Reducing Quasi-equal Clocks in Networks of Timed Automata

Christian Herrera, Bernd Westphal, Sergio Feo-Arenis, Marco Muñiz and Andreas Podelski

Albert-Ludwigs-Universität Freiburg, 79110 Freiburg, Germany

Abstract. We introduce the novel notion of *quasi-equal* clocks and use it to improve the verification time of networks of timed automata. Intuitively, two clocks are *quasi-equal* if, during each run of the system, they have the same valuation except for those points in time where they are reset. We propose a transformation that takes a network of timed automata and yields a network of timed automata which has a smaller set of clocks and preserves properties up to those not comparing *quasi-equal* clocks. Our experiments demonstrate that the verification time in three transformed real world examples is much lower compared to the original.

1 Introduction

Modelling the local timing behaviour of components and the synchronisation between them is the natural way to model distributed real-time systems by networks of timed automata [1]. This is achieved in a straightforward manner by using independent clocks.

For designs where a set of clocks is intended to be synchronized, e.g., the class of TDMA-based protocols [2], using independent clocks causes unnecessary verification overhead for those specifications where the order of resets of synchronised clocks is not relevant. For instance, in network traversal time requirements, resetting synchronised clocks does not contribute to the time lapses being measured. The unnecessary overhead is caused by the interleaving semantics of timed-automata where the automata do their resets one by one and thereby induce a set of reachable intermediate configurations which grows exponentially in the number of components in the system. Although the interleaving semantics of timed-automata offers a practical solution for model checking in tools like *Uppaal* [3], it artificially introduces intermediate states that are explored when verifying models of physical systems, although they may be irrelevant for the property being verified.

The overhead could be eliminated by manually optimising models for verification. Nonetheless, modelling without technicalities — in particular without manual optimizations for verification — is desired to improve on the readability and maintainability of the models. We aim to bridge the gap between efficiency and readability by enabling the modelling engineer to use more natural representations of a system. Unnecessary overhead can be mechanically removed as per our approach, thus enabling both readable models and efficient model checking.

To this end, we characterise clocks intended to be synchronised in the real world by the novel notion of *quasi-equality.* Intuitively, two clocks are *quasi-equal* if, during each run of the system, they have the same valuation except for those points in time where they are reset. We call the properties which, for a given network of timed automata, are independent from the ordering in which the *quasi-equal* clocks are reset, *validable.* Sets of quasi-equal clocks induce *equivalence classes* in networks of timed-automata. We present an algorithm that replaces all clocks from an *equivalence class* of *quasi-equal* clocks by a representative clock. The result is a network of timed automata with a smaller set of clocks. It is weakly bisimilar to the original network and thus preserves validable properties. We show that when performing clock replacement alone, properties are not necessarily preserved. Our algorithm introduces a new automaton in the network and uses auxiliary variables and *broadcast* synchronization channels to ensure that the semantics of the original network are preserved, up to configurations where quasi-equal clocks have different valuations. The use of our algorithm can lead to significant improvements in the verification cost of validable properties compared to the cost of verifying them in the original network.

This paper is organized as follows. In Section 2, we provide basic definitions. Section 3 introduces the formal definition of *quasi-equal* clocks, presents the algorithm that implements our approach on the set of well-formed networks, and proves its correctness. In Section 4, we compare the verification time of three real world examples before and after applying our approach. In Section 5, we draw conclusions and propose future work.

1.1 Related Work

The reduction of the state space to be explored in order to speed up the verification of properties of a system, is a well-known research topic. Diverse techniques have been proposed to achieve such a reduction, many of them by using static analysis over timed automata [4–7]. One method that uses static analysis is presented in [8], originally defined for single automata and later generalized for networks of timed automata [9]. This method reduces the number of clocks in single timed automata by detecting *equal* clocks. Two clocks are equal in a location if both are reset at the same time and by the same edge, or both are set to clocks that are themselves equal in the source location. Equal clocks always have the same valuation, so just one clock for every set of equal clocks in a given location is necessary to determine the behavior of the system at that location. The case studies we considered for experiments do not have equal clocks therefore applying the method in [8] would not reduce clocks.

In sequential timed automata [10], one set of quasi-equal clocks is syntactically declared. Those quasi-equal clocks are implicitly reduced by applying the sequential composition operator, which also exploits other properties of sequential timed automata, and thereby achieves further improvements in verification time.

In [11], clocks are reduced while abstracting systems composed of timed components which represent processes. Each process uses one internal clock for its internal operations. The approach exploits the fact that each component works in a sort of sequence in order to process events, and each internal clock is only used for a small fraction of time during such a sequence, thus only one clock can be used instead. The networks we consider can in general not be reduced to this approach as we do not assume a working sequence.

The technique in [4, 5] is based on so called *observers*, which are single components representing safety or liveness requirements of a given network of timed automata. For each location of the observer, the technique deactivates (or ignores) irrelevant components (clocks or even a whole automaton) if such components do not play a role in the future evolution of such an observer. The networks of our experiments do not have observers, therefore we can not use this technique. However, their case studies may benefit from our approach if observer clocks are quasi equal to component clocks.

2 Preliminaries

Following the presentation in [12], we here recall timed automata definitions.

Let $\mathcal{X}$ be a set of *clocks*. The set $\Phi(\mathcal{X})$ of *simple clock constraints* over $\mathcal{X}$ is defined by the grammar $\varphi ::= x \sim y \mid x - y \sim c \mid \varphi_1 \wedge \varphi_2$ where $x, y \in \mathcal{X}$, $c \in \mathbb{Q}_{\geq 0}$, and $\sim \in \{<, \leq, \geq, >\}$. Let $\Phi(\mathcal{V})$ be a set of *integer constraints* over *variables* $\mathcal{V}$. The set $\Phi(\mathcal{X}, \mathcal{V})$ of *constraints* comprises $\Phi(\mathcal{X})$, $\Phi(\mathcal{V})$, and conjunctions of clock and integer constraints. We use $\mathit{clocks}(\varphi)$ to denote the set of clocks occurring in a constraint φ. We assume the canonical satisfaction relation "$\models$" between *valuations* $\nu : \mathcal{X} \cup \mathcal{V} \to \mathit{Time} \cup \mathbb{Z}$ and constraints, with $\mathit{Time} = \mathbb{R}_{\geq 0}$.

A timed automaton $\mathcal{A}$ is a tuple $(L, B, \mathcal{X}, \mathcal{V}, I, E, \ell_{ini})$, which consists of a finite set of *locations* L, a finite set B of *actions* comprising the *internal action* τ, finite sets $\mathcal{X}$ and $\mathcal{V}$ of clocks and variables, a mapping $I : L \mapsto \Phi(\mathcal{X})$, that assigns to each location a *clock constraint*, and a set of *edges* $E \subseteq L \times B \times \Phi(\mathcal{X}, \mathcal{V}) \times \mathcal{R}(\mathcal{X}, \mathcal{V}) \times L$. An edge $e = (\ell, \alpha, \varphi, \vec{r}, \ell') \in E$ from location ℓ to ℓ' involves an action $\alpha \in B$, a *guard* $\varphi \in \Phi(\mathcal{X}, \mathcal{V})$, and a *reset vector* $\vec{r} \in \mathcal{R}(\mathcal{X}, \mathcal{V})$. A reset vector is a finite, possibly empty sequence of *clock resets* $x := 0$, $x \in \mathcal{X}$, and *assignments* $v := \psi_{int}$, where $v \in \mathcal{V}$ and ψ_{int} is an integer expression over $\mathcal{V}$. We write $\mathcal{X}(\mathcal{A})$, $\ell_{ini}(\mathcal{A})$, etc. to denote the set of clocks, the initial location, etc. of $\mathcal{A}$, and $\mathit{clocks}(\vec{r})$ to denote the set of clocks occurring in $\vec{r}$.

