Overview of classical and recent decidability and complexity results for networks of timed communicating automata (tutorial)

Lorenzo Clemente, University of Warsaw, Poland

October 10, 2018

Abstract

We present an overview of decidability and undecidability results for the control state reachability problem of systems of timed communicating automata.

1 Communicating automata (untimed)

In order to present the full picture for timed communicating automata, it is convenient to start from some classic results in the untimed setting. Let Σ be a finite alphabet, let P be a finite set of *process names*, let C be a finite set of *channel names*, and let r be a *ownership* function assigning to each channel $c \in C$ an ordered pair of processes $r(c) = pq \in P \times P$. A system of *communicating automata* (CA) consists of a family of nondeterministic finite automata $(A_p)_{p \in P}$, where each automaton A_p has transitions labelled by either a *send action* $c!a$ with $r(c)$ of the form pq or a *receive action* $c?a$ with $r(c)$ of the form qp. A channel c is interpreted as a string $w_c \in \Sigma^*$. A send action $c!a$ transforms w_c into aw_c, and a receive action $c?a$ transforms aw_c into w_c. Thus channels follow a *first-in first-out* policy (FIFO) and are *perfect* in the sense that they do not lose messages. The semantics of a CA is an infinite transition system recording the current control location of each automaton and channel contents for each channel. From an intuitive point of view, the only logical constraint that is imposed by communication is that a message cannot be received before it is sent, and this simple and basic intuition is sufficient for a complete understanding of CA. The *control state reachability problem* (often abbreviated as *reachability* in the following) amounts to determine whether, given initial and final control locations for each process, there exists a run where channels are empty both at the beginning and at the end of the run.

It is a classic result that reachability is undecidable in general for CA [5]. In order to understand the border between decidability and undecidability, it is convenient to introduce the notion of *communication topology*, which is the directed multigraph $T = (P, E, r)$ where P is the set of vertices, $E \subseteq C$ is the set of edges consisting of all channels c appearing in communication actions of the A_p's[1], and r is the ownership mapping. As a refinement of the undecidability above, in fact the following basic topologies are already undecidable:

$$p \to p \qquad p \rightleftarrows q \qquad p \rightrightarrows q.$$

This is shown by using the channel(s) to simulate directly the behaviour of a Turing machine. Any message inside the channel can be accessed without destroying the channel's contents by performing suitable rotations.

In fact, the border between decidable and undecidable topologies is well understood:

Theorem 1 ([21, 23]) *Reachability of CA is decidable if, and only if, the topology is a polyforest.*

[1] For example, if there exists a message a s.t. p can do $c!a$ and q can do $c?a$.

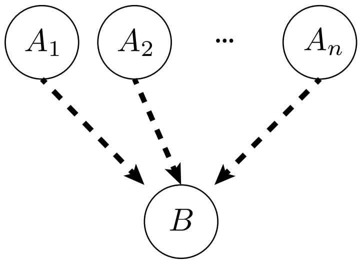


Figure 1: Polyforest topology solving the intersection non-emptiness problem for finite automata.

A *polyforest* is a directed multigraph which does not have any undirected cycle; a weakly connected component of a polyforest (i.e., connected in the underlying undirected graph) is called a *polytree*, and thus a polyforest is a disjoint union of polytrees. The crucial idea to show decidability for polyforest topologies is to observe that an accepting run exists if, and only if, an accepting run exists where all channels are bounded to size at most 1. This is achieved by using a scheduler that gives priority to receivers over senders: A send action is executed only if a matching receive action can be immediately fired, thus keeping the channels bounded. Since there are no cyclical send-receive dependencies, such *eager runs* suffice to decide the reachability problem. In the following, we will call this the *reordering trick*.

It is interesting to note that, while channels are oriented by definition, regarding decidability of the reachability problem it does not actually matter the orientation of channels.

