Quasi-equal Clock Reduction: Eliminating Assumptions on Networks

Christian Herrera[1] and Bernd Westphal

Albert-Ludwigs-Universität Freiburg, 79110 Freiburg, Germany

Abstract. Quasi-equal clock reduction for networks of timed automata replaces clocks in equivalence classes by representative clocks. An existing approach which reduces quasi-equal clocks and does not constrain the support of properties of networks, yields significant reductions of the overall verification time of properties. However, this approach requires strong assumptions on networks in order to soundly apply the reduction of clocks. In this work we propose a transformation which does not require assumptions on networks, and does not constrain the support of properties of networks. We demonstrate that the cost of verifying properties is much lower in transformed networks than in their original counterparts with quasi-equal clocks.

1 Introduction

Real-time systems can be modelled and verified using *networks of timed automata* [1]. Often the local timing behaviour and synchronisation activity of distributed components in a network are modelled by (local) clocks. If the valuations of those clocks only differ in points in time in which those clocks are reset, then we call them *quasi-equal* clocks [2]. Quasi-equality of clocks induces *equivalence classes* in networks of timed automata.

In systems using quasi-equal clocks, those clocks are often reset one by one at a given point in time. For instance, in TDMA [3] protocols, automata interleave when they reset quasi-equal clocks at the end of each communication phase. This interleaving induces sets of reachable intermediate configurations. These sets grow exponentially in the number of quasi-equal clocks in equivalence classes. Model checking tools like *Uppaal* [4] explore those configurations when a property being verified queries them. However, this exploration may also increase the memory consumption and verification time for properties which do not query those intermediate configurations.

In [5] a technique that reduces the number of quasi-equal clocks is presented. This technique can yield savings in the overall memory consumption and verification time of properties in transformed networks for two reasons. The first reason is that by using only the representative clock of each equivalence class, we reduce the size of the *Difference Bound Matrices (DBMs)* that Uppaal uses to represent *zones* [6]. The size of a DBM is determined by the size of the set of clocks

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

in a given system. A more compact DBM can be more efficiently represented, stored and accessed in memory. The second reason is that Uppaal explores less configurations when checking a property, since we eliminate intermediate configurations if and only if those configurations are reachable by taking edges which exclusively reset quasi-equal clocks.

In order to soundly reduce quasi-equal clocks, strong assumptions are required for networks in [5]. One of those assumptions states that there must be a delay greater than zero time units in the origin location of an edge resetting a quasi-equal clock. In networks which model the Foundation Fieldbus Data Link Layer protocol (FDLL) [7], there are edges resetting quasi-equal clocks which can be taken at any time, even without delaying at the origin locations of those edges. Hence we cannot transform those networks by using the technique from [5].

In this work we revisit the quasi-equal clock reduction approach, and we present an approach that does not require assumptions on networks with quasi-equal clocks. We now enforce a strong distinction of edges that reset quasi-equal clocks. Namely, edges which exclusively reset one quasi-equal clock, have no synchronisation with other edges, and are taken after a delay greater than zero time units, are called *simple edges*. All other edges resetting quasi-equal clocks are called *complex edges*. In general our approach allows us to yield savings related to having a more compact DBM in memory. Furthermore, for simple edges we also provide savings related to eliminating intermediate configurations.

As in [5] we delegate the reset of representative clocks to newly added *resetter* automata. Now automata with transformed simple or complex edges indicate resetters when to execute that reset which is part of a mechanism that, reduces the number of configurations reachable by taking transformed simple edges, and preserves all configurations reachable by taking transformed complex edges. Similar to [5], for those configurations that we eliminate, properties are restated in terms of an existing dedicated location in each resetter which encodes all the information about those configurations.

In this work we extend the applicability of the quasi-equal clock reduction approach by eliminating the assumptions on networks presented in [5]. Now we can transform any network with sets of quasi-equal clocks. Our new approach allows us to include three new case studies *FB* [7], *TA* [8] and *PG* [9], which cannot be transformed by the technique from [5]. In general, the cost of verifying properties is much lower in networks transformed with our new approach than in their original counterparts with quasi-equal clocks. Furthermore, our new approach allows us to prove in a much simpler and more elegant way that transformed networks are weakly bisimilar to their original counterparts. We show that properties wrt. an original network are fully reflected 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 nine real world examples, three of them new.

Related Work. The methods in [10, 11] detect and reduce *equal* and *active* clocks by using static analysis over single timed automata and networks of timed automata, respectively. 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. 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 [12, 13] 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 as required for our benchmark [14].

In sequential timed automata [15], one set of quasi-equal clocks is syntactically declared. Those quasi-equal clocks are implicitly reduced by applying the sequential composition operator. The approach in [16] 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 [17], 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 $\mathit{clocks}(\varphi)$ and $\mathit{vars}(\varphi)$ 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 use $L^u, L^c \subseteq L$ to denote the set of urgent and committed locations in L, respectively. 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; $\mathit{clocks}(\vec{r})$ and $\mathit{vars}(\vec{r})$ to denote the sets of clocks and variables occurring in $\vec{r}$, respectively.

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 sets $\mathcal{B}^r, \mathcal{B}^b, \mathcal{B}^u \subseteq \bigcup_{i=1}^{N} B(\mathcal{A}_i)$ of *rendez-vous, broadcast* and *urgent channels*, respectively. 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 $\bigcup_{1 \leq i \leq N} \mathcal{X}(\mathcal{A}_i) \cup \mathcal{V}(\mathcal{A}_i)$ 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 $s = \langle \vec{\ell}_s, \nu_s \rangle$ 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* $\langle \vec{\ell}_s, \nu_s \rangle \xrightarrow{t} \langle \vec{\ell}_s, \nu_s + t \rangle$ if for all $t' \in [0, t]$ and for all $1 \leq i \neq j \leq N$, $\nu_s + t' \models \bigwedge_{k=1}^{N} I_k(\ell_{s,k})$ (where $\nu_s + t'$ denotes the valuation obtained from ν_s by time shift t'), and (1) $\ell_{s,i} \notin L^u \cup L^c$, and (2) $\nu_s + t' \not\models \varphi(e)$, $e \in E(\mathcal{A})$, such that $\alpha(e) = b!$, $b \in \mathcal{B}^u \cap \mathcal{B}^b$ and $\ell_{s,i} = \ell(e)$; and (3) $\nu_s + t' \not\models \varphi(e_i) \wedge \varphi(e_j)$, with $e_i \in E(\mathcal{A}_i)$ and $e_j \in E(\mathcal{A}_j)$, such that $\alpha(e_i) = u!, \alpha(e_j) = u?, u \in \mathcal{B}^u(\mathcal{N}) \cap \mathcal{B}^r(\mathcal{N})$, $\ell_{s,i} = \ell(e_i)$ and $\ell_{s,j} = \ell(e_j)$. 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 $(\ell_{s,i}, \tau, \varphi, \vec{r}, \ell_{s',i}) \in E(\mathcal{A}_i)$, $1 \leq i \leq N$, 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 $\nu_{s'} \models I_i(\ell_{s',i})$, and if $\ell_{s,k} \in L^c$ for some $1 \leq k \leq N$ then $\ell_{s,i} \in L^c$. There is a *synchronization transition* $\langle \vec{\ell}_s, \nu_s \rangle \xrightarrow{\tau} \langle \vec{\ell}_{s'}, \nu_{s'} \rangle$ if there are $1 \leq i \neq j \leq N$, 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 $\nu_{s'} \models I_i(\ell_{s',i}) \wedge I_j(\ell_{s',j})$, and if $\ell_{s,k} \in L^c$ for some $1 \leq k \leq N$ then $\ell_{s,i} \in L^c$ or $\ell_{s,j} \in L^c$. 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 a *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 $\nu_{s'} \models \bigwedge_{j=0}^{k} I_{i_j}(\ell_{s',i_j})$, and if $\ell_{s,\hat{k}} \in L^c$ for some $\hat{k}, \bar{k} \in \{i_0, i_1, \ldots, i_k\}$ then $\ell_{s,\bar{k}} \in L^c$. $\mathcal{C}_{ini} = \{\langle \vec{\ell}_{ini}, \nu_{ini} \rangle\} \cap \mathit{Conf}(\mathcal{N})$, where $\vec{\ell}_{ini} = \langle \ell_{ini,1}, \ldots, \ell_{ini,N} \rangle$, $\nu_{ini}(x) = 0$ for each $x \in \mathcal{X}(\mathcal{A}_i)$, and $1 \leq i \leq N$. A finite or infinite sequence $\sigma = s_0 \xrightarrow{\lambda_1} s_1 \xrightarrow{\lambda_2} s_2 \cdots$ of configurations 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 finite 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 $\beta ::= \ell \mid \varphi$ where $\ell \in L(\mathcal{A}_i)$, $1 \leq i \leq n$, and $\varphi \in \Phi(\mathcal{X}, \mathcal{V})$. Basic formula β is satisfied by configuration $s \in \mathit{Conf}(\mathcal{N})$ if and only if $\ell_{s,i} = \ell$, or $\nu_s \models \varphi$, respectively. A *reachability query* over $\mathcal{N}$ is $\exists\Diamond\, \mathit{CF}$ where CF is a *configuration formula* over $\mathcal{N}$, i.e. any logical connector of basic formulae. We use $\beta(\mathit{CF})$ to denote the set of basic formulae in CF. $\mathcal{N}$ 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 $\mathcal{T}(\mathcal{N})$ s.t. $s \models \mathit{CF}$.

