Communicating Timed Automata: The More Synchronous, the More Difficult to Verify*

Pavel Krcal and Wang Yi

Uppsala University, Sweden
{pavelk, yi}@it.uu.se

Abstract. We study channel systems whose behaviour (sending and receiving messages via unbounded FIFO channels) must follow given timing constraints specifying the execution speeds of the local components. We propose Communicating Timed Automata (CTA) to model such systems. The goal is to study the borderline between decidable and undecidable classes of channel systems in the timed setting. Our technical results include: (1) CTA with one channel without shared states in the form $(A_1, A_2, c_{1,2})$ is equivalent to one-counter machine, implying that verification problems such as checking state reachability and channel boundedness are decidable, and (2) CTA with two channels without sharing states in the form $(A_1, A_2, A_3, c_{1,2}, c_{2,3})$ has the power of Turing machines. Note that in the untimed setting, these systems are no more expressive than finite state machines. This shows that the capability of synchronizing on time makes it substantially more difficult to verify channel systems.

1 Introduction

FIFO channels (i.e., unbounded buffers) are widely used as a communication mechanism in concurrent systems. In many applications, channels are a critical element for the correct functioning of such systems. In this work, we study timed systems whose components communicate through (unbounded) channels. An example of such systems is illustrated in Figure 1, where A_1 is a producer (or sender) which generates messages and puts them into the buffer $c_{1,2}$ and A_2 is a consumer (or receiver) which gets messages from the buffer. Assume that the production and consumption of messages must follow given timing constraints (specifying the relative execution speeds of the producer and the consumer). A relevant question to ask is whether the channel is bounded, and if it is, what is the maximal size of the buffer. This is a typical scenario in designing embedded systems, where it is desirable to know a priori the maximal size of a buffer needed to avoid buffer overflow and over-allocation of memory blocks in the final implementation.

* Partially supported by the European Research Training Network GAMES.

T. Ball and R.B. Jones (Eds.): CAV 2006, LNCS 4144, pp. 249–262, 2006.

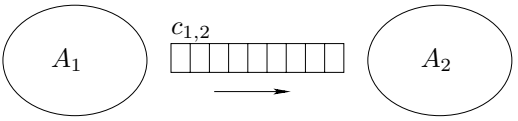

Fig. 1. A schema of a CTA with one channel

In the literature, channel systems have been studied intensively in the untimed setting, within the context of verification of infinite state systems (see below for related work which provides a brief summary of known results). To our best knowledge, this is the first attempt to study channel systems in the timed setting. The existing works address mainly channel systems that are a finite set of Communicating Finite State Machines (CFSMs). In the CFSM model, no notion of time is assumed and systems run in a fully asynchronous manner in the sense that any local move of a machine is allowed at any time. We observe that for systems modeled as CFSMs, the source of infiniteness is in not only *unbounded channels* but also the capability of *synchronization* or exchanging information between the machines. In fact, asynchronous systems – as illustrated in Figure 1 and 2 with only one-directional communication, where the receivers are not allowed to inform directly or indirectly the senders about the receipt of messages – are no more expressive than finite state machines [Pac03, CF05], and thus all properties such as reachability and channel boundedness are decidable. Roughly speaking, synchronization within CFSMs may be achieved through either shared states [BZ83], or two-direction communication [FM97] or combination of accepting conditions and doubled one-direction channels [Pac03, Pac82]. The synchronization features together with the unboundedness of channels are the essential source of undecidability for channel systems in the untimed setting.

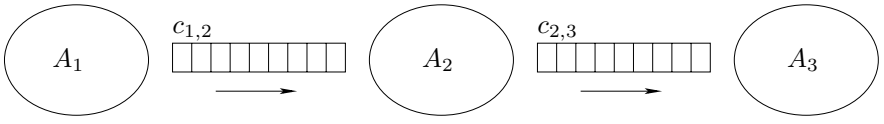

Fig. 2. A schema of a CTA with two channels

As a model for timed systems communicating via channels, we propose and study Communicating Timed Automata (CTA), i.e., networks of timed automata extended with (unbounded) channels. A CTA is a channel system where the sending and receiving transitions of machines are constrained with clock constraints. We shall show that channel systems (with one channel) as illustrated in Figure 1, which accept only regular languages in the untimed setting, are expressive enough to simulate one-counter machines in the timed setting. However, the density of time adds no more expressive power (than discrete time), and many questions of interests such as reachability and channel boundedness are still decidable for CTA with one channel. As a main technical contribution,

we present a novel proof showing that CTAs with one channel without sharing states are no more expressive than one-counter machines. The proof uses the notion of CDR (Clock Difference Relations) developed in [KP05]. To study the borderline of decidability and undecidability for CTA, we have shown that CTAs with two channels, as illustrated in Figure 2, can simulate Turing machines. By this we show the theoretical limits of analysis of timed systems with unbounded channels.

Related Work. Channel systems, i.e., networks of communicating finite state machines (*CFSMs*) have been widely studied in the untimed setting, as a model for communication protocols, in which no global notion of time is assumed and any local move of FSMs at any time is allowed. The first undecidability results for the untimed setting were presented in [BZ83] showing that two FSMs with shared states and one channel can simulate Turing machines. Further results consider even more restricted settings, showing that two identical simple FSMs with one channel in both directions are powerful enough to simulate a Turing machine [FM97]. A suprising result due to [Pac03, Pac82] is that two FSMs connected by two channels going in the same direction can simulate Initial Post's Correspondence Problem, and therefore have the power of Turing machines. Classes of CFSMs with decidable reachability problems have been identified in [CF05] (half-duplex systems), [Pac03] (cyclic systems with one channel bounded), and [PP92] (cyclic systems with one-type messages). Abstractions of CFSMs for acceleration in reachability analysis are presented in [FPS03]. Another recent work [GMK04] shows the equivalence of several formalisms when the communication is existentially bounded. Apart from work on systems with perfect channels, systems with unreliable channels have been studied in [AJ96a, AJ96b]. An excellent survey on work in this direction can be found in [CFP96].

