Quasi-Equal Clock Reduction On-the-Fly *

Bernd Westphal[0000−0002−6824−0567]

Albert-Ludwigs-Universität Freiburg, Germany
westphal@informatik.uni-freiburg.de

Abstract. For timed automata, there is the notion of quasi-equal clocks. Two clocks are quasi-equal if, in each reachable configuration, they are equal or at least one has value 0. There are approaches to exploit quasi-equality of clocks to speed up reachability checking for timed automata. There is a procedure to detect quasi-equal clocks and a syntactical transformation of networks of timed automata where quasi-equal clocks are encoded by one representative clock and one boolean token per clock.
In this work, we integrate reachability checking for timed automata with an on-the-fly analysis of quasi-equality. Our approach uses a new data-structure that combines Difference Bound Matrices (DBMs) for representative clocks with a vector of boolean tokens and a representation of equivalence classes. Our approach achieves space-savings similar to the syntactical transformation and exploits phase-wise quasi-equality of clocks, where existing approaches only consider quasi-equality in all reachable configurations.

1 Introduction

Space systems often use wireless communication based on Time Division Multiple Access (TDMA) schemes for reliability and efficiency as in, e.g., the RideSharing data aggregation protocol [1, 2]. Each participating component usually has a local timer that is most naturally represented in timed models by local clocks. The number of clocks in a model can substantially affect the space and time consumption of model-checking tasks, so a common goal for models of timed systems is to keep the number of clocks low or reduce the number of clocks in a model while preserving the system behaviour.

A number of properties of clocks have been identified that allow us to reduce the number of clocks in a model. The property considered in this article is *quasi-equality*: Two clocks are quasi-equal if and only if, in each reachable model configuration, they are equal or one has value 0. Quasi-equal (but not equal) clocks are in particular observed in protocol designs that employ some kind of time division: Local clocks define cycles, frames, or slots of the time division and each component is responsible for its clocks so clock resets may interleave [12]. Thereby, clocks become unequal but may be quasi-equal. In previous work [11–14] we have developed a source-to-source transformation of descriptions of networks of timed automata. The transformed network reflects all properties of the

* Supported by the German Research Council (DFG) under grant WE 6198/1-1.

original network but only has as many clocks as there are equivalence classes of quasi-equal clocks. A strong advantage of this syntactical approach is that the timed model need not be optimised for model-checking and can naturally model the timed system and thereby ease validation with the protocol engineers (e.g., by simulation, cf. [3]) and limit maintenance effort to one model (as opposed to one model for validation and one for model-checking, obtained following guidelines such as [18, 19]). A drawback is that the equivalence classes need to be known a priori and the efficient quasi-equality detection algorithm [15] is sound but not complete.

In this article, we address the question whether the representation of quasi-equal clocks by tokens and representative clocks (as employed in [12]) has a semantical correspondence which allows us to *integrate* detection of quasi-equalities and space-efficient model-checking. To this end, we introduce a new data structure called DBM_T and adapt the classical timed automata model-checking algorithm to the new data structure. The new algorithm avoids the two passes (detection and transformation) and supports the new, more general notion of *local* quasi-equality that has a potential for further space savings.

Related work includes [8, 7], where a static analysis of a timed automaton is performed to detect *equal* clocks (which can then be reduced by syntactical substitution). This approach does not support quasi-equality. In the same line of work, [8, 7] consider *activity* of clocks (similar to live variables in program analysis); also see, e.g., [4]. Considering activity is orthogonal to our approach. A different branch of research is concerned with minimisations of the number of clocks. The general case (whether there exists a language equivalent timed automaton with fewer clocks than a given one) is undecidable [9]. Guha et al. [10] have shown that a minimal timed bisimilar automaton is effectively constructable, the procedure is expensive in the number of locations and time. Saeedloei et al. [17] minimise under further constraints (same graph, same pattern of clock resets and uses). Our aim is to exploit the quasi-equality of clocks to improve the space consumption of model-checking networks of timed-automata in an integrated algorithm, we do not address minimality of the number of clocks in general.

The paper is structured as follows. Based on preliminaries from Section 2, we introduce the new data-structure DBM_T in Section 3. In Sections 4 and 5, we define operations on DBM_T that we use in Section 6 to adapt the classical reachability checking algorithm for networks of timed automata to DBM_T. Section 7 reports evaluation results and Section 8 concludes.

2 Preliminaries

The definition of the new data-structure of DBM_T relies strongly on the theory of Difference Bound Matrices (DBM), so we need to recall some definitions for self-containedness. The presentation follows [6] and is standard with the exception that we impose some assumptions on zones towards DBMs for convenience. We then briefly recall the syntax and semantics of timed automata, following [16].

2.1 Clocks, Zones, and Difference-Bound-Matrices

Given a finite, non-empty set $X = \{x_0, x_1, \ldots, x_n\}$, $n \geq 0$, of *clocks*, a *clock constraints* (over X) is an expression of the form $\phi = x - y \sim c$ with $\sim \in \{<, \leq\}$ and $c \in \mathbb{Z}$. Assuming a dedicated clock x_0 that always has value 0, clock constraints can express lower and upper bounds on individual clocks. We use $\Phi(X)$ to denote the set of clock constraints and φ to denote subsets of $\Phi(X)$. Functions $\nu : X \to \mathbb{R}_0^+$ are called *valuation* of X.

A *zone* Z is a set of clock constraints, i.e., $Z \subseteq \Phi(X)$. In the following, we assume that for each two clocks x and y there is at most one constraint of the form $x - y \sim c$ in Z, that a zone implies $x \geq 0$ for each clock x, and that a zone includes the constraint $x - x \leq 0$ for each clock $x \in X$. We write $[\![Z]\!]$ to denote the set of valuations that satisfy all constraints in Z, i.e., $[\![Z]\!] = \{\nu \mid \forall\, \phi \in Z \bullet \nu \models \phi\}$.

Zones can be represented by Difference Bounds Matrices (DBM). The DBM of Z over $X = \{x_0, x_1, \ldots, x_n\}$ is the $(n+1) \times (n+1)$ matrix with $D_{i,j} = (c, \sim)$ if $x_i - x_j \sim c \in Z$ and $D_{i,j} = \infty$ otherwise. We may use D_{x_i,x_j} to denote $D_{i,j}$ and $|D|$ to denote the dimension of the matrix. Figure 2a (on page 5) shows an example DBM that represents the clock valuation $\{x_0 \mapsto 0, x \mapsto 0, y \mapsto 10\}$. We write $[\![D]\!]$ to denote the set of valuations that satisfy all constraints in D, i.e., $[\![D]\!] = \{\nu \mid \forall\, i, j \bullet D_{i,j} = \infty \vee D_{i,j} = (c, \sim) \wedge \nu \models x_i - x_j \sim c\}$. If D is the DBM of zone Z, then $[\![D]\!] = [\![Z]\!]$. In the following, we assume that DBMs correspond to zones as introduced above, e.g., $D_{x,x} = (0, \leq)$ for all $x \in X$ with the exception that D_{x_0,x_0} may be $(-1, \leq)$ to represent zone Z with $[\![Z]\!] = \emptyset$.

DBM entries are compared as follows. We have $(m, \sim) < \infty$, $(m_1, \sim_1) < (m_2, \sim_2)$ if $m_1 < m_2$, and $(m, <) < (m, \leq)$. Addition of DBM entries is defined as $\infty + b = \infty$ for each DBM entry b, $(m_1, <) + (m_2, \sim) = (m_1 + m_2, <)$, and $(m_1, \leq) + (m_2, \leq) = (m_1 + m_2, \leq)$. It is computable whether any valuation in $[\![D]\!]$ satisfies a clock constraint ϕ, and whether $[\![D_1]\!] \subseteq [\![D_2]\!]$ (then D_2 is said to *subsume* D_1). There are algorithms to compute on DBMs the intersection of $[\![D]\!]$ with a clock constraint, the effect of arbitrary delay (freeing), and the effect of clock resets. There are a notion of *canonicity*, i.e., encoding each zone by exactly one canonical DBM, and canonicity-preserving variants of the algorithms.

