,b})$ for any $(l, \nu') \in Q'$ since $x_{i,a}$ and $x_{i,b}$ are reset on the same transition. Consider a state $q' = (l, \nu') \in Q'$ that is reachable from q_0', let $\rho' : q_0' \xrightarrow{\tau_1} q_0' \to q_1' \xrightarrow{\tau_2 - \tau_1} \ldots \xrightarrow{\tau_n - \tau_{n-1}} q_{n-1}' \to q_n' = q'$ be a run from q_0' to q'. Notice that $\nu'(x_{1,j}), \ldots, \nu'(x_{|resets(x)|,j})$ measures the time passed since each reset of x on $\mathscr{A}$ and the minimum of them represents the time passed since the last reset along the run. Therefore, mapping clock valuations via M^C defines a run $\rho : q_0 \xrightarrow{\tau_1} q_0 \to q_1 \xrightarrow{\tau_2 - \tau_1} \ldots \xrightarrow{\tau_n - \tau_{n-1}} q_{n-1} \to q_n = q$ of $\mathscr{A}$ from q_0 to q, where $q_i = (l_i, M^C(\nu_i'))$ and $q_i' = (l_i, \nu_i')$ for each $i = 1, \ldots, n$. In the following, we formalize this intuition and prove the bisimulation property for R. Let $l \in controls(x)$ be the j^{th} control for x and let $x \sim c$ be corresponding constraint. Consider $((l, \nu), (l, \nu')) \in R$ and transformed constraints:

$$\sim \in \{<, \leq\} : \pi_\Phi(x \sim c) = x_{1,j} \sim c \vee \ldots \vee x_{|resets(x)|,j} \sim c \tag{24}$$

$$\sim \in \{>, \geq\} : \pi_\Phi(x \sim c) = x_{1,j} \sim c \wedge \ldots \wedge x_{|resets(x)|,j} \sim c \tag{25}$$

Constraint $x \sim c$ holds for ν if and only if $\pi_\Phi(x \sim c)$ (24) (or (25)) holds for ν' since $\nu(x) = \min\{\nu(x_{i,j}) \mid i = 1, \ldots, |resets(x)|\}$.

CP: Given the argument above, considering that the constraints $C \setminus \{x\}$ are the same on both $\mathscr{A}$ and $\mathscr{A}_{s,x}$ and mapping M^C (22) preserves values of clocks $C \setminus \{x\}$, we conclude that a clock constraint ϕ holds for ν if and only if $\pi_\Phi(\phi)$ holds for ν' for any $((l, \nu), (l, \nu')) \in R$.

Consider $((l, \nu), (l, \nu')) \in R$:

Location change, Proof of *P1*: Assume that $(l, \nu) \rightarrow (l_t, \nu_t)$. Then there is a transition $(l, l_t, \lambda, \phi) \in T$ such that ϕ holds for ν, and $\nu_t = \nu[\lambda := 0]$. Then, $\pi_\Phi(\phi)$ holds for ν' via **CP**. The construction of $\mathscr{A}_{s,x}$ guarantees that $(l, l_t, \phi_\Lambda((l, l_t, \lambda, \phi)), \pi_\Phi(\phi)) \in T_s$. Therefore, $(l, \nu') \rightarrow' (l_t, \nu'_t)$ where $\nu'_t = \nu'[\phi_\Lambda((l, l_t, \lambda, \phi) := 0]$. By (20), it holds that $\lambda \setminus \{x\} = \phi_\Lambda((l, l_t, \lambda, \phi)) \setminus \pi_C(x)$. Therefore, for any $y \in C \setminus \{x\}$, $\nu'_t(y) = \nu_t(y)$. For clock x, consider two cases.

i. $x \notin \lambda$. Then $\nu_t(x) = \nu(x)$, and $\phi_\Lambda((l, l_t, \lambda, \phi)) \cap \pi_C(x) = \emptyset$ (see (20)). Consequently, $\nu'(x_{i,j}) = \nu'_t(x_{i,j}), \forall x_{i,j} \in \pi_C(x)$. As $\nu_t = M^C(\nu'_t)$, $((l_t, \nu_t), (l_t, \nu'_t)) \in R$.

ii. $x \in \lambda$. Then, $x_{i,j} \in \phi_\Lambda((l_s, l_t, \lambda, \phi))$ for some $x_{i,j}$. In other words, a clock obtained via split of x is reset on the transition. Then, by (22), $\nu_t(x) = \nu'_t(x_{i,j}) = 0$. Consequently, $((l_t, \nu_t), (l_t, \nu'_t)) \in R$.