We recall from [2] 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 $rep(Y)$. An edge e of a timed automaton $\mathcal{A}$ in network $\mathcal{N}$ is called *resetting edge* if and only if e resets a clock, i.e. if $\exists\, e = (\ell, \alpha, \varphi, \vec{r}, \ell') \in E(\mathcal{A}) \bullet clocks(\vec{r}) \neq \emptyset$. A location ℓ (ℓ') 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 $E(\mathcal{N})$ from (to) ℓ (ℓ'). 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 $\mathcal{SC}_{\mathcal{N}}^{Y}$ to denote the set of all configurations that are stable wrt. Y. A configuration not in $\mathcal{SC}_{\mathcal{N}}^{Y}$ is called *unstable* wrt. Y.

3 Reducing Clocks in Networks of Timed Automata

Consider the following motivating example of the network $\mathcal{N}_1$ depicted in Figure 1. Network $\mathcal{N}_1$ consists of automata $\mathcal{A}_1$ and $\mathcal{A}_2$ with respective clocks x and y, rendez-vous channel c, and global variable a. Note that after delaying ten time units at their respective initial locations, automata $\mathcal{A}_1$ and $\mathcal{A}_2$ interleave by taking their simple edges which exclusively reset their respective clocks. This interleaving induces intermediate configurations where clocks x and y differ on their valuations. Automata $\mathcal{A}_1$ and $\mathcal{A}_2$ after a delay of five time units at locations ℓ_1 and ℓ_5 interleave once again by taking their complex edges which reset their respective clocks together with updates of the variable a. Note that automata $\mathcal{A}_1$ and $\mathcal{A}_2$ can reset once again their respective clocks and transit simultaneously to their respective locations ℓ_3 and ℓ_7 at any time, even without delaying at locations ℓ_2 and ℓ_6. Since the valuations of clocks x and y only differ at the point in time when they are reset, therefore they are quasi-equal clocks.

Note that network $\mathcal{N}_1$ cannot be transformed by the approach from [5], since by that approach: (a) the outgoing edges of locations ℓ_2 and ℓ_6 do not fulfill the syntactical pattern of an edge resetting quasi-equal clocks, i.e. there are no clock constraints that guard those edges, and the origin locations of those edges have no invariants; and (b) there must be a delay greater than zero time units at the origin location of any edge resetting a quasi-equal clock.

In this paper we present an approach which: (1) does not require that a network with quasi-equal clocks fulfill certain syntactical assumptions; (2) does not require any delay before resetting quasi-equal clocks; (3) does not restrict the point in time at which quasi-equal clocks are reset; (4) eliminates configurations reachable by taking simple edges, e.g. configurations reachable by taking the simple edges of locations ℓ_0 and ℓ_4 and, (5) preserves configurations reachable by taking complex edges, e.g. configurations reachable by taking the complex edges of locations ℓ_1 and ℓ_5.

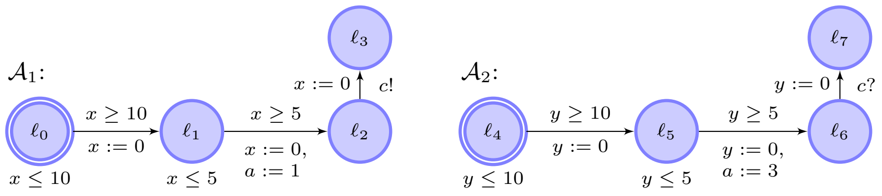


Fig. 1. Network $\mathcal{N}_1$ with quasi-equal clocks x and y.

In the following we introduce two definitions that allow us to classify edges that reset quasi-equal clocks into *simple* and *complex* edges. Intuitively, an edge resetting quasi-equal clocks is called simple if that edge resets exclusively one clock, does not synchronise with other edges and time must pass before taking that edge, otherwise is called complex.

Definition 1 (Pre-delayed edge). *An edge $e = (\ell, \alpha, \varphi, \vec{r}, \ell') \in E(\mathcal{N})$ is called* pre-delayed *if and only if time must pass in ℓ before e is taken, i.e. if $\Pi(\mathcal{N}) = \Pi(Z(\mathcal{N}))$, where Z is a transformation that adds a fresh clock x in $\mathcal{N}$, and for each edge incoming to ℓ, a reset $x := 0$, and to the guard φ the condition $x > 0$. We use $\mathcal{DE}_{\mathcal{N}}$ to denote the set of pre-delayed edges of $\mathcal{N}$.*

There are *sufficient* syntactic criteria for an edge $e = (\ell_1, \alpha_1, \varphi_1, \vec{r}_1, \ell_2)$ being pre-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 $\varphi_0 = (x \geq C)$ and $\varphi_1 = (x \geq D)$ and $C < D$, then e is pre-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 FS case-study (cf. Section 5). 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 resetting edges are pre-delayed in FS.

Definition 2 (Simple and Complex (Resetting) Edges). *Let edge $e = (\ell, \alpha, \varphi, \vec{r}, \ell')$ be an edge which resets at least one quasi-equal clock, i.e. $clocks(\vec{r}) \cap Y \neq \emptyset$ for some $Y \in \mathcal{EC}_{\mathcal{N}}$. Edge e is called* simple edge *if and only if*

– *it is of the form $(\ell, \tau, x \geq c, \langle x := 0 \rangle, \ell')$ for some local clock $x \in \mathcal{X}(\mathcal{A})$,*
– *the invariant of ℓ is $x \leq c$,*
– *it is pre-delayed, i.e. $e \in \mathcal{DE}_{\mathcal{N}}$,*
– *it is the only outgoing edge of ℓ, i.e. $\forall\, e_1 = (\ell_1, \alpha_1, \varphi_1, \vec{r}_1, \ell_1') \in E(\mathcal{A}) \bullet \ell_1 = \ell \implies e = e_1$, and it is the only incoming edge into ℓ', i.e. $\forall\, e_1 = (\ell_1, \alpha_1, \varphi_1, \vec{r}_1, \ell_1') \in E(\mathcal{A}) \bullet \ell_1' = \ell' \implies e = e_1$.*

