Quasi-Equal Clock Reduction: More Networks, More Queries

Christian Herrera, Bernd Westphal, and Andreas Podelski

Albert-Ludwigs-Universität Freiburg, 79110 Freiburg, Germany

Abstract. Quasi-equal clock reduction for networks of timed automata replaces equivalence classes of clocks which are equal except for *unstable phases*, i.e., points in time where these clocks differ on their valuation, by a single representative clock. An existing approach yields significant reductions of the overall verification time but is limited to so-called well-formed networks and *local queries*, i.e., queries which refer to a single timed automaton only. In this work we present two new transformations. The first, for networks of timed automata, summarises unstable phases without losing information under weaker well-formedness assumptions than needed by the existing approach. The second, for queries, now supports the full query language of Uppaal. We demonstrate that the cost of verifying non-local properties is much lower in transformed networks than in their original counterparts with quasi-equal clocks.

1 Introduction

Real-time systems often use distributed architectures and communication protocols to exchange data in real-time. Examples of such protocols are the classes of TDMA-based protocols [1] and EPL-based protocols [2].

Real-time systems can be modelled and verified by using *networks of timed automata* [3]. In [4] a technique that reduces the number of clocks that model the local timing behaviour and synchronisation activity of distributed components is presented in order to reduce the verification runtime of properties in networks of timed automata that fulfill a set of syntactical criteria called *well-formedness*. In systems implementing, e.g., TDMA or EPL protocols this technique eliminates the unnecessary verification overhead caused by the interleaving semantics of timed automata, where the automata reset their clocks one by one at the end of each communication phase. This interleaving induces sets of reachable intermediate configurations which grow exponentially in the number of components in the system. Model checking tools like *Uppaal* [5] explore these configurations even when they are irrelevant for the property being verified. This exploration unnecessarily increases the overall memory consumption and runtime verification of the property.

The notion of *quasi-equal* clocks was presented in [4] to characterise clocks that evolve at the same rate and whose valuation only differs in *unstable phases*,

[1] CONACYT (Mexico) and DAAD (Germany) sponsor the work of the first author.

E. Ábrahám and K. Havelund (Eds.): TACAS 2014, LNCS 8413, pp. 295–309, 2014.

i.e., points in time where these clocks are reset one by one. Sets of quasi-equal clocks induce *equivalence classes* in networks of timed automata.

Although the technique introduced in [4] shows promising results for transformed networks, the technique has two severe drawbacks. Namely, it loses all the information from intermediate configurations and it supports only *local queries*, i.e., properties defined over single timed automaton of well-formed networks. A concrete consequence of these drawbacks can be observed in the system with quasi-equal clocks presented in [6] which implements an EPL protocol. In the transformed model of this system it is not possible to perform the sanity check that a given automaton receives configuration data from other system components right before this automaton resets its quasi-equal clock. The check involves querying information of several automata from intermediate configurations. System properties are quite often expressed in terms of several automata.

To overcome these limitations, in this work we revisit the reduction of quasi-equal clocks in networks of timed automata, and we present an approach based on the following new idea. For each set of quasi-equal clocks we summarise unstable configurations using dedicated locations of automata introduced during network transformation. Queries which explicitly refer to unstable configurations are rewritten to refer to the newly introduced summary location instead. The dedicated summary locations also allow us to support *complex* resetting edges in the original model, i.e. edges with synchronisation of assignments other than clock resets. This allows us to extend the queries that we support as per our new approach which is also a source-to-source transformation, i.e. our approach can be used with a wide range of model-checking tools.

Our approach aims to provide the modelling engineer with a system optimisation technique which allows him to naturally model systems without caring to optimise them for verification. Our contributions are: (1) We now support properties referring to multiple timed automata, in particular properties which query (possibly overlapping) unstable configurations. (2) We enlarge the applicability of our new approach by relaxing the well-formedness criteria presented in [4]. Our approach allows us to prove in a much simpler and more elegant way (without a need for the reordering lemma from [4]) that transformed networks are weakly bisimilar to their original counterparts. We show that properties wrt. an original network are fully preserved in the transformed network, i.e., the transformed network satisfies a transformed property if and only if the original network satisfies the original property. We evaluate our approach on six real world examples, three of them new, where we observe significant improvements in the verification cost of non-local queries compared to the cost of verifying them in the original networks.

The paper is organized as follows. In Section 2, we provide basic definitions. Section 3 introduces the formal definition of *well-formed* networks and presents the algorithm that implements our approach. In Section 4, we formalise the relation of a well-formed network and its transformed network and prove the correctness of our approach. In Section 5, we compare the verification time of six real world examples before and after applying our approach. Section 6 concludes.

Related Work. The methods in [7–9] eliminate clocks by using static analysis over single timed automaton, networks of timed automata and parametric timed automata, respectively. The approaches in [7, 8] reduce the number of clocks in timed automata by detecting *equal* and *active* clocks. Two clocks are equal in a location if both are reset by the same incoming edge, so just one clock for each set of equal clocks is necessary to determine the future behavior of the system. A clock is active at a certain location if this clock appears in the invariant of that location, or in the guard of an outgoing edge of such a location, or another active clock takes its value when taking an outgoing edge. Non-active clocks play no role in the future evolution of the system and therefore can be eliminated. In [9] the same principle of active clocks is used in parametric timed automata. Our benchmarks use at most one clock per component which is always active, hence the equal and active approach is not applicable on them.

The work in [10, 11] uses *observers*, i.e., single components encoding properties of a system, to reduce clocks in systems. For each location of the observer, the technique can deactivate clocks if they do not play a role in the future evolution of this observer. Processing our benchmarks in order to encode properties as per the observers approach may be more expensive than our method (one observer per property), and may not guarantee the preservation of information from intermediate configurations which in the case of our EPL benchmark is needed. In general using observers to characterise non-local queries is not straightforward.

In sequential timed automata [12], one set of quasi-equal clocks is syntactically declared. Those quasi-equal clocks are implicitly reduced by applying the sequential composition operator. The work in [13] avoids the use of shared clocks in single timed automaton by replacing shared clocks with fresh ones if the evolution of these automata does not depend on these clocks. This approach increments the number of clocks (in contrast to ours). Our benchmarks do not use shared clocks. The approach in [14] detects quasi-equal clocks in networks of timed automata. Interestingly, the authors demonstrate the feasibility of their approach in benchmarks that we also use in this paper.

2 Preliminaries

Following the presentation in [15], we here recall the following 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 $\varphi ::= x \sim c \mid x - y \sim c \mid \varphi_1 \wedge \varphi_2$ 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 $clocks(\varphi)$ and $vars(\varphi)$ to respectively denote the set of clocks and variables 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, where $\ell_{ini} \in L$ is the initial location, a finite set B of *actions* comprising the *internal action* τ, finite sets $\mathcal{X}$ and $\mathcal{V}$ of clocks and variables, a mapping

$I : L \to \Phi(\mathcal{X})$, 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 $\mathcal{A}$; *clocks*$(\vec{r})$ and *vars*$(\vec{r})$ to denote the sets of clocks and variables occurring in $\vec{r}$, respectively. We use $\beta(e)$ to denote the set of basic elements (locations, reset vector, etc.) of an edge $e \in E(\mathcal{A})$. We use the following operation of complementation on actions $\overline{\cdot}$, which is defined by $\overline{\alpha!} = \alpha?$, $\overline{\alpha?} = \alpha!$ and $\overline{\tau} = \tau$. A *network* $\mathcal{N}$ *(of timed automata)* consists of a finite set $\mathcal{A}_1, \ldots, \mathcal{A}_N$ of timed automata with pairwise disjoint sets of clocks and pairwise disjoint sets of locations and a set $\mathcal{B}(\mathcal{N}) \subseteq \bigcup_{i=1}^{N} B(\mathcal{A}_i)$ of *broadcast channels*. We write $\mathcal{A} \in \mathcal{N}$ if and only if $\mathcal{A} \in \{\mathcal{A}_1, \ldots, \mathcal{A}_N\}$.