Location change, Proof of *P2*: Assume that $(l, \nu') \rightarrow (l_t, \nu'_t)$. Then there is a transition $(l, l_t, \phi_\Lambda((l, l_t, \lambda, \phi)), \pi_\Phi(\phi)) \in T_s$ such that $\pi_\Phi(\phi)$ holds for ν', and $\nu'_t = \nu'[\phi_\Lambda((l, l_t, \lambda, \phi) := 0]$. Then, ϕ holds for ν via **CP**. Therefore, $(l, \nu) \rightarrow (l_t, \nu_t)$ where $\nu_t = \nu[\lambda := 0]$. Similar to the proof of ***P1***, for any $y \in C \setminus \{x\}$, $\nu'_t(y) = \nu_t(y)$ by (20). Furthermore, $\phi_\Lambda((l_s, l_t, \lambda, \phi)) \cap \pi_C(x) \neq \emptyset$ if and only if $x \in \lambda$. Therefore, it follows from **i.** and **ii.** that $((l_t, \nu_t), (l_t, \nu'_t)) \in R$, which completes the proof for ***P2***.

Proofs for time elapse transitions for ***P1*** and ***P2*** directly follow from the application of **CP** on invariant constraints and constraint mapping (19). Observing that $(q_0, q'_0) \in R$, completes the proof. ∎

In some cases, naively applying the splitting operation may result into redundant reset-constraint pairs, i.e. clocks with constraints that are not reachable from their related resets. We avoid such cases in our solution by checking the reachability of each constraint from its related reset before assigning a unique clock to it. Essentially, in splitting operation, instead of $\pi_C(x)$, we use $\pi'_C(x) = \{x_{i,j} \in \pi_C(x) \mid \textbf{RC} \text{ holds for } x_{i,j}\}$ where **RC** is the reachability condition, i.e. **RC:** There is a path from $l_{i,r}$ to $l_{c,j}$ on which x is not reset, where $l_{i,r}$ is i^{th} reset location and $l_{c,j}$ is the j^{th} control location of the clock x. The reachability condition guarantees that using $\pi'_C(x)$ instead of $\pi_C(x)$ does not change the semantics of TA.

Splitting a clock may produce two dependent clocks which may result in increased number of clocks after the merge operation. We present such a problematic case in Ex 6.

Fig. 5. The TA $\bar{\mathscr{A}} = (\bar{L}, \bar{l}_0, \bar{C}_s, \bar{I}_s, \bar{T}_s)$.

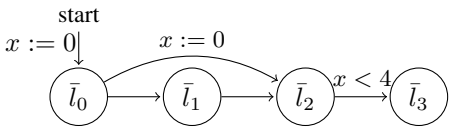
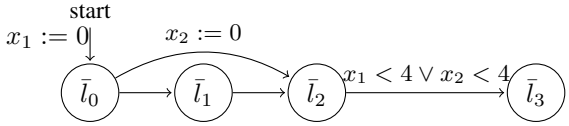


Fig. 6. The TA $\bar{\mathscr{A}}' = (\bar{L}', \bar{l}'_0, \bar{C}'_s, \bar{I}'_s, \bar{T}'_s)$.



Example 6: The TA $\bar{\mathscr{A}} = (\bar{L}, \bar{l}_0, \bar{C}_s, \bar{I}_s, \bar{T}_s)$ given in Fig. 5 has only one clock x. After applying splitting operation as introduced above, we obtain an new TA $\bar{\mathscr{A}}' = (\bar{L}', \bar{l}'_0, \bar{C}'_s, \bar{I}'_s, \bar{T}'_s)$ given in Fig. 6 with two dependent clocks x_1 and x_2.

As Ex.6 implies, splitting clocks of a TA may increase the dependent number of clocks; hence, increase total number of clocks at the end of the analysis. In our solution we avoid splitting such clocks and we apply splitting operation if all clocks in the set of clocks obtained after splitting are independent. Hence, we safely conclude that our solution does not increase the number of clocks in the model.

Fig. 7. The TA $\mathscr{A}_s = (L, l_0, C_s, I_s, T_s)$, obtained after splitting the clocks of the TA in Fig. 1.