2.2 Networks of Timed Automata

A *timed automaton* $\mathcal{A}$ is a tuple $(L, A, X, I, E, \ell_{ini})$ with a finite set of *locations* L (including the *initial location* ℓ_{ini}), a set of *channels* A, and a finite set of clocks X. Each location $\ell \in L$ is assigned a downward-closed *location invariant* $I(\ell) \subseteq \Phi(X)$ and E is a finite set of *edges*. An edge $(\ell, \alpha, \varphi, \varrho, \ell') \in E$ has *source* and *destination location* ℓ and ℓ', a *guard* $\varphi \subseteq \Phi(X)$, an *action* $\alpha \in \{a!, a? \mid a \in A\} \cup \{\tau\}$, and a *reset* $\varrho \subseteq X$. A *network (of timed automata)* $\mathcal{N} = \mathcal{A}_1 \| \ldots \| \mathcal{A}_n$ consists of finitely many automata with pairwise equal channel and clock sets for simplicity (a clock is called local if it is used in location invariants and edges of at most one automaton).

The operational semantics of $\mathcal{N}$ is the transition system $(\mathit{Conf}(\mathcal{N}), \{\xrightarrow{\lambda} \mid \lambda \in \mathbb{R}_0^+ \cup \{\tau\}\}, C_0)$ where $\mathit{Conf}(\mathcal{N})$ is the set of *configurations*. A configuration

Fig. 1: Clocks x and y are quasi-equal but not equal for $c = 10$.

$\langle \vec{\ell}, Z \rangle$ consists of a location vector $\vec{\ell} \in L_1 \times \cdots \times L_n$ and a zone Z such that $Z \models I(\vec{\ell}_i)$. Let $Z_0 = Z_{ini}{\uparrow} \wedge I(\vec{\ell}_0)$, where Z_{ini} implies that each clock $x \in X$ has value 0, and $\vec{\ell}_0$ the vector consisting of all initial locations,. The set of *initial configurations* is $C_0 = \{\langle \vec{\ell}_0, Z_0 \rangle\}$ if $[\![Z_0]\!] \neq \emptyset$, and $C_0 = \emptyset$ otherwise. There is a *(time abstract) transition* $\langle \vec{\ell}, Z \rangle \xrightarrow{\tau} \langle \vec{\ell}', Z' \rangle$ if and only if there are edges $e_i = (\ell_i, \alpha_i, \varphi_i, \varrho_i, \ell_i') \in E(\mathcal{A}_i)$ and $e_j = (\ell_j, \alpha_j, \varphi_j, \varrho_j, \ell_j') \in E(\mathcal{A}_j)$, $0 \leq i, j \leq n$, s.t. $Z \models \varphi_i \wedge \varphi_j$, $\vec{\ell}' = \vec{\ell}[\ell_i := \ell_i', \ell_j := \ell_j']$, and $Z' = Z[\varrho_i \cup \varrho_j]{\uparrow} \wedge I(\vec{\ell}')$ where either $i = j$ and $\alpha_i = \tau$, or $i \neq j$ and $\alpha_i = a!$ and $\alpha_j = a?$ for some $a \in A$. Edges e_i and e_j are called *enabled* in this case.

A *computation path* (of $\mathcal{N}$) is an initial and consecutive sequence $\langle \vec{\ell}_0, Z_0 \rangle \xrightarrow{\tau} \langle \vec{\ell}_1, Z_1 \rangle \xrightarrow{\tau} \ldots \xrightarrow{\tau} \langle \vec{\ell}_n, Z_n \rangle$, i.e., if $\langle \vec{\ell}_0, Z_0 \rangle \in C_0$ and $(\langle \vec{\ell}_i, Z_i \rangle, \langle \vec{\ell}_{i+1}, Z_{i+1} \rangle) \in \xrightarrow{\tau}$ for all $0 \leq i < n$. A configuration $\langle \vec{\ell}, Z \rangle$ (or $\langle \vec{\ell}, \nu \rangle$) is called *reachable* (in $\mathcal{N}$) if and only if there is a computation path with $\langle \vec{\ell}, Z \rangle = \langle \vec{\ell}_n, Z_n \rangle$ (or $\nu \in [\![Z]\!]$). We use $\mathit{Reach}(\mathcal{N})$ to denote the set of all reachable configurations of $\mathcal{N}$.

2.3 Quasi-Equal Clocks

To support *local* quasi-equality (not only network-wide), we define global quasi-equality as a special case of quasi-equality in a set of valuations.

Definition 1. *Two clocks $x, y \in X$ are called* quasi-equal (locally, or phase-wise) in the set of valuations $N \subseteq X \to \mathbb{R}_0^+$ *if and only if, for each $\nu \in N$, $\nu \models x = y \vee x = 0 \vee y = 0$. Clocks x and y are called* quasi-equal in network $\mathcal{N}$ (or network-wide) *if and only if x and y are quasi-equal in the all reachable valuations of $\mathcal{N}$. The set of equivalence classes [13] of network-wide quasi-equality is denoted by $\mathcal{EC}_{\mathcal{N}}$, in the following exclusive of $\{x_0\}$.* ♢

In Figure 1, clocks x and y are quasi-equal for $c = 10$ (and unequal otherwise). In Figure 5 on page 14, clocks x and y are locally but not globally quasi-equal.

3 DBM$_T$: DBMs with Clock Partitions and Tokens

In this section, we introduce a new data-structure called DBM$_T$ that realises the idea from [11–14] to only store representative clocks for sets of quasi-equal clocks together with boolean tokens that encode whether a clock is equal to the representative or has value 0. While the syntactical transformation of [11–14] has a set of equivalence-classes wrt. quasi-equality as an input, our approach aims at

$$\begin{array}{c} \\ x_0 \\ x \\ y \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x & y \end{array} \\ \begin{pmatrix} (0,\leq) & (0,\leq) & (-10,\leq) \\ (0,\leq) & (0,\leq) & (-10,\leq) \\ (10,\leq) & (10,\leq) & (0,\leq) \end{pmatrix} \end{array}$$

(a) Example DBM D.

$$\begin{pmatrix} (0,\leq) & (-10,\leq) \\ (10,\leq) & (0,\leq) \end{pmatrix}, \{t_{x_0}\}, \{\bar{t}_x, t_y\}$$

(b) Example DBM_T Q.

$$D(Q) = \begin{pmatrix} C_{[x_0],[x_0]} & C_{[x_0],0} & C_{[x_0],[x]} \\ C_{0,[x_0]} & C_{0,0} & C_{0,[x]} \\ C_{[y],[x_0]} & C_{[y],0} & C_{[y],[y]} \end{pmatrix} = \begin{pmatrix} (0,\leq) & (0,\leq) & (-10,\leq) \\ (0,\leq) & (0,\leq) & (-10,\leq) \\ (10,\leq) & (10,\leq) & (0,\leq) \end{pmatrix}$$

(c) The DBM of DBM_T Q.

Fig. 2: DBM and DBM_T examples.

detecting quasi-equality on-the-fly and exploiting phase-wise quasi-equality. To this end, DBM_T consist of a DBM to represent values of representative clocks, a set of sets of quasi-equal clocks, and a vector of boolean tokens.