Otherwise, e is called complex edge. *We use $\mathcal{SE}_Y(\mathcal{A})$ to denote the set of simple edges of $\mathcal{A}$ using a clock from $Y \in \mathcal{EC}_{\mathcal{N}}$. We use $\mathcal{CE}_Y(\mathcal{A})$ to denote the set of those complex edges which reset at least one clock from $Y \in \mathcal{EC}_{\mathcal{N}}$.*

To avoid really special corner cases in the following we assume that time is not stopped at origin and destination locations of simple edges. We use $\mathcal{RES}_Y(\mathcal{N})$

to denote the set of automata in $\mathcal{N}$ which have simple or complex resetting edges wrt. $Y \in \mathcal{EC}_{\mathcal{N}}$, i.e. $\mathcal{RES}_Y(\mathcal{N}) = \{\mathcal{A} \in \mathcal{N} \mid \mathcal{SE}_Y(\mathcal{A}) \cup \mathcal{CE}_Y(\mathcal{A}) \neq \emptyset\}$. For simplicity we could classify each edge resetting a clock $x \in Y$, with $Y \in \mathcal{EC}_{\mathcal{N}}$, as complex. However, with the above definition we distinguish complex edges from simple ones, and we provide a transformation for networks where interleavings of complex edges are preserved, while interleavings of simple edges are eliminated.

3.1 Algorithm for Transformation of Networks

In the following we present our transformation algorithm. It takes two inputs, a network $\mathcal{N}$ and a set of equivalence classes of quasi-equal clocks $\mathcal{EC}_{\mathcal{N}}$ (which can be obtained by [16]), and outputs a transformed network $\mathcal{N}'$ which from each $Y \in \mathcal{EC}_{\mathcal{N}}$ uses only the representative clock $rep(Y)$ and reflects the truth-value of all queries on $\mathcal{N}$.

Recall that we distinguish stable and unstable configurations per equivalence class Y. In stable configurations, all clocks from Y have the same value, thus in particular the same value as the representative $rep(Y)$. In unstable configurations, some clocks from Y have been reset and some not yet, so each clock from Y either has the value 0 or the same value as $rep(Y)$. We use a fresh boolean token t_x for each quasi-equal clock x to encode clock values in unstable configurations. Configurations in $\mathcal{N}'$ where token t_x is *true* encode configurations of $\mathcal{N}$ where $x = rep(x)$ holds, while the token being *false* encodes that x has already been reset at the current point in time and thus has value 0. Function $\mathit{\Gamma}$ (cf. Definition 3) transforms guards and conditions based on this encoding.

Definition 3 (Function $\mathit{\Gamma}$). *Let $\mathcal{N}$ be a network. Let $Y, W \in \mathcal{EC}_{\mathcal{N}}$ be sets of quasi-equal clocks of $\mathcal{N}$, $x \in Y$ and $y \in W$ clocks, and z a non-quasi-equal clock. Let $t_x, t_y \notin \mathcal{V}(\mathcal{N})$ be boolean variables. Given a clock constraint φ_{clk}, we define:*

$$\mathit{\Gamma}_0(\varphi_{clk}) := \begin{cases} ((rep(x) \sim c \ \wedge \ t_x) \vee (0 \sim c \ \wedge \ \neg t_x)) & \text{, if } \varphi_{clk} = x \sim c, \\ \big((rep(x) - rep(y) \sim c \ \wedge \ t_x \ \wedge \ t_y) & \text{, if } \varphi_{clk} = x - y \sim c, \\ \quad \vee \ (0 - rep(y) \sim c \ \wedge \ \neg t_x \ \wedge \ t_y) & \\ \quad \vee \ (rep(x) - 0 \sim c \ \wedge \ t_x \ \wedge \ \neg t_y) & \\ \quad \vee \ (0 \sim c \ \wedge \ \neg t_x \ \wedge \ \neg t_y)\big) & \\ \big((rep(x) - z \sim c \ \wedge \ t_x) & \text{, if } \varphi_{clk} = x - z \sim c, \\ \quad \vee \ (0 - z \sim c \ \wedge \ \neg t_x)\big) & \\ \mathit{\Gamma}_0(\varphi_1) \wedge \mathit{\Gamma}_0(\varphi_2) & \text{, if } \varphi_{clk} = \varphi_1 \wedge \varphi_2. \end{cases}$$

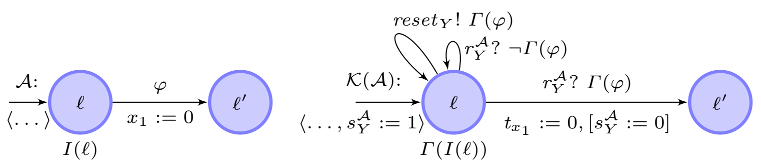
We obtain the transformation $\mathit{\Gamma}$ by setting $\mathit{\Gamma}(\varphi_{clk} \wedge \psi_{int}) := \mathit{\Gamma}_0(\varphi_{clk}) \wedge \psi_{int}$.

Following [5], we add to transformed networks a resetter automaton $\mathcal{R}_Y$ to whom we delegate the reset of clock $rep(Y)$. Resetter $\mathcal{R}_Y$ has the location $\ell_{nst\mathcal{R}_Y}$ which, as in [5], encodes unstable configurations wrt. Y. In contrast to [5], where the time points for resets were encoded in the resetters, our new approach lets the transformed automata indicate $\mathcal{R}_Y$ when to reset the clock $rep(Y)$ using the following two mechanisms (cf. Figure 2):

1. *Rendez-vous channel $reset_Y$.* This mechanism is used if at least one transformed automaton assumes the origin location of a simple edge. The origin locations of simple edges obtain self-loops which can synchronise with $\mathcal{R}_Y$ on $reset_Y$ exactly at those points in time in which the simple edge would be taken in the original network. Since multiple automata may have an edge synchronising on $reset_Y$ enabled, there is a slight verification time overhead, but all edges induce the exact same follow-up configuration where only $\mathcal{R}_Y$ changes its location. The location which $\mathcal{R}_Y$ reaches by the synchronisation on $reset_Y$ is the first of a chain of locations. The edges along the chain basically simulate a broadcast to all transformed automata which assume the origin location of a simple edge (as indicated by the flag $s_Y^{\mathcal{A}}$) using the rendezvous channels r_Y. The synchronisation on r_Y involves the transformed simple edge if and only if would be enabled in $\mathcal{N}$. Thereby, $\mathcal{N}'$ realises exactly one fixed sequence of simple edges as opposed to the full interleaving of simple edges possible in $\mathcal{N}$. To avoid costly and unnecessary interleavings between these "pseudo-broadcasts", all intermediate locations are committed.
   Note that a better option could be the use of a single broadcast channel on which automata assuming an origin location of a simple edge would be able to send *and* listen, and on which the corresponding $\mathcal{R}_Y$ would listen. Then, all interleavings of simple edges wrt. Y possible in $\mathcal{N}$ would be simulated by only one transition in $\mathcal{N}'$. Unfortunately, the version of Uppaal used in our experiments does not allow clock constraints on edges with inputs on broadcast channels, which is necessary since being at the origin location of a simple edge does not imply that that edge is enabled.
2. *Urgent broadcast channel u_Y.* For the case that no transformed simple edge is ready to indicate the reset time, the $\mathcal{R}_Y$ also (indirectly) observes whether complex edges are taken. If the first transformed complex edge is taken at reset time, then the sum of tokens will decrease. The resetter $\mathcal{R}_Y$ uses the urgent broadcast channel u_Y in order to transit to $\ell_{nst\mathcal{R}_Y}$ as soon as the sum of tokens is below $|Y|$. By transiting to the *urgent* location $\ell_{nst\mathcal{R}_Y}$, we ensure that no time elapses unless a configuration corresponding to stability wrt. Y is reached, i.e. until all tokens wrt. Y are 0.