The operational semantics of the network $\mathcal{N}$ is the labelled transition system $\mathcal{T}(\mathcal{N}) = (\mathit{Conf}(\mathcal{N}), \mathit{Time} \cup \{\tau\}, \{\xrightarrow{\lambda} \mid \lambda \in \mathit{Time} \cup \{\tau\}\}, \mathcal{C}_{ini})$. The set of configurations $\mathit{Conf}(\mathcal{N})$ consists of pairs of *location vectors* $\langle \ell_1, \ldots, \ell_N \rangle$ from $\times_{i=1}^{N} L(\mathcal{A}_i)$ and valuations of $\bigcup_{1 \le i \le N} \mathcal{X}(\mathcal{A}_i) \cup \mathcal{V}(\mathcal{A}_i)$ which satisfy the constraint $\bigwedge_{i=1}^{N} I(\ell_i)$. We write $\ell_{s,i}$, $1 \le i \le N$, to denote the location which automaton $\mathcal{A}_i$ assumes in configuration $s = \langle \vec{\ell}_s, \nu_s \rangle$ 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 four kinds of transitions. There is a *delay transition* $\langle \vec{\ell}_s, \nu_s \rangle \xrightarrow{t} \langle \vec{\ell}_{s'}, \nu_{s'} \rangle$ 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 *local transition* $\langle \vec{\ell}_s, \nu_s \rangle \xrightarrow{\tau} \langle \vec{\ell}_{s'}, \nu_{s'} \rangle$ if there is an edge $(\ell_{s,i}, \tau, \varphi, \vec{r}, \ell_{s',i}) \in E(\mathcal{A}_i)$, $1 \le i \le N$, such that $\vec{\ell}_{s'} = \vec{\ell}_s[\ell_{s,i} := \ell_{s',i}]$, $\nu_s \models \varphi$, $\nu_{s'} = \nu_s[\vec{r}]$, and $\nu_{s'} \models I_i(\ell_{s',i})$. There is a *synchronization transition* $\langle \vec{\ell}_s, \nu_s \rangle \xrightarrow{\tau} \langle \vec{\ell}_{s'}, \nu_{s'} \rangle$ if there are $1 \le i, j \le N$, $i \ne j$, 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 $\vec{\ell}_{s'} = \vec{\ell}_s[\ell_{s,i} := \ell_{s',i}][\ell_{s,j} := \ell_{s',j}]$, $\nu_s \models \varphi_i \wedge \varphi_j$, $\nu_{s'} = \nu_s[\vec{r}_i][\vec{r}_j]$, 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 $1 \le i_0 \le N$ 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 $1 \le i_1, \ldots, i_k \le N$, $k \ge 0$, 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* $\langle \vec{\ell}_s, \nu_s \rangle \xrightarrow{\tau} \langle \vec{\ell}_{s'}, \nu_{s'} \rangle$ in $\mathcal{T}(\mathcal{N})$ if $\vec{\ell}_{s'} = \vec{\ell}_s[\ell_{s,i_0} := \ell_{s',i_0}] \cdots [\ell_{s,i_k} := \ell_{s',i_k}]$, $\nu_s \models \bigwedge_{j=0}^{k} \varphi_{i_j}$, $\nu_{s'} = \nu_s[\vec{r}_{i_0}] \cdots [\vec{r}_{i_k}]$, and $\nu_{s'} \models \bigwedge_{j=0}^{k} I_{i_j}(\ell_{s',i_j})$. $\mathcal{C}_{ini} = \{\langle \vec{\ell}_{ini}, \nu_{ini} \rangle\} \cap \mathit{Conf}(\mathcal{N})$, where $\vec{\ell}_{ini} = \langle \ell_{ini,1}, \ldots, \ell_{ini,N} \rangle$ and $\nu_{ini}(x) = 0$ for each $x \in \mathcal{X}(\mathcal{A}_i), 1 \le i \le N$. 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.

The set of *basic formulae* over $\mathcal{N}$ is given by the grammar $\beta ::= \ell \mid \neg\ell \mid \varphi$ where $\ell \in L(\mathcal{A}_i), 1 \le i \le n$, and $\varphi \in \Phi(\mathcal{X}(\mathcal{N}), \mathcal{V}(\mathcal{N}))$. Basic formula β is satisfied by configuration $s \in \mathit{Conf}(\mathcal{N})$ if and only if $\ell_{s,i} = \ell$, $\ell_{s,i} \ne \ell$, or $\nu_s \models \varphi$, resp. 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. We use $\beta(CF)$ to denote the set of basic formulae in CF. $\mathcal{N}$ satisfies $\exists\Diamond\, CF$, denoted by $\mathcal{N} \models \exists\Diamond\, CF$, if and only if there is a configuration s reachable in $\mathcal{T}(\mathcal{N})$ s.t. $s \models CF$.

We recall from [4] the following definitions. Given a network $\mathcal{N}$ 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 $\forall\, s_0 \xrightarrow{\lambda_1} s_1 \xrightarrow{\lambda_2} s_2 \cdots \in \Pi(\mathcal{N})\ \forall\, i \in \mathbb{N}_0 \bullet \nu_{s_i} \models (x = 0 \vee y = 0 \vee x = y)$. 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 denoted by $rep(Y)$. For $x \in Y$, we use $rep(x)$ to denote $rep(Y)$. Given a constraint $\varphi \in \Phi(\mathcal{X}, \mathcal{V})$, we write $\Gamma(\varphi)$ to denote the constraint that is obtained by syntactically replacing each occurrence of a clock $x \in \mathcal{X}$ in φ, by the representative $rep(x)$. Given an automaton $\mathcal{A} \in \mathcal{N}$, a set of clocks $X \subseteq \mathcal{X}(\mathcal{A})$, and a set of variables $V \subseteq \mathcal{V}(\mathcal{A})$, we use $\mathcal{SE}_X(\mathcal{A})$ to denote the set of *simple resetting edges* of $\mathcal{A}$ which reset clocks from X, have action τ, no variables occur in their guards, and do not update any variables, i.e., $\mathcal{SE}_X(\mathcal{A}) = \{(\ell, \alpha, \varphi, \vec{r}, \ell') \in E(\mathcal{A}) \mid clocks(\vec{r}) \cap X \neq \emptyset \wedge \alpha = \tau \wedge vars(\varphi) = \emptyset \wedge vars(\vec{r}) = \emptyset\}$. We use $\mathcal{CE}_X(\mathcal{A})$ to denote the set of *complex resetting edges* of $\mathcal{A}$ which reset clocks from X and have an action different from τ or update some variables, i.e., $\mathcal{CE}_X(\mathcal{A}) = \{(\ell, \alpha, \varphi, \vec{r}, \ell') \in E(\mathcal{A}) \mid clocks(\vec{r}) \cap X \neq \emptyset \wedge (vars(\vec{r}) \cap V \neq \emptyset \ \vee \ \alpha \neq \tau)\}$. We use $\mathcal{LS}_X(\mathcal{A})$ and $\mathcal{LC}_X(\mathcal{A})$ to respectively denote the set of locations (source and destination) of simple and complex resetting edges wrt. X of $\mathcal{A}$. We use $\mathcal{E}_X(\mathcal{A}) = \mathcal{SE}_X(\mathcal{A}) \cup \mathcal{CE}_X(\mathcal{A})$ to denote the set of resetting edges of $\mathcal{A}$ which reset clocks from X, and $\mathcal{RESS}_X(\mathcal{N})$ to denote the set of automata in $\mathcal{N}$ which have a resetting edge, i.e., $\mathcal{RESS}_X(\mathcal{N}) = \{\mathcal{A} \in \mathcal{N} \mid \mathcal{E}_X(\mathcal{A}) \neq \emptyset\}$. A location ℓ (ℓ') is called is called *reset (successor) location* wrt. $Y \in \mathcal{EC}_{\mathcal{N}}$ in $\mathcal{N}$ if and only if there is a resetting edge in $\mathcal{SE}_Y(\mathcal{A}) \cup \mathcal{CE}_Y(\mathcal{A})$ from (to) ℓ (ℓ'). We use $\mathcal{RL}_Y(\mathcal{N})$ ($\mathcal{RL}^+_Y(\mathcal{N})$) to denote the set of reset (successor) locations wrt. Y in $\mathcal{N}$ and we set $\mathcal{RL}_{\mathcal{EC}_{\mathcal{N}}}(\mathcal{N}) := \bigcup_{Y \in \mathcal{EC}_{\mathcal{N}}} \mathcal{RL}_Y(\mathcal{N})$ and similarly $\mathcal{RL}^+_{\mathcal{EC}_{\mathcal{N}}}(\mathcal{N})$.

