Perfect Timed Communication Is Hard

Parosh Aziz Abdulla[1], Mohamed Faouzi Atig[1(✉)], and Shankara Narayanan Krishna[2]

[1] Uppsala University, Uppsala, Sweden
{parosh,mohamed_faouzi.atig}@it.uu.se
[2] IIT Bombay, Mumbai, India
krishnas@cse.iitb.ac.in

Abstract. We introduce the model of communicating timed automata (CTA) that extends the classical models of finite-state processes communicating through FIFO perfect channels and timed automata, in the sense that the finite-state processes are replaced by timed automata, and messages inside the perfect channels are equipped with clocks representing their ages. In addition to the standard operations (resetting clocks, checking guards of clocks) each automaton can either (1) append a message to the tail of a channel with an initial age or (2) receive the message at the head of a channel if its age satisfies a set of given constraints. In this paper, we show that the reachability problem is undecidable even in the case of two timed automata connected by one unidirectional timed channel if one allows global clocks (that the two automata can check and manipulate). We prove that this undecidability still holds even for CTA consisting of three timed automata and two unidirectional timed channels (and without any global clock). However, the reachability problem becomes decidable (in EXPTIME) in the case of two automata linked with one unidirectional timed channel and with no global clock. Finally, we consider the bounded-context case, where in each context, only one timed automaton is allowed to receive messages from one channel while being able to send messages to all the other timed channels. In this case we show that the reachability problem is decidable.

1 Introduction

In the last few years, several papers have been devoted to extend classical infinite-state systems such as pushdown systems, (lossy) channel systems and Petri nets with timed behaviors in order to obtain more accurate and precise formal models (e.g., [1,2,4,7,8,10–13,15,19–25,28]). In particular, *perfect channel systems* have been extensively studied as a formal model for communicating protocols [16,27]. Unfortunately, perfect channel systems are in general Turing powerful, and hence all basic decision problems (e.g., the reachability problem) are undecidable for them [16]. To circumvent this undecidability obstacle, several approximate techniques have been proposed in the literature including making the channels lossy [5,18], restricting the communication topology to polyforest architectures [26,27], or using half-duplex communication [17]. The decidability of the reachability problem can be also obtained by restricting the analysis

D. N. Jansen and P. Prabhakar (Eds.): FORMATS 2018, LNCS 11022, pp. 91–107, 2018.
https://doi.org/10.1007/978-3-030-00151-3_6

to only executions performing at most some fixed number of context switches (where in each context only one process is allowed to receive messages from one channel while being able to send messages to all the other channels) [26]. Another well-known technique used in the verification of perfect channel systems is that of loop acceleration where the effect of iterating a loop is computed [14].

In this paper, we introduce the model of *Communicating Timed Automata* which extends the classical models of finite-state processes communicating through FIFO perfect channels and timed automata, in the sense that the finite-state processes are replaced by timed automata, and messages inside the perfect channels are equipped with clocks representing their ages. In addition to the standard operations of timed automaton, each automaton can either (1) append a message to the tail of a channel with an initial age or (2) receive the message at the head of a channel if its age satisfies a set of given constraints. In a timed transition, the clock values and the ages of all the messages inside the perfect channels are increased uniformly. Our model subsumes both timed automata and perfect channel systems. More precisely, we obtain the latter if we do not allow the use of timed information (i.e., all the timing constraints trivially hold); and we obtain the former if we do not use the perfect channels (no message is sent or received from the channels). Observe that this model is infinite in multiple dimensions, namely we have a number of channels that may contain an unbounded number of messages each of which is equipped with a real number.

We show that the reachability problem is undecidable even in the case of two discrete timed automata connected by one unidirectional timed channel if one allows global clocks. We prove that this undecidability still holds even for the case when we have three timed automata and two unidirectional timed channels (and without any global clock). However, the reachability problem becomes decidable (in EXPTIME) in the case of two discrete timed automata linked with one unidirectional discrete timed channel and with no global clock. Finally, we consider the bounded-context case, where in each context only one timed automaton is allowed to receive messages from one channel while being able to send messages to all the other timed channels. In this case we show that the reachability is decidable even when the timed automata and the timed channels involved deal with dense time. This is quite surprising since the reachability problem for unidirectional polyforest architectures can be easily reduced to its corresponding problem in the bounded-context case in the untimed settings; however in the presence of time, polyforest architectures already give undecidability, while bounded-context stay decidable.

Related Work. Several extensions of infinite-state systems with time behaviours have been proposed in the literature (e.g., [1,2,4,6–8,10–13,15,19–25,28]). The two closest to ours are those presented in [19,25]. Both works extend perfect channel systems with time behaviours but do not associate a clock to each message (i.e., the content of each channel is still a word over a finite alphabet) as in our case. The work presented [19] shows that the reachability problem is decidable if and only if the communication topology is a polyforest while for our model the reachability problem is undecidable for polyforest architectures in general.

Furthermore, there is no simple reduction of our results to the results presented in [19]. The work presented in [25] considers dense clocks with urgent semantics. In [25], the authors show (as in our model) that the reachability problem is undecidable for three timed automata and two unidirectional timed channels; while it becomes decidable when considering two automata linked with one unidirectional timed channel. However, the used techniques for these results are quite different since we do not allow the urgent semantics.

2 Preliminaries

In this section, we introduce some notations and preliminaries which will be used throughout the paper. We use standard notation $\mathbb{N}$ for the set of naturals, along with ∞. Let $\mathcal{X}$ be a finite set of variables called *clocks*, taking on values from $\mathbb{N}$. A *valuation* on $\mathcal{X}$ is a function $\nu : \mathcal{X} \to \mathbb{N}$. We assume an arbitrary but fixed ordering on the clocks and write x_i for the clock with order i. This allows us to treat a valuation ν as a point $(\nu(x_1), \nu(x_2), \ldots, \nu(x_n)) \in \mathbb{N}^{|\mathcal{X}|}$. For a subset of clocks $X \in 2^{\mathcal{X}}$ and valuation $\nu \in {\mathbb{R}_{\geq 0}}^{|\mathcal{X}|}$, we write $\nu[X{:=}0]$ for the valuation where $\nu[X{:=}0](x) = 0$ if $x \in X$, and $\nu[X{:=}0](x) = \nu(x)$ otherwise. For $t \in \mathbb{N}$, write $\nu + t$ for the valuation defined by $\nu(x) + t$ for all $x \in X$. The valuation $\mathbf{0} \in {\mathbb{R}_{\geq 0}}^{|\mathcal{X}|}$ is a special valuation such that $\mathbf{0}(x) = 0$ for all $x \in \mathcal{X}$. A clock constraint over $\mathcal{X}$ is defined by a (finite) conjunction of constraints of the form $x \bowtie k$, where $k \in \mathbb{N}$, $x \in \mathcal{X}$, and $\bowtie \in \{<, \leq, =, >, \geq\}$. We write $\varphi(\mathcal{X})$ for the set of clock constraints. For a constraint $g \in \varphi(\mathcal{X})$, and a valuation $\nu \in \mathbb{N}^{|\mathcal{X}|}$, we write $\nu \models g$ to represent the fact that valuation ν satisfies constraint g. For example, $(1, 0, 10) \models (x_1 < 2) \wedge (x_2 = 0) \wedge (x_3 > 1)$.