Definition 2. *A* DBM_T Q *over* X *is a triple* (C, P, T) *where* $P = X_0, \ldots, X_m$ *is a partitioning of* X *(that is,* $X_i \neq \emptyset$*,* $X_i \cap X_j = \emptyset$ *for* $i \neq j$*,* $X_0 \cup \cdots \cup X_m = X$*),* $T : X \to \{0, 1\}$*, and* C *is a DBM over clocks* $\{r_0, \ldots, r_m\}$*. We call* r_i *the* representative (clock) *of the clocks in* X_i *and write* $[x] = r_i$ *for* $x \in X_i$*.*

We call $T(x)$ *the* token *of* x *in* Q*. The token is called* positive *if and only if* $T(x) = 1$ *and* negative *otherwise. In shorthand notation, we write* DBM_T *as* $Q = C, \{\hat{t}_{x_{0,1}}, \ldots, \hat{t}_{x_{0,k_0}}\}, \ldots, \{\hat{t}_{x_{m,1}}, \ldots, \hat{t}_{x_{m,k_m}}\}$ *if* $X_i = \{x_{i,1}, \ldots, x_{i,k_i}\}$ *and* $\hat{t}_x = t_x$ *if* $T(x) = 1$ *and* $\hat{t}_x = \bar{t}_x$ *if* $T(x) = 0$*.* ♢

Definition 3. *Let* $Q = (C, P, T)$ *be a* DBM_T *with* $P = X_0, \ldots, X_m$*. Partition* X_i *from* P *is called* unstable *(in* Q*) if and only if some tokens of clocks in* X_i *are positive and some negative in* T*, i.e. if* $0 < \sum_{x \in X_i} T(x) < |X_i|$*. Otherwise,* X_i *is called* stable*. The partition* X_i *is called* strongly stable *if and only if all tokens of clocks in* X_i *are positive, i.e. if.* $\sum_{x \in X_i} T(x) = |X_i|$*. If* X_i *is stable but not strongly stable, it is called* weakly stable*. We call* Q *stable etc. if and only if all partitions from* P *are stable.* ♢

Figure 2b shows an example DBM_T in shorthand notation, i.e., we have $P = \{x_0\}, \{x, y\}$ and $T = \{x_0 \mapsto 1, x \mapsto 0, y \mapsto 1\}$. In Q, $X_0 = \{x_0\}$ is strongly stable and $X_1 = \{x, y\}$ is unstable, hence Q is unstable

Definition 4. *Let* $Q = (C, P, T)$ *be a* DBM_T *over* X*. We use* $[\![Q]\!]$ *to denote the set of clock valuations where the values of clocks with positive token coincide with the value of their representative in* C *and where the values of clocks with negative tokens are* 0*, i.e.*

$$[\![Q]\!] = \{\nu : X \to \mathbb{R}_0^+ \mid \exists\, \nu_0 \in [\![C]\!]\ \forall\, x \in X \bullet \nu(x) = \nu_0([x]) \cdot T(x)\}. \qquad ♢$$

DBM_T represent sets of clock valuations just like DBMs. In the following, we define a mapping from DBM_T to DBMs and show that the DBM of a DBM_T represents the same set of clock valuations. Note that the (somewhat unusual) function notation in Definition 5 is introduced to be used in Lemma 4. Figure 2c shows (the computation of) the DBM of DBM_T Q from Figure 2b.

Definition 5. *Let $Q = (C, P, T)$ be a DBM_T over X. The DBM over X that is point-wise defined as $D(Q)_{x,y} := (f(P,T)(C))_{x,y}$, where*

$$(f(P,T)(C))_{x,y} = \begin{cases} C_{[x],[y]} & \text{, if } t_x, t_y \\ C_{[x],0} & \text{, if } t_x, \bar{t}_y \end{cases} \quad (f(P,T)(C))_{x,y} = \begin{cases} C_{0,[y]} & \text{, if } \bar{t}_x, t_y \\ C_{0,0} & \text{, if } \bar{t}_x, \bar{t}_y \end{cases},$$

is called the DBM of Q *and denoted by* $D(Q)$. ♢

Lemma 1. *Let $Q = (C, P, T)$ be a DBM_T over X and $D(Q)$ the DBM of Q.*

1. *For all $\nu \in [\![D(Q)]\!]$, clocks from the same partition and with positive token are equal in ν, i.e., for all X_i from P we have $\forall\, x, y \in X_i \bullet T(x) = 1 \wedge T(y) = 1 \implies \nu(x) = \nu(y)$.*
2. *For all $\nu \in [\![D(Q)]\!]$, clocks with negative token have value 0 in ν, i.e. $\forall\, x \in X \bullet T(x) = 0 \implies \nu(x) = 0$.*

Proof. 1. Let $\nu \in [\![D(Q)]\!]$. Then $D(Q)_{i,i} = (0, \leq)$ (cf. Section 2.1). Let $x \neq y \in X_i$ and $T(x) = T(y) = 1$. By Definition 5, we have $D(Q)_{x,y} = D_{[x],[y]}$ and $D(Q)_{y,x} = D_{[y],[x]}$. Since x and y are from the same partition, we have $[x] = [y]$ and hence $D_{[x],[y]} = D_{[y],[x]} = (0, \leq)$. Valuation ν satisfies both constraints, $x - y \leq 0$ and $y - x \leq 0$, hence $\nu(x) = \nu(y)$.

2. Let $\nu \in [\![D(Q)]\!]$ and $x \in X$ with $T(x) = 0$. By Definition 5, we have $D(Q)_{x_0,x} = D_{[x_0],0} = (0, \leq) = D_{0,[x_0]} = D(Q)_{x,x_0}$, hence $\nu(x) = \nu(x_0) = 0$.

□

Lemma 2. *For each DBM_T Q, $[\![D(Q)]\!] = [\![Q]\!]$.*

Proof. Let $\nu \in [\![D(Q)]\!]$. Construct a valuation ν_0 of the representative clocks as follows. Set $\nu_0(r) = \nu(x)$ if $[x] = r$ and $T(x) = 1$, and set $\nu_0(r) = 0$ otherwise. Valuation ν_0 is well-defined because, by Lemma 1, clocks from the same partition (hence with the same representative) have the same value in ν. For ν_0, we have $\nu(x) = \nu_0([x]) \cdot T(x)$ by construction and by Lemma 1, hence $\nu \in [\![Q]\!]$.

Let $\nu \in [\![Q]\!]$. Then there exists $\nu_0 \in [\![C]\!]$ such that $\nu(x) = \nu_0([x]) \cdot T(x)$. To show $\nu \in [\![D(Q)]\!]$, we have to show $\nu \models D(Q)_{x,y}$ for all $x, y \in X$. Distinguish four cases by the token values:

- If $T(x) = T(y) = 1$, then $D(Q)_{x,y} = C_{[x],[y]}$. The case $C_{[x],[y]} = \infty$ is trivial, hence consider $C_{[x],[y]} = (m, \sim)$. We have $\nu \models x - y \sim m$ iff $\nu(x) - \nu(y) \sim m$ iff $1 \cdot \nu(x) - 1 \cdot \nu(y) \sim m$ iff $\nu_0([x]) - \nu_0([y]) \sim m$ iff $\nu_0 \models [x] - [y] \sim m$. The latter holds because $\nu_0 \in [\![C]\!]$.
- The case $T(x) = 1$ and $T(y) = 0$ follows similarly to the previous one. We have $D(Q)_{x,y} = C_{[x],[x_0]}$ by definition (x is compared to the dedicated clock x_0), and can use $\nu(y) = 0$ from Lemma 1.