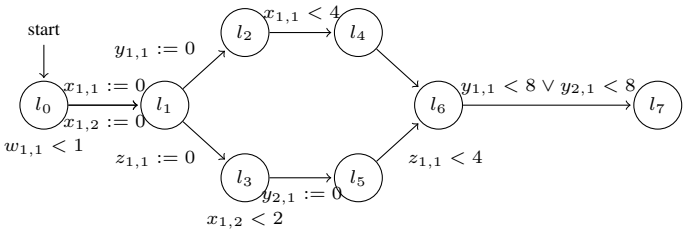
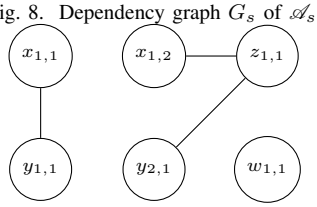


Fig. 8. Dependency graph G_s of $\mathscr{A}_s$.



Example 7: For the given TA $\mathscr{A} = (L, l_0, C, I, T)$ in Fig. 1, after the splitting operation we obtain TA $\mathscr{A}_s = (L, l_0, C_s, I_s, T_s)$ shown in Fig. 7. The dependency graph G_s of $\mathscr{A}_s$ is given in Fig. 8. Note that before splitting operation, unnecessary resets are removed from $\mathscr{A}$.

C. Clock Reduction Algorithm

In this section, we combine dependency analysis and splitting method and prove that the proposed approach solves Prob. 1. First, we remove all unnecessary resets and split clocks that does not introduce dependent clocks, then we apply dependency analysis and graph coloring on clocks, finally we merge the clocks according to the partition obtained from graph coloring. The algorithm is given in Alg. 1. Notice that, in the last step, we remove t_0 and add a new initiation transition.

Theorem 3: The TA $\mathscr{A}_m = (L, l_0, C_m, I_m, T_m)$ obtained from Alg. 1 is timed bisimilar to the TA $\mathscr{A} = (L, l_0, C, I, T)$.

Proof: $\mathscr{A}_r \approx \mathscr{A}$ via repeated applications of Prop. 1. $\mathscr{A}_s \approx \mathscr{A}_r$ by repeated applications of Thm. 2. $\mathscr{A}_m^{x_i \leftarrow \mathcal{C}_i} \approx \mathscr{A}_m^{x_{i-1} \leftarrow \mathcal{C}_{i-1}}$ for each $i = 1, \ldots, n$ by Cor. 1. Since initially $\mathscr{A}_m = \mathscr{A}_s$ and lastly $\mathscr{A}_m = \mathscr{A}_m^{x_n \leftarrow \mathcal{C}_n}$, it follows from the transitivity of timed bisimulation relation that $\mathscr{A}_m \approx \mathscr{A}$. ∎

Theorem 4: The TA $\mathscr{A}_m = (L, l_0, C_m, I_m, T_m)$, result of Alg. 1 on $\mathscr{A} = (L, l_0, C, I, T)$, is a solution for Prob. 1.

Proof: The resulting TA $\mathscr{A}_m = (L, l_0, C_m, I_m, T_m)$ has the same set of locations and the same initial transition. Since we only apply transformations changing set of clocks that are reset on a transition and constraints on a transition, for each $(l_s, l_t, \lambda, \phi) \in T$, there exists a $(l_s, l_t, \lambda', \phi') \in T_m$

Algorithm 1 Clock Reduction Algorithm

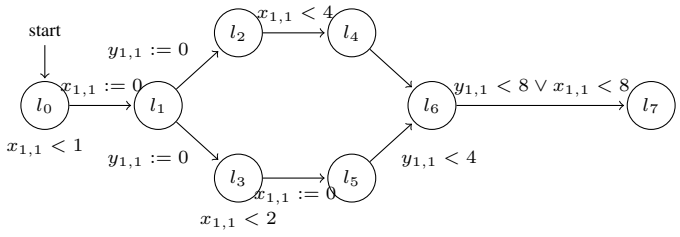
Require: A TA $\mathscr{A} = (L, l_0, C, I, T)$.
Ensure: A TA $\mathscr{A}_m = (L, l_0, C_m, I_m, T_m)$.