A configuration $s \in Conf(\mathcal{N})$ is called *stable* wrt. $Y \in \mathcal{EC}_{\mathcal{N}}$ if and only if all clocks in Y have the same value in s, i.e., if $\forall\, x \in Y \bullet \nu_s(x) = \nu_s(rep(x))$. We use $\mathcal{SC}^Y_{\mathcal{N}}$ to denote the set of all configurations that are stable wrt. Y and $\mathcal{SC}_{\mathcal{N}}$ to denote the set $\bigcap_{Y \in \mathcal{EC}_{\mathcal{N}}} \mathcal{SC}^Y_{\mathcal{N}}$ of *globally stable* configurations of $\mathcal{N}$. Configurations not in $\mathcal{SC}_{\mathcal{N}}$ are called *unstable*. 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 $\forall\, s_0 \xrightarrow{\lambda_1}_{E_1} s_1 \ldots s_{n-1} \xrightarrow{\lambda_n}_{E_n} s_n \in \Pi(\mathcal{N}) \bullet e \in E_n \implies \exists\, 0 \leq j < n \bullet \lambda_j \in Time \setminus \{0\} \wedge \forall\, j \leq i < n \bullet E(\mathcal{A}) \cap E_i = \emptyset$. 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 $\lambda_i \in Time$. We say *$\mathcal{EC}_{\mathcal{N}}$-reset edges are pre/post delayed in network $\mathcal{N}$* if and only if all edges originating in reset or reset successor locations are delayed, i.e., if $\forall\, e = (\ell, \alpha, \varphi, \vec{r}, \ell') \in E(\mathcal{N}) \bullet \ell \in \mathcal{RL}_{\mathcal{EC}_{\mathcal{N}}}(\mathcal{N}) \cup \mathcal{RL}^+_{\mathcal{EC}_{\mathcal{N}}}(\mathcal{N}) \implies e$ is delayed.

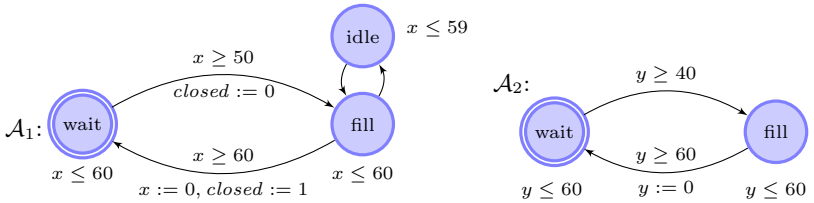


Fig. 1. Model of a chemical plant controller with quasi-equal clocks

3 Reducing Clocks in Networks of Timed Automata

Consider the following motivating example of 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 the other for at most 20 seconds. Figure 1 shows a model of this system in form of the network $\mathcal{N}_1$ which is composed of automata $\mathcal{A}_1$ and $\mathcal{A}_2$ with respective clocks x and y. Additionally, automaton $\mathcal{A}_1$ has the boolean variable *closed* that is set to true, i.e., $closed := 1$, when $\mathcal{A}_1$ has filled its container. Both automata start in a waiting phase at the point in time 0 and after filling the containers they wait for the next round. Both clocks x and y, together with the variable *closed* are respectively reset and updated at the point in time 60. Yet, in the strict interleaving semantics of networks of timed automata, these resets occur one after the other.

According to the definition of quasi-equal clocks, clocks x and y are quasi-equal because their valuations are only different from each other when they are reset at the point in time 60. Now consider verifying in $\mathcal{N}_1$, whether the container of automaton $\mathcal{A}_1$ is closed before automaton $\mathcal{A}_2$ resets its clock. A query that states this property is $\exists\Diamond\, \phi$ with configuration formula $\phi : closed = 1 \wedge y \geq 60$. Clearly in $\mathcal{N}_1$, this query is satisfied only when clocks x and y have different valuations, i.e., in unstable configurations. Property $\exists\Diamond\, \phi$ cannot be treated by the approach in [4] since that approach supports only local queries, i.e., queries which refer to properties of at most one automaton. The approach in [4] completely eliminates all unstable configurations, those where quasi-equal clocks have different valuations, since no alternative representation of them was proposed for transformed models. Furthermore, $\mathcal{N}_1$ does not satisfy the well-formedness criteria of [4] because the resetting edge also assigns a variable.

3.1 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 and preserves all possible queries. For simplicity, we impose a set of syntactical criteria called well-formedness rules over networks of timed automata.

Definition 1 (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 resets at most one clock $x \in Y$, in the constraint (guard) of this edge there is a clause of the form $x \geq C_Y$, and the source location of that edge has an invariant $x \leq C_Y$ for some constant $C_Y > 0$, i.e.,*

$$\begin{aligned}\exists\, C_Y \in \mathbb{N}^{>0}\ \forall\, \mathcal{A} \in \mathcal{N}\ \forall\, (\ell, \alpha, \varphi, \vec{r}, \ell') \in \mathcal{E}_Y(\mathcal{A})\ \exists\, x \in Y, \varphi_0 \in \beta(\varphi) \bullet \\ \mathit{clocks}(\vec{r}) = \{x\} \wedge I(\ell) = x \leq C_Y \wedge \varphi_0 = x \geq C_Y \\ \wedge\, \forall\, \varphi_1 \in \beta(\varphi) \bullet \mathit{clocks}(\varphi_1) \neq \emptyset \implies \varphi_1 = \varphi_0.\end{aligned}$$

(R2) *Resetting edges do not coincide on source locations.*

$$\forall\, \mathcal{A} \in \mathcal{N}\ \forall\, (\ell_i, \alpha_i, \varphi_i, \vec{r}_i, \ell_i') \neq (\ell_j, \alpha_j, \varphi_j, \vec{r}_j, \ell_j') \in \mathcal{E}_Y(\mathcal{A})\ \bullet\ \ell_i \neq \ell_j.$$

(R3) *For pairs of edges that synchronise on some channel $a \in B(\mathcal{N})$, either all edges reset a clock from Y, or none of these edges resets a clock from Y, or the output $a!$ is in one edge resetting a clock from Y, and the inputs $a?$ are in the edges of automata which do not reset clocks from Y, i.e.,*

$$\begin{aligned}\forall\, \mathcal{A}_1 \neq \mathcal{A}_2 \in \mathcal{N}\ \forall\, e_i = (\ell_i, \alpha_i, \varphi_i, \vec{r}_i, \ell_i') \in E(\mathcal{A}_i), i = 1, 2, \alpha_1 = \overline{\alpha_2} \bullet \\ (e_1 \notin \mathcal{E}_Y(\mathcal{A}_1) \wedge e_2 \notin \mathcal{E}_Y(\mathcal{A}_2)) \vee (e_1 \in \mathcal{E}_Y(\mathcal{A}_1) \wedge e_2 \in \mathcal{E}_Y(\mathcal{A}_2)) \\ \vee\, (\exists\, i \in \{1, 2\}, a \in B(\mathcal{N}) \bullet \alpha_i = a! \wedge e_i \in \mathcal{E}_Y(\mathcal{A}_i) \wedge \mathcal{A}_{3-i} \notin \mathcal{RES}_Y(\mathcal{N})).\end{aligned}$$