Timed Automata. Let *Act* denote a finite set called actions. A timed automaton (TA) is a tuple $\mathcal{A} = (L, L^0, Act, \mathcal{X}, E, F)$ such that (i) L is a finite set of locations, (ii) $\mathcal{X}$ is a finite set of clocks, (iii) *Act* is a finite alphabet called an action set, (iv) $E \subseteq L \times \varphi(\mathcal{X}) \times Act \times 2^{\mathcal{X}} \times L$ is a finite set of transitions, and (v) $L^0, F \subseteq L$ are respectively the sets of initial and final locations and *Act* is a finite set of actions. A state s of a timed automaton is a pair $s = (\ell, \nu) \in L \times \mathbb{N}^{|\mathcal{X}|}$. A transition (t, e) from a state $s = (\ell, \nu)$ to a state $s' = (\ell', \nu')$ is written as $s \xrightarrow{t,e} s'$ if $e = (\ell, g, a, Y, \ell') \in E$, such that $a \in Act$, $\nu + t \models g$, and $\nu' = (\nu + t)[Y{:=}0]$. A run is a finite sequence $\rho = s_0 \xrightarrow{t_1,e_1} s_1 \xrightarrow{t_2,e_2} s_2 \cdots \xrightarrow{t_n,e_n} s_n$ of states and transitions. $\mathcal{A}$ is non-empty iff there is a run from an initial state $(l_0, \mathbf{0})$ to some state (f, ν) where $f \in F$. Note that we have defined discrete timed automata, a subclass of Alur-Dill automata [9], where clocks assume only integral values.

Region Automata. If $\mathcal{A}$ is a timed automaton, the region automaton corresponding to $\mathcal{A}$ denoted by $Reg(\mathcal{A})$ is an untimed automaton defined as follows. Let K be the maximal constant used in the constraints of A and let $[K] = \{0, 1, \ldots, K, \infty\}$. The locations of $Reg(\mathcal{A})$ are of the form $L \times [K]^{|\mathcal{X}|}$. The set of initial locations of $Reg(\mathcal{A})$ is $L_0 \times \mathbf{0}$. The transitions in $Reg(\mathcal{A})$ are of the following kinds: (i) $(l, \nu) \xrightarrow{\checkmark} (l, \nu + 1)$ denotes a time elapse of 1. If $\nu(x) + 1$

exceeds K for any clock x, then it is replaced with ∞. (ii) For each transition $e = (\ell, g, a, Y, \ell')$, we have the transition $(l, \nu) \xrightarrow{a} (l', \nu')$ if $\nu \models g$, $\nu' = \nu[Y{:=}0]$. It is known [9] that $Reg(\mathcal{A})$ is empty iff $\mathcal{A}$ is.

3 Communicating Timed Automata (CTA)

A communicating timed automata (CTA) $\mathcal{N} = (\mathcal{A}_1, \ldots, \mathcal{A}_n, C, \Sigma, \mathcal{T})$ consists of timed automata $\mathcal{A}_i$, a finite set C of FIFO *channels*, a finite set Σ called the *channel alphabet*, and a *network topology* $\mathcal{T}$. The network topology is a directed graph $(\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}, C)$ comprising of the finite set of timed automata $\mathcal{A}_i$ as nodes, and the channels C as edges. C is given as a tuple $(c_{i,j})$; the channel from $\mathcal{A}_i$ to $\mathcal{A}_j$ is denoted by $c_{i,j}$, with the intended meaning that $\mathcal{A}_i$ writes a message from Σ to channel $c_{i,j}$ and $\mathcal{A}_j$ reads from channel $c_{i,j}$. We assume that there is atmost one channel $c_{i,j}$ from $\mathcal{A}_i$ to $\mathcal{A}_j$, for any pair $(\mathcal{A}_i, \mathcal{A}_j)$ of timed automata. Figure 1 illustrates the definition.

Each timed automaton $\mathcal{A}_i = (L_i, L_i^0, Act, \mathcal{X}_i, E_i, F_i)$ in the CTA is as explained before, with the only difference being in the transitions E_i. We assume that $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ for $i \neq j$. A transition in E_i has the form (l_i, g, op, Y, l_i') where g, Y have the same definition as in that of a timed automaton, while $op \in Act$ is one of the following operation on the channels $c_{i,j}$:

1. nop is an empty operation that does not check or update the channel contents. Transitions having the empty operation nop are called *internal transitions*. Internal transitions of $\mathcal{A}_i$ do not change any channel contents.
2. $c_{i,j}!a$ is a write operation on channel $c_{i,j}$. The operation $c_{i,j}!a$ appends the message $a \in \Sigma$ to the tail of the channel $c_{i,j}$, and sets the age of a to be 0. The timed automaton $\mathcal{A}_i$ moves from location l_i to l_i', checking guard g, resetting clocks Y and writes message a on channel $c_{i,j}$.
3. $c_{j,i}?(a{\in}I)$ is a read operation on channel $c_{j,i}$. The operation $c_{j,i}?(a{\in}I)$ removes the message a from the head of the channel $c_{j,i}$ if its age lies in the interval I. The interval I has the form $<\ell, u>$ with $u \in \mathbb{N}$ and $\ell \in \mathbb{N}\backslash\{\infty\}$, "<" stands for left-open or left-closed and ">" for right-open or right-closed. In this case, the timed automaton $\mathcal{A}_i$ moves from location l_i to l_i', checking guard g, resetting clocks Y and reads off the oldest message a from channel $c_{j,i}$ if its age is in interval I.

Global Clocks. A clock x is said to be global in a CTA if it can be checked any of the timed automata in the CTA, and can also be reset by any of them on a transition. Note that if a clock x is not global, then it can be checked and reset only by the automata which "owns" it. The automaton A_i owns x iff $x \in \mathcal{X}_i$ (recall that $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$). The convention $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ applies to non-global (or local) clocks. Thus, if a CTA consisting of automata $A_1, \ldots, A_n$ has global clocks, then its set of clocks can be thought of as $\biguplus \mathcal{X}_i \uplus \mathcal{G}$ where $\mathcal{G}$ is a set of global clocks, which are accessed by all of $A_1, \ldots, A_n$, while clocks of $\mathcal{X}_i$ are accessible only to A_i.

Configurations. The semantics of $\mathcal{N}$ is given by a labeled transition system $\mathcal{L}_{\mathcal{N}}$. A configuration γ of $\mathcal{N}$ is a tuple $((l_i, \nu_i)_{1\leq i\leq n}, c)$ where l_i is the current control location of $\mathcal{A}_i$, and ν_i gives the valuations of clocks $\mathcal{X}_i$, $1 \leq i \leq n$, where $\nu_i \in \mathbb{N}^{|\mathcal{X}_i|}$. $c = (c_{i,j})$, and each channel $c_{i,j}$ is represented as a monotonic timed word $(a_1, t_1)(a_2, t_2)\ldots(a_n, t_n)$ where $a \in \Sigma$ and $t_i \leq t_{i+1}$, and $t_i \in \mathbb{N}$. Given a word $c_{i,j}$ and a time $t \in \mathbb{N}$, $c_{i,j} + t$ is obtained by adding t to the ages of all messages in channel $c_{i,j}$. For $c = (c_{i,j})$, $c + t$ denotes the tuple $(c_{i,j} + t)$. The states of $\mathcal{L}_{\mathcal{N}}$ are the configurations.
Transition Relation of $\mathcal{L}_{\mathcal{N}}$ Let $\gamma_1 = ((l_1, \nu_1), \ldots, (l_n, \nu_n), c)$ and $\gamma_2 = ((l'_1, \nu'_1), \ldots, (l'_n, \nu'_n), c')$ be two configurations. The transitions $\rightarrow$ in $\mathcal{L}_{\mathcal{N}}$ are of two kinds:

1. Timed transitions $\xrightarrow{t}$: These transitions denote the passage of time $t \in \mathbb{N}$. $\gamma_1 \xrightarrow{t} \gamma_2$ iff $l_i = l'_i$, and $\nu'_i = \nu_i + t$, for all i and $c' = c + t$.
2. Discrete transitions $\xrightarrow{D}$. These are of the following kinds:
   (1) $\gamma_1 \xrightarrow{g,\mathsf{nop},Y} \gamma_2$: there is a transition $l_i \xrightarrow{g,\mathsf{nop},Y} l'_i$ in E_i, $\nu_i \models g$, $\nu'_i = \nu_i[Y := 0]$, for some i. Also, $l_k = l'_k$, $\nu_k = \nu'_k$ for all $k \neq i$, and $c_{d,h} = c'_{d,h}$ for all d, h. None of the channel contents are changed.
   (2) $\gamma_1 \xrightarrow{g,c_{i,j}!a,Y} \gamma_2$: Then, $l_k = l'_k$, $\nu_k = \nu'_k$ for all $k \neq i$, and $c_{d,h} = c'_{d,h}$ for all $(d, h) \neq (i, j)$. The transition $l_i \xrightarrow{g,c_{i,j}!a,Y} l'_i$ is in E_i, $\nu_i \models g$, $\nu'_i = \nu_i[Y := 0]$, $c_{i,j} = w \in (\Sigma \times \mathbb{N})^*$ and $c'_{i,j} = (a, 0).w$.
   (3) $\gamma_1 \xrightarrow{g,c_{j,i}?(a\in I),Y} \gamma_2$: Then, $l_k = l'_k$, $\nu_k = \nu'_k$ for all $k \neq i$, and $c_{d,h} = c'_{d,h}$ for all $(d, h) \neq (j, i)$. The transition $l_i \xrightarrow{g,c_{j,i}?(a\in I),Y} l'_i$ is in E_i, $\nu_i \models g$, $\nu'_i = \nu_i[Y := 0]$, $c_{j,i} = w.(a, t) \in (\Sigma \times \mathbb{N})^+$, $t \in I$ and $c'_{j,i} = w \in (\Sigma \times \mathbb{N})^*$.

The Reachability Problem. The initial location of $\mathcal{L}_{\mathcal{N}}$ is given by the tuple $\gamma_0 = ((l^0_1, \nu^0_1), \ldots, (l^0_n, \nu^0_n), c^0)$ where l^0_i is the initial location of A_i, $\nu^0_i = \mathbf{0}$ for all i, and c^0 is the tuple of empty channels $(\epsilon, \ldots, \epsilon)$. A control location $l_i \in L_i$ is reachable if $\gamma_0 \xrightarrow{*} ((s_i, \nu_i)_{1\leq i\leq n}, c)$ such that $s_i = l_i$ (It does not matter what $(\nu_1, \ldots, \nu_n)$ and c are). An instance of the reachability problem asks whether given a CTA $\mathcal{N}$ with initial configuration γ_0, we can reach a configuration γ.

4 Acyclic CTA

In this section, we look at the reachability problem in CTA whose underlying network topology $\mathcal{T}$ is somewhat restrictive. An *acyclic CTA* is a CTA $\mathcal{N} = (A_1, \ldots, A_n, C, \Sigma, \mathcal{T})$ which has no cycles in the underlying undirected graph of $\mathcal{T}$[1]. Such topologies are called polyforest topologies in [26] (left of Fig. 1). In this section, we answer the reachability question in acyclic CTA with and without global clocks by finding the thin boundary line which separates decidable and undecidable acyclic CTAs.

[1] Recall that the network topology $(\{A_1, \ldots, A_n\}, C)$ is a directed graph; the underlying undirected graph is obtained by considering all edges as undirected in this graph.

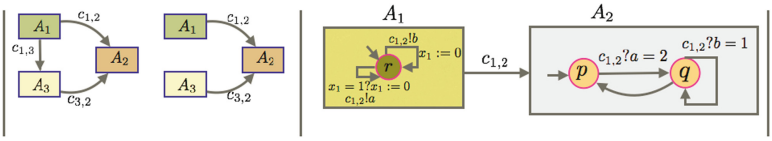

Fig. 1. The left half of the figure contains one cyclic and one acyclic topology. The right half of the figure illustrates an acyclic CTA which is not bounded context.

Theorem 1. *In the presence of global clocks, reachability is undecidable for CTA consisting of two timed automata A_1, A_2 connected by a single channel.*

A single global clock suffices; the proof can be found in [3].

Undecidable Reachability with no Global Clocks

Theorem 2. *Reachability is undecidable for acyclic CTA consisting of three one-clock timed automata without global clocks.*

Proof. We prove the undecidability by reducing the halting problem for deterministic two counter machines (see [3] for a formal definition). We consider the case of a discCTA consisting of timed automata A_1, A_2, A_3 with channels $c_{1,2}$ from A_1 to A_2 and $c_{2,3}$ from A_2 to A_3. The undecidability for the other possible topologies are discussed in [3].

The Encoding. Given a two counter machine $\mathcal{C}$, we build a discCTA $\mathcal{N}$ consisting of timed automata A_1, A_2, A_3 with channels $c_{1,2}$ from A_1 to A_2 and $c_{2,3}$ from A_2 to A_3. Corresponding to each increment, decrement and zero check instruction, we have a widget in each A_i. A widget is a "small" timed automaton, consisting of some locations and transitions between them. Corresponding to each increment/decrement instruction ℓ_i : inc or dec c, goto ℓ_j, or a zero check instruction ℓ_i : if $c = 0$, goto ℓ_j else goto ℓ_k, we have a widget $\mathcal{W}_i^{A_m}$ in each $A_m, m \in \{1, 2, 3\}$. The widgets $\mathcal{W}_i^{A_m}$ begin in a location labelled ℓ_i, and terminate in a location ℓ_j for increments/decrements, while for zero check, they begin in a location labelled ℓ_i, and terminate in a location ℓ_j or ℓ_k. Each A_m is hence obtained by superimposing (one of) the terminal location ℓ_j of a widget $\mathcal{W}_i^{A_m}$ to the initial location ℓ_j of widget $\mathcal{W}_j^{A_m}$.

We refer to initial as well as terminal locations (labelled p_{init}, p_{term}) in each $\mathcal{W}_i^{A_m}$ using the notation $(\mathcal{W}_i^{A_m}, p)$, $p \in \{p_{init}, p_{term}\}$. Note that an instruction ℓ_i can appear as initial location in a widget and a terminal location in another; thus, it is useful to remember the location along with the widget we are talking about. x_1, y_1, z_1 respectively denote the clocks used in A_1, A_2, A_3. To argue the proof of correctness, we use clocks $g_{A_1}, g_{A_2}, g_{A_3}$ respectively in A_1, A_2, A_3 which are never used in any transitions (g_{A_i} represent the total time elapse at any point in A_i).