2 Communicating Timed Automata

We assume that the reader is familiar with timed automata [AD94]. A network of Communicating Timed Automata (*CTA*) is a tuple $(A_1, A_2, \ldots, A_n, c_{i_1,j_1}, c_{i_2,j_2}, \ldots, c_{i_m,j_m})$ where each $A_i = (Q_i, \mathcal{A}ct, \mathcal{C}_i, E_i, q_i^0, F_i)$ is a timed automaton and each $c_{i,j}$, $i, j \in \{1 \ldots n\}$ is an unidirectional unbounded channel containing messages sent from A_i to A_j. Mutually disjoint finite sets $Q_1, \ldots Q_n$ contain locations of A_i's. A finite set $\mathcal{A}ct$ denotes a communication alphabet common for all A_i's. In addition, we assume that automata may perform an internal transition denoted by ϵ. $\mathcal{C}_i$ is a finite set of real-valued clocks ($\mathcal{C}_i$, $\mathcal{C}_j$ are disjoint for $i \neq j$), $q_i^0 \in Q_i$ is an initial location, and $F_i \subseteq Q_i$ is a set of accepting locations. $E_i \subseteq Q_i \times (\{1 \ldots n\} \times \{?, !\} \times \mathcal{A}ct) \cup \{\epsilon\} \times \mathcal{G}(\mathcal{C}) \times 2^{\mathcal{C}} \times Q_i$ is the set of transitions of A_i, where $\mathcal{G}(\mathcal{C})$ and $2^{\mathcal{C}}$ are timed automata guards and resets, respectively. Transitions are labeled by not only a letter from $\mathcal{A}ct$, but also information about whether a letter is sent or received (! or ?, respectively) and to or from which channel. We write $q_i \xrightarrow{k!a,g,r} q_i'$ when $(q_i, k!a, g, r, q_i') \in E$. Channels are assumed to be perfect. We denote the contents of a channel by finite words over $\mathcal{A}ct$.

Let $\nu_i : \mathcal{C}_i \mapsto \mathcal{R}_{\geq 0}$ denote a valuation of clocks in A_i. Let $\nu_i \models g$ denote that the guard g is satisfied by ν_i and $r(\nu_i), r \subseteq \mathcal{C}_i$ denote a valuation where all clocks from r are reset and other clocks keep their values. A state of the system is a tuple $(q_1, \nu_1, \ldots, q_n, \nu_n, w_1, \ldots, w_m)$, where $q_i \in Q_i$ is a location of A_i and $w_k \in \mathcal{A}ct^*$ is the content of channel c_{i_k,j_k}. We define the semantics of CTA based on Labeled Transition System (LTS).

Definition 1 (Synchronized Semantics). *The semantics of a CTA* $(A_1, \ldots, A_n, c_{i_1,j_1}, \ldots, c_{i_m,j_m})$ *is a labeled transition system with initial state* $(q_1^0, \nu_1^0, q_2^0, \nu_2^0, \ldots, q_n^0, \nu_n^0, \epsilon, ..., \epsilon)$, *where* $\nu_i^0(x) = 0$ *for all* $x \in \mathcal{C}_i$ *and two types of transitions – time pass and discrete transition – defined as follows. Let* $s = (q_1, \nu_1, \ldots, q_n, \nu_n, w_1, \ldots, w_m)$ *and* $s' = (q_1', \nu_1', \ldots, q_n', \nu_n', w_1', \ldots, w_m')$.

- $s \xrightarrow{t} s'$ *if* $\nu_i' = \nu_i + t$, $q_i' = q_i$ *and* $w_j' = w_j$ *for all* $1 \leq i \leq n$ *and* $1 \leq j \leq m$.
- $s \xrightarrow{(a,i,k,!)} s'$ *if* $q_i \xrightarrow{k!a,g,r} q_i'$, $w_l' = a \cdot w_l$, *where* w_l *is the content of* $c_{i,k}$, $\nu_i \models g$, $\nu_i' = r(\nu_i)$, *and* $q_j' = q_j, \nu_j' = \nu_j, w_k' = w_k$ *for all* $j \neq i, k \neq l$,
- $s \xrightarrow{(a,i,k,?)} s'$ *if* $q_i \xrightarrow{k?a,g,r} q_i'$, $a \neq \epsilon$, $w_l' \cdot a = w_l$, *where* w_l *is the content of* $c_{k,i}$, $\nu_i \models g$, $\nu_i' = r(\nu_i)$, *and* $q_j' = q_j, \nu_j' = \nu_j, w_k' = w_k$ *for all* $j \neq i, k \neq l$, *and*
- $s \xrightarrow{(\epsilon,i)} s'$ *if* $q_i \xrightarrow{\epsilon,g,r} q_i'$, $\nu_i \models g$, $\nu_i' = r(\nu_i)$, $q_j' = q_j, \nu_j' = \nu_j, w_k' = w_k$ *for all* $j \neq i, k \in \{1, \ldots, m\}$, *and there is no* $q_i \xrightarrow{k?a,\bar{g},\bar{r}} q_i''$ *such that* $\nu_i \models \bar{g}$ *and* $w_l = w_l'' \cdot a$, *where* w_l *is the content of* $c_{k,i}$.