$\mathscr{A}_s \leftarrow \mathscr{A}_r$ ▷ initilization for iterative split, $\mathscr{A}_r$ is the TA obtained by iteratively removing all unnecessary resets.
for each $x \in C$ **do**
  $\mathscr{A}_{s,x}$ is obtained after applying Def. 9 to $\mathscr{A}_s$ for x.
  $\mathcal{D}_{s,x}$ is the dependency relation of $\mathscr{A}_{s,x}$ (Def. 7).
  if $\pi'_C(x) \times \pi'_C(x) \cap \mathcal{D}_{s,x} = \emptyset$ **then** $\mathscr{A}_s = \mathscr{A}_{s,x}$
end for
$\mathcal{D}_s$ is the dependency relation of $\mathscr{A}_s$ (Def. 7).
$\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\} = GraphColoring(C_s, \mathcal{D}_s)$ ▷ the minimum partition of C_s found by graph coloring.
$\mathscr{A}_m \leftarrow \mathscr{A}_s$ ▷ initilization for iterative merge.
for each $\mathcal{C}_i \in \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$ **do**
  $\mathscr{A}_m \leftarrow \mathscr{A}_m^{x_i \leftarrow \mathcal{C}_i}$ for some $x_i \in \mathcal{C}_i$
end for
$T_m \leftarrow \{T_m \setminus \{t_0\}\} \cup \{(\tilde{l}, l_0, C_m, True)\}$

and $|T| = |T_m|$. Hence, the underlying graph structure of $\mathscr{A}$ is preserved in $\mathscr{A}_m$. Similarly, all applied transformations preserves the constraint constants by definition; thus set of constants appearing on the invariants and the constraints of transitions are the same in both $\mathscr{A}$ and $\mathscr{A}_m$. Since in the splitting part, we do not split any clock that creates pseudo-dependencies between the new splitted clocks, i.e., we do not split any clock that cannot be merged into one in the later merging phase, the solution ensures that $|C_m| \leq |C|$. Finally, $\mathscr{A}_m \approx \mathscr{A}$ holds by Thm. 3 which completes the proof. ■

Complexity of Alg. 1 The redundancy check operation for a reset, split and merge operations can be done via a depth first search, thus have complexity $O(|L| + |T|)$. Therefore, these run in time $O((|L|+|T|)|C|)$. However, the scope computation for a clock requires enumerating all simple paths between two locations, which is exponential in $|L|$. The complexity of the graph coloring problem is exponential in the number of vertices of $\mathcal{D}_s$, that is $r \times c$, where $c = \max\{|controls(x)|x \in C\}, r = \max\{|resets(x)|x \in C\}$ due to the split operation.

Fig. 9. The TA $\mathscr{A}_m = (L, l_0, C_m, I_m, T_m)$ obtained after applying the solution to the TA in Fig. 1.



Example 8: For the given TA in Fig. 1, the TA obtained after applying proposed the solution is given in Fig. 9.

IV. Empirical Evaluation

This section answers the following questions by an empirical evaluation of a case study: (i) To what extend does the proposed solution reduce the number of clocks, and (ii) how does the runtime of the algorithm suffer from the complexity?

We implemented the proposed solution as a simple Python program. The program uses NetworkX [20], a well-known Python package for graph operations. For generating random TA as test cases, we fixed two graph structures (an acyclic and a cyclic graph). The first graph structure we used for testing is our running example from Fig.1 (named `ExpAcyc`) and the second one is a cyclic graph with ten nodes (i.e., locations) where locations are indexed from 1 to 10 with the following layout: each i^{th} location is connected to $(i+1)^{th}$ location (except for $i = 10$, it is connected to 1) and additionally we have the following three transitions (chosen randomly to make the graph more complex) $(6, 1), (8, 2)$, and $(3, 9)$ (named `ExpCyc`). In both of the experiments, we varied number of clocks from two to ten. For each clock, at most four resets and four constraints are randomly placed in the graph. Experiments are performed on a hardware with an Intel Core i5 processor clocked at 2.3 GHz and a 16 GB main memory.

In Tab. I, we report the results of the experiments. In both experiments, for each clock count, we generated 1000 random samples and applied the proposed algorithm. The initial clock count is given in the first column, average clock count of all samples after applying Alg. 1 is given in the second column for `ExpAcyc` and fourth column for `ExpCyc`, and the average runtime of the solution in milliseconds is given in the third column for `ExpAcyc` and fifth column for `ExpCyc`.