In order to avoid interleavings between resetters and, e.g., complex edges wrt. other equivalence classes which may be unstable at the same point in time, a resetter $\mathcal{R}_Y$ only transits back to $\ell_{ini\mathcal{R}_Y}$ (and resets the representative $rep(Y)$ and the tokens), if all other equivalence classes are stable as expressed by condition $blk(\mathcal{EC}_{\mathcal{N}})$ in the guard.

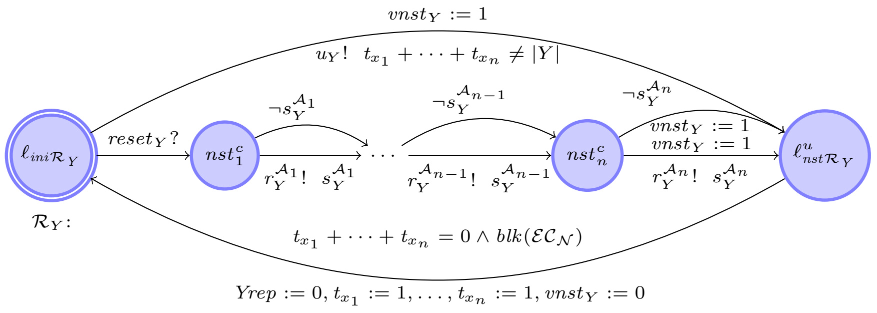
Note that simple edges are taken independently from all other edges, this allows us to take all transformed simple edges in $\mathcal{N}'$ before the first transformed complex one, which in turn allows us to support all queries which ask for configurations where some complex edges and none, only some, or all simple edges have been taken. Our choice for this order restricts broadcast synchronisation on edges where the receiver resets clocks from a given equivalence class, and the sender does not reset clocks from that class. To avoid unnecessarily interleavings, we enforce this order using $go(\vec{r})$ in guards of transformed complex edges



$\mathcal{A}$:
ℓ
φ
$x_1 := 0$
ℓ'
$\langle \dots \rangle$
$I(\ell)$
$reset_Y!\ \Gamma(\varphi)$
$r_Y^{\mathcal{A}}?\ \neg\Gamma(\varphi)$
$\mathcal{K}(\mathcal{A})$:
$r_Y^{\mathcal{A}}?\ \Gamma(\varphi)$
$\langle \dots, s_Y^{\mathcal{A}} := 1 \rangle$
$t_{x_1} := 0, [s_Y^{\mathcal{A}} := 0]$
$\Gamma(I(\ell))$

(a) Pattern of a simple edge wrt. Y in automaton $\mathcal{A}$.

(b) Pattern of the simple edge of $\mathcal{A}$ transformed in $\mathcal{K}(\mathcal{A})$.



$vnst_Y := 1$
$u_Y!\ \ t_{x_1} + \dots + t_{x_n} \neq |Y|$
$\neg s_Y^{\mathcal{A}_1}$
$\neg s_Y^{\mathcal{A}_{n-1}}$
$\neg s_Y^{\mathcal{A}_n}$
$vnst_Y := 1$
$vnst_Y := 1$
$\ell_{ini\mathcal{R}_Y}$
$reset_Y?$
nst_1^c
nst_n^c
$\ell^u_{nst\mathcal{R}_Y}$
$r_Y^{\mathcal{A}_1}!\ \ s_Y^{\mathcal{A}_1}$
$r_Y^{\mathcal{A}_{n-1}}!\ \ s_Y^{\mathcal{A}_{n-1}}$
$r_Y^{\mathcal{A}_n}!\ \ s_Y^{\mathcal{A}_n}$
$\mathcal{R}_Y$:
$t_{x_1} + \dots + t_{x_n} = 0 \wedge blk(\mathcal{EC_N})$
$Yrep := 0, t_{x_1} := 1, \dots, t_{x_n} := 1, vnst_Y := 0$

(c) Pattern of resetter $\mathcal{R}_Y$.

Fig. 2. Patterns used to transform a given network $\mathcal{N}$ with $\mathcal{EC_N}$. In figures (a), (b) and (c) we consider the following quasi-equal clocks $Y = \{x_1, \dots, x_n\}, Y \in \mathcal{EC_N}$. Urgent and committed locations are denoted with the superscript u and c in the name of those locations, respectively. We use $Yrep$ as representative clock of Y, and $blk(\mathcal{EC_N}) := \bigwedge_{W \in \mathcal{EC_N} \setminus \{Y\}} (\sum_{w \in W} t_w = 0 \vee \sum_{w \in W} t_w = |W|)$.

wrt. Y. The condition $go(\vec{r})$ refers to the sum of all variables $s_Y^{\mathcal{A}}$ as indicator of whether there are transformed simple edges wrt. Y which must be taken before transformed complex edges wrt. Y or, each of those transformed simple ones has been already taken, thus variable $vnst_Y$ holds value true.

Formally, the transformation algorithm $\mathcal{K}$ works with two given inputs, a network $\mathcal{N}$ and the set $\mathcal{EC_N}$ of equivalence classes of quasi-equal clocks. The output of $\mathcal{K}$ is the transformed network $\mathcal{N}' = \{\mathcal{K}(\mathcal{A}_1, \mathcal{EC_N}), \dots, \mathcal{K}(\mathcal{A}_n, \mathcal{EC_N})\} \cup \{\mathcal{R}_Y \mid Y \in \mathcal{EC_N}\}$ where $\mathcal{K}(\mathcal{A}, \mathcal{EC_N}) = (L(\mathcal{A}), B', \mathcal{X}', \mathcal{V}', I', E_c \cup E_s \cup E_n, \ell'_{ini})$.

- $B' = B(\mathcal{A}) \cup \{reset_Y, r_Y^{\mathcal{A}} \mid \mathcal{A} \in \mathcal{RES}_Y(\mathcal{N})\}$, i.e. the rendez-vous channels $reset_Y$ and $r_Y^{\mathcal{A}}$ are added for each equivalence class affected by $\mathcal{A}$,
- $\mathcal{X}' = (\mathcal{X}(\mathcal{A}) \setminus Y) \cup \{rep(Y)\}$, i.e. all quasi-equal clocks but the representative are removed,
- $\mathcal{V}' = \mathcal{V}(\mathcal{A}) \cup \{t_x \mid x \in Y, Y \in \mathcal{EC_N}\} \cup \{s_Y^{\mathcal{A}} \mid \mathcal{A} \in \mathcal{RES}_Y(\mathcal{N})\}$, i.e. one boolean (reset-)token for each quasi-equal clock is added (initial value is one), and a boolean simple-edge indicator $s_Y^{\mathcal{A}}$ (initial value is one iff the initial location of $\mathcal{A}$ is a reset location of a simple edge wrt. Y).
- $I' = \{\ell \mapsto \Gamma(I(\ell)) \mid \ell \in L(\mathcal{A})\}$, i.e. invariants are transformed with Γ to consider the representative and the reset-token of quasi-equal clocks,