A finite sequence $\mathcal{A}_1, \dots, \mathcal{A}_N$ of timed automata with pairwise disjoint sets of clocks and pairwise disjoint sets of locations together with a set $\mathcal{B} \subseteq \bigcup_{i=1}^{N} B(\mathcal{A}_i)$ of *broadcast channels* is called *network (of timed automata)*. To indicate that $\mathcal{N}$ consists of $\mathcal{A}_1, \dots, \mathcal{A}_N$, we write $\mathcal{N}(\mathcal{A}_1, \dots, \mathcal{A}_N)$, and we write $\mathcal{A} \in \mathcal{N}$ if and only if $\mathcal{A} \in \{\mathcal{A}_1, \dots, \mathcal{A}_N\}$. Given a set of clocks $X \subseteq \mathcal{X}(\mathcal{N})$, we use $\mathcal{RES}_X(\mathcal{N})$ to denote the set of automata in $\mathcal{N}$ which have an outgoing edge that resets a clock from X, i.e. $\mathcal{RES}_X(\mathcal{N}) = \{\mathcal{A} \in \mathcal{N} \mid \exists\, (\ell, \alpha, \varphi, \vec{r}, \ell') \in E(\mathcal{A}) \bullet \mathit{clocks}(\vec{r}) \cap X \neq \emptyset\}$.

The operational semantics of the network $\mathcal{N}$ is the labelled transition system $\mathcal{T}(\mathcal{N}) = (\mathit{Conf}(\mathcal{N}), \mathit{Time} \cup B, \{\xrightarrow{\lambda} \mid \lambda \in \mathit{Time} \cup B\}, \mathcal{C}_{ini})$. The set of configurations $\mathit{Conf}(\mathcal{N})$ consists of pairs of *location vectors* $\langle \ell_1, \dots, \ell_N \rangle$ from $\times_{i=1}^{N} L(\mathcal{A}_i)$ and valuations of $\bigcup_{1 \leq i \leq N} \mathcal{X}(\mathcal{A}_i) \cup \mathcal{V}(\mathcal{A}_i)$ which satisfy the constraint $\bigwedge_{i=1}^{N} I(\ell_i)$.

We write $\ell_{s,i}$, $1 \leq i \leq N$, to denote the location which automaton $\mathcal{A}_i$ assumes in configuration $s = \langle \ell_s, \nu_s \rangle$ and $\nu_{s,i}$ to denote $\nu_s|_{\mathcal{V}(\mathcal{A}_i) \cup \mathcal{X}(\mathcal{A}_i)}$. Between two configurations $s, s' \in \mathit{Conf}(\mathcal{N})$ there can be three kinds of transitions. There is a *delay transition* $\langle \ell_s, \nu_s \rangle \xrightarrow{t} \langle \ell_{s'}, \nu_{s'} \rangle$ if $\nu_s + t' \models \bigwedge_{i=1}^{N} I_i(\ell_{s,i})$ for all $t' \in [0, t]$, where $\nu_s + t'$ denotes the valuation obtained from ν_s by time shift t'. There is a *synchronization transition* $\langle \ell_s, \nu_s \rangle \xrightarrow{\tau} \langle \ell_{s'}, \nu_{s'} \rangle$ if there are $1 \leq i, j \leq N$, $i \neq j$, a channel $b \in B(\mathcal{A}_i) \cap B(\mathcal{A}_j)$, and edges $(\ell_{s,i}, b!, \varphi_i, \vec{r}_i, \ell_{s',i}) \in E(\mathcal{A}_i)$ and $(\ell_{s,j}, b?, \varphi_j, \vec{r}_j, \ell_{s',j}) \in E(\mathcal{A}_j)$ such that $\ell_{s'} = \ell_s[\ell_{s,i} := \ell_{s',i}][\ell_{s,j} := \ell_{s',j}]$, $\nu_s \models \varphi_i \wedge \varphi_j$, $\nu_{s'} = \nu_s[\vec{r}_i][\vec{r}_j]$, and $\nu_{s'} \models I_i(\ell_{s',i}) \wedge I_j(\ell_{s',j})$. Let $b \in \mathcal{B}$ be a broadcast channel and $1 \leq i_0 \leq N$ such that $(\ell_{s,i_0}, b!, \varphi_{i_0}, \vec{r}_{i_0}, \ell_{s',i_0}) \in E(\mathcal{A}_{i_0})$. Let $1 \leq i_1, \ldots, i_k \leq N$, $k \geq 0$, be those indices different from i_0 such that there is an edge $(\ell_{s,i_j}, b?, \varphi_{i_j}, \vec{r}_{i_j}, \ell_{s',i_j}) \in E(\mathcal{A}_{i_j})$. There is *broadcast transition* $\langle \ell_s, \nu_s \rangle \xrightarrow{\tau} \langle \ell_{s'}, \nu_{s'} \rangle$ in $\mathcal{T}(\mathcal{N})$ if $\ell_{s'} = \ell_s[\ell_{s,i_0} := \ell_{s',i_0}] \cdots [\ell_{s,i_k} := \ell_{s',i_k}]$, $\nu_s \models \bigwedge_{j=0}^{k} \varphi_{i_j}$, $\nu_{s'} = \nu_s[\vec{r}_{i_0}] \cdots [\vec{r}_{i_k}]$, and $\nu_{s'} \models \bigwedge_{j=0}^{k} I_{i_j}(\ell_{s',i_j})$.

A finite or infinite sequence $\sigma = s_0 \xrightarrow{\lambda_1} s_1 \xrightarrow{\lambda_2} s_2 \ldots$ is called *transition sequence* (starting in $s_0 \in \mathcal{C}_{ini}$) of $\mathcal{N}$. Sequence σ is called *computation* of $\mathcal{N}$ if and only if it is infinite and $s_0 \in \mathcal{C}_{ini}$. We denote the set of all computations of $\mathcal{N}$ by $\Pi(\mathcal{N})$. A configuration s is called *reachable* (in $\mathcal{T}(\mathcal{N})$) if and only if there exists a computation $\sigma \in \Pi(\mathcal{N})$ such that s occurs in σ. A timed automaton or network configuration s is called *timelocked* if and only if there is no delay transition in any transition sequence starting at s.

A *basic formula* over $\mathcal{N}$ is either $\mathcal{A}_i.\ell$, $1 \leq i \leq n$, $\ell \in L(\mathcal{A}_i)$, or a constraint φ from $\Phi(\bigcup_{i=1}^{N} \mathcal{X}(\mathcal{A}_i), \bigcup_{i=1}^{N} \mathcal{V}(\mathcal{A}_i))$. It is satisfied by a configuration $s \in \mathit{Conf}(\mathcal{N})$ if and only if $\ell_{s,i} = \ell$ or $\nu_s \models \varphi$, respectively. A *reachability query EPF* over $\mathcal{N}$ is $\exists\Diamond\, \mathit{CF}$ where CF is a *configuration formula* over $\mathcal{N}$, i.e. any logical connection of basic formulae. $\mathcal{N}$ satisfies $\exists\Diamond\, \mathit{CF}$, denoted by $\mathcal{N} \models \exists\Diamond\, \mathit{CF}$, if and only if there is a configuration s reachable in $\mathcal{T}(\mathcal{N})$ such that $s \models \mathit{CF}$. We write $\mathcal{N} \models_{\neg\text{timelock}} \exists\Diamond\, \mathit{CF}$ if and only if CF is satisfied by a reachable, not timelocked configuration of $\mathcal{T}(\mathcal{N})$.

3 Reducing Clocks in Networks of Timed Automata

3.1 Quasi-Equal Clocks

Definition 1 (Quasi-Equal Clocks). *Let $\mathcal{N}$ be a network with clocks $\mathcal{X}$. Two clocks $x, y \in \mathcal{X}$ are called quasi-equal, denoted by $x \simeq y$, if and only if for all computation paths of $\mathcal{N}$, the valuations of x and y are equal, or the valuation of one of them is equal to* 0, *i.e., if*

$$\forall\, s_0 \xrightarrow{\lambda_1} s_1 \xrightarrow{\lambda_2} s_2 \cdots \in \Pi(\mathcal{N})\ \forall\, i \in \mathbb{N}_0 \bullet \nu_{s_i} \models (x = 0 \vee y = 0 \vee x = y).$$

For example, consider a distributed chemical plant controller. At the end of every minute, the controller fills two containers with gas, one for at most 10 seconds and one for at most 20 seconds. In Figure 1, a model of the system, the network $\mathcal{N}$ which is composed of automata $\mathcal{A}_1$ and $\mathcal{A}_2$, is shown. Both automata

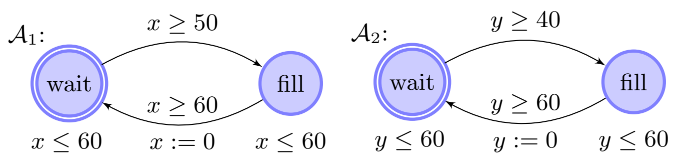


Fig. 1. The model of a chemical plant controller with quasi-equal clocks.

start in a waiting phase and after filling the containers, they wait for the next round. Both clocks, x and y, are reset when their valuation is equal to 60. Yet, in the strict interleaving semantics of networks of timed automata, the resets occur one after the other. According to Definition 1, x and y are *quasi-equal* because their valuations are only different from each other when one of the clocks has already been reset and the other still has value 60.