TABLE I
Experiment Results

Initial Clock Count	`ExpAcyc` Result	`ExpAcyc` Runtime (ms)	`ExpCyc` Result	`ExpCyc` Runtime (ms)
2	1.549	0.916	1.872	2.091
3	1.964	1.787	2.692	3.978
4	2.348	2.947	3.467	6.459
5	2.676	4.359	4.187	10.165
6	3.015	5.953	4.879	14.064
7	3.261	8.059	5.593	19.133
8	3.476	10.091	6.246	24.993
9	3.871	12.972	6.782	32.226
10	4.115	17.837	7.457	38.383

Given results of the empirical evaluation shows that (i) as the number of clocks in the model increases, the proposed solution continues to reduce number of clocks in the TA; hence, it gives promising results for the applicational concerns, and (ii) even for considerably large clock counts, the average runtime of the algorithm is under one second which shows that although, the proposed solution has exponential complexity, it is still applicable to practical areas.

V. Comparison with Related Work

In [13], Daws and Yovine proposed an algorithm for reducing the number of clocks of a TA by combining two methods. The first one is based on detecting the active set of clocks in a location and locally renaming them to obtain a timed bisimilar TA. The second one is based on detecting equal clocks and deleting redundant ones. Due to the local renaming approach, their solution does not always yield to the minimum number of

clocks. Moreover, they consider a variation of traditional TA, where assigning clocks to each other is allowed; however, in this work, clocks can only be reset to zero as in the traditional TA. This nuance has a significant effect on the solution.

In [14], Guha et al. proposed an algorithm for finding a new TA that is timed bisimilar to the original one with the minimum number of clocks. While their solution generates the TA with minimal number of clocks, it can substantially increase the number of locations and transitions, i.e. they significantly change the design parameters of the original TA. Consequently, in some cases, their solution may result into a slower verification time as it increases number of locations and transitions [1], [2] while they sacrifice from the understandability of the model at the same time. Moreover, their solution has 2-EXPTIME computational complexity.

In [15], Saeedloei et al. proposed an algorithm for reducing the number of clocks of a TA without changing the underlying graph. Their solution is based on renaming the clocks. They do liveliness analysis for clocks and use graph coloring for reducing number of clocks. Due to the graph coloring, their solution has exponential complexity on the number of clocks except a sub-class of TA. Below, we present the differences between [15] and our solution.

Liveness analysis: We define the scope of a clock as the set of all paths in between the target location of a reset and the source location of its corresponding constraint, i.e., we do not include the transition on which a clock is reset. In [15], they include reset transitions to their range definition, which can cause redundant dependencies, since a clock can be tested and reset on the same transition. During the execution of such transition, first the constraint is checked, then if the constraint is satisfied, the clocks are reset. Due to this nuance in our definition, we are able to avoid such redundant dependencies: hence, our solution further reduces the number of clocks.

Graph coloring: Applying graph coloring to clock minimization problem has been first introduced in [14], then used in [15] with a different approach. We apply graph coloring in a similar way to [15] since the intuitions behind both of the solutions root from identifying the independent clocks that can be merged, which trivially maps to the graph coloring problem.

Splitting: In [15], splitting has been addressed as a problem which cannot be handled by their solution. We formally define the splitting and use it to manipulate TA to avoid redundant dependencies. We consider the formal definition of splitting by preserving bisimulation property as a contribution, since it was just vaguely mentioned as a problematic concept in [15].

Optimality: The reduction algorithm of [15] generates the TA with minimal number of clocks for the class of TA with the following restrictive properties: (i) at most one clock can be reset on a transition and (ii) if a clock is reset on a transition leaving a location l, then it must be reset on every transition leaving l. Note that for this class of TA, our solution also generates the minimal number of clocks. Essentially, the number of clocks generated by our solution is upper bounded by [15], since we improve it in two directions: splitting and a more precise scope definition. For our running example, [15] reduces the clock count to 3, whereas we reduce it to 2. As another example, our method reduces the clock count of TA shown in Fig. 4 of [15] to 2, whereas they reduced it to 3. Note that our example violates property (ii), and the example from [15] violates property (i). Hence, neither of them belongs to the restricted class considered in [15] for the minimality claim. In particular, we have a stronger minimality claim.