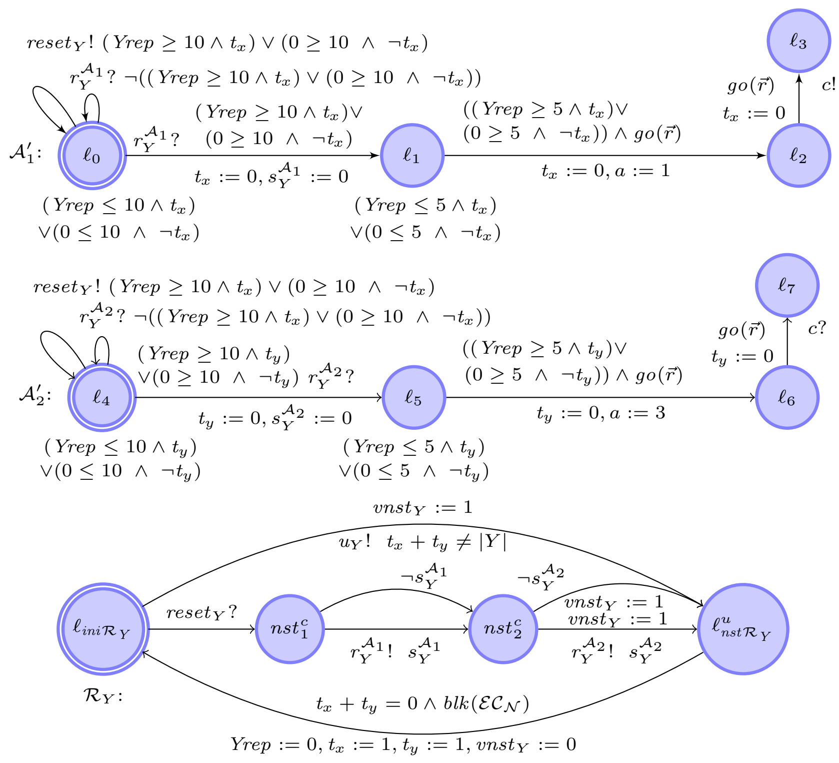

$reset_Y!\ (Yrep \geq 10 \wedge t_x) \vee (0 \geq 10\ \wedge\ \neg t_x)$
$r_Y^{\mathcal{A}_1}?\ \neg((Yrep \geq 10 \wedge t_x) \vee (0 \geq 10\ \wedge\ \neg t_x))$
$\mathcal{A}_1'$:
ℓ_0
$r_Y^{\mathcal{A}_1}?$
$(Yrep \geq 10 \wedge t_x) \vee$
$(0 \geq 10\ \wedge\ \neg t_x)$
$t_x := 0, s_Y^{\mathcal{A}_1} := 0$
ℓ_1
$((Yrep \geq 5 \wedge t_x) \vee$
$(0 \geq 5\ \wedge\ \neg t_x)) \wedge go(\vec{r})$
$t_x := 0, a := 1$
ℓ_2
ℓ_3
$go(\vec{r})$
$t_x := 0$
$c!$
$(Yrep \leq 10 \wedge t_x)$
$\vee(0 \leq 10\ \wedge\ \neg t_x)$
$(Yrep \leq 5 \wedge t_x)$
$\vee(0 \leq 5\ \wedge\ \neg t_x)$
$reset_Y!\ (Yrep \geq 10 \wedge t_x) \vee (0 \geq 10\ \wedge\ \neg t_x)$
$r_Y^{\mathcal{A}_2}?\ \neg((Yrep \geq 10 \wedge t_x) \vee (0 \geq 10\ \wedge\ \neg t_x))$
$\mathcal{A}_2'$:
ℓ_4
$(Yrep \geq 10 \wedge t_y)$
$\vee(0 \geq 10\ \wedge\ \neg t_y)\ r_Y^{\mathcal{A}_2}?$
$t_y := 0, s_Y^{\mathcal{A}_2} := 0$
ℓ_5
$((Yrep \geq 5 \wedge t_y) \vee$
$(0 \geq 5\ \wedge\ \neg t_y)) \wedge go(\vec{r})$
$t_y := 0, a := 3$
ℓ_6
ℓ_7
$go(\vec{r})$
$t_y := 0$
$c?$
$(Yrep \leq 10 \wedge t_y)$
$\vee(0 \leq 10\ \wedge\ \neg t_y)$
$(Yrep \leq 5 \wedge t_y)$
$\vee(0 \leq 5\ \wedge\ \neg t_y)$
$vnst_Y := 1$
$u_Y!\ \ t_x + t_y \neq |Y|$
$\neg s_Y^{\mathcal{A}_1}$
$\neg s_Y^{\mathcal{A}_2}$
$vnst_Y := 1$
$vnst_Y := 1$
$\ell_{ini\mathcal{R}_Y}$
$reset_Y?$
nst_1^c
nst_2^c
$\ell^u_{nst\mathcal{R}_Y}$
$r_Y^{\mathcal{A}_1}!\ \ s_Y^{\mathcal{A}_1}$
$r_Y^{\mathcal{A}_2}!\ \ s_Y^{\mathcal{A}_2}$
$\mathcal{R}_Y$:
$t_x + t_y = 0 \wedge blk(\mathcal{EC}_{\mathcal{N}})$
$Yrep := 0, t_x := 1, t_y := 1, vnst_Y := 0$

Fig. 3. Transformed network $\mathcal{N}_1' = \mathcal{K}(\mathcal{N}_1, \mathcal{EC}_{\mathcal{N}})$.

– Complex and non-resetting edges are transformed as follows, and the resulting edges contained in E_c and E_n, respectively. Guards are also transformed using Γ and, for complex edges, extended by the blocking condition $go(\vec{r}) := \bigwedge_{Y \in \mathcal{EC}_{\mathcal{N}}, clocks(\vec{r}) \cap Y \neq \emptyset} \sum_{\mathcal{A} \in \mathcal{N}} s_Y^{\mathcal{A}} = 0 \vee vnst_Y$. which ensures that simple edges are pushed first. Reset vectors are transformed to consider reset-tokens instead of the original clock, and extended by r_1 as book-keeping for the simple-edge indicator, where $r_1(\ell')$ is the update $s_Y^{\mathcal{A}} := 1$ if ℓ' is the origin location of a simple resetting edge, and ϵ otherwise.

$$E_c = \{(\ell, \alpha, \Gamma(\varphi) \wedge go(\vec{r}), \vec{r}[y := 0/t_y := 0 \mid y \in Y, Y \in \mathcal{EC}_{\mathcal{N}}]; r_1(\ell'), \ell') \mid (\ell, \alpha, \varphi, \vec{r}, \ell') \in E(\mathcal{A}) \setminus \mathcal{SE}_Y(\mathcal{A})\},$$

The transformation of simple edges and the construction of the *resetter* $\mathcal{R}_Y$ for equivalence class Y is depicted in Figure 2. Transformed simple edges contained in E_s.

Example 1. Applying $\mathcal{K}$ to network $\mathcal{N}_1$ from Figure 1 yields network $\mathcal{N}_1'$ (cf. Figure 3). Similar to the algorithm in [5], only the representative clock of each equivalence class remains, in our example we use the fresh clock *Yrep* as representative of Y which is reset by resetter $\mathcal{R}_Y$. Note that each guard and invariant in automata $\mathcal{A}_1'$ and $\mathcal{A}_2'$ is transformed by Γ into a disjunction of clauses. For instance, the guard $x \geq 10$ of automaton $\mathcal{A}_1$ in $\mathcal{N}_1$, is transformed in $\mathcal{N}_1'$ into the encoding $(\mathit{Yrep} \geq 10 \wedge t_x) \vee (0 \geq 10 \wedge \neg t_x)$. Then the clause $(\mathit{Yrep} \geq 10 \wedge t_x)$ is effective in configurations in $\mathcal{N}'$ where t_x is true (encoding that clock x has the same value as *Yrep*), while the clause $(0 \geq 10 \wedge \neg t_x)$ is effective in configurations where t_x is false (encoding that x has already been reset and thus has value 0).

Note that in $\mathcal{N}_1'$ the pair of transformed complex edges from locations ℓ_1 and ℓ_5 preserve their original interleavings. Furthermore, the other pair of transformed complex edges, from locations ℓ_2 and ℓ_6, are taken simultaneously even without delaying at their origin locations.