(R4) *At most one clock from Y occurs in the guard of any edge, i.e.,*

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

The transformation algorithm presented here which was developed in order to support all queries and in particular those interested in unstable configurations, allows us to easily relax the syntactical restrictions presented in [4]. The relaxations done in this work are the following. By restriction *R1*, now looped edges or those edges from initial locations can reset clocks from $Y \in \mathcal{EC}_\mathcal{N}$ as well as update variables, and we now allow the guard of such edges to conjoin integer constraints over variables. By *R2* we now allow more edges from a reset location (but still only one resetting edge from it). By *R3*, we now allow a resetting edge to have a limited but still useful synchronisation. The new well-formedness criteria are less restrictive then they look on first sight. They allow us to extend the applicability of our new approach by treating three new case studies. Note that the network in Figure 1 satisfies the new well-formedness criteria.

In the following we describe the transformation algorithm $\mathcal{K}$. It works with two given inputs, a well-formed network $\mathcal{N}$ and a set of equivalence classes $\mathcal{EC}_\mathcal{N} = \{Y_1, \ldots, Y_n\}$ of quasi-equal clocks. The output of $\mathcal{K}$ is the transformed network $\mathcal{N}' = \{\mathcal{A}_1', \ldots, \mathcal{A}_n'\} \cup \{\mathcal{R}_Y \mid Y \in \mathcal{EC}_\mathcal{N}\}$ with broadcast channels $\mathcal{B}(\mathcal{N}') = \mathcal{B}(\mathcal{N}) \cup \{\mathit{reset}_Y \mid Y \in \mathcal{EC}_\mathcal{N}\}$. The automata $\mathcal{A}_i'$ are obtained by applying repeatedly (in any order) the algorithm $\mathcal{K}_0$ to $\mathcal{A}_i$ for each equivalence class in $\mathcal{EC}_\mathcal{N}$, i.e., $\mathcal{A}_i' = \mathcal{K}_0(\ldots \mathcal{K}_0(\mathcal{A}_i, Y_1), \ldots Y_n)$. Algorithm $\mathcal{K}_0$ is defined as follows.

$$\mathcal{K}_0(\mathcal{A}, Y) = \begin{cases} \mathcal{A} & \text{, if } \mathcal{A} \notin \mathcal{RES}_Y(\mathcal{N}), \\ (L', B', \mathcal{X}', \mathcal{V}', I', E', \ell_{ini}') & \text{, otherwise} \end{cases}$$

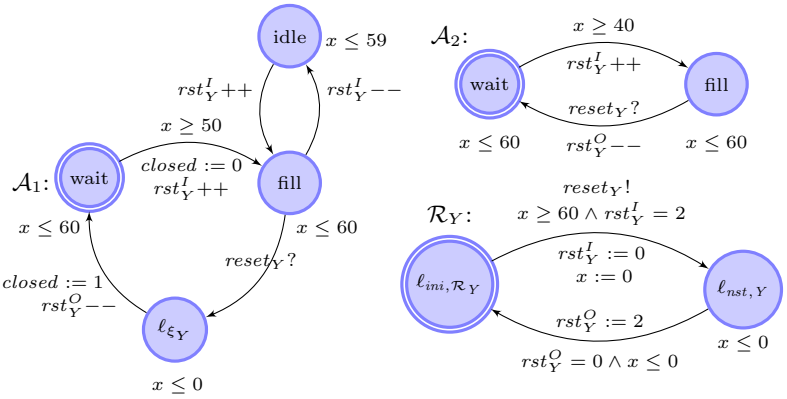


Fig. 2. The model the chemical plant controller after applying $\mathcal{K}$

where per equivalence class

- intermediate locations for each complex resetting edge are added, $L' = L(\mathcal{A}) \cup \Xi_Y(\mathcal{A})$ with $\Xi_Y(\mathcal{A}) = \{\ell_{\xi_{Y,e}} \mid e \in \mathcal{CE}_Y(\mathcal{A})\}$, $\ell'_{ini} = \ell_{ini}(\mathcal{A})$,
- the broadcast channel $reset_Y$ is added, $B' = B(\mathcal{A}) \cup \{reset_Y\}$; clocks except for each representative clock are deleted, $\mathcal{X}' = (\mathcal{X}(\mathcal{A}) \setminus Y) \cup \{rep(Y)\}$; and rst variables are added, $\mathcal{V}' = \mathcal{V} \cup \{rst^I_Y, rst^O_Y\}$,
- quasi-equal clocks occurring in invariants are replaced by the respective representative clock, $I'(\ell) = I(\ell)[y/rep(y) \mid y \in Y]$ for $\ell \in L(\mathcal{A})$; and a zero delay invariant is added to each intermediate location $I'(\ell) = rep(y) \leq 0$ for $\ell \in \Xi_Y(\mathcal{A})$; On non-resetting edges each quasi-equal clock is replaced by the respective representative clock; the input $reset_Y?$ is placed and the reset of quasi-equal clocks is removed from simple edges; and intermediate locations and reset successor locations are linked, respectively,

$$
\begin{aligned}
E' = \{&(\ell, \alpha, \varphi[y/rep(y) \mid y \in Y], \vec{r}; \rho_e, \ell') \mid e = (\ell, \alpha, \varphi, \vec{r}, \ell') \in E(\mathcal{A}) \setminus \mathcal{E}_Y(\mathcal{A})\} \\
&\cup \{(\ell, reset_Y?, \varphi[y \sim c/true \mid y \in Y], \vec{r}[y := 0/\epsilon \mid y \in Y]; \rho_e, \ell') \mid \\
&\qquad e = (\ell, \tau, \varphi, \vec{r}, \ell') \in \mathcal{SE}_Y(\mathcal{A})\} \\
&\cup \{(\ell_{\xi_{Y,e}}, \alpha, \varphi[y \sim c/true \mid y \in Y], \vec{r}[y := 0/\epsilon \mid y \in Y]; \rho_e, \ell'), \\
&\qquad (\ell, reset_Y?, true, \epsilon, \ell_{\xi_{Y,e}}) \mid e = (\ell, \alpha, \varphi, \vec{r}, \ell') \in \mathcal{CE}_Y(\mathcal{A})\}
\end{aligned}
$$

where the reset sequence $\rho_e = r_1; r_2; r_3$ depends on the edge e as follows:

- $r_1 = rst^I_Y := rst^I_Y + 1$ if e is to a reset location in $\mathcal{RL}_Y(\mathcal{N})$, and $r_1 = \epsilon$ otherwise,
- $r_2 = rst^I_Y := rst^I_Y - 1$ if e is from a reset location in $\mathcal{RL}_Y(\mathcal{N})$ and $e \notin \mathcal{E}_Y(\mathcal{A})$, and $r_2 = \epsilon$ otherwise, and
- $r_3 = rst^O_Y := rst^O_Y - 1$ if $e \in \mathcal{E}_Y(\mathcal{A})$, and $r_3 = \epsilon$ otherwise.