All automata move synchronously; time passes at the same pace for all of them. The automata read from the channels in an *urgent* manner, an automaton is not allowed to take an ϵ-transition if it can take a receiving a-transition and a is at the head of the corresponding channel. Another possibility is to define reading as non-urgent, i.e., there are no restrictions on taking ϵ transitions. In Section 3, we show by an example that CTA's even with non-urgent reading from the channels have strictly more expressive power than CFSMs in the untimed setting.

Let S be a CTA and T_S be its corresponding LTS. By ρ we denote a finite path in T_S, by $[\rho]$ a sequence of labels occurring along ρ, and by $[\rho]_i^!$ ($[\rho]_i^?$) a sequence of letters from $\mathcal{A}ct$ which is a projection of $[\rho]$ to letters sent (received) by an automaton A_i. If the location vector $(q_1, \ldots, q_n)$ of the last global state of ρ is accepting (i.e., $\forall i. q_i \in F_i$) then we say that the run is accepting, denoted $\rho \rhd T_S$. A language accepted by a CTA S is a set $L_S(S) = \{[\rho]_1^! \mid \rho \rhd T_S\}$.

Note that we can model CFSMs by CTA. Therefore, all negative results proved for CFSMs apply also to our model. In the following, we study the expressive power of the model by identifying decidable and undecidable classes of CTA.

3 CTA with One Channel

Let us first consider a system $(A_1, A_2, c_{1,2})$ schematically depicted in Figure 1. It has been shown that CFSMs with such topology accept regular languages and reachability and boundedness problems are decidable [Pac03, CF05]. We show

that CTA of this form can accept also some non-regular context-free languages. Moreover, we show that for such a CTA there is a one-counter machine which accepts the same language. Therefore, state reachability and channel boundedness problems are decidable, which follows from the decidability of emptiness and infiniteness for context-free languages.

To establish the proof, we propose an alternative (*desynchronized concrete*) semantics for CTA which resembles the reordering technique [Pac03] for CFSMs and the local time semantics for timed systems [BJLY98]. However, states in this semantics still contain concrete valuations of clocks. Therefore, we define a (*desynchronized symbolic*) semantics where the continuous part of the state has a finite symbolic representation. This symbolic semantics can be easily simulated by a one-counter machine. We also show that instructions of a one-counter machine can be simulated by a CTA of the form $(A_1, A_2, c_{1,2})$ and thus the expressive power of CTA with this topology is equivalent to one-counter machine.

Intuitively, we let the automata to desynchronize so that there is at most one message in the channel during the first part of the computation and that only the producing automaton runs during the second part of the computation. Local time (time from the beginning of the computation) can be different in A_1 and A_2. We keep track of the difference between local times of automata in a real valued variable. The acceptance condition is extended by a requirement that the system should be synchronized, i.e., the value of this variable is equal to 0.

In the following, we denote A_1 as A and A_2 as B. We also write $!a$ instead of $2!a$ and $?a$ instead of $1?a$. Without loss of generality, we assume that there is a clock t_i in each A_i which is never reset. The reason is to simplify the notation later. A state in the concrete desynchronized semantics is a tuple $(q_A, \nu_A, q_B, \nu_B, w, T)$, where $q_A \in Q_A, q_B \in Q_B, w \in \mathcal{A}ct^*$, valuations ν_A, ν_B are as in the original semantics, and $T \in \mathcal{R}$ is the lag of B behind A (it is negative if B is ahead). By $(q_A, \nu_A) \xrightarrow{lab} (q'_A, \nu'_A)$ we mean that there is a transition from (q_A, ν_A) to (q'_A, ν'_A) labeled by lab in the standard timed automata semantics LTS. We need to take special care about reading – a letter should not be read before it has been produced.

We let the automata to alternate in running as long as the size of the channel content does not exceed 1. When it contains at least two letters then only A can move. We assume $a \in \mathcal{A}ct$ and $w \in \mathcal{A}ct^*$ in the following definition.

Definition 2 (Desynchronized Concrete Semantics). *The* desynchronized concrete semantics *of a CTA $(A, B, q_{A,B})$ is a labeled transition system with initial state $(q_A^0, \nu_A^0, q_B^0, \nu_B^0, \epsilon, 0)$ and transitions induced by the following rules:*

- $(q_A, \nu_A, q_B, \nu_B, w, T) \xrightarrow{t}_{dc} (q_A, \nu_A + t, q_B, \nu_B, w, T + t)$ *if* $(q_A, \nu_A) \xrightarrow{t} (q_A, \nu_A + t)$,
- $(q_A, \nu_A, q_B, \nu_B, w, T) \xrightarrow{(a,1,2,!)}_{dc} (q'_A, \nu'_A, q_B, \nu_B, a \cdot w, T)$ *if* $(q_A, \nu_A) \xrightarrow{!a} (q'_A, \nu'_A)$,
- $(q_A, \nu_A, q_B, \nu_B, w, T) \xrightarrow{t}_{dc} (q_A, \nu_A, q_B, \nu_B + t, w, T - t)$ *if* $(q_B, \nu_B) \xrightarrow{t} (q_B, \nu_B + t)$ *and* $|w| \leq 1$,
- $(q_A, \nu_A, q_B, \nu_B, a, T) \xrightarrow{(a,2,1,?)}_{dc} (q_A, \nu_A, q'_B, \nu'_B, \epsilon, T)$ *if* $(q_B, \nu_B) \xrightarrow{?a} (q'_B, \nu'_B)$ *and* $T \leq 0$,

– $(q_A, \nu_A, q_B, \nu_B, w, T) \xrightarrow{\epsilon}_{dc} (q_A, \nu_A, q'_B, \nu'_B, w, T)$ *if* $(q_B, \nu_B) \xrightarrow{\epsilon} (q'_B, \nu'_B)$, $|w| \leq 1$, *if* $T \geq 0$ *then* $(w = a \Rightarrow (q_B, \nu_B) \overset{?a}{\nrightarrow})$, *and if* $T < 0$ *then* $(w = a \wedge (q_B, \nu_B) \overset{?a}{\nrightarrow})$.