Counter Values. The value of counter c_1 after i steps, denoted c_1^i is stored as the difference between the value of clock g_{A_2} after i steps and the value of clock

g_{A_1} after i steps. Denoting l_i to be the instruction reached after i steps, and thanks to the fact that we have locations l_i in each of A_1, A_2, A_3 corresponding to the instruction l_i, the value $c_1^i = $(value of clock g_{A_2} at location l_i of A_2) - (value of clock g_{A_1} at location l_i of A_1). Note that A_1, A_2 are not always in sync while simulating the two counter machine: A_1 can simulate the jth instruction l_j while A_2 is simulating the ith instruction l_i for $j \geq i$, thanks to the invariant maintaining the value of c_1. When they are in sync, the value of c_1 is 0. Thus, A_1 is always ahead of A_2 or at the same step as A_2 in the simulation. The value of counter c_2 is maintained in a similar manner by A_2 and A_3. To maintain the values of c_1, c_2 correctly, the speeds of A_1, A_2, A_3 are adjusted while doing increments/decrements. For instance, to increment c_1, A_2 takes 2 units of time to go from ℓ_i to ℓ_j while A_1 takes just one unit; then the value of g_{A_2} at ℓ_j is two more than what it was at ℓ_i; likewise, the value of g_{A_1} at ℓ_j is one more than what it was at ℓ_i. The channel alphabet is $\{(\ell_i, c^+, \ell_j) \mid \ell_i : \mathsf{inc}\ c\ \mathsf{goto}\ \ell_j\} \cup \{(\ell_i, c^-, \ell_j) \mid \ell_i : \mathsf{dec}\ c\ \mathsf{goto}\ \ell_j\} \cup \{(\ell_i, c{=}0, \ell_j), (\ell_i, c{>}0, \ell_k) \mid \ell_i : \mathsf{if}\ c = 0,\ \mathsf{then\ goto}\ \ell_j,\ \mathsf{else\ goto}\ \ell_k\} \cup \{zero_1, zero_2\}$.

1. Consider an increment instruction ℓ_i:inc c goto ℓ_j. The widgets $\mathcal{W}_i^{A_m}$ for $m = 1, 2, 3$ are described in Fig. 2. The one on the left is while incrementing c_1, while the one on the right is obtained while incrementing c_2.
2. The case of a decrement instruction is similar, and is obtained by swapping the speeds of the two automata (A_1, A_2 and A_2, A_3 respectively) in reaching ℓ_j from ℓ_i. Note that we preserve the invariant that A_1 is ahead of (or same as) A_2 which is ahead of (or same as) A_3 in the simulation of the two counter machine.
3. We finally consider a zero check instruction of the form ℓ_i:if $c_1 = 0$, then goto ℓ_j, else goto ℓ_k. The widgets $\mathcal{W}_i^{A_m}$ for $m = 1, 2, 3$ are described in Fig. 3. The one on the left is a zero check of c_1, while the one on the right is a zero check of c_2.

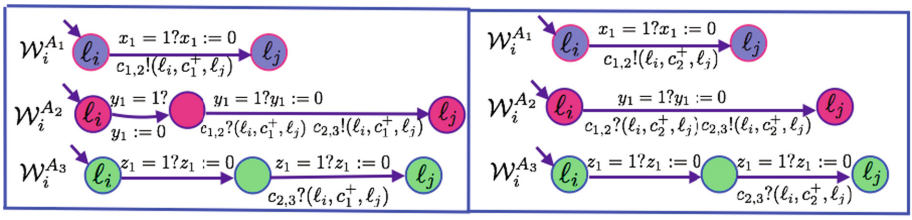

Fig. 2. Widgets corresponding to an increment c_1, c_2 instruction in A_1, A_2, A_3

Let $(\ell_0, 0, 0), (\ell_1, c_1^1, c_2^1), \ldots, (\ell_h, c_1^h, c_2^h) \ldots$ be the run of the two counter machine. ℓ_i denotes the instruction seen at the ith step and c_1^i, c_2^i respectively are the values of counters c_1, c_2 after i steps. Denote a block of transitions in A_m leading from the ith to the $(i{+}1)$st instruction as

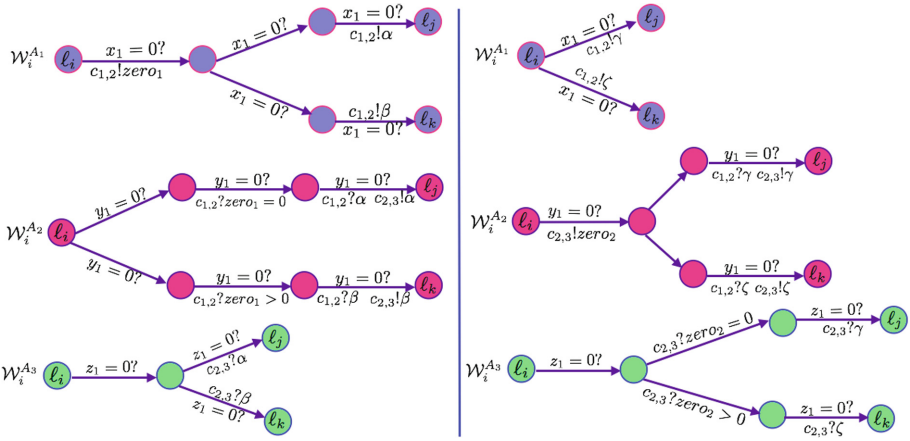

Fig. 3. Widgets corresponding to checking c_1, c_2 is 0. Let $\alpha = (\ell_i, c_1 = 0, \ell_j), \beta = (\ell_i, c_1 > 0, \ell_k), \gamma = (\ell_i, c_2 = 0, \ell_j), \zeta = (\ell_i, c_2 > 0, \ell_k)$.

$\mathcal{B}_{i,i+1} = [((\mathcal{W}_i^{A_m}, \ell_i), \nu_i^{A_m}), \ldots, ((\mathcal{W}_i^{A_m}, \ell_{i+1}), \nu_{i+1}^{A_m})]$. A run in each A_m is $\mathcal{B}_{0,1}, \mathcal{B}_{1,2}, \ldots, \mathcal{B}_{h,h+1}, \ldots$, where each block $\mathcal{B}_{h,h+1}$ of transitions in the widget $\mathcal{W}_h^{A_m}$ simulate the instruction ℓ_h, and shifts control to ℓ_{h+1}. For each m, $((\mathcal{W}_i^{A_m}, \ell_j), \nu_j^{A_m})$ represents A_m is at location ℓ_j of widget $\mathcal{W}_i^{A_m}$ with clock valuation $\nu_j^{A_m}$.

Lemma 1. *Let $\mathcal{C}$ be a two counter machine. Let c_1^h, c_2^h be the values of counters c_1, c_2 at the end of the hth instruction ℓ_h. Then there is a run of $\mathcal{N}$ which passes through widgets $\mathcal{W}_0^{A_m}, \mathcal{W}_1^{A_m}, \ldots, \mathcal{W}_h^{A_m}$ in $A_m, m \in \{1, 2, 3\}$ such that*

1. *c_1^h is the difference between the value of clock g_{A_2} on reaching location $(\mathcal{W}_h^{A_2}, \ell_h)$ and the value of clock g_{A_1} on reaching location $(\mathcal{W}_h^{A_1}, \ell_h)$. c_2^h is the difference between the value of clock g_{A_3} on reaching location $(\mathcal{W}_h^{A_3}, \ell_h)$ and the value of clock g_{A_2} on reaching location $(\mathcal{W}_h^{A_2}, \ell_h)$.*
2. *If $\mathcal{W}_h^{A_1}$ is a zero check widget for c_1 (c_2) then c_1^h (c_2^h) is 0 iff one reaches a terminal location of $\mathcal{W}_h^{A_2}$ reading α (γ) and $zero_1$ ($zero_2$) with age 0. Likewise, c_1^h (c_2^h) is > 0 iff one reaches a terminal location of $\mathcal{W}_h^{A_2}$ reading β (ζ) and $zero_1$ ($zero_2$) with age > 0.*

Machine $\mathcal{C}$ halts iff the halt widget $\mathcal{W}_{halt}^{A_m}$ is reached in $\mathcal{N}$, $m = 1, 2, 3$. [3] has the full proof.