Theorem 2 *In the case of polyforest topologies, the reachability problem for CA is* PSPACE*-complete.*

Membership in PSPACE follows by taking the synchronized product of all processes and deciding reachability in NL in the resulting (exponentially larger) transition system (plus Savitch's theorem NPSPACE $\subseteq$ PSPACE). Hardness for PSPACE follows from the fact that the following classical problem is PSPACE-hard (folklore): Given k DFA's $A_1, \ldots, A_k$, decide whether $\bigcap_{i=1}^{k} L(A_i) \neq \emptyset$. Indeed, the latter problem reduces to the CA reachability problem over the polyforest topology shown in Fig. 1: Each process A_i's sends to a new process B a word in its own language, and the process B just checks that all received words are the same, with transitions of the form (for every input symbol a)

$$p \xrightarrow{A_1?a; A_2?a; \cdots A_n?a} p.$$

Extensions. In order to recover decidability for arbitrary or more general topologies, various approaches have been proposed in the literature:

- Most famously, arbitrary messages can be lost from the channels (weaker semantics), giving rise to *lossy channel systems*, for which reachability is solved with well-quasi ordering techniques and is decidable for every topology [7, 3, 13].
- Alternatively, messages inside the channel can be reordered (thus it suffices to count the number of messages; weaker semantics), giving rise to a model equivalent to *Petri nets*, for which reachability is again decidable for every topology [20, 16, 19].
- Communication can be restricted to be half-duplex (stronger semantics: at most one channel is empty for each reachable configuration [6]), or, even more generally, *mutex* (at most one non-empty channel per weakly connected component [15]).
- Communication can be restricted to *bounded context switching* (stronger semantics), where a context is an arbitrarily long execution where only one process is reading from one queue (and unrestricted writings) [23].

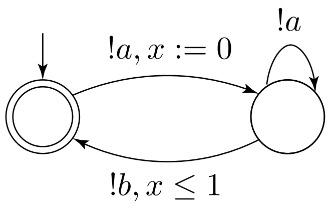


Figure 2: A communicating timed automaton A recognising L from (1).

In general, over-approximations (such as the first two points above) are sound for witnessing safety properties of the system, while under-approximations (such as the last two points above) are sound to catch bugs violating safety properties of the system.

Some works also studied the trade-offs between the perfect, FIFO semantics and its weaker lossy and bag semantics, and their interactions with the communication topologies:

- Topologies of FIFO channels mixing the perfect and lossy semantics have been studied in [8].
- Topologies of perfect channels mixing the FIFO and bag semantics have been studied in [11]

Another line of research considers more powerful communicating processes, such as counter automata [14], pushdown automata [15], and, in the context of lossy channel systems, register [1], and timed automata [2]. We focus in the rest of this note on communicating timed automata, in the context of perfect, FIFO channel systems.

2 Timed communicating automata in discrete time

2.1 Timed automata

A *timed automaton* (TA) is a finite-state automaton equipped with a finite set of real-valued *clocks* which can be reset to zero, tested with boolean combinations of diagonal constraints of the form $x - y \leq k$ with $k \in \mathbb{Z}$, and globally increased by the same amount by the *elapse of time.* As an example of timed properties that a TA can specify, consider the following property (timed language):

$$L = \text{“every symbol } a \text{ is followed by some } b \text{ after at most one timed unit”}.$$

Formally,

$$L = \{(a_1, t_1) \cdots (a_n, t_n) \in (\Sigma \times \mathbb{R}_{\geq 0})^* \mid \forall i \cdot a_i = a \text{ implies } \exists j \cdot a_j = b \text{ and } t_j - t_i \leq 1\}\,. \qquad (1)$$

The timed automaton A in Fig. 2 using one clock x recognises the language L above (ignoring the “!” symbol for the time being).

2.2 Untimed channels

We consider *timed communicating automata* (TCA), which are networks of timed automata communicating over perfect, FIFO channels [18]. Each timed automaton has its own set of *local clocks* that constrain its timed executions. Time is global and homogeneous, meaning that all local clocks of all timed processes elapse at the same time. Thus, additionally to explicit synchronisation due to the exchange of messages (or at least partial ordering), the processes also implicitly synchronise due to the elapse of time, which makes the model closer to the undecidability border.