– In case $T(x) = T(y) = 0$, we have $D(Q)_{x,y} = C_{[x_0],[x_0]} = (0, \leq)$ and we can use $\nu(x) = \nu(y) = 0$ from Lemma 1. □

In the following, we establish a relation between quasi-equality and DBM_T. As DBM_T include sets of clocks, we are in particular able to represent phase-wise quasi-equality as opposed to network-wide quasi-equality.

Lemma 3. *Let $Q = (C, P, T)$ be a DBM_T and X_i a partition from P. Then all clocks in X_i are quasi-equal in all valuations $\nu \in [\![Q]\!]$.* ♢

Proof. Let $x, y \in X_i$ and $\nu \in [\![Q]\!]$. If $T(x) = T(y) = 1$, then $\nu(x) = \nu_0([x]) = \nu(y)$. If $T(x) = 0$ or $T(y) = 0$, then the value of this clock is 0 in ν. Hence x and y are quasi-equal in ν. □

The following Lemmata 4 and 5 can be seen as a minimality result. It is not surprising that each DBM can be principally represented with a DBM_T by using singletons as partitions, one for each clock. Then the DBM inside the DBM_T is exactly the DBM to be represented. The stronger result here is that for each partitioning of the set of clocks into quasi-equal clocks (in the valuations represented by a DBM), there is an equivalent DBM_T whose dimension is the number of partitions. And if a partitioning P of the set of clocks X is maximal wrt. to quasi-equality, i.e. if there is no partitioning with strictly fewer partitions than P has and with quasi-equal clocks per partition, then there is not for each DBM D over X a DBM_T of size strictly smaller than $|P|$ equivalent to D.

Lemma 4. *Let $X_0, \ldots, X_m$ be a partitioning of the set of clocks X and let D be a DBM s.t. clocks from the same partition are quasi-equal in all valuations of D, i.e. where $\forall\, 0 \leq i \leq m\ \forall\, x, y \in X_i\ \forall\, \nu \in [\![D]\!] \bullet \nu \models x = y \vee x = 0 \vee y = 0$.*

Then there is a DBM_T $Q = (C, P, T)$ with $|C| = (m + 1)$ and $[\![Q]\!] = [\![D]\!]$. ♢

Proof. Construct a token function $T : X \to \{0, 1\}$ as follows. For each $0 \leq k \leq m$, if for all $x, y \in X_k$ and for all $\nu \in [\![D]\!]$, $\nu(x) = \nu(y)$, then set $T(x) := 1$ for all $x \in X_k$. Otherwise, consider $x \in X_k$ individually. Set $T(x) := 1$ if there is $\nu \in [\![D]\!]$ s.t. $\nu(x) > 0$, and set $T(x) := 0$ otherwise.

Let C be the DBM of dimension $m + 1$ defined by $f^{-1}(P, T)$, the inverse function of $f(\,\cdot\,,\,\cdot\,)$ from Definition 5. The inversion is well-defined with the above construction of T and the assumption of quasi-equality as follows. For the case of positive tokens, let $x_1, x_2 \in X_i$ and $y \in X_j$ be clocks with positive token in T. Then $Q_{x_1,y} = Q_{x_2,y}$, because, by construction of T, clocks x_1 and x_2 are equal in all valuations from $[\![D]\!]$ hence they must satisfy the same difference relation to y. The other three cases follow similarly. Then, with $P = X_0, \ldots, X_m$, we have $D(C, P, T) = f(P, T)(f^{-1}(P, T)(D)) = D$, hence $[\![Q]\!] = [\![D]\!]$. □

Lemma 5. *Let $N \subseteq X \to \mathbb{R}_0^+$ be a set of valuations of clocks X and let $X_0, \ldots, X_m$ be a partitioning of X that is maximal wrt. quasi-equality in N (not considering the dedicated partition $X_0 = \{x_0\}$). Let $Q = (C, P, T)$ be a DBM_T with $[\![Q]\!] = N$. Then $|C| \geq m + 1$.* ♢

Proof. Assume $Q = (C, P, T)$ with $[\![Q]\!] = N$ and $|C| < m + 1$. Then we can choose x, y from one partition of P (excluding the dedicated partition X_0) such that $x \in X_{i_1}$ and $y \in X_{i_2}$ with $i_1 \neq i_2$. Since $X_1, \ldots, X_m$ is maximal wrt. quasi-equality in N, there is $\nu \in N$ s.t. $\nu(x) > 0$, $\nu(y) > 0$, and $\nu(x) \neq \nu(y)$.

If we had $T(x) = T(y) = 1$, then for each $\nu' \in [\![Q]\!]$, we had $\nu'(x) = \nu'(y)$, hence $\nu \notin [\![Q]\!]$. If we had, e.g., $T(x) = 0$, then for each $\nu' \in [\![Q]\!]$, we had $\nu'(x) = 0$, hence $\nu \notin [\![Q]\!]$. □

4 Standard Operations on DBMx

In the following, we define the standard operations for constraint satisfaction, the intersection of DBM_T with a clock constraint, the effect of arbitrary delay (freeing), and the effect of clock resets on DBM_T. We show that applying these operations (except for freeing) on DBM_T and then decoding the DBM_T into a DBM yields the same result as first decoding into a DBM and then applying the standard operation there. Freeing only commutes for strongly stable DBM_T because in an unstable DBM_T the negative tokens enforce that the corresponding clocks have value 0 and not any larger value, which is the effect of freeing.

The valuations encoded by a DBM do not satisfy a constraint ϕ, denoted by $\mathit{sat}(D, \phi) = \mathit{false}$, if and only if $\phi = x - y \sim m$ and $D_{y,x} + (m, \sim) < (0, \leq)$. We write $D \models \phi$ if and only if $\mathit{sat}(D, \phi) = \mathit{true}$. For DBM_T $Q = (C, P, T)$, we define $\mathit{sat}(Q, \phi) := \mathit{sat}(C, \Gamma_Q(\phi))$ using the constraint transformation from Definition 6 below. Both operations are canonically lifted to sets of constraints $\varphi \subseteq \Phi(X)$.

Definition 6. *Let $Q = (C, P, T)$ be a DBM_T over X and $\phi = x - y \sim m \in \Phi(X)$ a clock constraint. Then the Q-*transformation of ϕ *is defined as follows:*

$$\Gamma_Q(\phi) = \begin{cases} [x] - [y] \sim m & \text{, if } t_x, t_y \\ [x] - [x_0] \sim m & \text{, if } t_x, \bar{t}_y \end{cases} \qquad \Gamma_Q(\phi) = \begin{cases} [x_0] - [y] \sim m & \text{, if } \bar{t}_x, t_y \\ [x_0] - [x_0] \sim m & \text{, if } \bar{t}_x, \bar{t}_y \end{cases},$$

where T provides token values and P representative clocks for clocks X. Sets of clock constraints are Q-transformed element-wise. ♢

Lemma 6. *For DBM_T Q over X and $\phi \in \Phi(X)$, $D(Q) \models \phi$ iff $Q \models \phi$.* ♢

Proof. Let $Q = (C, P, T)$ and $\phi = x - y \sim m$. If $T(x) = T(y) = 1$ then $D(Q) \models \phi = \mathit{false}$ iff $D(Q)_{y,x} + (m, \sim) < (0, \leq)$ iff $C_{[y],[x]} + (m, \sim) < (0, \leq)$ iff $C \models \Gamma_Q(\phi) = \mathit{false}$, because $\Gamma_Q(\phi) = [x] - [y] \sim m$. If $T(x) = 0$ and $T(y) = 1$ then $D(Q) \models \phi = \mathit{false}$ iff $C_{[y],0} + (m, \sim) < (0, \leq)$ iff $C \models \Gamma_Q(\phi) = \mathit{false}$, because $\Gamma_Q(\phi) = [x_0] - [y] \sim m$. The other cases follow similarly. □

A DBM D is subsumed by D' (written $D \subseteq D'$) if and only if $\forall\, x, y \in X \bullet D_{x,y} \leq D'_{x,y}$. For DBM_T, we say Q' subsumes Q (and write $Q \subseteq Q'$) if and only if $\forall\, x, y \in X \bullet D(Q)_{x,y} \leq D(Q')_{x,y}$. Note that $D(Q)_{x,y}$ can be computed point-wise, that is, we need not construct the full $D(Q)$.