Decidable Reachability

Theorem 3. *The reachability problem is decidable (in* EXPTIME*) for acyclic CTA consisting of two timed automata without global clocks.*

The proof proceeds by a reachability preserving reduction of the discCTA to a pushdown automaton. We give the proof idea here, correctness arguments and an example can be found in [3]. Given discCTA $\mathcal{N}$ consisting of $A = (L_A, L_A^0, \mathcal{X}_A, \Sigma, E_A, F_A)$ and $B = (L_B, L_B^0, \mathcal{X}_B, \Sigma, E_B, F_B)$, with a channel $c_{A,B}$ from A to B, we simulate $\mathcal{N}$ using a pushdown automaton $\mathcal{O}$ as follows.

Intermediate Notations. We start with $Reg(A)$ and $Reg(B)$, the corresponding region automata, and run them in an interleaved fashion. Let K be the maximal constant used in the guards of A, B. Let $[K] = \{0, 1, \ldots, K, \infty\}$. The locations Q_A (Q_B) of $Reg(A)$ ($Reg(B)$) are of the form $L_A \times [K]^{|\mathcal{X}_A|}$ ($L_B \times [K]^{|\mathcal{X}_B|}$). The transitions in $Reg(A), Reg(B)$ are as follows:

(i) A transition $(l, \nu) \xrightarrow{\checkmark} (l, \nu + 1)$ denotes a time elapse of 1 in both $Reg(A), Reg(B)$. If $\nu(x) + 1$ exceeds K for any clock x, then it is replaced with ∞. (ii) For each transition $e = (\ell, g, c_{A,B}!a, Y, \ell')$ in A we have the transition $(l, \nu) \xrightarrow{a} (l', \nu')$ in $Reg(A)$ if $\nu \models g$, and $\nu' = \nu[Y{:=}0]$. (iii) For each transition $e = (\ell, g, c_{A,B}?(a \in I), Y, \ell')$ in B we have the transition $(l, \nu) \xrightarrow{a \in I} (l', \nu')$ in $Reg(B)$ if $\nu \models g$, and $\nu' = \nu[Y{:=}0]$. (iv) For each internal transition $e = (\ell, g, \mathsf{nop}, Y, \ell')$ in A, B we have the transition $(l, \nu) \xrightarrow{\mathsf{nop}} (l', \nu')$ in $Reg(A), Reg(B)$ if $\nu \models g$, and $\nu' = \nu[Y{:=}0]$. Note that the above is an intermediate notation which will be used in the construction of the pushdown automaton $\mathcal{O}$. There is no channel between $Reg(A), Reg(B)$, and we have symbolically encoded all transitions of A, B in $Reg(A), Reg(B)$ as above.

Construction of $\mathcal{O}$: In the reduction from discCTA $\mathcal{N}$ to the pushdown automaton $\mathcal{O}$, the global time difference between A and B is stored in a counter which is part of the finite control, such that B is always ahead of A, or at the same time as A. Thus, a value $i \geq 0$ stored in the counter of finite control means that B is i units of time ahead of A. The state space of $\mathcal{O}$ is constructed using the locations of $Reg(A), Reg(B)$, and the transitions of $\mathcal{O}$ will make use of the transitions described above of $Reg(A), Reg(B)$. Internal transitions of A, B are simulated by updating the respective control locations in $Reg(A), Reg(B)$. Each unit time elapse in B results in incrementing the counter by 1, while each unit time elapse in A results in decrementing the counter. Consider a transition in A where a message m is written on the channel. The counter value when m is written tells us the time difference between B, A, and hence also the age of the message as seen from B. Assume the counter value is $i \geq 0$. If indeed m must be read in B when its age is exactly i, then B can move towards a transition where m is read, without any further time elapse. In case m must be read when its age is $j > i$, then B can execute internal transitions as well a time elapse $j - i$ so that the transition to read m is enabled. However, if m must have been read when its age is some $k < i$, then B will be unable to read m. By our interleaved execution, each time A writes a message, we make B read it before A writes further messages and proceeds. Note that this does not disallow A writing multiple messages with the same time stamp.

To ensure that the state space of $\mathcal{O}$ is finite, only counter values $\leq K$ are kept as part of the finite control of $\mathcal{O}$. When the value exceeds K, we start using the

stack (with stack alphabet $\{1\}$) to keep track of the exact value $> K$. Note that we have to keep track of the exact time difference between B, A since otherwise we will not be able to check age requirements of messages correctly.

State Space of $\mathcal{O}$: Let $\hat{Q}_x = \{q_\perp, q_1, q'_\perp, q'_1 \mid q \in Q_x, x \in \{A, B\}\}$[2]. Let $O_x = Q_x \cup \hat{Q}_x$ for $x \in \{A, B\}$. The state space of $\mathcal{O}$ is $O_A \times (O_B \times (\Sigma \cup \{\epsilon\})) \times ([K] \backslash \{\infty\})$, where the $\Sigma \cup \{\epsilon\}$ in $(O_B \times (\Sigma \cup \{\epsilon\}))$ is to remember the message (if any) written by A, which has to be read by B, and the last entry in the triple denotes the counter value. The stack alphabet is $\{\perp, 1\}$. The initial location of $\mathcal{O}$ is $\{((l^0_A, 0^{|\mathcal{X}_A|}), (l^0_B, 0^{|\mathcal{X}_B|}, \epsilon), 0) \mid l^0_A \in L^0_A, l^0_B \in L^0_B\}$ and the stack has the bottom of stack symbol $\perp$ in the initial configuration.

Transitions in $\mathcal{O}$: The transitions in $\mathcal{O}$ are as follows : For states l, l' of $\mathcal{O}$, internal transitions Δ_{int} consist of transitions of the form (l, l'); push transitions Δ_{push} consist of transitions of the form (l, a, l') for $a \in \{1, \perp\}$. Finally, we also have pop transitions Δ_{pop} of the form (l, a, l') for $a \in \{1, \perp\}$. We now describe the transitions.

1. Pop transitions Δ_{pop} : Pop transitions simulate time elapse in $Reg(A)$ as well as checking the age of a symbol being K or $> K$ while it is read from the channel.
   (a) If $(p, \nu_1) \xrightarrow{\checkmark} (p, \nu_1 + 1)$ in $Reg(A)$, and if the counter value as stored in the finite control is K, and if the stack is non-empty, then we pop the top of the stack to decrement the counter. For $l = ((p, \nu_1), (q, \nu_2, \alpha), K)$, $l' = ((p, \nu_1 + 1), (q, \nu_2, \alpha), K)$, $(l, 1, l') \in \Delta_{pop}$.
   (b) If $(p, \nu_1) \xrightarrow{\checkmark} (p, \nu_1 + 1)$ in $Reg(A)$, and if the counter value as stored in the finite control is K, and if the stack is empty, we pop $\perp$, reduce K in the finite control to $K - 1$, and push back $\perp$ to the stack. We remember that $\perp$ has been popped in the finite control, so that we push it back immediately. For $l = ((p, \nu_1), (q, \nu_2, \alpha), K), l' = ((p_\perp, \nu_1 + 1), (q, \nu_2, \alpha), K - 1)$, $(l, \perp, l') \in \Delta_{pop}$. The location $p_\perp$ tells us that $\perp$ has to be pushed back immediately.
   (c) To check that a message has age K when read, we need $i = K$, along with the fact that the stack is empty (top of stack $= \perp$). In this case, we pop $\perp$ and remember it in the finite control, and push it back. For $l = ((p, \nu_1), (q, \nu_2, \alpha), K)$, $l' = ((p, \nu_1), (q_\perp, \nu_2, \alpha), K)$, $(l, \perp, l') \in \Delta_{pop}$.
   (d) To check that a message has age $> K$ when read, we need $i = K$, along with the fact that the stack is non-empty (top of stack $= 1$). In this case, we pop 1 and remember it in the finite control, and push it back. For $l = ((p, \nu_1), (q, \nu_2, \alpha), K), l' = ((p, \nu_1), (q_1, \nu_2, \alpha), K)$, $(l, 1, l') \in \Delta_{pop}$.
2. Push transitions Δ_{push} : Push transitions simulate time elapse in $Reg(B)$, and also aid in simulating checking the age of a symbol being K or $> K$ while being read from the channel.
   (a) Push $\perp$ to the stack while reducing counter value from K to $K-1$ (1(b)). For $l = ((p_\perp, \nu_1), (q, \nu_2, \alpha), K-1)$ and $l' = ((p, \nu_1), (q, \nu_2, \alpha), K-1)$, $(l, \perp, l') \in \Delta_{push}$.