For instance, turning the previous timed automaton A into a TCA sending messages to a new TCA B (depicted in Fig. 3), shows that, even if the topology does not have cycles, the execution

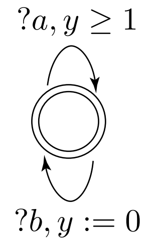


Figure 3: Communicating timed automaton B.

reordering trick leading to decidability in Theorem 1 now fails, since A can send unboundedly many messages a's within the first time unit, but B cannot receive any such a until the first time unit is elapsed. The topology

$$A \to B$$

above turns out to be decidable (with untimed channels). The main obstacle to the reordering trick is that B needs some time to elapse before being able to receive messages. The idea is to remove global time and replace it with *local time*, whereby both processes can elapse their local time independently; time synchronisations occur when communication happens. We could allow B to move to the future in order to receive messages as soon as they are sent. This is sound since it respects the causality of communication: Each message is received *after* it is sent. We keep track of the desynchronisation between A and B by introducing a new structure. In the case of discrete time, this would be just an integer counter c. Since B should not be allowed to move to the past w.r.t. A, the counter is always a non-negative integer. Every time B elapses one time unit we increment the counter

$$c := c + 1.$$

Every time A elapses one time unit, we instead decrement the counter

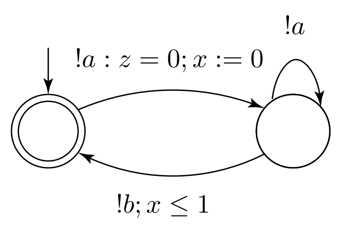
$$c := c - 1.$$

Since c is constrained to take non-negative values, this automatically ensures that B cannot lag behind A. In this way, we can always schedule B far enough in the future as to keep the channel bounded by matching a send $!m$ of A with a matching receive $?m$ of B. Note the trade-off: We bound the length of the channel at the cost of introducing a potentially unbounded counter. Finally, we want no desynchronisation at the beginning $c = 0$ and, if we allow processes to elapse an arbitrary amount of time at the end of the run (which is usually the case for timed automata, i.e., without location invariants), we are happy with any counter value $c \geq 0$ at the end of the run, which leads to the coverability semantics of a counter automaton with no zero tests. Putting these considerations together and generalising it to an arbitrary number of processes, we obtain the following characterisation of decidability for TCA with untimed channels, which, perhaps surprisingly, is the same as in the untimed setting.

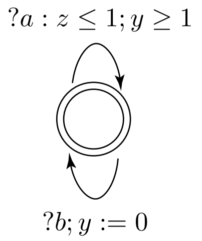
Theorem 3 ([10]) *The reachability problem for TCA with untimed channels is decidable if, and only if, the topology is a polyforest.*

Decidability for more processes is established by introducing one non-negative integer counter for each channel, keeping track of the desynchronisation between sender and receiver (in order to schedule receptions eagerly and keep the channel bounded). In fact, for weakly connected topologies, the reachability problem for TCA with untimed channels is equivalent to the *coverability problem for Petri nets*, showing that such counters are in a sense unavoidable. Since the coverability problem for Petri nets is EXPSPACE-complete [22], this yields the following result.

Theorem 4 ([10]) *The complexity of the reachability problem for TCA with untimed channels over polyforest topologies is EXPSPACE-complete.*

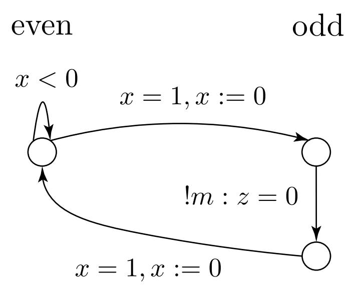


(a) Communicating timed automaton A'.

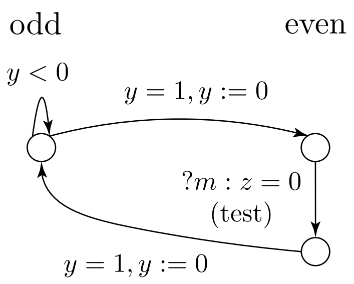


(b) Communicating timed automaton B'.

Figure 4: The increased expressive power of timing channel constraints.



(a) Communicating timed automaton C.



(b) Communicating timed automaton D.

Figure 5: Simulating channel emptiness tests with timed channels.

On the other hand, if we forbid timed processes to elapse arbitrary amounts of time at the end of the run, then we obtain equivalence (always for decidable topologies) with the arguably harder *reachability problem* for Petri nets, which has recently shown to be non-elementary [12].