Lemma 1. *Let $\mathcal{N}$ be a network with clocks $\mathcal{X}$. The quasi-equality relation $\simeq \subseteq \mathcal{X} \times \mathcal{X}$ is an equivalence relation.*

Proof. For transitivity, show $(x = 0 \lor y = 0 \lor x = y) \land (x = 0 \lor z = 0 \lor x = z) \land (y = 0 \lor z = 0 \lor y = z)$ by induction over indices in a computation. □

In the following, we use $\mathcal{EC}_\mathcal{N}$ to denote the set $\{Y \in \mathcal{X}/\simeq \mid 1 < |Y|\}$ of equivalence classes of *quasi-equal* clocks of $\mathcal{N}$ with at least two elements. For each $Y \in \mathcal{X}/\simeq$, we assume a designated representative $\mathit{rep}(Y) \in Y$. We may write $\mathit{rep}(x)$ to denote $\mathit{rep}(Y)$ if $x \in Y$ is the representative clock of Y.

Given a constraint $\varphi \in \Phi(\mathcal{X}, \mathcal{V})$, we write $\Gamma(\varphi)$ to denote the constraint that is obtained by syntactically replacing in φ each occurrence of a clock $x \in \mathcal{X}$ by the representative $\mathit{rep}(x)$.

3.2 Transformational Reduction of Quasi-equal Clocks

In the following we present an algorithm which reduces a given set of quasi-equal clocks in networks of timed automata. For simplicity, we limit the discussion to the syntactically characterised class of well-formed networks. The syntactical rules, although restrictive at first sight, still enabled us to apply our approach to relevant real world examples.

Definition 2 (Well-formed Network). *A network $\mathcal{N}$ is called* well-formed *if and only if it satisfies the following restrictions for each set of quasi-equal clocks $Y \in \mathcal{EC}_\mathcal{N}$:*

(R1) *An edge resetting a clock $x \in Y$ is not a loop and has a guard of the form $x \geq C_Y$, and the source location of such an edge has an invariant $x \leq C_Y$ for some constant $C_Y > 0$, i.e.,*

$$\begin{aligned}&\exists\, C_Y \in \mathbb{N}^{>0}\ \forall \mathcal{A} \in \mathcal{N}\ \forall (\ell, \alpha, \varphi, \vec{r}, \ell') \in E(\mathcal{A})\ \forall x \in \mathit{clocks}(\vec{r}) \bullet \\ &\qquad (\mathit{clocks}(\vec{r}) \cap Y \neq \emptyset) \\ &\qquad\quad \implies (\varphi = (x \geq C_Y) \land I(\ell) = (x \leq C_Y) \land \ell \neq \ell_{ini}(\mathcal{A}) \land \ell \neq \ell').\end{aligned}$$

(R2) *A location having an outgoing edge resetting a clock $x \in Y$, does not have other outgoing edges, and such an edge only resets a single clock, i.e.,*

$$\begin{aligned}&\forall\, e_1 = (\ell_1, \alpha_1, \varphi_1, \vec{r}_1, \ell'_1), e_2 = (\ell_2, \alpha_2, \varphi_2, \vec{r}_2, \ell'_2) \in E(\mathcal{N}) \bullet \\ &\quad (\ell_1 = \ell_2 \wedge (\mathit{clocks}(\vec{r}_1) \cup \mathit{clocks}(\vec{r}_2)) \cap Y \neq \emptyset) \\ &\qquad \implies e_1 = e_2 \wedge |\mathit{clocks}(\vec{r}_1)| = 1 \wedge \exists\, x \in Y \bullet \vec{r}_1 = (x := 0).\end{aligned}$$

(R3) *An edge resetting a clock $x \in Y$ is unique per automaton, i.e.,*

$$\begin{aligned}&\forall\, \mathcal{A} \in \mathcal{N}\ \forall\, (\ell_1, \alpha_1, \varphi_1, \vec{r}_1, \ell'_1), (\ell_2, \alpha_2, \varphi_2, \vec{r}_2, \ell'_2) \in E(\mathcal{A}) \bullet \\ &\quad ((\mathit{clocks}(\vec{r}_1) \cup \mathit{clocks}(\vec{r}_2)) \cap Y \neq \emptyset) \implies e_1 = e_2.\end{aligned}$$

(R4) *A location having an outgoing edge resetting a clock $x \in Y$ has at least one incoming, non-looped edge, i.e.,*

$$\begin{aligned}&\forall\, \mathcal{A} \in \mathcal{N}\ \forall\, (\ell_2, \alpha_2, \varphi_2, \vec{r}_2, \ell'_2) \in E(\mathcal{A}) \bullet \\ &\quad (\mathit{clocks}(\vec{r}_2) \cap Y) \neq \emptyset \implies \exists\, (\ell_1, \alpha_1, \varphi_1, \vec{r}_1, \ell'_1) \in E(\mathcal{A}) \bullet \ell'_1 = \ell_2 \wedge \ell_1 \neq \ell_2.\end{aligned}$$

(R5) *The action of an edge resetting a clock $x \in Y$ is τ, i.e.,*

$$\forall\, (\ell, \alpha, \varphi, \vec{r}, \ell') \in E(\mathcal{N}) \bullet (\mathit{clocks}(\vec{r}) \cap Y \neq \emptyset) \implies \alpha = \tau.$$

(R6) *At most one clock from Y occurs in the constraint of any edge, i.e.,*

$$\forall\, (\ell, \alpha, \varphi, \vec{r}, \ell') \in E(\mathcal{N}) \bullet |\mathit{clocks}(\varphi) \cap Y| \leq 1.$$

By rules *R1*, *R2*, and *R3* there is a unique reset edge per equivalence class and automaton, and a constant describing the reset times of quasi-equal clocks from the same equivalence class. By *R4* guarantees the existence of an edge that can be used to encode blocking multicast synchronisation. Rules *R2*, *R5*, and *R6* guarantee that the behaviour of a well-formed network is independent from the order of resets of quasi-equal clocks.

These rules should be relaxed to cover a broader class of networks of timed automata. For example, *R3* could be weakened to allow quasi-equal clocks with more than one reset point to be reduced. This would make, however, the transformation algorithm and its prove of correctness more involved.

Our transformation mainly operates on the source and destination locations of the edges resetting quasi-equal clocks, so-called reset locations.

Definition 3 (Reset Location). *Let $\mathcal{N}$ be a well-formed network. Let $Y \in \mathcal{EC}_\mathcal{N}$ be a set of clocks of $\mathcal{N}$. Let $(\ell, \alpha, \varphi, \vec{r}, \ell') \in E(\mathcal{N})$ be an edge that resets a clock from Y, i.e. $\mathit{clocks}(\vec{r}) \cap Y \neq \emptyset$. Then ℓ (ℓ') is called* reset (successor) location *wrt. Y. We use $\mathcal{RL}_Y$ ($\mathcal{RL}^+_Y$) to denote the set of reset (successor) locations wrt. Y in $\mathcal{N}$ and we set $\mathcal{RL}_\mathcal{N} := \bigcup_{Y \in \mathcal{EC}_\mathcal{N}} \mathcal{RL}_Y$ and similarly $\mathcal{RL}^+_\mathcal{N}$.*

In the following we describe the transformation function $\mathcal{K}$. It works with two given inputs: a well-formed network $\mathcal{N}$ and the set of equivalence classes $\mathcal{EC}_\mathcal{N}$ of quasi-equal clocks in $\mathcal{N}$. $\mathcal{K}$ outputs a transformed network $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_\mathcal{N})$ by performing in $\mathcal{N}$ the following steps for each equivalence class $Y \in \mathcal{EC}_\mathcal{N}$:

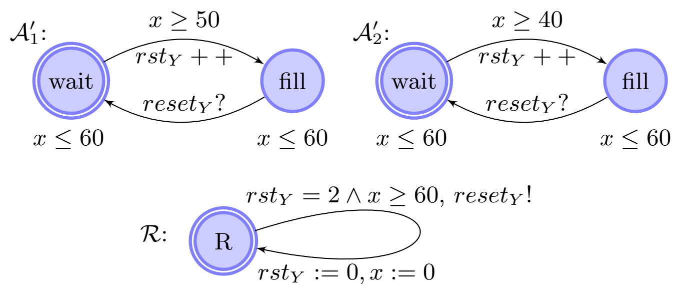


Fig. 2. The model of the chemical plant controller after applying $\mathcal{K}$.