VI. Conclusion

In this paper, we aimed at reducing number of clocks of a TA without changing the underlying graph structure and the design parameters of the modeled system. We developed a novel solution to this problem that it applicable to whole class of TA. First, we presented precise concepts for analyzing relations between clocks and demonstrated how these concepts can be used to reduce number of clocks in a TA. Then, we formalized the splitting operation that is a crucial phase of the solution for further reducing number of clocks in a TA. Finally, we presented a promising empirical evaluation of the algorithm to show its applicability and runtime performance.

References

[1] R. Alur, C. Courcoubetis, and D. Dill, "Model-checking in dense real-time," *Information and computation*, vol. 104, no. 1, pp. 2–34, 1993.

[2] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

[3] R. Alur, *Principles of Cyber-Physical Systems*. The MIT Press, 2015.

[4] C. Heitmeyer and N. Lynch, "The generalized railroad crossing: a case study in formal verification of real-time systems," in *1994 Proceedings Real-Time Systems Symposium*, Dec 1994, pp. 120–131.

[5] F. Wang, "Formal verification of timed systems: a survey and perspective," *Proceedings of the IEEE*, vol. 92, no. 8, pp. 1283–1305, Aug 2004.

[6] M. Kwiatkowska, A. Mereacre, N. Paoletti, and A. Patanè, "Synthesising robust and optimal parameters for cardiac pacemakers using symbolic and evolutionary computation techniques," in *Hybrid Systems Biology*, A. Abate and D. Šafránek, Eds. Cham: Springer International Publishing, 2015, pp. 119–140.

[7] A. David, J. Illum, K. G. Larsen, and A. Skou, "Model-based framework for schedulability analysis using UPPAAL 4.1," in *Model-based design for embedded systems*, 2009, pp. 117–144.

[8] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi, "Times: a tool for schedulability analysis and code generation of real-time systems," in *Proc. of Formal Modeling and Analysis of Timed Systems*, 2003, pp. 60–72.

[9] N. Guan, Z. Gu, Q. Deng, S. Gao, and G. Yu, "Exact schedulability analysis for static-priority global multiprocessor scheduling using model-checking," in *Proc. of SEUS*, 2007, pp. 263–272.

[10] G. Behrmann, A. David, K. G. Larsen, J. Hakansson, P. Petterson, W. Yi, and M. Hendriks, "Uppaal 4.0," in *Proceedings of the 3rd International Conference on the Quantitative Evaluation of Systems*, ser. QEST '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 125–126.

[11] S. Tripakis, "Folk theorems on the determinization and minimization of timed automata," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2003, pp. 182–188.

[12] O. Finkel, "Undecidable problems about timed automata," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2006, pp. 187–199.

[13] C. Daws and S. Yovine, "Reducing the number of clock variables of timed automata," in *Real-Time Systems Symposium, 1996., 17th IEEE*. IEEE, 1996, pp. 73–81.

[14] S. Guha, C. Narayan, and S. Arun-Kumar, "Reducing clocks in timed automata while preserving bisimulation," in *International Conference on Concurrency Theory*. Springer, 2014, pp. 527–543.

[15] N. Saeedloei and F. Kluźniak, "Clock allocation in timed automata and graph colouring," in *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control*. ACM, 2018, pp. 71–80.

[16] J. Bengtsson and W. Yi, "Timed automata: Semantics, algorithms and tools," in *Advanced Course on Petri Nets*. Springer, 2003, pp. 87–124.
[17] R. Alur, "Timed automata," in *International Conference on Computer Aided Verification*. Springer, 1999, pp. 8–22.
[18] K. G. Larsen and W. Yi, "Time abstracted bisimulation: Implicit specifications and decidability," in *International Conference on Mathematical Foundations of Programming Semantics*. Springer, 1993, pp. 160–176.
[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.
[20] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

APPENDIX

Proof of Thm. 1: We will prove the bisimulation properties ***P1*** and ***P2*** from Def. 3 for R for both time elapse and location change transitions. Consider any$((l,\nu),(l,\nu')) \in R$.