A run with the last state $(q_A, \nu_A, q_B, \nu_B, w, T)$ is accepting if $q_A \in F_A, q_B \in F_B$, and $T = 0$. Definition of the accepted language $L_{DC}(S)$ for a given CTA S is the same as for synchronized semantics. The set of reachable states of a given CTA is equal to the set of states reachable in its desynchronized concrete semantics where $T = 0$. Also, the language accepted by a CTA is the same in both semantics.

Lemma 1. *For a given CTA* S *of the form* $(A, B, c_{A,B})$, *the reachability set* $\{(q_A, \nu_A, q_B, \nu_B, w) \mid (q_A^0, \nu_A^0, q_B^0, \nu_B^0, \epsilon) \rightarrow^* (q_A, \nu_A, q_B, \nu_B, w)\}$ *is equal to the set* $\{(q_A, \nu_A, q_B, \nu_B, w) \mid (q_A^0, \nu_A^0, q_B^0, \nu_B^0, \epsilon, 0) \longrightarrow_{dc}^* (q_A, \nu_A, q_B, \nu_B, w, 0)\}$. *Moreover,* $L_S(S) = L_{DC}(S)$.

The basic idea of the proof of this lemma is the same as in [Pac03]. Desynchronized concrete semantics cannot reach more states where $T = 0$ or accept more words because the counter gives us a possibility to check the following conditions on the transitions of B. A letter can be read only after it has been produced and ϵ-transitions can be taken only when no enabled transition is labeled by the head of the buffer.

The desynchronization semantics shows how to avoid necessity to remember the whole content of the buffer during the run of a CTA. Note that one does not have to remember the content of the channel when its size exceeds 1, because it will never be read. The price we have to pay is an additional real number as a part of the state. In case of discrete time, T is an integer and therefore one can replace such a system by a language equivalent (in fact, bisimilar) one-counter machine. To be able to prove that there is a one-counter machine which is language equivalent to such a system in the dense time, we need to handle real valued clocks and T in a symbolic way, such that we get a finite state control unit and one counter.

The first step is to use regions [AD94] instead of valuations for each automaton. We denote regions by D, D_A, D_B. When D is a region over clocks of two automata A and B then by $(\nu_A, \nu_B) \in D$ we mean that $\nu \in D$ where $\nu(x) = \nu_A(x)$ for all $x \in \mathcal{C}_A$ and $\nu(y) = \nu_B(y)$ for all $y \in \mathcal{C}_B$. We write $D \Rightarrow D_A$ if D is a region over clocks of A, B, D_A is a region over clocks of A, and for all $(\nu, \nu') \in D$ it holds that $\nu \in D_A$.

Now we need to take care of T. There are two sources of infinity in T – its integral part, which can grow arbitrarily large, and its fractional part. We remember the integral part of T in a counter, denoted N. To remember the fractional part of T, we use the extra local clocks t_A and t_B of A and B. We observe that the difference of their fractional parts is equal to the fractional part of T (we do not use their integral parts). More precisely, if $(q_A, \nu_A, q_B, \nu_B, w, T)$ is reachable and $N = \lceil T \rceil$ then $T = N + (\mathsf{fr}(\nu_A(t_A)) - \mathsf{fr}(\nu_B(t_B)))$ if $\nu_A(t_A) \geq \nu_B(t_B)$ and $T = N + (1 - (\mathsf{fr}(\nu_B(t_B)) - \mathsf{fr}(\nu_A(t_A))))$ if $\nu_A(t_A) < \nu_B(t_B)$.

The fractional parts of t_A and t_B are then symbolically represented by regions and we remember their relative order as a constraint of the form $t_A \bowtie t_B$, where $\bowtie \in \{<, =, >\}$. Assume that local regions D_A, D_B were reached during the standard reachability analysis. For two given local regions D_A, D_B, our goal is to find a global region D which contains only valuations reachable in the desynchronized concrete semantics. We can define D as an ordering of the fractional parts of clocks which is consistent with D_A, D_B ($D \Rightarrow D_A, D \Rightarrow D_B$), and with $t_A \bowtie t_B$.

However, such constraints on global regions are not sufficient. There are CTA for which symbolic analysis reaches $D_A, D_B, t_A \bowtie t_B$, but there is a global region D consistent with $D_A, D_B, t_A \bowtie t_B$ which contains unreachable valuations.

To eliminate such global regions, we will remember also relations between clock differences. We use the fact that t_A and t_B are never reset and relate all other clocks to them. The concept of *clock difference relations* has been used before in [KP05] to characterize reachability relations. Here we give a slightly modified definition which suits our purposes better. To differentiate this definition from the original one, we call it *desynchronized* clock difference relations here, but later we will use only an abbreviation CDR or clock difference relation.