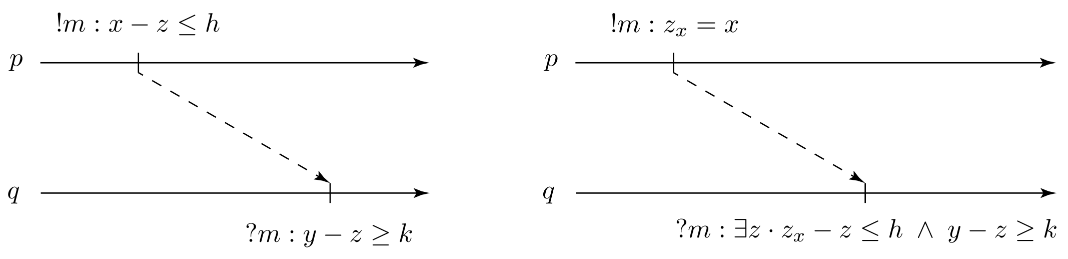
2.3 Timed channels

We now consider a more general setting, where also the channels themselves contain timing information. In other words, when a process sends a message, it can additionally initialise a number of *channel clocks* according to some clock constraint between local and channel clocks, and similarly, upon reception, the channel clocks of the message being received are checked according to some constraint of the same form: which are boolean combinations of basic constraints

$$x - z \leq k, \tag{2}$$

where x is a local clock, z is a channel clock, and $k \in \mathbb{Z}$. For example, consider the two TCA A' and B' shown in Fig. 4. Now A' sends a message a to B' with initial age 0, and B can receive a's only if their age is at most 1. This is in conflict with its local timing constraint that at least one time unit elapsed since the last b was received. Altogether, the first a is received after exactly one time unit of permanence in the channel. This shows the subtle interaction of local and channel timing constraints of TCA.

To show the power of the extra synchronisation facility offered by having timed channels, consider solving the problem of testing a channel for *emptiness*. This cannot be done in untimed CA (where only non-emptiness can be tested by the receiver), and even in TCA with untimed channels. However, with timed channels the receiver can test whether the channel is empty by following the following protocol, in coordination with the sender: At even moments, the two processes simulate the original transitions. At odd moments, the sender always sends a special control message m with initial age 0;



(a) Original communication.

(b) Simulation by simplified copy-send.

Figure 6: Copy-send simulate arbitrary transmission constraints.

if the receiver wants to check whether the channel is empty, it suffices to receive a control message m with age 0 at an odd instant: Since this is the only thing which is sent on a odd position, it follows that the channel was empty in the original system. Otherwise, the receiver just discards the control message m. Cf. Fig. 5 for the sending gadget C and the testing gadget of the receiver D, where x and y are local clocks, and z is a channel clock.

Thus, in general timed channel exhibit more intricate behaviours than their untimed counterparts. Nonetheless, in some simple situations we can still decide the reachability problem for TCA with timed channels. Assume for simplicity that a transmission occurs where the initial age of the message is set to 0,

$$!m : z = 0,$$

and that, at the time of reception, the following non-diagonal constraint on the channel clock z needs to be tested:

$$?m : z \leq k.$$

By following the rescheduling trick and introducing a non-negative integer counter c measuring the desynchronisation between sender and receiver, we readily notice that the value of c at the time of reception is in fact the age of m that we want to test. Thus, the channel reception above is simulated by a counter test

$$c \leq k.$$

Similar tests arise for other non-diagonal constraints $\geq k$, and so on.

It remains to address how to simplify transmissions constraints to the form $z = 0$ above, and how to remove diagonal reception constraints $x - z \leq k$ in favour of non-diagonal ones of the form $z \leq k$ above. We call a TCA in the form above *simple*. While we do not present the involved transformations in detail, the basic intuitions are simple:

1. First we simplify transmissions constraints to be of the form

   $$z_x = x,$$

   i.e., copies of local clocks are sent along the channel. A pair of send and receive actions

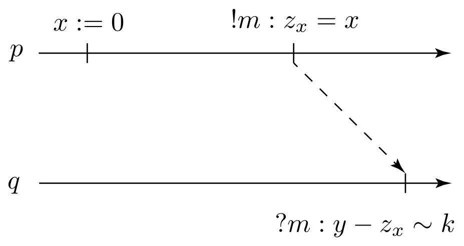
   $$!m : x - z \leq h \qquad \text{and} \qquad ?m : y - z \geq k$$