The *resetter* $\mathcal{R}_Y$ for equivalence class Y is the timed automaton

$$(\{\ell_{ini,\mathcal{R}_Y}, \ell_{nst,Y}\}, \{reset_Y\}, \{rep(Y)\}, \{rst^I_Y := iL_Y, rst^O_Y := n_Y\}, I, E, \ell_{ini,\mathcal{R}_Y}).$$

It initializes the variable rst^I_Y to $iL_Y := |\{\mathcal{A} \in \mathcal{N} \mid \ell_{ini,\mathcal{A}} \in \mathcal{RL}_Y(\mathcal{N})\}|$, i.e. the number of automata whose initial location is a reset location of Y, and rst^O_Y to $n_Y := |\mathcal{RES}_Y(\mathcal{N})|$, i.e. the number of automata that reset the clocks of Y. There are two locations with the invariants $I(\ell_{ini,\mathcal{R}_Y}) = \mathit{true}$ and $I(\ell_{nst,Y}) = \mathit{rep}(Y) \leq 0$. The set of edges E consists of

$$(\ell_{ini,\mathcal{R}_Y}, \mathit{reset}_Y!, (rst^I_Y = n_Y \wedge \mathit{rep}(Y) \geq C_Y), rst^I_Y := 0; \mathit{rep}(Y) := 0, \ell_{nst,Y})$$
$$\text{and } (\ell_{nst,Y}, \tau, (rst^O_Y = 0 \wedge \mathit{rep}(Y) \leq 0), rst^O_Y := n_Y, \ell_{ini,\mathcal{R}_Y})$$

where C_Y is the time at which the clocks in Y are reset (cf. *R1*).

Example 1. Applying $\mathcal{K}$ to $\mathcal{N}_1$ from Figure 1 yields network $\mathcal{N}'_1$ (cf. Figure 2). Similar to the algorithm in [4], only the representative clock of each equivalence class remains. All guards and invariants with quasi-equal clocks are re-written to refer to the representative clock, and the reset operation is delegated to the resetter. The variable rst^I_Y together with well-formedness enforces a *blocking* multicast synchronisation between resetter and the automata in $\mathcal{RES}_Y(\mathcal{N})$.

In order to support non-local queries, and in particular queries for possibly overlapping unstable configurations, the approach presented here introduces one resetter *per* equivalence class with *two* locations each. The location $\ell_{nst,Y}$ represents all unstable configuration wrt. Y. To support complex edges, and thus non-trivial behaviour during unstable phases, complex edges are basically split into two. The first one synchronises with the resetter and the second one carries out the actions of the original complex edge. As long as the second edge has not been taken, the system is unstable. The variable rst^O_Y is introduced to indicate to automaton $\mathcal{R}_Y$ when this unstability finishes. Its value gives the number of automata which still need to take their reset edge in the current unstable phase.

In $\mathcal{N}'_1$, we have thereby eliminated the interleaving induced by resetting the clocks x and y in $\mathcal{N}_1$, but the interleaving wrt. variable updates during reset of quasi-equal clocks is preserved by splitting the complex edge into two. Note that in transformed networks, configurations with the locations $\ell_{nst,Y_1}, \ldots, \ell_{nst,Y_n}$, where $1 < n$, reflect overlapping unstable phases, i.e. instability wrt. multiple equivalence classes at one point in time.

The following function Ω syntactically transforms properties over a well-formed network $\mathcal{N}$ into properties over $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}})$. Function Ω treats queries for source or destination locations of resetting edges special and outputs an equivalent property which can be verified in $\mathcal{N}'$.

For instance, consider a simple resetting edge $e \in \mathcal{SE}_Y(\mathcal{A})$ of some $\mathcal{A} \in \mathcal{N}$. The source location ℓ of e can be assumed in $\mathcal{N}$ in different configurations: either the reset time is not yet reached, or the reset time is reached but $\mathcal{A}$ did not reset yet, while other automata in $\mathcal{RES}_Y(\mathcal{N})$ may have reset their clocks already. In $\mathcal{N}'$, all edges resulting from simple edges fire at once on the broadcast synchronisation, so all source locations are left together. Because the resetter moves to $\ell_{nst,Y}$, a configuration of $\mathcal{N}'$ which assumes location $\ell_{nst,Y}$ represents all similar configurations of $\mathcal{N}$ where all simple edges are in their source or

destination location. Thus the location ℓ is reachable in $\mathcal{N}$ if and only if (i) $\mathcal{N}'$ reaches $\ell_{nst,Y}$, or (ii) if ℓ is reached while being stable, i.e., not being in $\ell_{nst,Y}$.

A similar reasoning is applied to properties querying elements of a complex resetting edge wrt. Y, but instead of using $\ell_{nst,Y}$ we use the intermediate location $\ell_{\xi_{Y,e}}$ from $\mathcal{N}'$, since this location represents unstability before updating any variable that occurs in a complex edge.

In this sense, configurations involving location $\ell_{nst,Y}$ summarise unstable phases of $\mathcal{N}$. Assuming $\ell_{nst,Y}$ in $\mathcal{N}'$ represents both cases for a simple edge, that it has already been taken or not, and that the clock x reset by this edge is still C_Y or already 0. Although involving two choices, there are essentially two cases (not four): having taken the reset edge and being unstable implies that, x is 0 and some other clocks are still C_Y, or x is still C_Y and some other clocks are already 0. To this end, we introduce fresh existentially quantified variables $\tilde{\ell}$ and $\tilde{x}$ in Ω_0 and conjoin it with a consistency conjunction. By *R1*, we only need to consider 0 and C_Y as values of $\tilde{x}$, thus the existential quantification can be rewritten into a big disjunction, and hence is a proper query.

Definition 2 (Function Ω). *Let $Y \in \mathcal{EC}_{\mathcal{N}}$ be sets of clocks of a well-formed network $\mathcal{N}$ and let $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}})$. Let C_Y be the constant described in restriction* R1*. Let $\ell_{nst,Y}$ be the unique non initial location of $\mathcal{R}_Y$, the resetter automaton wrt. Y in $\mathcal{N}'$. Let β be a basic formula over $\mathcal{N}$. Then the function Ω is defined as follows where $\mathcal{L}_Y = \mathcal{LS}_Y(\mathcal{N}) \cup (\mathcal{LC}_Y(\mathcal{N}) \cap \mathcal{RL}_Y(\mathcal{N}))$:*

$$\Omega_0(\beta) = \begin{cases} (\ell \wedge \neg\ell_{nst,Y}) \vee (\ell_{nst,Y} \wedge \tilde{\ell}) & \text{, if } \beta = \ell, \ell \in \mathcal{L}_Y. \\ (\neg\ell \wedge \neg\ell_{nst,Y}) \vee (\ell_{nst,Y} \wedge \neg\tilde{\ell}) & \text{, if } \beta = \neg\ell, \ell \in \mathcal{L}_Y. \\ \big(\Gamma(\varphi) \wedge \neg\ell_{nst,Y}\big) \vee \big(\ell_{nst,Y} \wedge \tilde{\varphi}\big) & \text{, if } \beta = \varphi, \tilde{\varphi} = \varphi[x/\tilde{x} \mid x \in \mathcal{X}(\mathcal{N})]. \\ \beta & \text{, otherwise} \end{cases}$$

$$\Omega(CF) = \exists\, \tilde{x}_1, .., \tilde{x}_k\ \exists\, \tilde{\ell}_1, .., \tilde{\ell}_m \bullet \Omega_0(CF) \wedge \bigwedge_{\substack{(\ell,\alpha,\varphi,\vec{r},\ell') \\ \in \mathcal{CE}_Y(\mathcal{A}), \\ \ell_i = \ell}} (\tilde{\ell}_i \implies \ell_{\xi_{Y,e}}) \wedge \bigwedge_{\substack{1 \le i \ne j \le m, \\ 1 \le p \le n \\ \ell_i, \ell_j \in L_p,}} \neg(\tilde{\ell}_i \wedge \tilde{\ell}_j)$$

$$\wedge \bigwedge_{\substack{1 \le i \le k, 1 \le j \le m, \\ x_j \in \mathcal{X}_p \cap Y, 1 \le p \le n, \\ \ell_i \in L_p \cap (\mathcal{RL}_Y(\mathcal{N}) \setminus \mathcal{RL}_Y^+(\mathcal{N}))}} (\tilde{\ell}_i \implies \tilde{x}_j = C_Y) \wedge \bigwedge_{\substack{1 \le i \le k, 1 \le j \le m, \\ x_j \in \mathcal{X}_p \cap Y, 1 \le p \le n, \\ \ell_i \in L_p \cap (\mathcal{RL}_Y^+(\mathcal{N}) \setminus \mathcal{RL}_Y(\mathcal{N}))}} (\tilde{\ell}_i \implies \tilde{x}_j = 0) \wedge \bigwedge_{\substack{(\ell,\alpha,\varphi,\vec{r},\ell') \\ \in \mathcal{SE}_Y(\mathcal{A}), \\ \ell_i \in \{\ell,\ell'\}}} (\tilde{\ell}_i \implies \ell')$$