- Delete each reset of a clock from Y.
- For each edge resetting a clock from Y, replace the guard by *true*.
- In each invariant and each guard, replace each clock $x \in Y$ by $rep(x)$.
- Add to $\mathcal{N}$ a broadcast channel $reset_Y$ and add the input action $reset_Y?$ to each edge resetting a clock from Y.
- Add a counter variable rst_Y to $\mathcal{N}$ and add the increment $rst_Y := rst_Y + 1$ to the reset sequence of each incoming edge of a reset location $\ell \in \mathcal{RL}_Y$.

As a final step, add a new automaton $\mathcal{R}$ with a single location $\ell_{ini,\mathcal{R}}$ *if* $\mathcal{EC}_\mathcal{N} \neq \emptyset$. For each $Y \in \mathcal{EC}_\mathcal{N}$, add an edge $(\ell_{ini,\mathcal{R}}, \alpha, \varphi, \vec{r}, \ell_{ini,\mathcal{R}})$ to $\mathcal{R}$ with action $reset_Y!$, guard $\varphi = (rst_Y = n_Y \wedge rep(Y) \geq C_Y)$, and reset vector $\vec{r} = rst_Y := 0, rep(Y) := 0$. In the guard φ, n_Y is the number of automata that reset the clocks of Y, i.e. $n_Y = |\mathcal{RES}_Y(\mathcal{N})|$, and C_Y is the time at which the clocks in Y are reset, i.e., the constant C_Y as described in R1. The result of applying the transformation to the example from Figure 1 is shown in Figure 2.

Note that well-formedness together with the counter variables rst_Y enforce *blocking* multicast synchronisation, that is, always all automata from $\mathcal{RES}_Y(\mathcal{N})$ participate in the reset. Furthermore, $\mathcal{N}'$ is equal to $\mathcal{N}$ if there are no quasi-equal clocks in $\mathcal{N}$.

3.3 A Semantical Characterisation of $\mathcal{N}'$

Following our discussion, we can distinguish two kinds of configurations in the transition system of a well-formed network $\mathcal{N}$. A configuration is unstable if there are quasi-equal clocks with different values, and stable otherwise.

In the following, we observe that our algorithm yields a network whose configurations directly correspond to the stable configurations of $\mathcal{N}$. In addition, we observe that transition sequences in $\mathcal{N}'$ correspond to transition sequences in $\mathcal{N}$ where reset phases of different equivalence classes do not overlap. To this end, we formally define stability of configurations and different notions of reset sequences, in particular full pure reset sequences, i.e., those where reset phases do not overlap.

Definition 4 (Stable Configuration). *Let $\mathcal{N}$ be a well-formed network and let $Y \in \mathcal{EC}_\mathcal{N}$ be a set of quasi-equal clocks.*

A configuration $s \in \mathit{Conf}(\mathcal{N})$ is called stable wrt. Y *if and only if all clocks in Y have the same value in s, i.e., if $\forall\, x \in Y \bullet \nu_s(x) = \nu_s(\mathit{rep}(x))$.*

We use $\mathcal{SC}_Y$ to denote the set of all configurations that are stable wrt. Y and $\mathcal{SC}_{\mathcal{N}}$ to denote the set $\bigcap_{Y \in \mathcal{EC}_{\mathcal{N}}} \mathcal{SC}_Y$ of globally stable *configurations.*

Definition 5 (Reset Sequence). *Let $\mathcal{N}$ be a well-formed network and let $Y \in \mathcal{EC}_{\mathcal{N}}$ be a set of quasi-equal clocks. Let $\sigma = s_0 \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_n} s_n$ be a transition sequence of $\mathcal{N}$ such that s_0 and s_n are stable wrt. Y, $s_1, \ldots, s_{n-1}$ are unstable wrt. Y, and the valuation of every clock in Y at s_n is 0. Then σ is called Y*-reset sequence. *We use $\mathcal{RS}_Y$ to denote the set of Y-reset sequences of $\mathcal{N}$.*

A suffix of σ starting at s_i, $1 \leq i \leq n$, is called a pure Y-reset sequence *if and only if the valuation for some clock of Y changes with every transition, i.e. if $\forall\, i < j \leq n \bullet \nu_{s_{j-1}}|_Y \neq \nu_{s_j}|_Y$. We use $\mathcal{RS}_Y^{pure}$ to denote the set of pure Y-reset sequences of $\mathcal{N}$.*

If σ is in $\mathcal{RS}_Y^{pure}$ and starts in a globally stable configuration, i.e., $s_0 \in \mathcal{SC}_{\mathcal{N}}$, then σ is called a full pure Y-reset sequence. *We use $\mathcal{RS}_Y^{full}$ to denote the set of full pure Y-reset sequences of $\mathcal{N}$. The smallest suffix of σ which is not in $\mathcal{RS}_Y^{pure}$ is called an* impure Y-reset sequence. *We use $\mathcal{RS}_Y^{impure}$ to denote the set of impure Y-reset sequences of $\mathcal{N}$.*

Let $\sigma = s_0 \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_n} s_n \xrightarrow{\lambda_{n+1}} s_{n+1}$ be a transition sequence of $\mathcal{N}$ such that the prefix of σ up to and including s_n is in $\mathcal{RS}_Y^{pure}$ and s_n and s_{n+1} coincide on Y, i.e. $\nu_{s_n}|_Y = \nu_{s_{n+1}}|_Y$. Then σ is called pure Y-reset sequence-δ.

Proposition 1. *For all well-formed networks $\mathcal{N}$, if $s_0 \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_n} s_n$ is a full pure Y-reset sequence of $\mathcal{N}$, then s_n is globally stable, i.e., $s_n \in \mathcal{SC}_{\mathcal{N}}$, and the valuation at s_n of each clock $x \in Y$ is 0, i.e., $\nu_{s_n}|_Y = 0$.*

Formally, the relation between a stable configuration $s \in \mathit{Conf}(\mathcal{N})$ of a well-formed network $\mathcal{N}$ and a configuration $r \in \mathit{Conf}(\mathcal{N}')$ of the network $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}})$ is characterised by the function *reverseQE*. It removes the following from r: the unique location of the automaton $\mathcal{R}$; each counter variable rst_Y, $Y \in \mathcal{EC}_{\mathcal{N}}$; and it assigns to each clock $x \in Y$ the value of the clock $\mathit{rep}(x) \in Y$.