   is simulated by

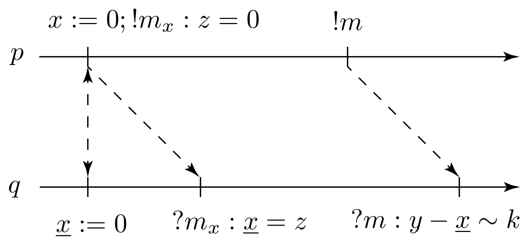
   $$!m : z_x = x \qquad \text{and} \qquad ?m : \exists z \cdot z_x - z \leq h \ \wedge \ y - z \geq k,$$

   where the receiver guesses the *final value* of channel clock z, checks that the transmission constraint $x - z \leq h$ was initially satisfied by checking the equivalent constraint $z_x - z \leq h$ at the time of reception, and checks that the reception constraint $y - z \geq k$ is also satisfied; cf. Fig. 6 Note that

   $$\exists z \cdot z_x - z \leq h \ \wedge \ y - z \geq k$$



(a) Original copy-send communication.



(b) Simulation by sending only the age of the message.

Figure 7: Ages simulate copy-send transmissions.

is not a constraint, since it contains a quantifier. However, one can show that this is logically equivalent to the constraint

$$z_x - y \leq h - k.$$

This is true in general, so the technique above generalises to arbitrary constraints.

Lemma 5 *The logic of clock constraints over the structure* $(\mathbb{Q}, \leq, +1, \equiv_k)$ *admits quantifier elimination.*

2. We further simplify transmission constraints to be of the even more restricted form

$$z = 0,$$

i.e., the only timing information attached to a message is its *age*. Consider the following pair of copy-transmission and matching reception

$$!m : z_x = x \qquad \text{and} \qquad ?m : y - z_x \geq k$$

What really counts is the value that clock x had at the time of reception, which in turn is determined by the last time it was reset before the transmission happened. The idea is to let the receiver have a local copy $\underline{x}$ of x, and use communication to keep it sufficiently synchronised with the original x: Both the sender and the receiver guess the last time x is reset before the actual transmission above. At this moment, the sender will additionally send a control message m_x with age 0, and at a later moment the receiver will receive this control message and check that it has age equal to its local copy of $\underline{x}$: In this way, now they agree on the value of x. The original transmission is then performed without any timing constraint just as $!m$, and the orignal reception is replaced by

$$?m; y - \underline{x} \sim k,$$

where now the former is just a constraint on local clocks y and $\underline{x}$; cf. Fig. 7. Thus, we can further assume that the only timing information necessary to send is the age of the message.

3. Finally, we would like to simplify reception constraints of the form

$$\underline{x} = z \tag{3}$$

to the non-diagonal form $z \sim k$. It is well known that diagonal constraints for TA can be removed in favour of their non-diagonal counterparts [4]. We thus first remove all diagonal constraints of the receiver, such as $y - \underline{x} \sim k$. Then, we notice that, since $\underline{x}$ now appears only in non-diagonal constraints of the form $\underline{x} \sim k$ we can guess the set of such constraints that $\underline{x}$ is required to later satisfy, and replace the reception constraint (3) with non-diagonal constraints of the form

$$z \sim k$$

in order to verify such guess.

Altogether, this allows us to have non-diagonal reception constraints on message ages, which is what we purported to show.

Notice that lower-bound counter tests $c \geq k$ do not increase the expressive power of Petri nets, since they can be simulated by performing

$$c := c - k; c := c + k,$$

exploiting the non-negative semantics of Petri net counters. On the other hand, upper-bound counter tests $c \leq k$ do increase the expressiveness of Petri nets; they can be simulated a decrement $c := c - h$, followed by a zero test $c = 0$, followed by an increment $c := c + h$, for some (guessed) $h \leq k$. Thus, we have a direct correspondence between reception constraints on timed channels and zero-tests.

The ideas above lead to a complete characterisation of decidable topologies for TCA with timed channels. We say that a channel is *untimed* if no send/receive action uses any channel clock. We say that a channel c has *upper bound constraints* if it has some receive action of the form $c?m : z \leq k$ for some channel clock z; lower bound constraints are defined similarly.

Theorem 6 *[9] The reachability problem for TCA with timed channels is decidable, if, and only if, the communication topology is a polyforest, and, additionally, in each polytree thereof there is at most one timed channel with upper bound constraints.*

It turns out that zero tests are unavoidable to simulate upper bound constraints.