Lemma 7. *For DBM_T Q, Q' over X, $Q \subseteq Q'$ if and only if $D(Q) \subseteq D(Q')$.* ♢

```
1: if  D_{x,y} + (m,∼) < (0,≤)  then
2:     D_{0,0} ← (−1,≤)
3: else if  (m,∼) < D_{x,y}  then
4:     D_{x,y} ← (m,∼)
5:     for  i,j ∈ {0,...,n}  do
6:         D_{i,j} ← min(D_{i,j}, D_{i,x} + D_{x,j})
7:         D_{i,j} ← min(D_{i,j}, D_{i,y} + D_{y,j})
8:     end for
9: end if
```

Fig. 3: Algorithm $and(D, x - y \sim m)$, $|D| = n + 1$.

Proof. Definition 5. □

The intersection of a DBM D with a constraint ϕ, written as $D \wedge \varphi$, is defined by the algorithm in Figure 3. For DBM_T, we define $Q \wedge \phi := C \wedge \Gamma_Q(\phi)$. Both operations are canonically lifted to sets of constraints $\varphi \subseteq \Phi(X)$.

Lemma 8. *For DBM_T Q over X and $\phi \in \Phi(X)$, $D(Q \wedge \phi) = D(Q) \wedge \phi$.* ♢

Proof. Let $\phi = x - y \sim m$. For the case $T(x) = T(y) = 1$, we have $\Gamma_Q(x - y \sim m) = [x] - [y] \sim m$. In Lines 1 and 2 of the algorithm, we check $C_{[x],[y]} + (m, \sim) < (0, \leq)$ in both, $C \wedge \Gamma_Q(\phi)$ and $D(Q) \wedge \phi$, and get the same result. In Line 3, we have $(m, \sim) < C_{[x],[y]}$ iff $(m, \sim) < D(Q)_{x,y}$ and in Line 4, $C'_{[x],[y]} = (m, \sim)$ iff $D(Q)'_{x,y} = (m, \sim)$. Lines 6 and 7 preserve the loop invariant $C'_{[i],[x]} = D(Q)'_{i,x}$ and $C'_{[x],[j]} = D(Q)'_{x,j}$ and $C'_{[i],[j]} = D(Q)'_{i,j}$. Assuming the loop invariant, we have $C'_{[i],[x]} + C'_{[x],[j]} = D(Q)'_{i,x} + D(Q)'_{x,j}$ and

$$\min(C'_{[i],[j]}, D(Q)'_{i,x} + D(Q)'_{x,j}) = \min(D(Q)'_{i,j}, D(Q)'_{i,x} + D(Q)'_{x,j}),$$

hence after Line 6, $C'_{[i],[j]} = D(Q)'_{i,j}$ and similarly for Line 7. The other three cases are argued similarly. □

The effect of resetting clock x on DBM D, written $reset(D, x)$ (or $D[x := 0]$ for short), is entry-wise defined as follows. $reset(D, x)_{i,j}$ is $D_{i,0}$ if $j = x$, $D_{0,j}$ if $i = x$, and $D_{i,j}$ otherwise. For DBM_T, we define $(C, P, T)[x := 0] = (C, P, T[x := 0])$ if $x \in X_i$, $\sum_{y \neq x \in X_i} T(y) > 0$ and $\nu(x) > 0$ for some $\nu \in [\![Q]\!]$ $(*)$, and $(C[[x] := 0], P, T[y := 1 \mid y \in X_i])$ otherwise.

Condition $(*)$ above ensures that, if the representative clock has value 0, then all tokens become positive to avoid weakly unstable DBM_T.

Lemma 9. *For each DBM_T Q over X, $x \in X$, $D(Q[x := 0]) = D(Q)[x := 0]$.*

Proof. For i, j with $i \neq x$ or $j \neq x$, we have $D(Q)[x := 0]_{i,j} = D(Q)_{i,j} = D(Q[x := 0])_{i,j}$ because $Q[x := 0]$ leaves these entries unchanged.

Consider the case $i = x$ and $j = y$. Then $D(Q[x := 0])_{i,j}$ is $b = D(C, P, T')_{i,j}$ with $T' = T[x := 0]$ if $\sum_{z \neq x \in X_i} T(z) > 0$. If $T'(y) = 1$, then $b = C_{0,[y]} = D(Q)_{0,j}$

which is $D(Q)[x := 0]_{i,j}$. If $T'(y) = 0$, then $b = C_{0,0} = D(Q)_{0,0}$ which is $D(Q)[x := 0]_{i,j}$. Otherwise, $b = D(C[x := 0], P, T')_{i,j}$ with $T' = T[X_i := 1]$ where $C[x := 0]_{[x],[y]} = C_{0,[y]}$. If $T[X_i := 1](y) = 1$, then $b = C_{0,[y]} = D(Q)_{0,j} = D(Q)[x := 0]_{i,j}$. If $T[X_i := 1](y) = 0$, then $b = C_{0,0} = D(Q)_{0,0} = D(Q)[x := 0]_{i,j}$. The case $j = x$ and $i = y$ is symmetric. □

Proposition 1. *Let $Q = (C, P, T)$ be a DBM$_T$ over X and $x \in X$. Then $D((C, P, T[x := 0])) = D(Q)[x := 0]$.*

Proof. Reseting x in $D(Q)$ copies over the row and column of the designated clock x_0, $D((C, P, T[x := 0]))$ yields the corresponding entries from the row and column of the (representative of) x_0. □

The following proposition observes that each weakly stable DBM$_T$ has an equivalent strongly stable DBM$_T$.

Proposition 2. *For DBM$_T$ $Q = (C, P, T)$ with weakly stable partition X_i in P whose representative clock is r, $D(Q) = D((C[r := 0], P, T[X_i := 1]))$.* ♢

Proof. As resets only change DBM entries of x, let $x \in X_i$ and $y \in X$. Then $D(Q)_{x,y} = C_{0,[y]} = C[r := 0]_{[x],[y]} = D((C[r := 0], P, T[X_i := 1]))_{x,y}$. □

The effect of removing upper bounds on clocks (or freeing) in DBM D, written $up(D)$ (or $D{\uparrow}$ for short), is entry-wise defined as $up(D)_{i,j} = \infty$ if $i > 0$ and $j = 0$, and $D_{i,j}$ otherwise. For DBM$_T$ $Q = (C, P, T)$, we define $up(C, P, T) = (C{\uparrow}, P, T)$ if Q is strongly stable, and $up(Q) = Q$ otherwise. That is, if Q is not strongly stable, the up operation is effectively the identity.

Lemma 10. *Let Q be a strongly stable DBM$_T$. Then $D(Q{\uparrow}) = D(Q){\uparrow}$.* ♢

Proof. Let $x \in X \backslash \{x_0\}$. Then $D(Q{\uparrow})_{x,0} = Q{\uparrow}_{[x],[0]} = C{\uparrow}_{[x],[0]} = \infty = D(Q){\uparrow}_{x,0}$. □

If a DBM$_T$ is not strongly stable, then freeing does not commute with the operation on DBM as illustrated by the network in Figure 1 with $c = 11$. At time 10, clock x is reset, and then at time 11, clock y is reset, hence x and y are not quasi-equal. This observation matches the intuition of quasi-equality: Quasi-equal clocks are reset at the same point in time, yet not necessarily with the same transition (this would be equality).