Definition 6 (*reverseQE* and Consistency). *Let $\mathcal{N}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ be a well-formed network and let $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}})$. The function reverseQE : $\mathit{Conf}(\mathcal{N}') \to \mathit{Conf}(\mathcal{N})$ is defined point-wise as follows. Let $r = \langle (\ell_1, \ldots, \ell_n, \ell_{ini,\mathcal{R}}), \nu \rangle \in \mathit{Conf}(\mathcal{N}')$. Then reverseQE$(r) = \langle (\ell_1, \ldots, \ell_n), \tilde{\nu} \rangle$ where*

$$\tilde{\nu} = \big(\nu \cup \textstyle\bigcup_{Y \in \mathcal{EC}_{\mathcal{N}}} \{x \mapsto \nu(\mathit{rep}(x)) \mid x \in Y\}\big) \setminus \{\mathit{rst}_Y \mapsto \nu(\mathit{rst}_Y) \mid Y \in \mathcal{EC}_{\mathcal{N}}\}.$$

The configuration r is called Y-consistent *if and only if $\nu_r(\mathit{rst}_Y)$ is the number of reset locations wrt. Y assumed in r, i.e., if $\nu_r(\mathit{rst}_Y) = |\{\ell_1, \ldots, \ell_n, \ell_{ini,\mathcal{R}}\} \cap \mathcal{RL}_Y|$. We use CONS_Y to denote the set of Y-consistent configurations of $\mathcal{N}'$ and $\mathit{CONS}_{\mathcal{N}'}$ to denote the set $\bigcap_{Y \in \mathcal{EC}_{\mathcal{N}}} \mathit{CONS}_Y$ of* consistent *configurations.*

Proposition 2. *Let $\mathcal{N}$ be a well-formed network. Then reverseQE is a bijection between $\mathit{CONS}_{\mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}})}$ and $\mathcal{SC}_{\mathcal{N}}$.*

In the following we define a special transition relation for well-formed networks, which relates stable configurations by collapsing full pure reset sequences. A special transition in $\mathcal{N}$ corresponds to a single transition in $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_\mathcal{N})$.

Definition 7 ($\overset{\lambda}{\Rightarrow}$). *Let $\mathcal{N}$ be a well-formed network and let $s, s' \in \mathcal{SC}_\mathcal{N}$ be two globally stable configurations of $\mathcal{N}$. There is a transition $s \overset{\lambda}{\Rightarrow} s'$ if and only if there is either a delay or τ-transition, or a full pure Y-reset sequence for some $Y \in \mathcal{EC}_\mathcal{N}$ from s to s' in $\mathcal{T}(\mathcal{N})$, i.e., if*

$$\begin{aligned} & s \xrightarrow{\lambda} s' \wedge \lambda \in \mathit{Time} \cup \{\tau\} \\ & \vee \Big(\lambda = \tau \wedge \exists\, s_0 \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_n} s_n \in \mathcal{RS}_Y^{\mathit{full}} \bullet s = s_0 \wedge s_n = s'\Big). \end{aligned}$$

Configuration s is called $\Rightarrow$-reachable *if and only if there are configurations $s_0, \ldots, s_n \in \mathit{Conf}(\mathcal{N})$ such that $s_0 \in \mathcal{C}_{ini}$, $s_n = s$, and $s_0 \overset{\lambda_1}{\Longrightarrow} s_1 \ldots s_{n-1} \overset{\lambda_n}{\Longrightarrow} s_n$ for some $\lambda_1, \ldots, \lambda_n \in \mathit{Time} \cup \{\tau\}$.*

Definition 8 (Weak Bisimulation). *Let $\mathcal{N}_1$ be a well-formed network and $\mathcal{N}_2$ a network with the same set of locations, i.e., $L(\mathcal{N}_1) = L(\mathcal{N}_2)$. Let $\mathcal{T}(\mathcal{N}_i) = (\mathit{Conf}(\mathcal{N}_i), \Lambda, \{\xrightarrow{\lambda}_i \mid \lambda \in \Lambda\}, \mathcal{C}_{ini,i})$, $i = 1, 2$, be the corresponding transition systems restricted to $\Lambda = \mathit{Time} \cup \{\tau\}$.*

A weak bisimulation *is a relation $\mathcal{S} \subseteq \mathit{Conf}(\mathcal{N}_1) \times \mathit{Conf}(\mathcal{N}_2)$ such that*

1. $\forall\, s \in \mathcal{C}_{ini,1}\ \exists\, r \in \mathcal{C}_{ini,2} \bullet (s, r) \in \mathcal{S}$ *and* $\forall\, r \in \mathcal{C}_{ini,2}\ \exists\, s \in \mathcal{C}_{ini,1} \bullet (s, r) \in \mathcal{S}$,
2. *for all $(s, r) \in \mathcal{S}$,*
   (a) *for all configuration formulae CF over $\mathcal{N}_i$, $s \models CF$ iff $r \models \Gamma(CF)$,*
   (b) *if $s \xrightarrow{\lambda}_1 s'$, there exists $r' \in \mathit{Conf}(\mathcal{N}_2)$ such that $r \xrightarrow{\lambda}_2 r'$ and $(s', r') \in \mathcal{S}$,*
   (c) *if $r \xrightarrow{\lambda}_2 r'$, there exists $s' \in \mathit{Conf}(\mathcal{N}_1)$ such that $s \xrightarrow{\lambda}_1 s'$ and $(s', r') \in \mathcal{S}$.*

The networks $\mathcal{N}_1, \mathcal{N}_2$ are called weakly bisimilar*, if and only if there exists a weak bisimulation $\mathcal{S}$ for $\mathcal{T}(\mathcal{N}_1)$ and $\mathcal{T}(\mathcal{N}_2)$.*

Proposition 3. *Let $\mathcal{N}$ be a well-formed network. Let $\mathcal{K}(\mathcal{N}, \mathcal{EC}_\mathcal{N})$ be the network output by $\mathcal{K}$. Let $r \in \mathit{Conf}(\mathcal{K}(\mathcal{N}, \mathcal{EC}_\mathcal{N}))$ and $s \in \mathit{Conf}(\mathcal{N})$ be two configurations, such that $s = \mathit{reverseQE}(r)$. Then, for every $\ell \in \mathcal{L}(\mathcal{N})$, ℓ is the i-th location of ℓ_s if and only if ℓ is the i-th location of ℓ_r, i.e., if*

$$\forall\, \ell \in \mathcal{L}(\mathcal{N}) \bullet \ell = \ell_{s,i} \Leftrightarrow \ell_{r,i} = \ell.$$

Proposition 4. *Let $\mathcal{N}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ be a well-formed network. Let $s \in \mathit{Conf}(\mathcal{N})$ and $r \in \mathit{Conf}(\mathcal{K}(\mathcal{N}, \mathcal{EC}_\mathcal{N}))$ be two configurations such that $s = \mathit{reverseQE}(r)$. Let $CF_\mathcal{N}$ be the set of configuration formulae over $\mathcal{N}$. Then*

$$\forall\, CF \in CF_\mathcal{N} \bullet s \models CF \iff r \models \Gamma(CF).$$

Theorem 1. *Any well-formed network $\mathcal{N}$ is weakly bisimilar to $\mathcal{K}(\mathcal{N}, \mathcal{EC}_\mathcal{N})$.*

Proof. $\{(\mathit{reverseQE}(r), r) \mid r \in \mathit{CONS}_{\mathcal{K}(\mathcal{N}, \mathcal{EC}_\mathcal{N})}\}$ is a weak bisimulation by Definition 8, and Propositions 2, 3 and 4. □

Corollary 1 (Reachability of Stable Configurations). *Let $s \in \mathcal{SC}_\mathcal{N}$ be a stable configuration of the well-formed network $\mathcal{N}$. s is $\Rightarrow$-reachable in $\mathcal{T}(\mathcal{N})$ if and only if $\mathit{reverseQE}^{-1}(s)$ is reachable in $\mathcal{T}(\mathcal{K}(\mathcal{N}, \mathcal{EC}_\mathcal{N}))$.*

Proof. Theorem 1 and Proposition 2. □

3.4 Handling Impurities and Unstable Configurations

While Corollary 1 allows us to conclude from reachability of a configuration in $\mathcal{N}'$ to the reachability of a corresponding configuration in $\mathcal{N}$, we cannot conclude in the opposite direction. The reason is that $\mathcal{N}'$ misses two things: firstly, computation paths of $\mathcal{N}$ that contain overlapping reset phases are not simulated by any computation path of $\mathcal{N}'$, i.e., $\mathcal{N}'$ would not reach a stable configuration if it were only reachable by computation paths with overlapping reset phases. Secondly, reachability of unstable configurations of $\mathcal{N}$ is not reflected in $\mathcal{N}'$.