[2] The $q_\perp, q_1$ are used to remember the topmost symbol of the stack while in location q.

(b) Push $\perp$ to the stack before checking the age of a message is K (1(c)). For $l = ((p,\nu_1),(q_\perp,\nu_2,\alpha),K)$ and $l' = ((p,\nu_1),(q'_\perp,\nu_2,\alpha),K))$, $(l,\perp,l') \in \Delta_{push}$.

(c) Push 1 to the stack before checking the age of a message is $> K$ (1(d)). For $l = ((p,\nu_1),(q_1,\nu_2,\alpha),K)$ and $l' = ((p,\nu_1),(q'_1,\nu_2,\alpha),K)$, $(l,1,l') \in \Delta_{push}$.

(d) If $(q,\nu_2) \xrightarrow{\checkmark} (q,\nu_2+1)$ in $Reg(B)$, and if the counter value as stored in the finite control is K, then we push a 1 on the stack to represent the counter value is $> K$. That is, $(l,1,l') \in \Delta_{push}$ for $l = ((p,\nu_1),(q,\nu_2,\alpha),K)$ and $l' = ((p,\nu_1),(q,\nu_2+1,\alpha),K)$.

3. Internal transitions Δ_{int}: Transitions of Δ_{int} simulate internal transitions of $Reg(A), Reg(B)$ as well as $\checkmark$- transitions as follows:

(a) Let $l = ((p,\nu_1),(q,\nu_2,\alpha),i)$, $l' = ((p',\nu'_1),(q,\nu_2,\alpha),i)$ be states of $\mathcal{O}$. $(l,l') \in \Delta_{int}$ if $(p,\nu_1) \xrightarrow{\mathsf{nop}} (p',\nu'_1)$ is an internal transition in $Reg(A)$. The same can be said of internal transitions in $Reg(B)$ updating q,ν_2, leaving α, i and (p,ν_1) unchanged.

(b) For $l = ((p,\nu_1),(q,\nu_2,\alpha),i)$ with $0 \le i < K$, and $l' = ((p,\nu_1),(q,\nu_2+1,\alpha),i+1)$, $(l,l') \in \Delta_{int}$ if $(q,\nu_2) \xrightarrow{\checkmark} (q,\nu_2+1)$ is a $\checkmark$-transition in $Reg(B)$. Note that $i+1 \le K$.

(c) For $l = ((p,\nu_1),(q,\nu_2,\alpha),i)$ with $0 < i \le K$, and $l' = ((p,\nu_1+1),(q,\nu_2,\alpha),i-1)$, $(l,l') \in \Delta_{int}$ if $(p,\nu_1) \xrightarrow{\checkmark} (p,\nu_1+1)$ is a $\checkmark$-transition in $Reg(A)$.

(d) For $l = ((p,\nu_1),(q,\nu_2,\epsilon),i), l' = ((p',\nu'_1),(q,\nu_2,a),i)$, $(l,l') \in \Delta_{int}$ if $(p,\nu_1) \xrightarrow{a} (p',\nu'_1)$ is a transition in $Reg(A)$ corresponding to a transition from p to p' which writes a onto the channel $c_{A,B}$.

(e) For $i < K$, and $i \in I$, $l = ((p,\nu_1),(q,\nu_2,a),i), l' = ((p,\nu_1),(q',\nu'_2,\epsilon),i)$, $(l,l') \in \Delta_{int}$ if $(q,\nu_2) \xrightarrow{a \in I} (q',\nu'_2)$ is a transition in $Reg(B)$ corresponding to a transition from q to q' which reads a from the channel $c_{A,B}$ and checks its age to be in interval I.

(f) To check that a message has age K when read, we need the counter value i to be K, along with the top of stack $= \perp$. See 1(c), 2(b), and then we use transition $(l,l') \in \Delta_{int}$ for $l = ((p,\nu_1),(q'_\perp,\nu_2,m),K), l' = ((p,\nu_1),(r,\nu'_2,\epsilon),K)$, if $(q,\nu_2) \xrightarrow{m \in [K,K]} (r,\nu'_2)$ is a read transition in $Reg(B)$.

(g) To check that a message has age $> K$ when read, we need $i = K$, along with the fact that the stack is non-empty (top of stack $= 1$). See 1(d), 2(c), and then $(l,l') \in \Delta_{int}$ for $l = ((p,\nu_1),(q'_1,\nu_2,m),K), l' = ((p,\nu_1),(r,\nu'_2,\epsilon),K)$, if $(q,\nu_2) \xrightarrow{m \in (K,\infty)} (r,\nu'_2)$ is a read transition in $Reg(B)$. (age requirements $\ge K$ are checked using this or the above).

The correctness of the construction is proved in [3] using Lemmas 2 and 3.

Lemma 2. *If $((l_A,\nu_A),(l_B,\nu_B,a),i)$ is a configuration in $\mathcal{O}$, along with a stack consisting of $1^j\perp$, then message a has age $i+j$, A is at l_A, B is at l_B, and B is $i+j$ time units ahead of A.*

Lemma 3. *Let $\mathcal{N}$ be a* disc*CTA with timed automata A, B connected by a channel $c_{A,B}$ from A to B. Assume that starting from an initial configuration $((l_A^0, 0^{|\mathcal{X}_A|}), (l_B^0, 0^{|\mathcal{X}_B|}), \epsilon)$ of $\mathcal{N}$, we reach configuration $((l_A, \nu_1), (l_B, \nu_2), w.(m, i))$ such that $w \in (\Sigma \times \{0, 1, \ldots, i\})^*$, and $(m, i) \in \Sigma \times [K]$ is read off by B from (l_B, ν_2). Then, from the initial configuration $((l_A^0, 0^{|\mathcal{X}_A|}), (l_B^0, 0^{|\mathcal{X}_B|}, \epsilon), 0)$ with stack contents $\perp$ of $\mathcal{O}$, we reach one of the following configurations*

(i) $((p_A, \nu'_A), (l_B, \nu_2, m), i)$ *with stack contents* $\perp$ *if* $i \leq K$,
(ii) $((p_A, \nu'_A), (l_B, \nu_2, m), h)$ *with stack contents* $1^j\perp$, $j > 0$ *if* $i > K$ *and* $h + j = i$.

Moreover, it is possible to reach (l_A, ν_1) from (p_A, ν'_A) in A after elapse of i units of time. The converse is also true.

Complexity: Upper and Lower bounds The EXPTIME upper bound is easy to see, thanks to the exponential blow up incurred in the construction of $\mathcal{O}$ using the regions of A and B, and the fact that reachability in a pushdown automaton is linear. The best possible lower bound we can achieve as of now is NP-hardness, as described in [3].