Proposition 3. *If DBM$_T$ Q is not strongly stable, $D(Q{\uparrow}) \neq D(Q){\uparrow}$.* ♢

Proof. Let $T(x) = 0$. $D(Q{\uparrow})_{x,0} = Q{\uparrow}_{0,0} = (0, \leq) \neq \infty = D(Q){\uparrow}_{x,0}$. □

DBM$_T$ do not have a strong notion of a canonical form in contrast to DBM. For example, for DBM$_T$ $Q = C, \{t_{x_0}\}, \{t_x, \bar{t}_y\}, \{z\}$ with $\nu(x) > \nu(z) > 0$ for all $\nu \in [\![Q]\!]$, we have $D(Q) = D(C, \{t_{x_0}\}, \{t_x\}, \{\bar{t}_y, z\})$. That is, clocks with negative token can be moved from one partition to another without changing the set of represented valuations. The algorithm presented in Section 6 does

not store multiple encodings of the same set of represented valuations due to the conducted subsumption check. Furthermore, two equal clocks with positive value can be represented by one or two partitions in a DBM_T. The merge operation as introduced in the next section avoids that case in our algorithm by unifying partitions whose representative clock is 0. Before two clocks reach a positive value, they need to have been reset and are unified into one partition then.

5 Splitting and Merging Equivalence Classes

DBM_T cannot be used as a "plug in" replacement in the classical algorithm for reachability checking because the freeing operation on DBM_T does not commute with the operation on DBM. In the following, we introduce a split operation on DBM_T that transforms a given unstable DBM_T into another DBM_T on which freeing can safely be applied. The subsequently introduced merge operation unifies partitions that are equal after a reset. The merge operation is not necessary for correctness but with this operation, our algorithm can achieve space-savings by exploiting phase-wise quasi-equality, where the transformation-based approach only exploits network-wide quasi-equality.

The *split operation* $split(Q, \varphi)$ on DBM_T $Q = (C, P, T)$ wrt. φ is defined as follows. If Q is stable or if no delay is possible from Q under constraint φ, i.e., if there is no $\nu \in [\![D(Q) \wedge \varphi]\!]$ such that $\nu + d \in [\![Q \wedge \varphi]\!]$ with $d > 0$, then $split(Q, \varphi) = Q$. Otherwise, given $P = X_0, \ldots, X_m$, let X_i' be $X_i \setminus \{x \in X_i \mid \bar{t}_x\}$ if X_i is unstable in Q, and X_i otherwise. Set $X_{m+1} = \bigcup_{i=1}^{m} X_i \setminus X_i'$. Note that X_{m+1} is not empty, because we assumed at least one unstable partition. Then $split(Q, \varphi) = (C'[r_{m+1} := 0], P', T')$ where C' is an $(m+1) \times (m+1)$ matrix with $C'_{i,j} = C_{i,j}$, $0 \leq i, j \leq m$, $P' = X_0', \ldots, X_m', X_{m+1}'$, and $T' = T$.

Lemma 11. *Let Q be a DBM$_T$ over X, $\varphi \subseteq \Phi(X)$, and $\varrho \subseteq X$. Then $D(Q) = D(split(Q, \varphi))$ and $D(split(Q, \varphi)[\varrho := 0]{\uparrow} \wedge \varphi) = D(Q)[\varrho := 0]{\uparrow} \wedge \varphi$.* ♢

Proof. Let $Q' = split(Q, \varphi)$. The split operation only affects clocks with negative token, so for the clocks with positive tokens, Q' yields the same bounds as copied over from $D(Q)$. For clocks with negative tokens, tokens remains negative and hence $D(Q')$ yields bounds of the dedicated clock.

For the second claim, distinguish two cases. If $Q'[\varrho := 0]$ is strongly stable, Lemmata 10 and 8 apply. If $Q'[\varrho := 0]$ is not strongly stable, then $Q'{\uparrow} = Q'$ by definition and Lemma 8 applies. □

Note that the split operation as defined above could be called lazy because clocks are kept together in one partition until the network dynamics require a split (by delay in an unstable configuration). A different split operation could group all clocks with negative tokens together as soon as they are reset. Whether a different split operation is more effective in practice needs further research.

The *merge operation* $merge(Q)$ on DBM_T $Q = (C, P, T)$ is defined as follows. Without loss of generality (as we can re-order DBM C and P), assume that $P = \{x_0\}, X_1, \ldots, X_k, X_{k+1}, \ldots, X_m$ such that $m > k$ and such that partitions

1: $P \leftarrow \emptyset$;
2: $W \leftarrow \{(\vec{\ell}_0, Q) \mid Q = Q_0{\uparrow} \wedge I(\vec{\ell}_0), Q \neq \emptyset\}$;
3: **while** $(\vec{\ell}, Q) \leftarrow \mathit{pick}(W)$ **do** ▷ *empty W terminates loop*
4: **if** $\forall (\vec{\ell}, Q') \in P : Q \not\sqsubseteq Q'$ **then**
5: $P \leftarrow P \cup \{(\vec{\ell}, Q)\}$;
6: $W \leftarrow W \cup \{(\vec{\ell}_s, Q_s) \mid (\vec{\ell}, Q) \overset{\tau}{\rightsquigarrow} (\vec{\ell}_s, Q'), Q_s = Q'{\uparrow} \wedge I(\vec{\ell}_s), Q_s \neq \emptyset\}$
7: **end if**
8: **end while**

Fig. 4: Reachable configurations computation with DBM_T.

$X_{k+1}, \ldots, X_m$ are exactly those partitions that are stable and whose representative clock gets value 0 in all valuations of C. Then $\mathit{merge}(Q) = (C', P', T')$ where C' is a $(k+1) \times (k+1)$ matrix with $C'_{i,j} = C_{i,j}$, $0 \leq i, j \leq k+1$, $T' = T$, $P' = X_0, \ldots, X_k, (X_{k+1} \cup \cdots \cup X_m)$. Use $\mathit{merge}(Q) = Q$ otherwise.

Lemma 12. *For each DBM_T Q, $D(Q) = D(\mathit{merge}(Q))$.* ♢

Proof. Consider the entries of $D(Q)$. For partitions $X_0, \ldots, X_k$, the entries in C' are the same as in C, hence $D(Q)$ and $D(\mathit{merge}(Q))$ yield the same bounds. All clocks from partitions $X_{k+1}, \ldots, X_m$ satisfied the same constraints wrt. each other and all other clocks. They continue to do so after merging as we copy over one row and column from C to C' □

6 Putting It All Together

Figure 4 shows an adaptation of the classical algorithm for reachability checking of networks of timed automata for DBM_T. In Line 2, we start with a DBM_T of dimension 2 and all tokens positive because all clocks are equal in the initial configuration by definition. More formally, we use $Q_0 = (C_0, P_0, T_0)$ where $C_0 = \mathbf{0}_{2\times 2}$ is a 2×2 matrix with all entries being $(0, \leq)$, $P_0 = \{x_0\}, \{x_1, \ldots, x_n\}$, and $T_0(x) = 1$ for each $x \in X$, hence Q_0 is strongly stable. Lemmata 10 and 8 ensure that $D(Q_0)$ is equal to the initial DBM in the classical algorithm. Line 4 conducts the subsumption check on DBM_T, Lemma 7 ensures that the operation corresponds to subsumption check on their DBMs. Line 6 implicitly includes the computation of the set of enabled edges according to their guards, Lemma 8 ensures that the enabledness check is sound on DBM_T.