3.2 Transformation of Properties

Definition 4 (Function Ω). *Let $\mathcal{N}$ be a network with a set $\mathcal{EC}_\mathcal{N}$. Let $\mathcal{A}_i$, with $1 \leq i \leq n$, be the i-th automaton of $\mathcal{N}$. Let $x \in Y$ be a clock. Let $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_\mathcal{N})$. Let β be a basic formula over $\mathcal{N}$. Let $\ell_{nst\mathcal{R}_Y}$ be the unique urgent location of resetter $\mathcal{R}_Y$. We define the function Ω as follows where $E^\star = E(\mathcal{A}) \setminus \mathcal{SE}_Y(\mathcal{A})$: $\Omega_0(\beta) =$*

$$\begin{cases} (\ell' \wedge \tilde{\ell}_i) \vee \ell & , \text{ if } \beta = \ell, (\ell, \alpha, \varphi, \langle x := 0 \rangle, \ell') \in \mathcal{SE}_Y(\mathcal{A}_i). \\ (\ell' \wedge \neg\tilde{\ell}_i) & , \text{ if } \beta = \ell', (\ell, \alpha, \varphi, \langle x := 0 \rangle, \ell') \in \mathcal{SE}_Y(\mathcal{A}_i). \\ \ell_0 & , \text{ if } \beta = \ell_0(\ell_0'), (\ell_0, \alpha_0, \varphi_0, \vec{r}_0, \ell_0') \in E^\star. \\ \Gamma_0(\varphi_{clk})[t_x/(t_x \vee \tilde{x})] \wedge \varphi_{int} & , \text{ if } \beta = \varphi_{clk} \wedge \varphi_{int}. \end{cases}$$

$$\Omega(CF) = \exists\, \tilde{\ell}_1, .., \tilde{\ell}_m \; \exists\, \tilde{x}_1, .., \tilde{x}_k \bullet \Omega_0(CF) \wedge$$
$$(\tilde{\ell}_i \implies \bigvee_{(\ell,\alpha,\varphi,\vec{r},\ell') \in \mathcal{SE}_Y(\mathcal{A}_i)} \ell' \wedge \ell_{nst\mathcal{R}_Y}) \;\wedge\; (\tilde{x}_j \implies \bigvee_{(\ell,\alpha,\varphi,\langle x_j := 0 \rangle,\ell') \in \mathcal{SE}_Y(\mathcal{A}_j)} \ell' \wedge \tilde{\ell}_j)$$

By structural induction Ω_0 transforms configuration formulas CF.

Function Ω syntactically transforms properties over a network $\mathcal{N}$ with a set of equivalence classes of quasi-equal clocks $\mathcal{EC}_\mathcal{N}$ into properties over $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_\mathcal{N})$. Function Ω treats queries for origin and destination locations of simple edges special, and outputs an equivalent property which can be verified in $\mathcal{N}'$. For instance, consider the simple edge $e = (\ell_0, \tau, (x \geq 10), \langle x := 0 \rangle, \ell_1)$ of automaton $\mathcal{A}_1$ of network $\mathcal{N}_1$. The query $\exists\Diamond\, \phi$, where $\phi : \ell_0$, is transformed after some simplifications into $\Omega(\phi) : \exists\, \tilde{\ell} \in \{0,1\} \bullet ((\ell_1 \wedge \tilde{\ell}) \vee \ell_0) \wedge (\tilde{\ell} \implies (\ell_1 \wedge \ell_{nst\mathcal{R}_Y}))$. The logical variable $\tilde{\ell}$ in the transformed query enforces consistent unstable configurations where the location ℓ_0 can be assumed.

The origin location ℓ_0 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 $\mathcal{A}_1$ has not reset its clock x 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 are taken in a fixed sequence by rendez-vous synchronisations, so each origin location is left

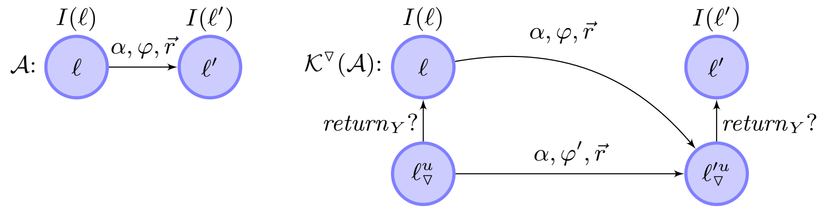

$I(\ell)$
$I(\ell')$
$\mathcal{A}$:
ℓ
$\alpha, \varphi, \vec{r}$
ℓ'
$\mathcal{K}^{\triangledown}(\mathcal{A})$:
$I(\ell)$
$\alpha, \varphi, \vec{r}$
$I(\ell')$
ℓ
ℓ'
$return_Y ?$
$return_Y ?$
$\ell^u_{\triangledown}$
$\alpha, \varphi', \vec{r}$
$\ell'^u_{\triangledown}$

Fig. 4. Transformation pattern of algorithm $\mathcal{K}^{\triangledown}$ over network $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}})$, where $\varphi' = \varphi \wedge \bigwedge_{Y \in \mathcal{EC}_{\mathcal{N}}} \sum_{x \in Y \cap \mathcal{X}(\mathcal{A})} t_x > 0$. Algorithm $\mathcal{K}^{\triangledown}$ takes each edge of network $\mathcal{N}'$ (excluding edges of resetters), cf. left-hand side and transforms it according to the right-hand side. The edge originally linking locations ℓ and ℓ' is redirected to $\ell'^u_{\triangledown}$ if and only if $\exists\, Y \in \mathcal{EC}_{\mathcal{N}}\ \exists\, x \in Y \bullet t_x \in vars(\vec{r})$. In addition, in each resetter $\mathcal{R}_Y$, an output action on broadcast channel $return_Y$ is added to the edge from $\ell_{nst\mathcal{R}_Y}$ to $\ell_{ini\mathcal{R}_Y}$.

one by one. Because the resetter finally moves to $\ell_{nst\mathcal{R}_Y}$ after synchronising with automata $\mathcal{A}'_1$ and $\mathcal{A}'_2$, a configuration of $\mathcal{N}'$ which assumes location $\ell_{nst\mathcal{R}_Y}$ represents all similar unstable configurations of $\mathcal{N}$ where all simple edges are in their origin or destination location. Therefore, from $\Omega(\phi)$, the clause ℓ_0 is effective in configurations of $\mathcal{N}'$ which represent those in $\mathcal{N}$ where the reset time is not yet reached; while the clause $(\ell_1 \wedge \tilde{\ell})$ is effective in configurations of $\mathcal{N}'$ which represent those in $\mathcal{N}$ where the reset time is reached but $\mathcal{A}_1$ has not reset its clock x yet, while other automata in $\mathcal{RES}_Y(\mathcal{N})$ may have reset their clocks already. The latter configurations in $\mathcal{N}$, enforced by the logical variable $\tilde{\ell}$, are assumed in $\mathcal{N}'$, in particular, when resetter $\mathcal{R}_Y$ is located at location $\ell_{nst\mathcal{R}_Y}$ and automaton $\mathcal{A}'_1$ at location ℓ_1.

4 Weak Bisimulation

In order to prove our approach correct we establish a weak bisimulation relation between a network $\mathcal{N}$ with a set of equivalence classes of quasi-equal clocks $\mathcal{EC}_{\mathcal{N}}$, and its respective transformed network $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}})$.

Recall that configurations induced when each clock x from $Y \in \mathcal{EC}_{\mathcal{N}}$ is reset in network $\mathcal{N}$, are summarised in $\mathcal{N}'$ in configurations where the $\ell_{nst\mathcal{R}_Y}$-location is assumed, in particular, when the values of variables $s^{\mathcal{A}}_Y$ and t_x reflect these resets. Hence with the valuations from those variables we unfold information summarised in these configurations from $\mathcal{N}'$.