In the following we approach the two issues as follows. We argue that, under the additional assumption that the reset edges in $\mathcal{N}$ are pre/post delayed, impure computation paths without timelock can always be reordered into pure ones. Regarding unstable configurations, we firstly observe that – not surprisingly – $\mathcal{N}'$ reflects reachability queries which explicitly ask for stable configurations. In addition, we syntactically characterise the class of local queries, which are reflected by $\mathcal{N}'$ because they cannot distinguish stable and unstable configurations.

Definition 9 (Delayed Edge). *An edge e of a timed automaton $\mathcal{A}$ in network $\mathcal{N}$ is called* delayed *if and only if time must pass before e can be taken, i.e., if*

$$\forall s_0 \xrightarrow{\lambda_1}_{E_1} s_1 \dots s_{n-1} \xrightarrow{\lambda_n}_{E_n} s_n \in \Pi(\mathcal{N}) \bullet e \in E_n$$
$$\implies \exists\, 0 \le j < n \bullet \lambda_j \in \mathit{Time} \setminus \{0\} \wedge \forall\, j \le i < n \bullet E(\mathcal{A}) \cap E_i = \emptyset$$

where we write $s_i \xrightarrow{\lambda_i}_{E_i} s_{i+1}$, $i \in \mathbb{N}^{>0}$, to denote that the transition $s_i \xrightarrow{\lambda_i} s_{i+1}$ is justified by the set of edges E_i; E_i is empty for delay transitions, i.e. if $\lambda_i \in \mathit{Time}$.

Definition 10 (Reset Pre/Post Delay). *Let $\mathcal{N}$ be a well-formed network. We say $\mathcal{EC}_\mathcal{N}$-*reset edges are pre/post delayed in $\mathcal{N}$ *if and only if all edges originating in reset or reset successor locations are delayed, i.e. if*

$$\forall\, e = (\ell, \alpha, \varphi, \vec{r}, \ell') \in E(\mathcal{N}) \bullet \ell \in \mathcal{RL}_\mathcal{N} \cup \mathcal{RL}_\mathcal{N}^+ \implies e \textit{ is delayed.}$$

There are *sufficient* syntactic criteria for an edge $e = (\ell_1, \alpha_1, \varphi_1, \vec{r}_1, \ell_2)$ being delayed. For instance, if $(\ell_0, \alpha_0, \varphi_0, \vec{r}_0, \ell_1)$ is the only incoming edge to ℓ_1 and if $\varphi_0 = (x \ge C \wedge x \le C)$ and $\varphi_1 = (x \ge D \wedge x \le D)$ and $C < D$, then e is delayed. It is also delayed if $(\ell_0, \alpha_0, \varphi_0, \vec{r}_0, \ell_1)$ is the only incoming edge to ℓ_1, $\vec{r}_0$ is resetting x, and $\varphi_1 = (x > 0)$.

Both patterns occur, e.g., in the FSN case-study (cf. Section 4). There, the reset location is entered via an edge following the former pattern, and the edges originating at the reset successor location follow the latter pattern. Thus $\mathcal{EC}_\mathcal{N}$-reset edges are pre/post delayed in FSN.

Proposition 5. *Let $\mathcal{N}(\mathcal{A}_1, \dots, \mathcal{A}_n)$ be a well-formed network where $\mathcal{EC}_\mathcal{N}$-resets are pre/post delayed and let $Y \in \mathcal{EC}_\mathcal{N}$ be a set of quasi-equal clocks.*

Let $s \in \mathit{Conf}(\mathcal{N})$ be a reachable configuration of $\mathcal{N}$ which is not timelocked and not stable wrt. Y. Then all automata in $\mathcal{RES}_Y(\mathcal{N})$ are either in a reset or in a reset successor location in s, i.e.

$$\forall\, Y \in \mathcal{EC}_\mathcal{N}\ \forall\, s \in \mathit{Conf}(\mathcal{N}) \setminus \mathcal{SC}_Y\ \forall\, 1 \le i \le n \bullet$$
$$\mathcal{A}_i \in \mathcal{RES}_Y(\mathcal{N}) \implies \ell_{s,i} \in \mathcal{RL}_Y \cup \mathcal{RL}_Y^+.$$

In order to precisely define the concept of reordering of configurations in computation paths, we introduce the following notion of congruence of configurations. Reordering then means that for each impure computation path there exists a pure computation path over congruent configurations.

Definition 11 (Congruent Modulo Y).
Let $\mathcal{N}(\mathcal{A}_1, \dots, \mathcal{A}_n)$ be a well-formed network with variables $\mathcal{V}$. Two configurations $s_1, s_2 \in \mathit{Conf}(\mathcal{N})$ are called congruent modulo $Y \in \mathcal{EC}_\mathcal{N}$, *denoted by $s_1 \equiv_Y s_2$, if and only if they coincide on values of all variables and if, for each $\mathcal{A} \in \mathcal{N}$, either s_1, s_2 coincide on the location of $\mathcal{A}$ and on the values of clocks from $\mathcal{X}(\mathcal{A})$, or $\mathcal{A}$ is in a reset location in s_1 and in the corresponding reset successor location in s_2, and s_2 has the effect of taking a reset edge, i.e. if*

$$\begin{aligned}&\nu_{s_1}|_\mathcal{V} = \nu_{s_2}|_\mathcal{V} \wedge \big(\forall 1 \leq i \leq n \bullet \ \big(\ell_{1,i} = \ell_{2,i} \wedge \nu_{s_1}|_{\mathcal{X}(\mathcal{A}_i)} = \nu_{s_2}|_{\mathcal{X}(\mathcal{A}_i)}\big)\\ &\quad \vee \textstyle\bigvee_{j=1,2}(\ell_{s_j,i} \in \mathcal{RL}_Y \wedge \ell_{s_{3-j},i} \in \mathcal{RL}_Y^+\\ &\qquad \wedge \exists\, (\ell_{s_j,i}, \alpha, \varphi, \vec{r}, \ell_{s_{3-j},i}) \in E(\mathcal{A}_i) \bullet \nu_{s_j}|_{\mathcal{X}(\mathcal{A}_i)}[\vec{r}] = \nu_{s_{3-j}}|_{\mathcal{X}(\mathcal{A}_i)})\big).\end{aligned}$$

We write $s_1 \equiv_{Y_1,\dots,Y_n} s_2$ if and only if $s_1 \equiv_{Y_1 \cup \dots \cup Y_n} s_2$ for $Y_1, \dots, Y_n \in \mathcal{EC}_\mathcal{N}$.

Lemma 2. *Let $\mathcal{N}(\mathcal{A}_1, \dots, \mathcal{A}_n)$ be a well-formed network where $\mathcal{EC}_\mathcal{N}$-resets are pre/post delayed and let $Y \in \mathcal{EC}_\mathcal{N}$ be a set of quasi-equal clocks.*

Each impure reset sequence which starts in a reachable and ends in a not timelocked configuration can be "reordered" into a reset sequence-δ of Y, i.e.,

$$\forall Y \in \mathcal{EC}_\mathcal{N}\ \forall s_0 \xrightarrow{\lambda_1} \cdots \xrightarrow{\lambda_n} s_n \in \mathcal{RS}_Y^{\mathit{impure}}\ \exists\, r_0, \dots, r_n \in \mathit{Conf}(\mathcal{N}) \bullet r_0 = s_0 \wedge$$

$$r_n = s_n \wedge (\forall 0 \leq i < n \bullet r_i \equiv_Y s_0) \wedge r_0 \xrightarrow{\lambda_2} \dots \xrightarrow{\lambda_n} r_{n-1} \in \mathcal{RS}_Y^{\mathit{pure}} \wedge r_{n-1} \xrightarrow{\lambda_1} r_n.$$

Proof. Proposition 5 and, by Definition 2 (well-formedness), the reset edges wrt. Y are independent from the edges justifying the transition from s_0 to s_1. □

In Figure 3 we provide an illustration of the reordering of reset sequences from Lemma 2. Subfigure a) represents a Y-reset sequence from s_0 to s_4, where filled circles represent stable configurations and stars unstable configurations. The transition marked with (!) represents an impurity. The suffix from s_1 to s_4 is an impure Y-reset sequence. Subfigure b) shows the result after reordering. The sequence from r_0 to r_4 is a pure Y-reset sequence-δ.