Definition 3. *A* desynchronized clock difference relation *(CDR) is a set of (in)equalities of the form $exp \bowtie exp$ or $exp \bowtie 1 - exp$ where exp is a clock difference (over the clocks of either A or B) in the form: $t_A - x$, $x - t_A$, $t_B - y$ or $y - t_B$, x is a clock of A, y is a clock of B, and $\bowtie \in \{<, >, =\}$.*

Definition 4. *The semantics of a CDR is defined as follows. Assume C is a CDR. We say that a pair of valuations (ν, ν') satisfies C ($(\nu, \nu') \vDash C$) if:*

- *if $x - y \bowtie u - v \in C$ then $\mathsf{fr}(\nu(x)) - \mathsf{fr}(\nu(y)) \bowtie \mathsf{fr}(\nu'(u)) - \mathsf{fr}(\nu'(v))$,*
- *if $x - y \bowtie 1 - (u - v) \in C$ then $\mathsf{fr}(\nu(x)) - \mathsf{fr}(\nu(y)) \bowtie 1 - (\mathsf{fr}(\nu'(u)) - \mathsf{fr}(\nu'(v)))$,*

Additionally, we require that for each $x - y$ (or $u - v$), $\mathsf{fr}(\nu(x)) - \mathsf{fr}(\nu(y)) > 0$.

We will use clock difference relations to restrict possible merges of regions over clocks of A and B. The merged regions represent only reachable concrete desynchronized valuations now.

States of the desynchronized symbolic system $(q_A, D_A, q_B, D_B, C, t_A \bowtie t_B, w, N)$ consist of locations and regions of A and B, respectively, clock difference relations, relation of t_A and t_B, $w \in \mathcal{A}ct^*$ is a content of the buffer, and N is an integer used to remember the difference between the integral parts of t_A and t_B.

We need some more technical definitions before the definition of the semantics. By $D \models C$ where D is a global region we mean that there exists $(\nu_A, \nu_B) \in D$ such that $(\nu_A, \nu_B) \models C$. We write e for a clock difference relation (a single (in)equality). We define a predicate $\mathsf{Consistent}(D_A, D_B, C, t_A \bowtie t_B) = \exists D. D(t_A) \bowtie D(t_B), D \models C, D \Rightarrow D_A, D \Rightarrow D_B$.

Definition 5 (Desynchronized Symbolic Semantics). *The* desynchronized symbolic semantics *of a CTA $(A, B, q_{A,B})$ is a labeled transition system with initial state $(q_A^0, D_A^0, q_B^0, D_B^0, \emptyset, t_A = t_B, \epsilon, 0)$, transition rules are given in Table 1, Table 2, and Table 3.*

Table 1. Rules for symbolic transitions induced by the region graph of A. For clarity, we omit locations in the rules for time pass.

Time Pass:		
$D_A \to D'_A$, $\exists x \in \mathit{integral}(D_A)$		
$(D_A, D_B, C, t_A < t_B, w, N)$	$\longrightarrow_{ds}$	$(D'_A, D_B, C, t_A < t_B, w, N)$
$(D_A, D_B, C, t_A = t_B, w, N)$	$\longrightarrow_{ds}$	$(D'_A, D_B, C, t_A > t_B, w, N)$
$(D_A, D_B, C, t_A > t_B, w, N)$	$\longrightarrow_{ds}$	$(D'_A, D_B, C, t_A > t_B, w, N)$
D_A, $\nexists x \in \mathit{integral}(D_A)$		
$(D_A, D_B, C, t_A < t_B, w, N)$	$\longrightarrow_{ds}$	$(D_A, D_B, C, t_A = t_B, w, N+1)$ if $\mathsf{Consistent}(D_A, D_B, C, t_A = t_B)$
$(D_A, D_B, C, t_A = t_B, w, N)$	$\longrightarrow_{ds}$	$(D_A, D_B, C, t_A > t_B, w, N)$
$D_A \to D'_A$, $\exists x \in \mathit{integral}(D'_A)$		
$(D_A, D_B, C, t_A < t_B, w, N)$	$\longrightarrow_{ds}$	$(D'_A, D_B, C', t_A = t_B, w, N+1)$ if $\mathsf{Consistent}(D'_A, D_B, C', t_A = t_B)$
$(D_A, D_B, C, t_A < t_B, w, N)$	$\longrightarrow_{ds}$	$(D'_A, D_B, C', t_A < t_B, w, N)$ if $\mathsf{Consistent}(D'_A, D_B, C', t_A < t_B)$
$(D_A, D_B, C, t_A > t_B, w, N)$	$\longrightarrow_{ds}$	$(D'_A, D_B, C', t_A < t_B, w, N)$ if $t_A \in \mathit{integral}(D'_A), t_B \notin \mathit{integral}(D_B)$
$(D_A, D_B, C, t_A > t_B, w, N)$	$\longrightarrow_{ds}$	$(D'_A, D_B, C', t_A = t_B, w, N+1)$ if $t_A \in \mathit{integral}(D'_A), t_B \in \mathit{integral}(D_B)$
Discrete Transition:		
$(q_A, D_A) \to (q'_A, D'_A)$, x is reset		
$(q_A, D_A, q_B, D_B, C, t_A \bowtie t_B, w, N)$	$\overset{(a,1,2,!)}{\longrightarrow_{ds}}$	$(q'_A, D'_A, q_B, D_B, C', t_A \bowtie t_B, a \cdot w, N)$ if $a \in \mathcal{A}ct \cup \{\epsilon\}$ is the label on the corresponding edge of A
$(q_A, D_A) \to (q'_A, D_A)$, no clock is reset		
$(q_A, D_A, q_B, D_B, C, t_A \bowtie t_B, w, N)$	$\overset{(a,1,2,!)}{\longrightarrow_{ds}}$	$(q'_A, D_A, q_B, D_B, C, t_A \bowtie t_B, a \cdot w, N)$ if $a \in \mathcal{A}ct \cup \{\epsilon\}$ is the label on the corresponding edge of A

A run with the last state $(q_A, D_A, q_B, D_B, C, t_A \bowtie t_B, w, N)$ is accepting if $q_A \in F_A, q_B \in F_B$, $N = 0$, and $t_A = t_B$. Definition of the accepted language $L_{DS}(S)$ for a given CTA S is the same as for synchronized semantics. Now we state that the desynchronized symbolic semantics is language equivalent to the desynchronized concrete one.