Theorem 7 *[9] The reachability problem for simple TCA with timed channels for a polyforest topology with no channel upper bound constraints is equivalent to coverability in Petri nets. In the case of polyforest topologies with one channel with upper bound constraints, we have equivalence to coverability in Petri nets with one zero test.*

3 Timed communicating automata in dense time

All the main intuitions explained above in the case of discrete time extend to dense time. One additional issue in the analysis of dense time is that we need to keep track of the *fractional value* of clocks. To see why this is so, even if constants k used in diagonal constraints $x - y \leq k$ are integers, if x, y take values in $\mathbb{Q}$, then we need to know the ordering of their fractional values to decide the truth value of the constraint.

This is usually done with the notion of *region*, which can be seen as a *total preorder* on the fractional values of all clocks of a TA: Indeed, for every pair of clocks x, y with $\{x\} \leq \{y\}$, we need to know exactly which of the following three cases hold:

$$0 = \{x\} = \{y\} \qquad \text{or} \qquad 0 = \{x\} < \{y\} \qquad \text{or} \qquad 0 < \{x\} = \{y\} \qquad \text{or} \qquad 0 < \{x\} < \{y\}\,,$$

and this information can be maintained during time elapse by performing suitable *cyclical rotations*, and clock resets can be kept track of by resetting fractional values to 0. For instance, if $0 < \{x\} < \{y\}$ and we elapse time, then the next interesting thing that happen is that y becomes an integer and thus $0 = \{y\} < \{x\}$ holds; from the latter situation, if we elapse time again we get $0 < \{y\} < \{x\}$, and if we now reset x we obtain $0 = \{x\} < \{y\}$.

Regions are sufficient in global time. However, in our simulations we let each process have a different local time, which can be elapsed independently of all the other processes. To this end, it does not suffice to consider total preorders of fractional clock values. To see this, consider two processes p with local clock x_0, x_1 and q with local clocks y_0, y_1, and assume the current global region is

$$0 < \{x_0\} < \{y_0\} < \{x_1\} < \{y_1\} .$$

If we now let the local time of process p elapse, then we enter xeither the first xor the second of the following two successor regions, depending on whether $\{y_0\} - \{y_0\} \leq \{y_1\} - \{x_0\}$ holds or not:

$$0 < \{x_0\} = \{y_0\} < \{x_1\} < \{y_1\} \qquad \text{or} \qquad 0 < \{x_0\} < \{y_0\} < \{x_1\} = \{y_1\} .$$

This naturally leads to the idea of considering a total preorder not just on fractional values of clocks, but on their *differences* instead. Such an object is called a *clock difference relation* (CDR) [17], which is a boolean combination of atomic statements of the form

$$\{x - y\} \leq \{z - t\} ,$$

where x, y, z, t are clocks, their primed versions x', or the constant 0. For instance, letting time elapse for process p (but not for every other process) gives rise to the following CDR

$$\varphi_{\mathsf{elapse},p}(\overline{x}, \overline{x}') \;\equiv\; \bigwedge_{x,y \text{ of } p} \{x' - y'\} = \{x - y\} \;\wedge\; \bigwedge_{x \text{ not of } p} \{x'\} = \{x\} . \tag{4}$$

Resetting clock x is represented by the following CDR:

$$\varphi_{x:=0}(\overline{x}, \overline{x}') \;\equiv\; \{x'\} = 0 \;\wedge\; \bigwedge_{y \neq x} \{y'\} = \{y\} . \tag{5}$$

For a fixed number of clocks, there are only finitely many CDR (up to logical equivalence). Moreover, CDR are closed under a natural notion of relational composition: If φ, ψ are CDRs, then $\xi \equiv \varphi \circ \psi$, where

$$\xi(\overline{x}, \overline{x}') \;\equiv\; \exists \overline{x}'' \cdot \varphi(\overline{x}, \overline{x}'') \wedge \psi(\overline{x}'', \overline{x}'),$$

is also (expressible as) a CDR, another instance of the principle of quantifier elimination, this time for CDRs. Therefore, one can abstract away fractional values of clocks by CDRs, starting with the zero CDR

$$\varphi_{\mathsf{zero}}(\overline{x}, \overline{x}') \;\equiv\; \bigwedge_{\text{clock } x} \{x'\} = \{x\} = 0,$$

and composing with the basic time elapse (4) and reset (5) CDRs to build all CDRs which are reachable in the system. In terms of complexities, there are exponentially many CDR (up to logical equivalence) in the number of clocks, thus keeping track of CDRs introduces a multiplicative exponential factor. While this does not affect the class of decidable TCA topologies in Theorem 3 (which thus stays the same even in dense time), the simple TCA reduce in exponential time to coverability in Petri nets if there are no upper bound channel constraints, and to Petri nets with one zero test if there is at most one upper bound channel constraint for each polytree.