Given one or two enabled edges, Q' is the effect of taking these edges. We define $(\vec{\ell}, Q) \overset{\tau}{\rightsquigarrow} (\vec{\ell}_s, Q_s)$ if and only if $(C', P', T') = Q \cap \mathit{\Gamma}_Q(\varphi)$, where $\varphi \subseteq \Phi(X)$ is the conjoined guard of the enabled edges, $\vec{\ell}_s$ is $\vec{\ell}$ updated to the destination locations of the considered edges, and

$$Q_s = \mathit{merge}(\mathit{split}(C', P', T'[\varrho := 0], I(\vec{\ell}_s))[\varrho := 0])$$

where ϱ is the union of the reset sets of the considered edges. Proposition 1 allows us to reset tokens without resetting representatives (yet). The split operation

ensures, by Lemma 11, that the subsequent freeing and intersection with the invariants of the destination location vector has the desired effect. Then the reset operation on the whole DBM_T resets the representative and changes the tokens to all-positive for the weakly stable partitions. The merge operation is not necessary for correctness, yet safe by Lemma 12.

Theorem 1. $\forall \mathcal{N}\ \forall\, (\vec{\ell}, Q) \in W\ \forall\, \nu \in [\![D(Q)]\!] \bullet \langle \vec{\ell}, \nu \rangle \in \mathit{Reach}(\mathcal{N})$. ♢

Proof. By induction over the loop iteration in which a DBM_T is added to W. The induction base is Line 2, where $D(Q)$ the initial D (which is reachable) or empty if C_0, and the loop is skipped. The induction step is Line 6. By induction assumption, there is a reachable $\langle \vec{\ell}, D \rangle$ corresponding to $(\vec{\ell}, Q)$ as picked from W. By Lemma 8, the same sets of edges are found enabled by their guard joint guard φ. Resetting tokens preserves a correspondence by Proposition 1, Lemma 11 ensures that the following freeing and intersection with destination location invariant preserves correspondence. Lemma 12 does not change correspondence, hence $D(Q_s) = (D \wedge \varphi)[\varrho := 0]{\uparrow} \wedge I(\vec{\ell}_s)$ which is reachable. □

Theorem 2. $\forall \mathcal{N}\ \forall\, \langle \vec{\ell}, \nu \rangle \in \mathit{Reach}(\mathcal{N})\ \exists\, (\vec{\ell}, Q) \in W @ L3 \bullet \nu \in [\![Q]\!]$. ♢

Proof. If $\langle \vec{\ell}, Z \rangle$ is reachable, then the classical algorithm has an execution that adds a corresponding $\langle \vec{\ell}, D \rangle$ to (properly intialised) W. The pick operation in Figure 4 can choose corresponding DBM_T from W, and thereby simulate each step of the classical algorithm, yielding a corresponding Q. □

The following theorem relates our algorithm to the transformation-based approach that exploits network-wide quasi-equality of clocks. The theorem implies that, if there are n equivalence classes of network-wide quasi-equality, then the DBM_T occurring in W during an execution of our algorithm will include DBMs for representative clocks of at most size n, possibly smaller.

Theorem 3. *Let $\mathcal{N}$ be a network over X and EC an equivalence class of (network-wide) quasi-equality. Let $(\vec{\ell}, (C, P, T))$ be a configuration occurring in W during execution of the algorithm from Figure 4* without *the merge operation. Then for all clocks $x, y \in EC$, there is a partition X_i in P s.t. $x, y \in X_i$.* ♢

Proof. In the induction base case, all clocks (except for x_0) are in one partition hence the claim holds. In the induction step, we can assume that the claim holds for $(\vec{\ell}, Q)$ as picked from W. Partitions are only changed by the split operation, which operates lazily on unstable partitions, that is, a clock x is only removed from a partition X_i if x has a negative token and a positive delay $d > 0$ is possible. If $y \in X_i$ has a negative token, then it is moved to the same partition as x. If y has a positive token, then the representative has a positive value d' otherwise X_i would not be unstable (by definition of the reset operation on DBM_T). Hence in Q_s, x can have value $d > 0$ and y can have value $d' + d > d$, hence x and y are not quasi-equal, i.e., not in equivalence class EC. □

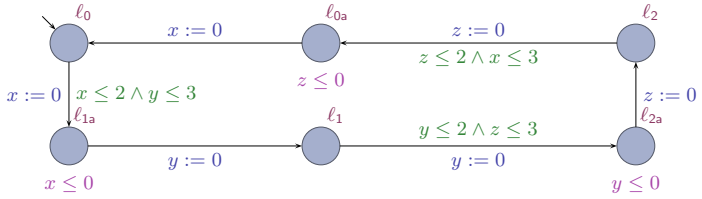

Fig. 5: Illustrative example for equal clock detection [8]; the original does not have locations $\ell_{1a}, \ell_{2a}, \ell_{0a}$ but resets, e.g., x and y on one edge from ℓ_0 to ℓ_1.

	$\mathcal{N}$	$\lvert\mathcal{N}\rvert$	$\lvert X\rvert$	$\lvert\mathcal{EC}\rvert$	$\lvert\mathcal{T}(\mathcal{N})\rvert$	D	t	C	T	t
1a	TTDA	5	5	1	145	7,105	0.124	1,125	1,015	0.116
1b	[1, 2]	6	6	1	341	16,709	0.496	1,364	2,387	0.464
1c		7	7	1	805	65,205	1.612	6,575	7,245	1.532
1d		8	8	1	1,835	148,635	4.696	7,340	16,515	4.500
2a	WFAS	3	2	1	17	153	0.004	68	51	0.004
2b	[3]	4	3	1	28	448	0.012	112	112	0.012
2c		5	4	1	43	1,075	0.016	172	215	0.020
2d		6	5	1	66	2,376	0.032	264	396	0.028
3a	Figure 1,	2	2	1	4	36	$< 10^{-3}$	16	12	$< 10^{-3}$
3b	$c = 10$	2	3	2	4	64	$< 10^{-3}$	21	16	$< 10^{-3}$
4a	Figure 1,	2	2	2	3	27	$< 10^{-3}$	22	9	$< 10^{-3}$
4b	$c = 11$	2	3	3	3	48	$< 10^{-3}$	29	12	$< 10^{-3}$
5	Fig. 5 [8]	1	3	3	8	128	$< 10^{-3}$	62	32	$< 10^{-3}$

Table 1: Evaluation results.

7 Evaluation

We have prototypically implemented the DBM_T data-structure and the algorithm given in Figure 4, and included the classical algorithm on DBM for comparison. Table 1 gives results from running the implementation on two well-known case-studies (the TDMA scheduling and message exchange of the Track Topology Data Aggregation (TTDA) [1, 2] and the self-monitoring protocol of the Wireless Fire Alarm System (WFAS) [3]), and a selection of artificial benchmarks that we referred to in this article.

Columns '$\mathcal{N}$', '$|\mathcal{N}|$', and '$|X|$' give model name, network size, and the number of clocks in the original system (without dedicated clock x_0), Columns '$\mathcal{EC}$' and '$|\mathcal{T}(\mathcal{N})|$ give the maximum number of equivalence classes of (locally) quasi-equal clocks and the number of reachable configurations. Columns 'D' and 't' give the number of DBM-entries and the runtime (in seconds) with the classical algorithm, and Columns 'C', 'T', 't' give the numbers of DBM-entries and tokens, and the runtime (in seconds) with the new on-the-fly quasi-equal clock reduction. We omit runtimes (indicated by '$< 10^{-3}$') if below the order of microseconds.