4.1 Bounded Context discCTA

Before winding up with discCTA, we consider bounded context discCTA and show that the reachability problem is decidable even when having global clocks. Given a discCTA, a *context* is a sequence of transitions in the discCTA where only one automaton is *active* viz., reading from at most one fixed channel, but possibly writing to many channels that it can write to (this cannot be the one it reads from). Thus, (a) a context is simply a sequence of transitions where a single automaton A_i performs channel operations, and (b) in a context, A_i can read from at most one channel. A *context switch* happens when either a different automaton A_j, $j \neq i$ performs channel operations, or when A_i reads from a different channel.

Definition 1. *A* disc*CTA $\mathcal{N}$ is bounded context (*disc*CTA-*bc*) if the number of context switches in any run of $\mathcal{N}$ is bounded above by some $B \in \mathbb{N}$.*

See the right part of Fig. 1 for an example of a discCTA consisting of two processes A_1, A_2, where A_1 writes on $c_{1,2}$ to A_2. This acyclic discCTA is not bounded context. There is a run where A_1 writes an a after every one time unit, and A_2 reads an a once in two time units. There is also a run where A_1 writes b onto the channel whenever it pleases and A_2 reads it one time unit after it is written.

Theorem 4. *Reachability is decidable for* disc*CTA-*bc*, even in the presence of global clocks.*

The Idea: Let K be the maximal constant used in the discCTA with bounded context $\leq B$, and let $[K] = \{0, 1, \ldots, K, \infty\}$. For $1 \leq i \leq n$, let $A_i =$

$(L_i, L_i^0, Act, \mathcal{X}_i, E_i, F_i)$ be the n automata in the discCTA. Let $c_{i,j}$ denote the channel to which A_i writes to and A_j reads from. We translate the discCTA into a bounded phase, multistack pushdown system (BMPS) $\mathcal{M}$ preserving reachability. A multistack pushdown system (MPS) is a timed automaton with multiple untimed stacks. A *phase* in an MPS is one where a fixed stack is popped, while pushes can happen to any number of stacks. A change of phase occurs when there is a change in the stack which is popped. See [3] for a formal definition. We use Lemma 4 (proof in [3]) to obtain decidability after our reduction.

Lemma 4. *The reachability problem is decidable for BMPS.*

Encoding into BMPS. The BMPS $\mathcal{M}$ uses two stacks $W_{i,j}$ and $R_{i,j}$ to simulate channel $c_{i,j}$. The control locations of $\mathcal{M}$ keeps track of the locations and clock valuations of all the A_i, as n pairs $(p_1, \nu_1), \ldots, (p_n, \nu_n)$ with $\nu_i \in [K]$ for all i;[3] in addition, we also keep an ordered pair (A_w, b) consisting of a number $b \leq B$ to count the context switch in the discCTA and also remember the active automaton $A_w, w \in \{1, 2, \ldots, n\}$. To simulate the transitions of each A_i, we use the pairs (p_i, ν_i), keeping all pairs (p_j, ν_j) unchanged for $j \neq i$. An initial location of $\mathcal{M}$ has the form $((l_1^0, \nu_1), \ldots, (l_n^0, \nu_n), (A_i, 0))$ where $l_i^0 \in L_i^0$, $\nu_i = 0^{|\mathcal{X}_i|}$; the pair $(A_i, 0)$ denotes context 0, and A_i is some automaton which is active in context 0 (A_i writes to some channels).

Transitions of $\mathcal{M}$. The internal transitions Δ_{in} of $\mathcal{M}$ correspond to any internal transition in any of the A_is and change some (p, ν) to (q, ν') where ν' is obtained by resetting some clocks from ν. These take place irrespective of context switch.

The push and pop transitions (Δ_{push} and Δ_{pop}) of $\mathcal{M}$ are more interesting. Consider the kth context where A_j is active in the discCTA. In $\mathcal{M}$, this information is stored as (A_j, k). In the kth context, A_j can read from atmost one fixed channel $c_{l,j}$; it can also write to several channels $c_{j,i_1}, \ldots, c_{j,i_k} \neq c_{l,j}$, apart from time elapse/internal transitions. All automata other than A_j participate only in time elapse and internal transitions. When A_j writes a message m to channel c_{j,i_h} in the discCTA, it is simulated by pushing message m to stack W_{j,i_h}. All time elapses $t \in [K]$ are captured by pushing t to all stacks. Δ_{push} has transitions pushing a message m on a stack W_{i,j_k}, or pushing time elapse $t \in [K]$ on all stacks.

When A_j is ready to read from channel $c_{l,j}$ (say), the contents of stack $W_{l,j}$ are shifted to stack $R_{l,j}$ if the stack $R_{l,j}$ is empty. Assuming $R_{l,j}$ is empty, we transfer contents of $W_{l,j}$ to $R_{l,j}$. The stack to be popped is remembered in the finite control of $\mathcal{M}$: the pair (p, ν), $p \in L_j$ is replaced with $(p^{W_{l,j}}, \nu)$. As long as we keep reading symbols $t \in [K]$ from $W_{l,j}$, we remember it in the finite control of $\mathcal{M}$ by adding a tag t to locations $(p^{W_{l,j}}, \nu)$ $(p \in L_j)$ making it $((p^{W_{l,j}})_t, \nu)$. When a message m is seen on top of $W_{l,j}$, with $((p^{W_{l,j}})_t, \nu)$ in the finite control of $\mathcal{M}$, we push (m, t) to stack $R_{l,j}$, since t is indeed the time that elapsed after m was written to channel $c_{l,j}$. When we obtain $t' \in [K]$ as the top of stack $W_{l,j}$,

[3] The global clock valuations are maintained as a separate tuple; for simplicity, we omit this detail here. Our proof works in the presence of global clocks easily.

with $((p^{W_{l,j}})_t, \nu)$ in the finite control, we add t' to the finite control obtaining $((p^{W_{l,j}})_{t+t'}, \nu)$. The next message m' has age $t + t'$ and so on, and stack $R_{l,j}$ is populated. When $W_{l,j}$ becomes empty, the finite control is updated to $(p^{R_{l,j}}, \nu)$ and A_j starts reading from $R_{l,j}$. If $R_{l,j}$ is already non-empty when A_j starts reading, it is read off first, and when it becomes empty, we transfer $W_{l,j}$ to $R_{l,j}$. A time elapse t'' between reads and/or reads/writes of A_j is simulated by pushing t'' on all stacks, to reflect the increase in age of all messages stored in all stacks.

Phases of $\mathcal{M}$ are bounded. Each context switch in the discCTA results in $\mathcal{M}$ simulating a different automaton, or simulating the read from a different channel. Assume that every context switch of the discCTA results in some automaton reading off from some channel. Correspondingly in $\mathcal{M}$, we pop the corresponding R-stack, and if it goes empty, pop the corresponding W-stack filling up the R-stack. Once the R-stack is filled up, we continue popping it. This results in atmost two phase changes (some $R_{i,j}$ to $W_{i,j}$ and $W_{i,j}$ to $R_{i,j}$) for each context in the discCTA. An additional phase change is incurred on each context switch (a different stack $R_{k,l}$ is popped in the next context). Note that $\mathcal{M}$ does not pop a stack unless a read takes place in some automaton, and the maximum number of stacks popped is 2 per context. $\mathcal{M}$ is hence a $3B$ bounded phase MPS. A detailed proof of correctness and an example can be seen in [3].

Table 1. Summary of results for discCTA. k-discCTA represents discCTA with k discrete timed automata, $k \in \mathbb{N}$. $* -$ discCTA has finitely many discrete timed automata involved.