References

[1] P. A. Abdulla, C. Aiswarya, and M. F. Atig. Data communicating processes with unreliable channels. In *Proc. of LICS'16*, pages 166–175, New York, NY, USA, 2016. ACM.

[2] P. A. Abdulla, M. F. Atig, and J. Cederberg. Timed lossy channel systems. In D. D'Souza, T. Kavitha, and J. Radhakrishnan, editors, *Proc. of FSTTCS'12*, volume 18 of *LIPIcs*, pages 374–386, 2012.

[3] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.

[4] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126:183–235, April 1994.

[5] D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, Apr. 1983.

[6] G. Cécé and A. Finkel. Verification of programs with half-duplex communication. *Information and Computation*, 202(2):166–190, 2005.

[7] G. Cece, A. Finkel, and S. P. Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, 1996.

[8] P. Chambart and P. Schnoebelen. Mixing lossy and perfect fifo channels. In F. van Breugel and M. Chechik, editors, *Proc. of CONCUR'08*, volume 5201 of *LNCS*, pages 340–355. Springer, 2008.

[9] L. Clemente. Decidability of timed communicating automata. 04 2018.

[10] L. Clemente, F. Herbreteau, A. Stainer, and G. Sutre. Reachability of communicating timed processes. In F. Pfenning, editor, *Proc. of FOSSACS'13*, volume 7794 of *LNCS*, pages 810–96. Springer, 2013.

[11] L. Clemente, F. Herbreteau, and G. Sutre. Decidable topologies for communicating automata with fifo and bag channels. In P. Baldan and D. Gorla, editors, *Proc. of CONCUR'14*, volume 8704 of *LNCS*, pages 281–296. Springer, 2014.

[12] W. Czerwinski, S. Lasota, R. Lazic, J. Leroux, and F. Mazowiecki. The Reachability Problem for Petri Nets is Not Elementary (Extended Abstract). *ArXiv e-prints*, Sept. 2018.

[13] C. Haase, S. Schmitz, and P. Schnoebelen. The power of priority channel systems. *Logical Methods in Computer Science*, 10(4), 2014.

[14] A. Heußner, T. Le Gall, and G. Sutre. Safety verification of communicating one-counter machines. In D. D'Souza, T. Kavitha, and J. Radhakrishnan, editors, *Proc. of FSTTCS'12*, volume 18 of *LIPIcs*, pages 224–235, 2012.

[15] A. Heußner, J. Leroux, A. Muscholl, and G. Sutre. Reachability analysis of communicating pushdown systems. *LMCS*, 8(3):1–20, September 2012.

[16] S. R. Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *Proc of. STOC'82*, pages 267–281. ACM, 1982.

[17] P. Krčál and R. Pelánek. On sampled semantics of timed systems. In S. Sarukkai and S. Sen, editors, *In Proc. of FSTTCS'05*, volume 3821 of *LNCS*, pages 310–321. Springer, 2005.

[18] P. Krcal and W. Yi. Communicating timed automata: the more synchronous, the more difficult to verify. In *Proc. of CAV'06*, LNCS, pages 249–262. Springer, 2006.

[19] J. L. Lambert. A structure to decide reachability in Petri nets. *Theor. Comput. Sci.*, 99(1):79–104, June 1992.

[20] E. W. Mayr. An algorithm for the general Petri net reachability problem. In *Proc. of STOC'81*, pages 238–246. ACM, 1981.

[21] J. Pachl. Reachability problems for communicating finite state machines. Technical Report CS-82-12, University of Waterloo, May 1982.

[22] C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223 – 231, 1978.

[23] S. L. Torre, P. Madhusudan, and G. Parlato. Context-bounded analysis of concurrent queue systems. In *Proc. of TACAS'08*, volume 4963 of *LNCS*, pages 299–314, 2008.