Time elapse, Proof of *P1*: Assume that $(l,\nu) \xrightarrow{\delta} (l,\nu+\delta)$. To prove ***P1*** , we will show that $((l,\nu+\delta),(l,\nu'+\delta)) \in R$ and $(l,\nu') \xrightarrow{\delta}{}' (l,\nu'+\delta)$ holds. By construction of $\mathcal{T}(\mathscr{A})$, $(l,\nu) \xrightarrow{\delta} (l,\nu+\delta)$ implies that $I(l)$ holds true for $\nu+\delta'$ for each $\delta' \in [0,\delta]$. By property **MP** (16), $I'(l)$ holds true for $\nu'+\delta'$ for each $\delta' \in [0,\delta]$. Consequently, $(l,\nu'+\delta) \in Q'$ and $(l,\nu') \xrightarrow{\delta}{}' (l,\nu'+\delta)$, and by (12), $((l,\nu+\delta),(l,\nu'+\delta)) \in R$, which completes the proof of ***P1*** for time elapse transitions.

Time elapse, Proof of *P2*: Now, assume that $(l,\nu') \xrightarrow{\delta}{}' (l,\nu'+\delta)$. To prove ***P2***, we will show that $((l,\nu+\delta),(l,\nu'+\delta)) \in R$ and $(l,\nu) \xrightarrow{\delta} (l,\nu+\delta)$ holds. By construction of $\mathcal{T}(\mathscr{A}^{x\leftarrow y})$, $(l,\nu') \xrightarrow{\delta}{}' (l,\nu'+\delta)$ implies that $I'(l)$ holds true for $\nu'+\delta'$ for each $\delta' \in [0,\delta]$. Then, by property **MP** (16), $I(l)$ holds true for $\nu+\delta'$ for each $\delta' \in [0,\delta]$. Consequently, $(l,\nu+\delta) \in Q$, and $(l,\nu) \xrightarrow{\delta} (l,\nu+\delta)$, and by (12), $((l,\nu+\delta),(l,\nu'+\delta)) \in R$, which completes the proof of ***P2*** for time elapse transitions.

Location change, Proof of *P1*: Assume that $(l,\nu) \rightarrow (l_t,\nu_t)$. To prove ***P1***, we will show that $((l_t,\nu_t),(l_t,\nu'_t)) \in R$, where

$$\nu'_t(z) = \begin{cases} \nu_t(z) & \forall z \in C \setminus \{x,y\} \\ M(l_t,(\nu_t(x),\nu_t(y))) & z \in \{x\} \end{cases} \tag{26}$$

and $(l,\nu') \rightarrow{}' (l_t,\nu'_t)$ holds.

By assumption $(l_t,\nu_t) \in Q$, which implies that $I(l_t)$ holds true for ν_t. Then, by property **MP** (16), $I'(l_t)$ holds true for ν'_t. Therefore, $(l_t,\nu'_t) \in Q'$, and by (12), $((l_t,\nu_t),(l_t,\nu'_t)) \in R$.

If $(l,\nu) \rightarrow (l_t,\nu_t)$, then there exists $(l,l_t,\lambda,\phi) \in T$ such that ϕ holds true for ν and $\nu_t = \nu[\lambda := 0]$. Then, by property **MP** (16), $\mu_\Phi(\phi,x,y)$ holds true for ν'. By construction of $\mathscr{A}^{x\leftarrow y}$ (see Def. 4),

$$(l,l_t,\mu_C(\lambda,x,y),\mu_\Phi(\phi,x,y)) \in T'$$

To complete the proof of ***P1*** for location change transitions, it is sufficient to show that

$$\nu'_t = \nu'[\mu_C(\lambda,x,y) := 0] \tag{27}$$

Since $(l,\nu') \rightarrow{}' (l_t,\nu'_t)$ holds via transition $(l,l_t,\mu_C(\lambda,x,y),\mu_\Phi(\phi,x,y))$ and (27). We prove (27) for each clock $z \in C \setminus \{y\}$ by considering the following five cases:

c1 $z \in C\setminus(\{x,y\}\cup\lambda)$: By (26), $\nu'_t(z) = \nu_t(z)$. Since $z \notin \lambda$, $\nu_t(z) = \nu(z)$. By (12), for $z \notin \{x,y\}$, $\nu(z) = \nu'(z)$. Finally, by (3) $\mu_C(\lambda,x,y) \subseteq \lambda \cup \{x\}$. Consequently, $z \notin \mu_C(\lambda,x,y)$ and $\nu'_t(z) = \nu'(z)$.

c2 $z \in \lambda \setminus \{x,y\}$: By (26), $\nu'_t(z) = \nu_t(z)$. $z \in \lambda$ implies that $\nu_t(z) = 0$. As $z \in \mu_C(\lambda,x,y)$ by (3), $\nu'_t(z) = 0$ and the equality holds.