Acyclic discCTA	Global clocks	Channels	Reachability	Where
2-discCTA	Yes (1 global clock)	1	Undecidable	[3]
3-discCTA time	No	2	Undecidable	Theorem 2
2-discCTA	No	1	Decidable	Theorem 3
$* -$ discCTA bounded context	Yes	any	Decidable	Theorem 4

4.2 discCTA Summary

Table 1 summarizes our exhaustive characterisation for discCTA. The tightness of the lower bound (NP−hardness) of our decidability result (Theorem 3) is open. We mention the possible extensions to the model of discCTA as studied in this paper which we conjecture will preserve the decidability result in Theorem 3.

1. If we allow diagonal constraints of the form $x - y \sim c$ where x, y are clocks and $c \in \mathbb{N}$, Theorem 3 continues to hold. In the proof, given a discCTA $\mathcal{N}$ consisting of timed automata A, B connected by the channel $c_{A,B}$ from A to B, we construct a one counter automaton $\mathcal{O}$ using $Reg(A)$ and $Reg(B)$. We can easily track the difference between two clocks x, y in $Reg(A)$ or $Reg(B)$, thereby handling diagonal constraints.

2. The initial age of a newly written message in a channel is set to 0. This can be generalized in two ways : (i) allowing the initial age of a message to be some $j \in \mathbb{N}$, or (ii) assigning the value of some clock x as the initial age. The construction of $\mathcal{O}$ is such that each time A writes a message $m \in \Sigma$ to the channel, m is remembered in the finite control of $\mathcal{O}$ (transition 3(d) in the proof of Theorem 3). While simulating the read by B of the message m (transitions 3(e), (f), (g) in the proof of Theorem 3), the value i in the finite control of $\mathcal{O}$ along with the top of the stack determines whether the age of m is $< K, = K$ or $> K$, where K is the maximal constant used in A, B. This is used to see if the age constraint of m is met; the age of m when it is read is same as the time difference between B, A. We can adapt this for an initial age $j > 0$, by remembering (m, j) in the finite control of $\mathcal{O}$. If the counter value is $i < K$, then the age of the message is $j + i$, while if it is K and the top of stack is $\perp$, then the age of m is $j + K$, and it is $> j + K$ if the top of stack is not $\perp$. Checking the age constraint of m correctly now boils down to using $j + i$ and verifying if the constraint is satisfied.

References

1. Abdulla, P.A., Nylén, A.: Timed Petri nets and BQOs. In: Colom, J.-M., Koutny, M. (eds.) ICATPN 2001. LNCS, vol. 2075, pp. 53–70. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45740-2_5
2. Abdulla, P.A., Atig, M.F., Cederberg, J.: Timed lossy channel systems. In: FSTTCS 2012, LIPIcs, 15–17 December 2012, Hyderabad, India, vol. 18, pp. 374–386. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2012)
3. Abdulla, P.A., Atig, M.F., Krishna, S.N.: What is decidable about perfect timed channels? CoRR, abs/1708.05063 (2017)
4. Abdulla, P.A., Atig, M.F., Stenman, J.: Dense-timed pushdown automata. In: Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, 25–28 June 2012, pp. 35–44. IEEE Computer Society (2012)
5. Abdulla, P.A., Jonsson, B.: Verifying programs with unreliable channels. In: LICS. IEEE Computer Society (1993)
6. Abdulla, P., Mahata, P., Mayr, R.: Dense-timed Petri nets: checking zenoness, token liveness and boundedness. Log. Methods Comput. Sci. **3**(1) (2007). https://doi.org/10.2168/LMCS-3(1:1)2007
7. Akshay, S., Gastin, P., Krishna, S.N.: Analyzing timed systems using tree automata. In: 27th International Conference on Concurrency Theory, CONCUR 2016, 23–26 August 2016, LIPIcs, Québec City, Canada, vol. 59, pp. 27:1–27:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
8. Akshay, S., Genest, B., Hélouët, L.: Decidable classes of unbounded Petri nets with time and urgency. In: Kordon, F., Moldt, D. (eds.) PETRI NETS 2016. LNCS, vol. 9698, pp. 301–322. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39086-4_18
9. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**(2), 183–235 (1994)

10. Bérard, B., Cassez, F., Haddad, S., Lime, D., Roux, O.H.: Comparison of different semantics for time Petri nets. In: Peled, D.A., Tsay, Y.-K. (eds.) ATVA 2005. LNCS, vol. 3707, pp. 293–307. Springer, Heidelberg (2005). https://doi.org/10.1007/11562948_23
11. Bhave, D., Dave, V., Krishna, S.N., Phawade, R., Trivedi, A.: A perfect class of context-sensitive timed languages. In: Brlek, S., Reutenauer, C. (eds.) DLT 2016. LNCS, vol. 9840, pp. 38–50. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53132-7_4
12. Bocchi, L., Lange, J., Yoshida, N.: Meeting deadlines together. In: Aceto, L., de Frutos Escrig, D. (eds.) 26th International Conference on Concurrency Theory (CONCUR 2015), Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, vol. 42, pp. 283–296. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2015)
13. Bouajjani, A., Echahed, R., Robbana, R.: On the automatic verification of systems with continuous variables and unbounded discrete data structures. In: Antsaklis, P., Kohn, W., Nerode, A., Sastry, S. (eds.) HS 1994. LNCS, vol. 999, pp. 64–85. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60472-3_4
14. Bouajjani, A., Habermehl, P.: Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations. Theor. Comput. Sci. **221**(1–2), 211–250 (1999)
15. Bouchy, F., Finkel, A., Sangnier, A.: Reachability in timed counter systems. Electron. Notes Theor. Comput. Sci. **239**, 167–178 (2009)
16. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM **30**(2), 323–342 (1983)
17. Cécé, G., Finkel, A.: Verification of programs with half-duplex communication. Inf. Comput. **202**(2), 166–190 (2005)
18. Chambart, P., Schnoebelen, P.: Mixing lossy and perfect FIFO channels. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 340–355. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85361-9_28
19. Clemente, L., Herbreteau, F., Stainer, A., Sutre, G.: Reachability of communicating timed processes. In: Pfenning, F. (ed.) FoSSaCS 2013. LNCS, vol. 7794, pp. 81–96. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37075-5_6
20. Clemente, L., Lasota, S.: Timed pushdown automata revisited. In: 30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, 6–10 July 2015, pp. 738–749. IEEE Computer Society (2015)
21. Clemente, L., Lasota, S., Lazic, R., Mazowiecki, F.: Timed pushdown automata and branching vector addition systems. In: 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, 20–23 June 2017, pp. 1–12. IEEE Computer Society (2017)
22. Dang, Z.: Pushdown timed automata: a binary reachability characterization and safety verification. Theor. Comput. Sci. **302**(1–3), 93–121 (2003)
23. Emmi, M., Majumdar, R.: Decision problems for the verification of real-time software. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 200–211. Springer, Heidelberg (2006). https://doi.org/10.1007/11730637_17
24. Ganty, P., Majumdar, R.: Analyzing real-time event-driven programs. In: Ouaknine, J., Vaandrager, F.W. (eds.) FORMATS 2009. LNCS, vol. 5813, pp. 164–178. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04368-0_14
25. Krcal, P., Yi, W.: Communicating timed automata: the more synchronous, the more difficult to verify. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 249–262. Springer, Heidelberg (2006). https://doi.org/10.1007/11817963_24

26. La Torre, S., Madhusudan, P., Parlato, G.: Context-bounded analysis of concurrent queue systems. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 299–314. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_21
27. Pachl, J.K.: Reachability problems for communicating finite state machines. Ph.D. thesis, Faculty of Mathematics, University of Waterloo, Ontario (1982)
28. Trivedi, A., Wojtczak, D.: Recursive timed automata. In: Bouajjani, A., Chin, W.-N. (eds.) ATVA 2010. LNCS, vol. 6252, pp. 306–324. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15643-4_23