Lemma 1. *Any network $\mathcal{N}$ with equivalence classes of quasi-equal clocks $\mathcal{EC}_{\mathcal{N}}$, 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. *$\forall\, s \in \mathcal{C}_{ini}(\mathcal{N})\ \exists\, r \in \mathcal{C}_{ini}(\mathcal{N}) \bullet (s, r) \in \mathcal{S}$ and $\forall\, r \in \mathcal{C}_{ini}(\mathcal{N}')\ \exists\, s \in \mathcal{C}_{ini}(\mathcal{N}) \bullet (s, r) \in \mathcal{S}$.*
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 $\forall\, r \in\downarrow_2 \mathcal{S} \bullet r \models \Omega(CF) \implies \exists\, s \in Conf(\mathcal{N}) \bullet (s, r) \in \mathcal{S} \wedge s \models CF$.*
3. *For all $(s, r) \in \mathcal{S}$,*

(a) if $s \xrightarrow{\lambda} s'$ and
   i. $\lambda = d > 0$, then there exists a sequence of transitions $r \xrightarrow{\tau}^{*} r' \xrightarrow{\lambda} r''$, with $(s', r'') \in \mathcal{S}$,
   ii. transition is justified by some edges (non-resetting, simple or complex edge wrt. $Y \in \mathcal{EC}_{\mathcal{N}}$). Then there exist $r \xrightarrow{\tau}^{*} r' \xrightarrow{\lambda} r'' \xrightarrow{\tau}^{*} r'''$, with $(s', r''') \in \mathcal{S}$.

(b) if $r \xrightarrow{\lambda} r'$ then there exists s', such that $s \xrightarrow{\lambda_1} s'$, with $(s', r') \in \mathcal{S}$.

Where $r \xrightarrow{\tau}^{*} r'$ denotes zero or more successive τ-transitions from configuration r to configuration r'.

Proof. (Sketch) For each $Y \in \mathcal{EC}_{\mathcal{N}}$ the weak bisimulation relation $\mathcal{S}$ which relates pairs of configurations $(s, r) \in \mathit{Conf}(\mathcal{N}) \times \mathit{Conf}(\mathcal{N}')$, is based in the following four aspects: (A1) the values of variables and non-quasi-equal clocks. (A2) configurations where both networks assume the same locations, and the value of each clock $x \in Y$ in $\mathcal{N}$ coincides with the value of that clock assumed by $\mathit{rep}(x)$ and token t_x in $\mathcal{N}'$. (A2) also considers configurations where simple edges in $\mathcal{N}$ are enabled, and their corresponding transformed simple edges in $\mathcal{N}'$ have been taken. (A3) stable configurations wrt. Y in $\mathcal{N}$ and configurations in $\mathcal{N}'$ where either no transformed resetting edge wrt. Y has been taken, or all transformed resetting edges wrt. Y have been taken. (A4) consistent values for variables $s_Y^{\mathcal{A}}$ and vnst_Y.

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) from $\mathcal{N}$ to $\mathcal{N}'$. For instance, the reset of a simple edge in $\mathcal{N}$ is simulated in $\mathcal{N}'$ with multiple steps. Artificial steps in $\mathcal{N}'$ such as reset_Y-synchronization, u_Y-output and return to $\ell_{ini\mathcal{R}_Y}$ in $\mathcal{R}_Y$ are simulated in $\mathcal{N}$ by a zero delay transition. Steps where $\mathcal{N}'$ takes transformed complex o simple edges are simulated in $\mathcal{N}$ by one step taking the corresponding resetting edge.

Theorem 1. *Let $\mathcal{N}$ be a network with a set $\mathcal{EC}_{\mathcal{N}}$. Let CF be a configuration formula over $\mathcal{N}$. Then $\mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}}) \models \exists\Diamond\, \Omega(CF) \iff \mathcal{N} \models \exists\Diamond\, CF$.*

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

For performance purposes we have transformed our benchmarks with algorithm $\mathcal{K}^{\nabla}$ which takes the output of algorithm $\mathcal{K}$ and applies the changes depicted in Figure 4. Algorithm $\mathcal{K}^{\nabla}$ together with functions *devirQE* and Ω^{∇} allow us to state in Lemma 2 a strong bisimulation relation between the networks $\mathcal{N}'$ and $\mathcal{N}^{\nabla}$ which are output by algorithms $\mathcal{K}$ and $\mathcal{K}^{\nabla}$, respectively.

Lemma 2. *Given networks: $\mathcal{N}$ with a set $\mathcal{EC}_{\mathcal{N}}$, $\mathcal{N}' = \mathcal{K}(\mathcal{N}, \mathcal{EC}_{\mathcal{N}})$ and $\mathcal{N}^{\nabla} = \mathcal{K}^{\nabla}(\mathcal{N}')$. Then $\mathcal{N}'$ is strongly bisimilar to $\mathcal{N}^{\nabla}$ with $\mathit{devirQE}(r) = r[\ell_{\nabla}/\ell \mid \ell_{\nabla} \in L(\mathcal{N}^{\nabla})]$, and $\mathcal{N}' \models \exists\Diamond\, \Omega(CF) \iff \mathcal{N}^{\nabla} \models \exists\Diamond\, \Omega^{\nabla}(CF)$, where Ω^{∇} is defined by replacing in Ω every occurrence of a location ℓ', $(\ell, \alpha, \varphi, \vec{r}, \ell') \in \mathcal{SE}_Y, Y \in \mathcal{EC}_{\mathcal{N}}$, by ℓ'_{∇}.*

Network	C	kStates	M	$t(s)$	Network	C	kStates	M	$t(s)$
EP-21	21	3,145.7	507.6	444.8	*FS-8*	14	5,084.3	160.8	1,007.1
EP-21K	1	3,145.7	525.7	89.8	*FS-8K*	5	1,892.7	78.0	80.3
EP-22	22	6,291.5	1,027.2	1,032.0	*FS-10*	16	17,474,6	518.6	4,734.0
EP-22K	1	6,291.5	1,060.5	193.8	*FS-10K*	5	2,152.1	83.7	97.7
EP-23	23	–	–	–	*FS-11*	17	–	–	–
EP-23K	1	12,582.9	2,146.9	427.3	*FS-126K*	5	28,510.8	905.6	3,963.3
TT-5	6	436.9	57.9	5.9	*CD-14*	29	7,078.1	591.7	1,388.0
TT-5K	1	327.1	79.5	4.5	*CD-14K*	15	1,327.3	142.0	179.1
TT-6	7	2,986.0	116.5	36.9	*CD-15*	31	8,945.7	1,186.9	1,785.7
TT-6K	1	1,916.6	467.1	30.2	*CD-15K*	16	6,062.6	529.6	978.9
TT-7	8	16,839.9	612.9	235.3	*CD-16*	33	–	–	–
TT-7K	1	11,054.9	2,527.7	198.2	*CD-16K*	17	17,892.1	1,954.9	3,703.0
LS-6	17	145.1	21.6	4.3	*CR-6*	6	264.5	20.3	2.8
LS-6K	3	151.2	23.0	2.2	*CR-6K*	1	67.7	12.3	0.8
LS-7	19	553.3	74.6	22.2	*CR-7*	7	7,223.7	497.5	132.8
LS-7K	3	554.7	81.0	10.1	*CR-7K*	1	1,300.6	165.2	20.9
LS-9	23	8,897.6	1,285.2	524.9	*CR-8*	8	–	–	–
LS-9K	3	9,008.2	1,450.8	224.1	*CR-8K*	1	2,569.7	359.0	52.4
FB-12	14	24.6	6.6	30.9	*TA-2*	7	42.1	6.3	0.3
FB-12K	3	24.6	6.1	0.4	*TA-2K*	2	40.1	6.1	0.3
FB-15	17	2,920.3	36.5	3,894.4	*TA-3*	8	921.5	97.5	10.6
FB-15K	3	196.6	31.7	4.8	*TA-3K*	2	917.4	59.5	9.7
FB-16	18	–	–	–	*TA-4*	9	33,547.6	1,827.8	630.0
FB-21K	3	12,582.9	2,138.4	647.5	*TA-4K*	2	31,397.2	1,405.7	412.7
PG-10	13	85.0	4.6	367.9					
PG-10K	3	160.8	7.1	1.5					
PG-12	15	389.1	9.6	9,560.9					
PG-12K	3	737.3	21.1	8.5					
PG-13	16	–	–	–					
PG-18K	3	65,273.9	1,732.4	1,165.3					

Experimental environment: Intel i3,
2.3 GHz, 3 GB, Ubuntu 11.04,
verifyta 4.1.3.4577 / default options.