For example, for $\Omega(\phi)$ we obtain, after some simplifications given that $\mathcal{A}_2$ has only simple resetting edges, the following transformed formula:

$$\exists\, \tilde{x} \in \{0, C_Y\} \bullet \mathit{closed} = 1 \wedge ((x \le 60 \wedge \neg\ell_{nst,Y}) \vee (\ell_{nst,Y} \wedge \tilde{x} \ge 60)).$$

4 Formal Relation of a Well-formed Network and Its Transformed Network

In order to prove our approach correct we establish a weak bisimulation relation between a well-formed network and its respective transformed network. To this end, we firstly extend the notion of (un)stability to $\mathcal{N}'$ as follows.

Definition 3 (Stable Configuration of $\mathcal{N}'$). *Let $\mathcal{N}$ be a network and let $Y \in \mathcal{EC}_{\mathcal{N}}$ be a set of quasi-equal clocks. Let $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}})$.*

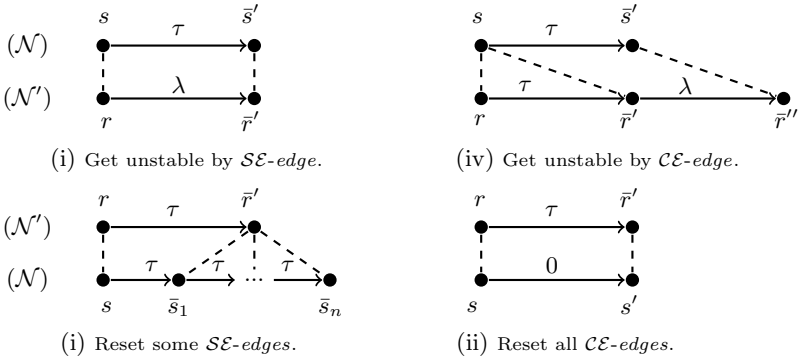
A configuration $r \in Conf(\mathcal{N}')$ is called stable wrt. *Y if and only if the initial location $\ell_{ini,\mathcal{R}_Y}$ of resetter $\mathcal{R}_Y \in \mathcal{N}'$ occurs in r, i.e., if $r \models \ell_{ini,\mathcal{R}_Y}$. We use $\mathcal{SC}^Y_{\mathcal{N}'}$ to denote the set of all configurations that are stable wrt. Y and $\mathcal{SC}_{\mathcal{N}'}$ to denote the set $\bigcap_{Y \in \mathcal{EC}_{\mathcal{N}}} \mathcal{SC}^Y_{\mathcal{N}'}$ of* globally stable *configurations of $\mathcal{N}'$. We call a configuration $r \notin \mathcal{SC}_{\mathcal{N}'}$* unstable.

We recall that configurations induced when each clock from $Y \in \mathcal{EC}_{\mathcal{N}}$ is reset in well-formed networks $\mathcal{N}$, are summarised in transformed networks $\mathcal{N}'$ in configurations where the $\ell_{nst,\mathcal{R}_Y}$-location occurs together with the valuations of rst^I_Y and rst^O_Y reflecting these resets. Hence with the valuations from rst^I_Y and rst^O_Y we unfold information summarised in these configurations from $\mathcal{N}'$.

Lemma 1 (Weak Bisimulation)
Any well-formed network $\mathcal{N}$ where $\mathcal{EC}_{\mathcal{N}}$-reset edges are pre/post delayed, is weakly bisimilar *to $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}})$, i.e., there is a* weak bisimulation relation *$\mathcal{S} \subseteq Conf(\mathcal{N}) \times Conf(\mathcal{N}')$ such that*

1. $\forall\, s \in \mathcal{C}_{ini}(\mathcal{N})\ \exists\, r \bullet (s,r) \in \mathcal{S}$ *and* $\forall\, r \in \mathcal{C}_{ini}(\mathcal{N}')\ \exists\, s \bullet (s,r) \in \mathcal{S}$.
2. *For all config. formulae CF over $\mathcal{N}$,* $\forall\, (s,r) \in \mathcal{S} \bullet s \models CF \implies r \models \Omega(CF)$ *and* $\forall\, r \in CONS_{\mathcal{EC}_{\mathcal{N}}} \bullet r \models \Omega(CF) \implies \exists\, s \in Conf(\mathcal{N}) \bullet (s,r) \in \mathcal{S} \wedge s \models CF$.
3. *For all $(s,r) \in \mathcal{S}$,*
   (a) *if $s \xrightarrow{\lambda} s'$ with*
      i. *$s \in \mathcal{SC}^Y_{\mathcal{N}}$, $s' \notin \mathcal{SC}^Y_{\mathcal{N}}$, where $Y \in \mathcal{EC}_{\mathcal{N}}$, and justified by a simple resetting edge, there is r' such that $r \xrightarrow{\lambda} r'$ and $(s',r') \in \mathcal{S}$.*
      ii. *$s, s' \notin \mathcal{SC}^Y_{\mathcal{N}}$, or $s \notin \mathcal{SC}^Y_{\mathcal{N}}$ and $s' \in \mathcal{SC}^Y_{\mathcal{N}}$, where $Y \in \mathcal{EC}_{\mathcal{N}}$, and justified by a simple resetting edge, then $r \xrightarrow{0} r$ and $(s',r) \in \mathcal{S}$.*
      iii. *$s, s' \in \mathcal{SC}^Y_{\mathcal{N}}$, or $s \in \mathcal{SC}^Y_{\mathcal{N}}$ and $s' \notin \mathcal{SC}^Y_{\mathcal{N}}$, where $Y \in \mathcal{EC}_{\mathcal{N}}$, and justified by the set $CE_Y \subseteq \mathcal{CE}_Y(\mathcal{N})$ of complex resetting edges wrt. Y, then there exist r', r'' such that $r \xrightarrow{\tau} r' \xrightarrow{\lambda} r''$ and $(s,r'), (s',r'') \in \mathcal{S}$.*
      iv. *$s \notin \mathcal{SC}^Y_{\mathcal{N}}$, $s' \in \mathcal{SC}^Y_{\mathcal{N}}$, where $Y \in \mathcal{EC}_{\mathcal{N}}$, and justified by $CE_Y \subseteq \mathcal{CE}_Y(\mathcal{N})$, there is r' s.t. $r \xrightarrow{\lambda} r'$ and $(s',r') \in \mathcal{S}$.*
      v. *$s, s' \in \mathcal{SC}^Y_{\mathcal{N}}$, $\ell_r = \ell_{nst,\mathcal{R}_Y}$ for some $Y \in \mathcal{EC}_{\mathcal{N}}$, and $\lambda = d > 0$, there exist r', r'' such that $r \xrightarrow{\tau} r' \xrightarrow{\lambda} r''$ and $(s,r'), (s',r'') \in \mathcal{S}$.*
      vi. *Otherwise there exists r' such that $r \xrightarrow{\lambda} r'$ and $(s',r') \in \mathcal{S}$.*
   (b) *if $r \xrightarrow{\lambda} r'$ with*
      i. *$r \in \mathcal{SC}^Y_{\mathcal{N}'}, r' \notin \mathcal{SC}^Y_{\mathcal{N}'}$, where $Y \in \mathcal{EC}_{\mathcal{N}}$, $\nu_{r'}(rst^O_Y) < N$, where $N = \nu_r(rst^O_Y)$, there exist $s_1, \ldots, s_n$ where $n = N - \nu_{r'}(rst^O_Y)$, such that $s \xrightarrow{\tau} s_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} s_n$ and $(s_i, r') \in \mathcal{S}$, $1 \le i \le n$.*
      ii. *$r \in \mathcal{SC}^Y_{\mathcal{N}'}, r' \notin \mathcal{SC}^Y_{\mathcal{N}'}$, $\nu_{r'}(rst^O_Y) = \nu_r(rst^O_Y)$, where $Y \in \mathcal{EC}_{\mathcal{N}}$, then $s \xrightarrow{0} s$ and $(s,r') \in \mathcal{S}$.*
      iii. *$\ell_r = \ell_{nst,\mathcal{R}_Y}$, $\ell_{r'} \neq \ell_{nst,\mathcal{R}_Y}$, $Y \in \mathcal{EC}_{\mathcal{N}}$, then $s \xrightarrow{0} s$ and $(s,r') \in \mathcal{S}$.*
      iv. *Otherwise there exists s' such that $s \xrightarrow{\lambda} s'$ and $(s',r') \in \mathcal{S}$.*