For TTDA and WFAS (Rows (1) and (2) in Table 1), we observe the expected savings in the number of DBM-entries: For the larger instances of these networks, the number of DBM-entries goes down by factors of about 20 or 9, respectively.

When including the number of tokens (which are, without further optimisations, stored as integers, thus half the size of a DBM entry) we still observe factors of about 9.5 and 5. Regarding verification time, we see some savings with TTDA and WFAS yet much less than for space. To appreciate verification times, recall that the implementation is prototypical and prefers correctness over speed; while space consumption is determined by data structure and algorithm, there is room for improvement in efficiency. Considering the latest figures from the source-to-source transformation approach [12], we see that there are models that reach only 5-10 % of verification time savings (when treating edges as *complex* [12]) and that time savings increase with the number of clocks in the original model, as in our case. A direct comparison to Figures from [12] is not possible because measurements in [12] *exclude* the costs for detection of quasi-equal clocks and the transformation (that are both included in one pass in the new algorithm) and *includes* special and efficiency-relevant treatment of so-called simple edges.

Rows (3) to (4) give the results for our running example from Figure 1. Depending on the constant c, we obtain one or two equivalence classes ((3a), (4a)), and with another, unrelated clock one equivalence class more ((3b), (4b)). Row (5) shows results from the analysis of Figure 5. The automaton in Figure 5 has been derived from a running example of [8] that demonstrates equality of clocks. In our modification, clocks x, y, and z are neither equal nor (globally) quasi-equal, but only locally quasi-equal during certain phases. As expected, we see three different equivalence-classes of quasi-equal clocks over time together with space savings. Neither [8] nor [12] achieve any clock reduction for this model because their equivalence notions are strictly stronger than local quasi-equality.

8 Conclusion

We have shown that the idea to consider representative clocks together with boolean tokens that encode the relation of clocks to the representatives (as used in syntactical transformations for quasi-equal clock reduction) has a direct semantical correspondence in form of DBM_T. Our adaptation of the classical reachability checking algorithm for timed automata to DBM_T allows us to integrate *detection* of quasi-equal clocks with a space-saving *verification*.

If detection is the only goal, the (incomplete) approach of [15] may be more efficient than ours and the syntactical transformation [11] has the advantage that it directly supports all features and strengths of the tool Uppaal [5]. Yet both approaches are limited to network-wide quasi-equality where our algorithm can benefit from phase-wise quasi-equality, and the new algorithm is complete for detection and works in one integrated pass.

Future work includes an extension of our algorithm on DBM_T with a correspondence of the offline partial-order reduction as included in [11–14] in order to also realise the savings in number of configurations that have been observed for models with so-called simple edges (cf. [12]).

References

1. Arenis, S.F., Westphal, B.: Formal verification of a parameterized data aggregation protocol. In: Brat, G., Rungta, N., Venet, A. (eds.) NFM. LNCS, vol. 7871, pp. 428–434. Springer (2013). https://doi.org/10.1007/978-3-642-38088-4_29
2. Arenis, S.F., Westphal, B.: Parameterized verification of track topology aggregation protocols. In: Beyer, D., Boreale, M. (eds.) FORTE. LNCS, vol. 7892, pp. 35–49. Springer (2013). https://doi.org/10.1007/978-3-642-38592-6_4
3. Arenis, S.F., Westphal, B., Dietsch, D., Muñiz, M., Andisha, A.S., Podelski, A.: Ready for testing: ensuring conformance to industrial standards through formal verification. Formal Asp. Comput. **28**(3), 499–527 (2016). https://doi.org/10.1007/s00165-016-0365-3
4. Behrmann, G., Bouyer, P., Fleury, E., Larsen, K.G.: Static guard analysis in timed automata verification. In: Garavel, H., Hatcliff, J. (eds.) TACAS. LNCS, vol. 2619, pp. 254–277. Springer (2003). https://doi.org/10.1007/3-540-36577-X_18
5. Behrmann, G., David, A., Larsen, K.G.: A tutorial on uppaal. In: Bernardo, M., Corradini, F. (eds.) SFM-RT. LNCS, vol. 3185, pp. 200–236. Springer (2004). https://doi.org/10.1007/978-3-540-30080-9_7
6. Bengtsson, J., Yi, W.: Timed automata: Semantics, algorithms and tools. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) ACPN. LNCS, vol. 3098, pp. 87–124. Springer (2003). https://doi.org/10.1007/978-3-540-27755-2_3
7. Daws, C., Tripakis, S.: Model checking of real-time reachability properties using abstractions. In: TACAS. LNCS, vol. 1384, pp. 313–329. Springer (1998). https://doi.org/http://dx.doi.org/10.1007/BFb0054180
8. Daws, C., Yovine, S.: Reducing the number of clock variables of timed automata. In: RTSS. pp. 73–81. IEEE (1996). https://doi.org/10.1109/REAL.1996.563702
9. Finkel, O.: Undecidable problems about timed automata. In: Asarin, E., Bouyer, P. (eds.) FORMATS. LNCS, vol. 4202, pp. 187–199. Springer (2006). https://doi.org/10.1007/11867340_14
10. Guha, S., Narayan, C., Arun-Kumar, S.: Reducing clocks in timed automata while preserving bisimulation. In: Baldan, P., et al. (eds.) CONCUR. LNCS, vol. 8704, pp. 527–543. Springer (2014). https://doi.org/10.1007/978-3-662-44584-6_36
11. Herrera, C., Westphal, B.: Quasi-equal clock reduction: Eliminating assumptions on networks. In: Piterman, N. (ed.) HVC. LNCS, vol. 9434, pp. 173–189. Springer (2015). https://doi.org/10.1007/978-3-319-26287-1_11
12. Herrera, C., Westphal, B.: The model checking problem in networks with quasi-equal clocks. In: Dyreson, C.E., Hansen, M.R., Hunsberger, L. (eds.) TIME. pp. 21–30. IEEE (2016). https://doi.org/10.1109/TIME.2016.10
13. Herrera, C., Westphal, B., Arenis, S.F., Muñiz, M., Podelski, A.: Reducing quasi-equal clocks in networks of timed automata. In: Jurdzinski, M., Nickovic, D. (eds.) FORMATS. LNCS, vol. 7595, pp. 155–170. Springer (2012). https://doi.org/10.1007/978-3-642-33365-1_12
14. Herrera, C., Westphal, B., Podelski, A.: Quasi-equal clock reduction: More networks, more queries. In: Ábrahám, E., et al. (eds.) TACAS. LNCS, vol. 8413, pp. 295–309. Springer (2014). https://doi.org/10.1007/978-3-642-54862-8_20
15. Muñiz, M., Westphal, B., Podelski, A.: Detecting quasi-equal clocks in timed automata. In: Braberman, V.A., Fribourg, L. (eds.) FORMATS. LNCS, vol. 8053, pp. 198–212. Springer (2013). https://doi.org/10.1007/978-3-642-40229-6_14
16. Olderog, E.R., Dierks, H.: Real-time systems - formal specification and automatic verification. Cambridge University Press (2008)

17. Saeedloei, N., Kluzniak, F.: Clock allocation in timed automata and graph colouring. In: Prandini, M., Deshmukh, J.V. (eds.) HSCC. pp. 71–80. ACM (2018). https://doi.org/10.1145/3178126.3178138
18. Salah, R., Bozga, M., et al.: Compositional timing analysis. In: EMSOFT. pp. 39–48. ACM (2009)
19. Waszniowski, L., Hanzalek, Z.: Over-approximate model of multitasking application based on timed automata using only one clock. In: IPDPS. pp. pp. 128a–128a. IEEE (April 2005)