Lemma 3. *Let $\mathcal{N}(\mathcal{A}_1, \dots, \mathcal{A}_n)$ be a well-formed network where $\mathcal{EC}_\mathcal{N}$-resets are pre/post delayed and let $Y \in \mathcal{EC}_\mathcal{N}$ be a set of quasi-equal clocks.*

Each reset sequence $s_0 \xrightarrow{\lambda_1} \dots \xrightarrow{\lambda_n} s_n \in \mathcal{RS}_Y$ of Y where s_0 is reachable and s_n is not timelocked can be "reordered" into a computation path

$$r_0 \xrightarrow{\lambda_{k_1}} r_1 \dots r_{n-1} \xrightarrow{\lambda_{k_n}} r_n$$

where $k_1, \dots, k_n$ is a reordering of $1, \dots, n$, $r_0 = s_0$, and where there exists an index $0 \leq j \leq n$ such that $s_0 \equiv_Y r_i$ for all $0 \leq i \leq j$, $s_i \equiv_Y r_{k_i}$ for all $j < i \leq n$, and $r_0 \xrightarrow{\lambda_{k_1}} \dots \xrightarrow{\lambda_{k_j}} r_j \in \mathcal{RS}_Y^{\mathit{pure}}$.

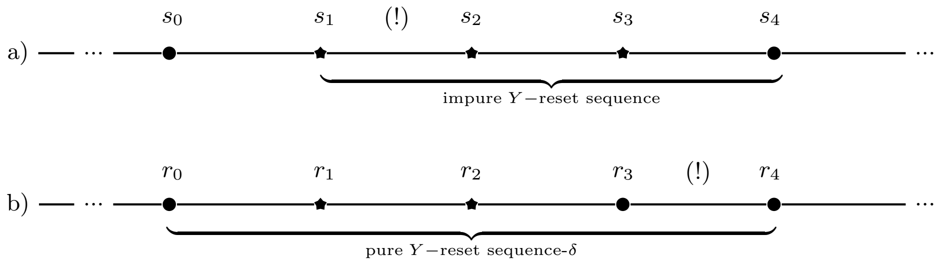


Fig. 3. Reordering of reset sequences.

Proof. Apply Lemma 2 inductively from right to left. That is, if the reset sequence of Y has m impure reset sequences of Y, we start the reordering with the m-th impure reset sequence (the shortest one), and finalize with the first (and longest) impure reset sequence of Y. $\square$

Definition 12 (Stability Query). *A configuration formula CF over the well-formed network $\mathcal{N}$ is called* stability query *iff CF is exactly satisfied in stable configurations of $\mathcal{N}$, i.e., if $\forall\, s \in Conf(\mathcal{N}) \bullet s \models CF \iff s \in \mathcal{SC}_{\mathcal{N}}$.*

Proposition 6. *Let $\mathcal{N}$ be a well-formed network.*
Then $CF = \bigwedge_{Y \in \mathcal{EC}_{\mathcal{N}}} \bigwedge_{x \in Y} (x - rep(x) = 0)$ is a stability query over $\mathcal{N}$.

Note that if reset locations are known for a specific network, a stability query could also be stated in terms of those.

Theorem 2. *Let $\mathcal{N}$ be a well-formed network where $\mathcal{EC}_{\mathcal{N}}$-resets are pre/post delayed, let CF be a configuration formula and 'stable' a stability query over $\mathcal{N}$. Then*

$$\mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}}) \models \exists\Diamond\, \Gamma(CF) \iff \mathcal{N} \models \exists\Diamond(CF) \wedge stable.$$

Proof. Use Lemma 3 to obtain a transition sequence which $\Rightarrow$-reaches the witness configuration, then Corollary 1. $\square$

In the following, we bring together Lemmata 2 and 3 to handle computation paths with arbitrary overlaps of reset phases. This allows us to conclude that $\mathcal{N}'$ reflects queries which cannot distinguish stable and unstable configurations.

Lemma 4. *Let $\mathcal{N}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ be a well-formed network where $\mathcal{EC}_{\mathcal{N}}$-edges are pre/post delayed. Let*

$$\sigma = s_0 \xrightarrow{\lambda_{1,1}} s_{1,1} \ldots \xrightarrow{\lambda_{1,m_1}} s_{1,m_1} \xrightarrow{\lambda_1} s_1 \ldots s_{n-1} \xrightarrow{\lambda_{n,1}} s_{n,1} \ldots \xrightarrow{\lambda_{n,m_n}} s_{n,m_n} \xrightarrow{\lambda_n} s_n$$

be a transition sequence where s_0 is reachable and s_n is not timelocked, where s_i is globally stable, i.e., $s_i \in \mathcal{SC}_{\mathcal{N}}$, $0 \le i \le n$, where for each sub-sequence $\sigma_i = s_{i-1} \xrightarrow{\lambda_{i,1}} s_{i,1} \ldots s_{i,m_i} \xrightarrow{\lambda_i} s_i$, $0 < i \le n$, either s_i has a globally stable *successor, i.e. $m_i = 0$, and $\lambda_i = \tau$, or σ_i is a full pure reset sequence of some $Y \in \mathcal{EC}_{\mathcal{N}}$, and where at s_0 starts a full pure reset sequence, i.e. $m_1 > 0$.*

Then σ can be "reordered" into the transition sequence

$$\sigma' = r_0 \xrightarrow{\hat{\lambda}_1} r_1 \dots r_{K-1} \xrightarrow{\hat{\lambda}_K} r_K$$
$$\xrightarrow{\hat{\lambda}_{K+1,1}} r_{K+1,1} \dots r_{K+1,m_{K+1}} \xrightarrow{\hat{\lambda}_{K+1}} r_{K+1} \dots r_{n-1} \xrightarrow{\hat{\lambda}_{n,1}} r_{n,1} \dots r_{n,p_n} \xrightarrow{\hat{\lambda}_{n,p_n}} r_n$$

where $1 \le i_1 < \dots < i_K \le n$ are the indices of the configurations with globally stable successors, where $0 \le j_1, \dots, j_N \le n$ are the indices such that there is a full pure reset sequence of $Y_k \in \mathcal{EC}_{\mathcal{N}}$ between s_{j_k} and s_{j_k+1} in σ, and where the actions not belonging to full pure reset sequences occur first, followed by the full pure reset sequences of $Y_1, \dots, Y_N$ in this order, i.e. $r_0 = s_0$, $r_k \equiv_{Y_1,\dots,Y_N} s_{i_k}$ for $1 \le k \le K$, and $r_k \xrightarrow{\hat{\lambda}_{k+1,1}} r_{k+1,1} \dots r_{k+1,m_{j_k}} \xrightarrow{\hat{\lambda}_{k+1,m_{j_k}}} r_{k+1} \in \mathcal{RS}^{pure}_{Y_{j_k}}$.

Proof. Similar to Lemma 2 using the independence of reset edges from other edges implied by Definition 2, then induction similar to Lemma 3. □

Definition 13 (Local Query). *Let $\mathcal{N}$ be a well-formed network. A reachability query $\exists\Diamond\, CF$ over $\mathcal{N}$ is called* local query *wrt. $\mathcal{A} \in \mathcal{N}$ if and only if CF is in disjunctive normal form, i.e. $CF = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} BF_{i,j}$, and if each atom $BF_{i,j}$ is of the form $\mathcal{A}.\ell$ with $\ell \in L(\mathcal{A})$, or a constraint φ with $clocks(\varphi) \subseteq \mathcal{X}(\mathcal{A})$.*

Theorem 3. *Let $\mathcal{N}$ be a well-formed network where $\mathcal{EC}_{\mathcal{N}}$-reset edges are pre/post delayed and let $\exists\Diamond\, CF$ be a reachability query which is local to $\mathcal{A} \in \mathcal{N}$.*

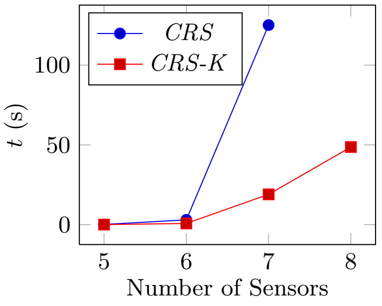
If there is a configuration reachable in $\mathcal{N}$ which satisfies CF and which is not timelocked, then there is a configuration reachable in $\mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}})$ which satisfies $\Gamma(CF)$, i.e. $\mathcal{N} \models_{\neg timelock} \exists\Diamond\, CF \implies \mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}}) \models \exists\Diamond\, \Gamma(CF)$.