Proof (sketch). Let $\mathcal{N}$ be a well-formed network and let $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}})$. For each $Y \in \mathcal{EC}_{\mathcal{N}}$ use the following six conditions to obtain a weak bisimulation



(i) Get unstable by $\mathcal{SE}$-*edge*. (iv) Get unstable by $\mathcal{CE}$-*edge*.

(i) Reset some $\mathcal{SE}$-*edges*. (ii) Reset all $\mathcal{CE}$-*edges*.

Fig. 3. Some involved weak bisimulations cases between the transition system (TS) of a well-formed network $\mathcal{N}$ and the TS of the network $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}})$. Dots with the legend $(\bar{s})s$ and $(\bar{r})r$ represent (unstable) stable configurations of $\mathcal{N}$ and $\mathcal{N}'$, respectively. Arrows represent transitions between configurations of the same TS. Configurations s and r are in transition simulation if they are linked by a dotted line.

relation $\mathcal{S}$ between configurations $s \in \mathit{Conf}(\mathcal{N})$ and *consistent* configurations $r \in \mathit{CONS}_{\mathcal{EC}_{\mathcal{N}}}$ of $\mathcal{N}'$, i.e., configurations whose valuations of variables rst^I_Y match the number of reset locations wrt. Y assumed in r, and if r is unstable, variables rst^O_Y match the number of intermediate locations assumed in r, otherwise match $|Y|$. (1) any pair (s, r) matches the valuations of variables and non-quasi-equal clocks. (2) automata from $\mathcal{N}$ and $\mathcal{N}'$ which do not reset clocks from Y must coincide on locations. (3) relate stable configurations from $\mathcal{N}$ where all quasi-equal clocks from Y have the valuation C_Y and the variable $rst^I_Y = |Y|$, to either stable configurations in $\mathcal{N}'$ where the clock $\mathit{rep}(Y) = C_Y$, or to unstable configurations which reflect the synchronisation on the channel reset_Y. (4) take unstable configurations from $\mathcal{N}$ and $\mathcal{N}'$ and relate them if they show the effect of taking the same complex resetting edge. (5) relate unstable configurations wrt. Y in $\mathcal{N}$ to the unstable configuration in $\mathcal{N}'$ whose variable $rst^O_Y = 0$ and the location $\ell_{r,\mathcal{R}_Y} = \ell_{nst,\mathcal{R}_Y}$. (6) relate stable configurations from $\mathcal{N}$ which show the effect of resetting each clock from Y, to the unstable configuration of $\mathcal{N}'$ which shows the same effect, i.e., the valuation of the variable $rst^O_Y = 0$. This last restriction allows $\mathcal{N}'$ to make the return transition to stability.

During stability phases there is a strong bisimulation (one-to-one) between the networks $\mathcal{N}$ and $\mathcal{N}'$. Only during unstability phases there is a weak bisimulation (one-to-many) in both directions. There are cases (reset of simple edges) where $\mathcal{N}$ simulates one step of $\mathcal{N}'$ with multiple steps, and cases (reset of complex edges) where $\mathcal{N}'$ simulates one step of $\mathcal{N}$ with multiple steps. Figure 3 shows some involved simulation steps between unstable phases in $\mathcal{N}$ and $\mathcal{N}'$. □

Theorem 1. *Let $\mathcal{N}$ be a well-formed network where $\mathcal{EC}_{\mathcal{N}}$-resets are pre/post delayed. Let CF be a configuration formula over $\mathcal{N}$. Then*

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

Table 1. Column XX-$N(K)$ gives the figures for case study XX with N sensors (and $\mathcal{K}$ applied). 'C' gives the number of clocks in the model, 'kStates' the number of visited states times 10^3, 'M' memory usage in MB, and '$t(s)$' verification time in seconds. FraTTA transformed each of our benchmarks in at most 5 seconds.
(Env.: Intel i3, 2.3GHz, 3GB, Ubuntu 11.04, verifyta 4.1.3.4577 with default options.)

Network	C	kStates	M	$t(s)$	Network	C	kStates	M	$t(s)$
EP-21	21	3,145.7	507.4	498.4	*FS-8*	14	5,217.7	181.4	758.7
EP-21K	1	5,242.9	624.9	167.5	*FS-8K*	5	1,081.9	41.6	32.8
EP-22	22	6,291.5	1,025.9	1,291.7	*FS-10*	16	17,951.3	568.2	4,271.2
EP-22K	1	10,485.8	1,269.6	358.3	*FS-10K*	5	1,215.0	44.1	39.0
EP-23	23	-	-	-	*FS-11*	17	-	-	-
EP-23K	1	18,431.8	2,490.2	646.8	*FS-126K*	5	9,512.3	300.5	1,529.8
TT-6	7	2,986.0	114.9	38.1	*CD-14*	16	7,078.1	591.8	1,384.3
TT-6K	1	2,759.6	106.7	30.6	*CD-14K*	2	442	52.5	42.3
TT-7	8	16,839.9	611.5	276.4	*CD-16*	18	13,441.1	1,996.4	3,806.3
TT-7K	1	15,746.7	577.3	262.7	*CD-16K*	2	2,031.9	240.9	389.0
TT-8	9	-	-	-	*CD-17*	19	-	-	-
TT-8K	1	66,265.9	2,367.7	1,227.6	*CD-18K*	2	9,1975.3	1,142.8	2,206.1
LS-7	18	553.3	74.5	22.7	*CR-6*	6	264.5	18.7	2.9
LS-7K	6	605.3	83.2	11.5	*CR-6K*	1	129.4	18.2	1.5
LS-9	22	8,897.6	1,283.5	686.6	*CR-7*	7	7,223.7	496.5	136.3
LS-9K	6	9,106.3	1,417.9	238.8	*CR-7K*	1	2,530.1	342.4	42.5
LS-11	26	-	-	-	*CR-8*	8	-	-	-
LS-11K	6	7,694.0	2,188.2	460.6	*CR-8K*	1	5,057.6	785.0	109.4

Proof. Use Lemma 1 and induction over the lenght of paths to show that *CF* holds in $\mathcal{N}$ if and only if $\Omega(CF)$ holds in $\mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}})$. □