Lemma 2. *For a given CTA S, $L_{DS}(S) = L_{DC}(S)$.*

Proof. Proof is given in the full version of this paper [KY06].

Obviously, this system can be replaced by a one-counter machine accepting the same language (actually, a bisimilar one-counter machine).

Theorem 1. *State reachability and channel boundedness problems are decidable for CTA of the form $(A_1, A_2, c_{1,2})$.*

Table 2. Rules for symbolic transitions induced by the region graph of B. All transitions are constrained by $|w| \leq 1$. Transitions for time pass are the same as for A except for that N is never incremented, but it is decremented when $t_A = t_B$ changes to $t_A < t_B$ and inequality signs in $t_A \bowtie t_B$ are inverted. Complete table is given in the full version of this paper [KY06].

Discrete Transition:	
$(q_B, D_B) \to (q'_B, D'_B)$, x is reset	
$(q_A, D_A, q_B, D_B, C, t_A \bowtie t_B, a, N)$	$\xrightarrow{(a,2,1,?)}_{ds}$ $(q_A, D_A, q'_B, D'_B, C', t_A \bowtie t_B, \epsilon, N)$ if $?a$, $a \in \mathcal{A}ct$ is the label on the corresponding edge of B, and $N < 0 \vee (N = 0 \wedge t_A = t_B)$
$(q_A, D_A, q_B, D_B, C, t_A \bowtie t_B, w, N)$	$\xrightarrow{\epsilon}_{ds}$ $(q_A, D_A, q'_B, D'_B, C', t_A \bowtie t_B, w, N)$ if ϵ is the label on the corresponding edge of B, if $N \geq 0$ then $(w = a \Rightarrow (q_B, \nu_B) \overset{?a}{\nrightarrow})$ and if $N < 0$ then $(w = a \wedge (q_B, \nu_B) \overset{?a}{\nrightarrow})$
Similarly when no clock is reset.	

Proof. Follows from Lemma 1, Lemma 2, and basic language theory.

Now we show that the instructions of a one-counter machine can be encoded in a CTA with one channel. The counter is encoded as the number of a's in the channel. Figure 3 shows how to encode incrementation of the counter q_i: C:=C+1; goto q_j and conditional decrementation of the counter q_i: if C=0 then goto q_j else C:=C-1; goto q_k. Each transition takes exactly one time unit. We omit clocks and guards on all other edges (they are labeled by $x = 1, x := 0$). Test for zero is performed by a nondeterministic choice for A. To check that the choice was correct, A produces b. If it was wrong then b is not consumed by the corresponding transition of B, stays in the channel and eventually blocks the computation of B. At the end of the computation, B has to check whether there is any b in the channel. If it is the case then it moves to an error location.

To illustrate the expressive power of CTA, Figure 4 shows a (schematic description of a) CTA which accepts a non-regular context-free language $a^n b a^n b$. Again, each transition takes exactly one time unit and we omit $x = 1, x := 0$ from all edges. The number of a's is remembered in the size of the channel content and we use different speed of production/consumption to maintain the correct number of a's in the channel. At the beginning, A produces twice faster than B reads. There are $n/2$ a's in the channel when B reads the first b and from this moment B reads twice faster then A produces.

From the point of view of the desynchronized semantics, the number of a's in the channel corresponds to the level of desynchronization. After reading the first n letters a the lag of B is $2n$ time units. Then it reads a dividing letter b and reads a's again. If there are n letters a then A and B get synchronized again and the accepting configuration is reachable after two more steps. If there are more a's then B gets stuck reading them, because it reads faster than A produces. If there are less a's then B can read b immediately and it has to go down to the error state. All locations of A are accepting, but the only accepting location of B is the next to the last one.

Table 3. Updates of the clock difference relations according to the type of the transition of the desynchronized symbolic system. We write e for a clock difference relation (a single (in)equality). We write exp for an expression of the form $x - y$ or $1 - (x - y)$ where x, y are clock from the automaton given by the context.

C'	Condition, A moves
$D_A \to D'_A$, $\exists x \in integral(D'_A)$	
e	$e \in C$, e does not contain any $x \in integral(D'_A)$
$y - x \bowtie^{-1} 1 - (exp)$	$x - y \bowtie exp \in C$, $x \in integral(D'_A)$
$D_A \to D'_A$, x is reset	
e	$e \in C$, e does not contain x
$t_A - x > exp$	$t_A - y \geq exp \in C$
$t_A - x < 1 - exp$	$z - t_A \geq exp \in C$
$t_A - x < t_B - y$	$t_A < t_B$, $y \in integral(D_B)$
$t_A - x < 1 - (y - t_B)$	$t_A < t_B$, $D_B(y) > D_B(t_B)$
$t_A - x = t_B - y$	$t_A = t_B$, $y \in integral(D_B)$
$t_A - x > t_B - y$	$t_A = t_B$, $y \notin integral(D_B)$, $D_B(y) < D_B(t_B)$
$t_A - x < 1 - (y - t_B)$	$t_A = t_B$, $D_B(y) > D_B(t_B)$
$t_A - x > t_B - y$	$t_A > t_B$, $D_B(y) < D_B(t_B)$