Table 1. Row X-$N(K)$ gives the figures for case study X with N components (and $\mathcal{K}$ applied). 'C' gives the number of clocks in the model, 'kStates' the number of 10^3 visited states, 'M' memory usage in MB, and '$t(s)$' verification time in seconds. sAsEt transformed each of our benchmarks in at most 5 seconds.

5 Experimental Results

We applied our approach to nine industrial case studies using *sAsEt* [16], our implementation of algorithms $\mathcal{K}$ and $\mathcal{K}^\nabla$ with integrated detection of equivalence classes of quasi-equal clocks and, simple and complex edges. Six case studies *FS* [18], *CR* [19], *CD* [20], *EP* [14], *TT* [21] and *LS* [22] appear in [5]. The interested reader can obtain from [5] more details of those case studies. The elimination of assumptions on networks allowed us to include three new case studies that [5] cannot transform: *FB* [7], *TA* [8] and *PG* [9]. We verified queries as proposed by the respective authors of each case study. Our motivating case study is inspired by the network from [7] which models the Foundation Fieldbus Data Link Layer protocol (FDLL). The network consists of N sensors and one master. Each of them with complex edges which are taken simultaneously by synchronising on a given broadcast channel at the command of the master. Both sender and receiver reset quasi-equal clocks of the same equivalence class. The point in time in which quasi-equal clocks are reset by those complex edges, is neither unique nor explicit in the syntax of those edges. Moreover, those complex edges can be taken even without delaying at their origin locations. Case study [8] is an implementation of a TDMA protocol. Case study [9] is an implementation of the Pragmatic General Multicast (PGM), which is a reliable multicast transport protocol for applications that require multicast data delivery from a single source to multiple receivers.

Table 1 gives figures for the verification of queries in instances of the original and the transformed model (denoted by the suffix K in the name). The rows without results indicate the smallest instances for which we did not obtain results within 24 hours. For all examples we achieved significant savings in verification time, sometimes of factor n. However, the verification time in transformed models is less meaningful in benchmarks *TA* and *TT*. The quasi-equal clocks in the *TA* and *TT* models are reset by complex edges, so all interleaving of resets in the original model are preserved in the transformed network, together with the artificial transitions that our transformation introduces. This can explain that our savings in these models are related to a more efficient DBM-management. Still, the verification of the transformed models of *TA* and *TT* *including* transformation time is faster than verification of the original ones.

The biggest savings in terms of verification time are obtained in the transformed models *FS*, *CD*, and *CR*. In these models we have simple edges whose interleaving is reduced to a fixed sequence. Regarding memory consumption, we observe the biggest savings again in the mentioned models for the reasons already explained. Note that the verification of the transformed models *EP*, *LS*, *PG* takes slightly more memory than the verification of the original counterparts. We argue that this is due to all resetting edges being *complex* in these three 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 $\ell_{nst\mathcal{R}_Y}$. Furthermore, we add extra variables in those networks, namely, boolean tokens for each quasi-equal clock whose management contributes in the overall memory consumption. The shown reduction of the verification time is due to a smaller size of the DBMs that Uppaal uses to represent zones [6] and whose size grows quadratically in the number of clocks.

Our new technique transforms any network with quasi-equal clocks. It reduces the verification time of properties in transformed networks, and represents all clocks from an equivalence class by one representative. This technique can reduce those configurations induced by automata that reset quasi-equal clocks one by one. Furthermore, our technique supports all properties reflected by original networks. We plan to implement our new approach in hybrid automata.

References

1. R. Alur and D. Dill. A theory of timed automata. *TCS*, 126(2):183–235, 1994.
2. C. Herrera, B. Westphal, et al. Reducing quasi-equal clocks in networks of timed automata. In *FORMATS*, volume 7595 of *LNCS*, pages 155–170. Springer, 2012.
3. T.S. Rappaport. *Wireless communications*, volume 2. Prentice Hall, 2002.
4. G. Behrmann, A. David, and K. Larsen. A tutorial on Uppaal. In *SFM*, volume 3185 of *LNCS*, pages 200–236. Springer, 2004.
5. C. Herrera, B. Westphal, and A. Podelski. *Quasi-Equal Clock Reduction: More Networks, More Queries.* In *TACAS*, volume 8413 of *LNCS*, pages 295–309. Springer, 2014.
6. J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *ACPN*, volume 3098 of *LNCS*, pages 87–124. Springer, 2003.
7. N. Petalidis. *Verification of a fieldbus scheduling protocol using timed automata. CI*, 28(5):655–672, 2009.
8. K. Godary. *Validation temporelle de réseaux embarqués critiques et fiables pour l'automobile.* PhD thesis, Institut National des Sciences Appliquées de Lyon, France, 2004.
9. B. Bérard, P. Bouyer, and A. Petit. *Analysing the PGM Protocol with UPPAAL. IJPR*, 42(14):2773–2791, 2004.
10. C. Daws and S. Yovine. Reducing the number of clock variables of timed automata. In *RTSS*, pages 73–81. IEEE, 1996.
11. C. Daws et al. Model checking of real-time reachability properties using abstractions. In *TACAS*, volume 1384 of *LNCS*, pages 313–329. Springer, 1998.
12. V. Braberman, D. Garbervetsky, N. Kicillof, D. Monteverde, 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.
13. V. Braberman et al. Improving the verification of timed systems using influence information. In *TACAS*, volume 2280 of *LNCS*, pages 21–36. Springer, 2002.
14. S. Limal, S. Potier, B. Denis, and J. Lesage. Formal verification of redundant media extension of ethernet powerlink. In *ETFA*, pages 1045–1052. IEEE, 2007.
15. M. Muñiz, B. Westphal, and A. Podelski. Timed automata with disjoint activity. In *FORMATS*, volume 7595 of *LNCS*, pages 188–203. Springer, 2012.
16. M. Muñiz, B. Westphal, and A. Podelski. Detecting quasi-equal clocks in timed automata. In *FORMATS*, pages 198–212. Springer, 2013.
17. E.-R. Olderog and H. Dierks. *Real-time systems - formal specification and automatic verification.* Cambridge University Press, 2008.
18. D. Dietsch, S. Feo-Arenis, et al. Disambiguation of industrial standards through formalization and graphical languages. In *RE*, pages 265–270. IEEE, 2011.
19. S. Gobriel, S. Khattab, D. Mossé, et al. RideSharing: Fault tolerant aggregation in sensor networks using corrective actions. In *SECON*, pages 595–604. IEEE, 2006.
20. H. Jensen, K. Larsen, and A. Skou. Modelling and analysis of a collision avoidance protocol using SPIN and Uppaal. In *2nd SPIN Workshop*, 1996.
21. W. Steiner and W. Elmenreich. Automatic recovery of the TTP/A sensor/actuator network. In *WISES*, pages 25–37. Vienna University of Technology, 2003.
22. P. Kordy, R. Langerak, et al. Re-verification of a lip synchronization protocol using robust reachability. In *FMA*, volume 20 of *EPTCS*, pages 49–62, 2009.