Proof. Use Lemma 4 to obtain a stable configuration which satisfies CF, then Theorem 2 applies. □

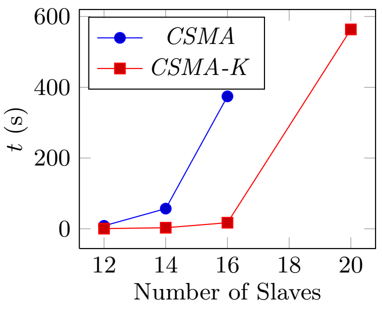
4 Experimental Results

We applied our approach manually to three real world case studies, one of which is an industrial case, and the other two were obtained from the scientific literature. Initially, the six restrictions of well-formedness were motivated by the industrial case, and later generalised to increase the applicability of our approach.
CRS-N is the cascaded ride sharing protocol [13] with N sensors organized in the form of a spanning tree. There exists a sink node that collects data from every sensor. We verified the local query *lessMaxFail*, which states that if a sensor has at least one working communication path, the data sent to the sink node is correctly aggregated. In *lessMaxFail* we only use variables and locations of the sink node.
CSMA-N is the model of the CSMA/CD protocol [14] with N slaves and one

Network	Clk.	States	M. (MB)	t (s)
CRS-5	5	14.9k	17.3	0.2
CRS-5K	1	5.0k	16.2	0.1
CRS-6	6	264.5k	74.7	3.0
CRS-6K	1	64.9k	43.1	0.8
CRS-7	7	7,223.8k	1,986.3	142.1
CRS-7K	1	1,266.4k	693.5	19.2
CRS-8	8	-	-	
CRS-8K	1	2,530.2k	1,543,200	48.7



Network	Clk.	States	M. (MB)	t (s)
CSMA-12	14	688.2k	230.1	8.5
CSMA-12K	1	49.3k	36.3	0.5
CSMA-14	16	3,670.1k	1,257.6	57.0
CSMA-14K	1	229.5k	135.6	3.1
CSMA-16	18	18,874.5k	7,051.7	374.3
CSMA-16K	1	1,048.7k	597.5	17.2
CSMA-20	22	-	-	-
CSMA-20K	1	20,971.7k	10,589.0	563.5



Network	Clk.	States	M. (MB)	t (s)
FSN-6	12	2,149.9k	302.1	176.6
FSN-6K	5	0.9k	16.6	0.1
FSN-8	14	5,084.3k	643.1	729.9
FSN-8K	5	0.9k	17.0	0.1
FSN-10	16	17,474.7k	2,069.4	4057.1
FSN-10K	5	0.9k	17.4	0.1
FSN-60K	5	4,239.4k	454.4	318.4
FSN-80K	5	5,604.4k	611.3	543.9

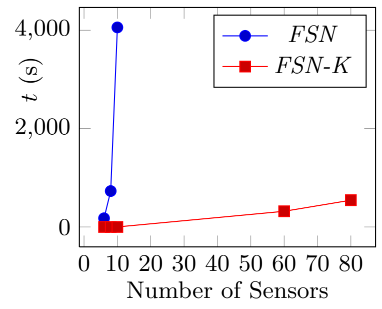


Table 1. Row 'Clk.' gives the number of clocks in the model, 'States' the number of visited states, 'M.' the memory usage in MB, and 't (s)' the runtime in seconds. (Env.: Intel i3, 2.3GHz, 3GB, Ubuntu 11.04, verifyta 4.1.3.4577 with default options.)

master. We verified the local query *noCollision*, which states that no collision occurs when slaves send data to the master. We have represented the occurrence of collisions by a location of the master.
FSN-N is a custom TDMA-based wireless fire alarm system with N sensors [15]. We verified the local query *300seconds*, which states that a sensor malfunction is detected by the central unit (main sensor) in at most 300 seconds. In this query we use a clock and a location from the central unit.

Table 1 gives the figures for verification before and after applying the transformation from Section 3.2, the latter figures are indicated by suffix K at the network name.

5 Conclusion and Future Work

We have presented a transformation approach to mechanically remove verification overhead from networks of timed automata where clocks in the real world are intended to be synchronized. We formally introduced the notion of *quasi-equal* clocks to characterise such clocks. We propose a transformation that goes beyond simple syntactical replacement. We formally prove the correctness of our approach and define a class of timed automata networks and reachability queries for which it is applicable. Although well-formedness imposes a set of restrictions over the networks where we can apply our approach, this is reasonable since the semantics of well-formed networks are preserved after transformation, up to configurations where quasi-equal clocks have different valuations.

Experiments with real-world case studies show the feasibility of reducing clocks in networks of timed automata based on quasi-equal clocks. Significant gains in the computational cost of model checking using *Uppaal* for transformed models are achieved, once eliminated the unnecessary overhead caused when well-formed networks generate intermediate configurations by resetting quasi-equal clocks one by one. We enable an increase in decoupling between modelling as a design and documentation activity, and model optimization for verification. Thus, the approach effectively narrows the gap between readable, maintainable models and model checking efficiency.

In the future, we would like to enlarge the spectrum of networks that can be treated by our approach by investigating relaxations of the well-formedness criteria presented. Additionally, we would like to extend the types of queries supported by the transformation by providing a broader syntax for validable queries beyond simple reachability. Finally, an automatic detection of quasi-equal clocks would increase the mechanisation of our approach.

References

1. R. Alur and D. Dill. A theory of timed automata. *TCS*, 126(2):183–235, 1994.
2. T.S. Rappaport. *Wireless communications: principles and practice*, volume 2. Prentice Hall, 2002.
3. G. Behrmann, A. David, and K. Larsen. A tutorial on uppaal. In M. Bernardo et al., editors, *SFM*, volume 3185 of *LNCS*, pages 200–236. Springer, 2004.
4. V. Braberman et al. Speeding up model checking of timed-models by combining scenario specialization and live component analysis. In *FORMATS*, volume 5813 of *LNCS*, pages 58–72. Springer, 2009.
5. V. Braberman, D. Garbervetsky, and A. Olivero. Improving the verification of timed systems using influence information. In J.-P. Katoen and P. Stevens, editors, *TACAS*, volume 2280 of *LNCS*, pages 21–36. Springer, 2002.
6. M. Bozga et al. State space reduction based on live variables analysis. In A. Cortesi et al., editors, *SAS*, volume 1694 of *LNCS*, pages 164–178. Springer, 1999.
7. D. Lugiez, P. Niebert, and S. Zennou. A partial order semantics approach to the clock explosion problem of timed automata. In K. Jensen and A. Podelski, editors, *TACAS*, volume 2988 of *LNCS*, pages 296–311. Springer, 2004.
8. C. Daws and S. Yovine. Reducing the number of clock variables of timed automata. In *RTSS*, pages 73–81. IEEE, 1996.
9. C. Daws et al. Model checking of real-time reachability properties using abstractions. In *TACAS*, volume 1384 of *LNCS*, pages 313–329. Springer, 1998.
10. M. Muñiz, B. Westphal, and A. Podelski. Timed automata with disjoint activity. In *FORMATS*, LNCS. Springer, 2012.
11. R. Salah, M. Bozga, and O. Maler. Compositional timing analysis. In *EMSOFT*, pages 39–48. ACM, 2009.
12. E.-R. Olderog and H. Dierks. *Real-time systems - formal specification and automatic verification*. Cambridge University Press, 2008.
13. S. Gobriel, S. Khattab, D. Mossé, J. Brustoloni, and R. Melhem. Ridesharing: Fault tolerant aggregation in sensor networks using corrective actions. the 3rd annual. In *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 595–604, 2006.
14. H. Jensen, K. Larsen, and A. Skou. Modelling and analysis of a collision avoidance protocol using SPIN and Uppaal. In *2nd SPIN Workshop*, 1996.
15. D. Dietsch, S. Feo-Arenis, B. Westphal, and A. Podelski. Disambiguation of industrial standards through formalization and graphical languages. In *RE*, pages 265–270, 2011.