C'	Condition, B moves
$D_B \to D'_B$, $\exists x \in integral(D'_B)$	
e	$e \in C$, e does not contain any $x \in integral(D'_B)$
$exp \bowtie y - x$	$exp \bowtie 1 - (x - y) \in C, x \in integral(D'_B)$
$exp \bowtie 1 - (y - x)$	$exp \bowtie x - y \in C$, $x \in integral(D'_B)$
$D_B \to D'_B$, x is reset	
e	$e \in C$, e does not contain x
$exp < t_B - x$	$exp \leq t_B - y \in C$
$exp < 1 - (t_B - x)$	$exp \leq z - t_B \in C$
$t_A - y > t_B - x$	$t_A > t_B$, $y \in integral(D_A)$
$y - t_A < 1 - (t_B - x)$	$t_A > t_B$, $D_A(y) > D_A(t_A)$
$t_A - y = t_B - x$	$t_A = t_B$, $y \in integral(D_A)$
$t_A - y < t_B - x$	$t_A = t_B$, $y \notin integral(D_A)$, $D_A(y) < D_A(t_A)$
$y - t_A < 1 - (t_B - x)$	$t_A = t_B$, $D_A(y) > D_A(t_A)$
$t_A - y < t_B - x$	$t_A < t_B$, $D_A(y) < D_A(t_A)$

This automaton accepts the same language also in discrete time. It also shows the expressive power of CTA with one channel without urgency in the semantics, i.e., ϵ-transitions of B are not restricted. The language accepted by the CTA in Figure 4 remains the same even for non-urgent semantics when the only accepting location of A is the location m.

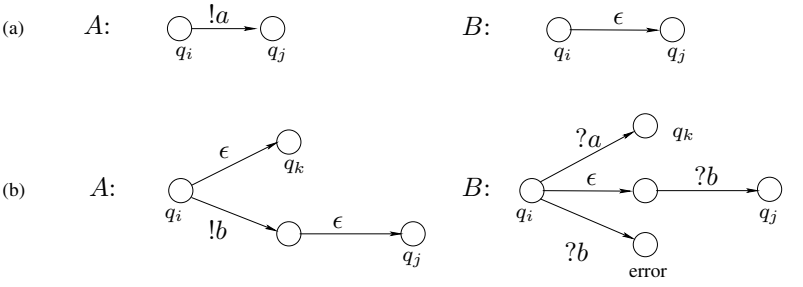

Fig. 3. A CTA encoding instructions of a one-counter machine. (a) encodes incrementation of the counter and (b) encodes conditional decrementation of the counter.

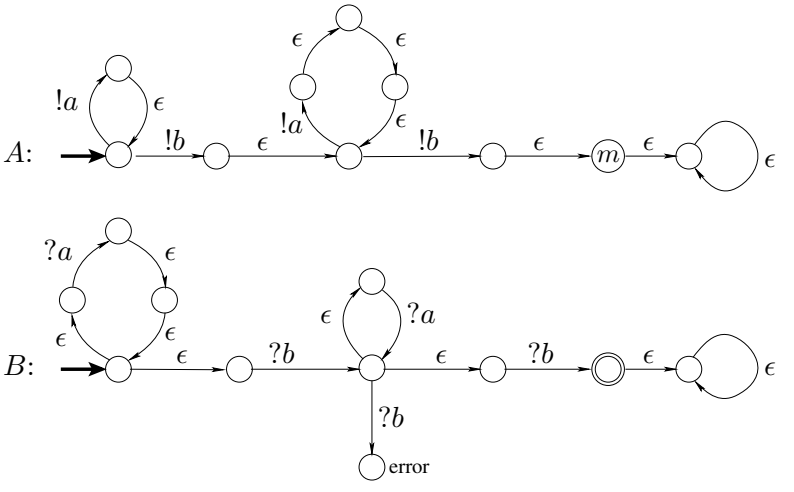

Fig. 4. A CTA accepting the language $a^n b a^n b$

4 CTA with Two Channels

Now we consider systems of the form $(A_1, A_2, A_3, c_{1,2}, c_{2,3})$ shown in Figure 2. We show that such CTA have the Turing power. This contrasts with the CFSMs, where systems of this form can accept only regular languages. The notion of the global time changes substantially the expressive power.

We cannot encode counters in the number of a's as we did it for one-counter machine, because there is no way how to verify nondeterministic choice of A_1 when deciding whether $c_{2,3}$ is empty. We will build on the construction from Figure 4. Again, we use different speed of production/consumption to maintain number of a's in the channels.

To show the simulation of a two-counter machine by a CTA with two channels we first notice that there is a system which accepts a language $a^n(a^n b a^n b)^*$. Therefore, there is a system which can keep the number of a's at the same level

during the whole computation. It works on the same principle as the system from Figure 4. Using the first channel ($c_{1,2}$) and the desynchronization of the automata we check that $2i$-th and $2i+1$-th sequence of a's have the same length and, at the same time, send the $2i+1$-th sequence to the second channel ($c_{2,3}$). Then the same construction is used to check that $2i+1$-th sequence has the same length as the $2i+2$-th sequence. A schematic description of this CTA is given in the full version of this paper [KY06].