5 Experimental Results

We applied our approach to six industrial case studies using *FraTTA* [16], our implementation of $\mathcal{K}$. The three case studies *FS* [17], *CR* [18], *CD* [19] are from the class of TDMA protocols and appear in [4]. The relaxed well-formedness criteria (compared to [4]) allowed us to include the three new case studies *EP* [6], *TT* [20], *LS* [21]. We verified non-local queries as proposed by the respective authors of these case studies. None of these queries could be verified with the approach presented in [4]. Our motivating case study is inspired by the network from [6] which use the Ethernet PowerLink protocol in Alstom Power Control Systems. The network consists of N sensors and one master. The sensors exchange information with the master in two phases, the first is isochronous and the second asynchronous. An error occurs if a sensor fails to update the configuration data as sent by the master in the beginning of the isochronous phase. Specifically, each sensor should update its internal data before the master has reset its clock. The query $\mathit{configData} := \forall\,\Box\ \mathcal{A}.\mathit{configData} = 1 \wedge \mathcal{A}.x = 0 \wedge \mathcal{M}.y > 0$, where $\mathcal{A}$ is a sensor and $\mathcal{M}$ is the master, x and y are quasi-equal clocks from

the same equivalence class, and *configData* is a boolean variable set to true by the edge that resets x when $\mathcal{A}$ has successfully updated its configuration data, checks whether this network is free from errors as explained before. Note that query *configData* is non-local and in addition refers to an unstable configuration. We refer the reader to [17–21] for more information on the other case studies.

Table 1 gives figures for the verification of the non-local queries in instances of the original and the transformed model. The rows without results indicate the smallest instances for which we did not obtain results within 24 hours. For all examples except for TT, we achieved significant reductions in verification time. The quasi-equal clocks in the TT model are reset by a broadcast transition so there is no interleaving of resets in the original model. Still, verification of the transformed TT instances *including* transformation time is faster than verification of the original ones. Regarding memory consumption, note that verification of the K-models of EP and LS takes slightly more memory than verification of the original counterparts. We argue that this is due to all resetting edges being *complex* in these two networks. Thus, our transformation preserves the full interleaving of clock resets and the whole set of unstable locations whose size is exponential in the number of participating automata, and it adds the transitions to and from location ℓ_{nst}. The shown reduction of the verification time is due to a smaller size of the DBMs that Uppaal uses to represent zones [22] and whose size grows quadratically in the number of clocks. If the resetting edges are *simple* (as in FS, CD, and CR), our transformation removes all those unstable configurations.

6 Conclusion

Our new technique reduces the verification time of networks of timed automata with quasi-equal clocks. It represents all clocks from an equivalence class by one representative, and it eliminates those configurations induced by automata that reset quasi-equal clocks one by one. All interleaving transitions which are induced by simple resetting edges are replaced by just two transitions in the transformed networks. We use *nst*-locations to summarise unstable configurations. This allows us to also reduce the runtime of non-local properties or properties explicitly querying unstable phases. With variables $rstI$, $rstO$ we unfold information summarised in *nst*-locations, and together with a careful syntactical transformation of properties, we reflect all properties of original networks in transformed ones. Our new approach fixes the two severe drawbacks of [4], which only supports local queries and whose strong well-formedness conditions rules out many industrial case-studies. Our experiments show the feasibility and potential of the new approach, even if some interleavings are preserved and only the number of clocks is reduced.

References

1. Rappaport, T.S.: Wireless communications, vol. 2. Prentice Hall (2002)
2. Cena, G., Seno, L., et al.: Performance analysis of ethernet powerlink networks for distributed control and automation systems. CSI 31(3), 566–572 (2009)

3. Alur, R., Dill, D.: A theory of timed automata. TCS 126(2), 183–235 (1994)
4. Herrera, C., Westphal, B., Feo-Arenis, S., Muñiz, M., Podelski, A.: Reducing quasi-equal clocks in networks of timed automata. In: Jurdziński, M., Ničković, D. (eds.) FORMATS 2012. LNCS, vol. 7595, pp. 155–170. Springer, Heidelberg (2012)
5. Behrmann, G., David, A., Larsen, K.G.: A tutorial on UPPAAL. In: Bernardo, M., Corradini, F. (eds.) SFM-RT 2004. LNCS, vol. 3185, pp. 200–236. Springer, Heidelberg (2004)
6. Limal, S., Potier, S., Denis, B., Lesage, J.: Formal verification of redundant media extension of ethernet powerlink. In: ETFA, pp. 1045–1052. IEEE (2007)
7. Daws, C., Yovine, S.: Reducing the number of clock variables of timed automata. In: RTSS, pp. 73–81. IEEE (1996)
8. Daws, C., Tripakis, S.: Model checking of real-time reachability properties using abstractions. In: Steffen, B. (ed.) TACAS 1998. LNCS, vol. 1384, pp. 313–329. Springer, Heidelberg (1998)
9. André, É.: Dynamic clock elimination in parametric timed automata. In: FSFMA, OASICS, pp. 18–31, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2013)
10. Braberman, V., Garbervestky, D., Kicillof, N., Monteverde, D., Olivero, A.: Speeding up model checking of timed-models by combining scenario specialization and live component analysis. In: Ouaknine, J., Vaandrager, F.W. (eds.) FORMATS 2009. LNCS, vol. 5813, pp. 58–72. Springer, Heidelberg (2009)
11. Braberman, V.A., Garbervetsky, D., Olivero, A.: Improving the verification of timed systems using influence information. In: Katoen, J.-P., Stevens, P. (eds.) TACAS 2002. LNCS, vol. 2280, pp. 21–36. Springer, Heidelberg (2002)
12. Muñiz, M., Westphal, B., Podelski, A.: Timed automata with disjoint activity. In: Jurdziński, M., Ničković, D. (eds.) FORMATS 2012. LNCS, vol. 7595, pp. 188–203. Springer, Heidelberg (2012)
13. Balaguer, S., Chatain, T.: Avoiding shared clocks in networks of timed automata. In: Koutny, M., Ulidowski, I. (eds.) CONCUR 2012. LNCS, vol. 7454, pp. 100–114. Springer, Heidelberg (2012)
14. Muñiz, M., Westphal, B., Podelski, A.: Detecting quasi-equal clocks in timed automata. In: Braberman, V., Fribourg, L. (eds.) FORMATS 2013. LNCS, vol. 8053, pp. 198–212. Springer, Heidelberg (2013)
15. Olderog, E.-R., Dierks, H.: Real-time systems - formal specification and automatic verification. Cambridge University Press (2008)
16. Fitriani, K.: FraTTA: Framework for transformation of timed automata, Master Team Project, Albert-Ludwigs-Universität Freiburg (2013)
17. Dietsch, D., Feo-Arenis, S., et al.: Disambiguation of industrial standards through formalization and graphical languages. In: RE, pp. 265–270. IEEE (2011)
18. Gobriel, S., Khattab, S., Mossé, D., et al.: RideSharing: Fault tolerant aggregation in sensor networks using corrective actions. In: SECON, pp. 595–604. IEEE (2006)
19. Jensen, H., Larsen, K., Skou, A.: Modelling and analysis of a collision avoidance protocol using SPIN and Uppaal. In: 2nd SPIN Workshop (1996)
20. Steiner, W., Elmenreich, W.: Automatic recovery of the TTP/A sensor/actuator network. In: WISES, pp. 25–37, Vienna University of Technology (2003)
21. Kordy, P., Langerak, R., et al.: Re-verification of a lip synchronization protocol using robust reachability. In: FMA. EPTCS, vol. 20, pp. 49–62 (2009)
22. Bengtsson, J., Yi, W.: Timed automata: Semantics, algorithms and tools. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) ACPN 2003. LNCS, vol. 3098, pp. 87–124. Springer, Heidelberg (2004)