c3 $z \in \{x\} \subseteq \lambda$ (i.e. z is x): By (3), $x \in \mu_C(\lambda,x,y)$. Therefore, $\nu'_t(x) = 0$ and (27) holds.

c4 $z \in \{x\}$, $x \notin \lambda$ and $x \notin \mu_C(\lambda,x,y)$: Since $x \notin \lambda$, $\nu_t(x) = \nu(x)$. By definition (26), $\nu'(x) = M(l,(\nu(x),\nu(y)))$. By (15), for each of the following cases **a-d**, $M(l,(a,b)) = M(l_t,(a,b))$ for any a,b (M chooses the same clock in both locations):

a: $l,l_t \in L_x \cap L_y$, **b:** $l,l_t \in L_x \setminus L_y$, **c:** $l,l_t \in L_y \setminus L_x$, **d:** $l,l_t \in L \setminus \{L_x \cup L_y\}$

Therefore, in these cases (27) holds as $x \notin \mu_C(\lambda,x,y)$ and $\nu'_t(x) = \nu'(x)$. Now consider

e. $l \in L_x\setminus L_y$ and $l_t \in L_y\setminus L_x$ **f.** $l \in L_y\setminus L_x$ and $l_t \in L_x\setminus L_y$

Condition **e.** implies that l is in the scope of x and l_t is in the scope of y. $(l,l_t,\lambda,\phi) \in T$, therefore $controls(y)$ is reachable from l. Consequently, it is reachable from a location where x was reset (as unnecessary resets are removed via (8)). Then, x and y is reset on the same transition, $\bar{\nu}(x) = \bar{\nu}(y)$ for any $(l,\bar{\nu}) \in Q$, hence (27) holds. Otherwise, x is reset in the scope of y, which contradicts with the independence assumption that $(x,y) \notin \mathcal{D}$. The same argument applies to case **f.**. Observing that cases **a-f** cover all possibilities completes the proof.

c5 $z \in \{x\}$, $x \notin \lambda$ and $x \in \mu_C(\lambda,x,y)$: In this case, $y \in \lambda$ and it is replaced with x by μ_C. As unnecessary resets are removed via (8), $l_t \in L_y$, $M(l,(\nu_t(x),\nu_t(y))$ is $\nu_t(y)$, which is 0, and $\nu'_t(x) = 0$.

It was shown that (27) always holds. Therefore, $(l,\nu') \rightarrow{}' (l_t,\nu'_t)$, which completes the proof for ***P1*** for location change transitions.

Location change, Proof of *P2*: Assume that $(l,\nu') \rightarrow{}' (l_t,\nu'_t)$. To prove ***P2***, we will show that there exists $(l_t,\nu_t) \in Q$ such that $(l,\nu) \rightarrow (l_t,\nu_t)$ and $((l_t,\nu_t),(l_t,\nu'_t)) \in R$. By construction of $\mathcal{T}(\mathscr{A}^{x\leftarrow y})$, there exists $(l,l_t,\lambda',\phi') \in T'$ such that ϕ' holds true at ν' and $\nu'_t = \nu'[\lambda' := 0]$. By Defn. 4, there exists $(l,l_t,\lambda,\phi) \in T$ such that $\lambda' = \mu_C(\lambda,x,y)$ and $\phi' = \mu_\Phi(\phi,x,y)$. By property **MP** (16), ϕ holds true for ν. Let ν_t be defined as $\nu_t = \nu[\lambda := 0]$. Hence, it holds that $(l_t,\nu_t) \in Q$ and $(l,\nu) \rightarrow (l_t,\nu_t)$. Considering the cases presented in the proof of location change transitions for ***P1***, it follows that $((l_t,\nu_t),(l_t,\nu'_t)) \in R$, which completes the proof for location change transitions for ***P2***.

Note that $q_0 = (l_0,\nu_0)$, $\nu_0(z) = 0 \forall z \in C$, $q'_0 = (l_0,\nu'_0)$, $\nu_0(z) = 0 \forall z \in C \setminus \{y\}$ and $((l_0,\nu_0),(l_0,\nu'_0)) \in R$. Hence, $R \in Q \times Q'$ is a timed bisimulation relation and $\mathscr{A}^{x\leftarrow y}$ and $\mathscr{A}$ are timed bisimilar.