The CTA simulating a two-counter machine accepts a language corresponding to the sequence of the encoded values of the counters during the computation of this machine. The values m, n of the two counters C_1, C_2 are encoded by the length of the sequence of a's – the corresponding sequence is $a^{2^n 3^m}$. Therefore, incrementation of the counter C_1 corresponds to doubling of the length of the sequence, decrementation of C_1 to halving, incrementation of C_2 to multiplying by 3, and decrementation of C_2 to dividing by 3. To test a counter for zero, we need to check whether the length of the sequence is divisible by 2 or 3. We use the same trick as for the language $a^n b(a^n b a^n b)^*$. Just the consecutive sequences can be of the form $a^n b a^{2n}$, $a^n b a^{3n}$, $a^{2n} b a^n$, or $a^{3n} b a^n$. This can be easily done, since each of these pairs are context-free languages, and the correct overlapping is secured by using both channels in an alternating manner.

Figure 5 depicts a fragment for doubling of the length of the sequence of a's, which corresponds to the incrementation of C_1. The relative speed of production and consumption is set so that A_2 does not end in the error sink only if the second

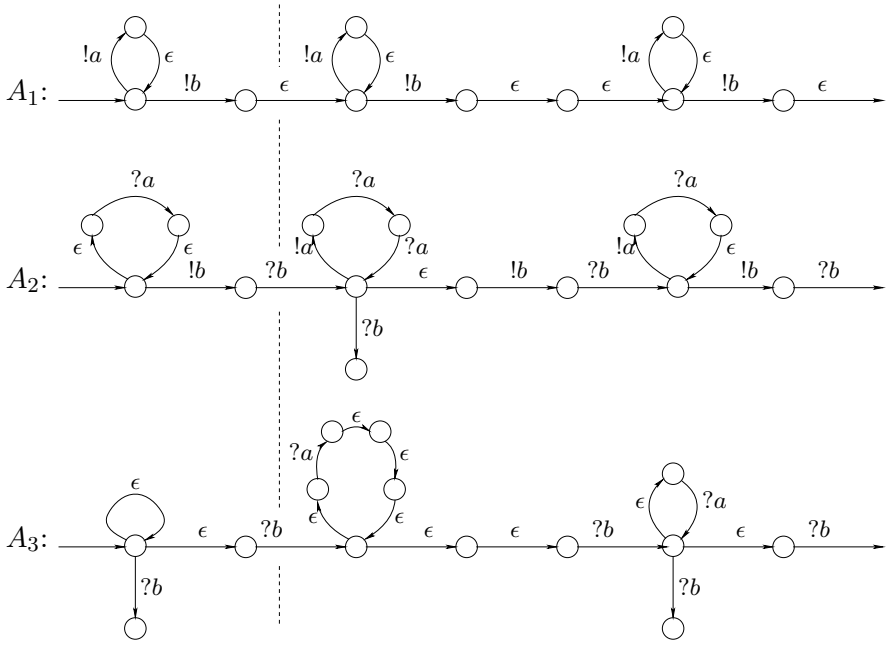

Fig. 5. A widget for doubling of the number of a's – incrementation of C_1

sequence is twice as long as the first one. The third sequence is as long as the second one (otherwise, A_3 ends up in the error sink), but A_2 gets desynchronized at the same time. This is a preparation for the next operation. Therefore, the simulation of the next instruction does not start with the first loop, but it goes directly to the second loop (behind the dashed line). This is to ensure overlapping of the length checking. Each transition takes one time unit, we omit guards and resets ($x = 1, x := 0$). All these constructions also work in the discrete time, but they do not work for non-urgent semantics. The encodings for halving and test for zero are given in the full version of this paper [KY06].

Theorem 2. *Reachability for networks of communicating timed automata of the form* $(A_1, A_2, A_3, c_{1,2}, c_{2,3})$ *is undecidable.*

5 Conclusions

To the best of our knowledge, this is the first attempt to study channel systems in the timed setting. We have proposed CTA as a general framework for modeling of channel systems in which the relative speeds of message production and consumption by local components must meet given timing constraints. Our goal is to mark the basic ground by identifying decidable and undecidable problems for such systems and raise relevant questions for future work. Our technical results can be summarized as follows: (1) CTA with one channel without sharing states in the form $(A_1, A_2, c_{1,2})$ (as shown in Figure 1) is equivalent to one-counter machine and therefore questions such as state reachability and channel boundedness are decidable for such systems, and (2) CTA with two channels without sharing states in the form $(A_1, A_2, A_3, c_{1,2}, c_{2,3})$ (as shown in Figure 2) has the power of Turing machines.

An interesting question related to the timed setting is whether one can synthesize the clock constraints of a CTA (or a controller for a CTA in general) under given liveness requirements such that the channel content remains bounded. As future work, we will also study and develop abstraction techniques for efficient analysis of timed channel systems.

References

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AJ96a] Parosh Aziz Abdulla and Bengt Jonsson. Undecidable verification problems for programs with unreliable channels. *Information and Computation*, 130(1):71–90, 1996.

[AJ96b] Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.

[BJLY98] Johan Bengtsson, Bengt Jonsson, Johan Lilius, and Wang Yi. Partial order reductions for timed systems. In *Proc. of CONCUR'98*, volume 1466 of *LNCS*, pages 485–500. Springer, 1998.

[BZ83] Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.

[CF05] Gérard Cécé and Alain Finkel. Verification of programs with half-duplex communication. *Information and Computation*, 202(2):166–190, 2005.

[CFP96] Gérard Cécé, Alain Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, 1996.

[FM97] Alain Finkel and Pierre McKenzie. Verifying identical communicating processes is undecidable. *Theoretical Computer Science*, 174(1-2):217–230, 1997.

[FPS03] Alain Finkel, S. Purushothaman Iyer, and Grégoire Sutre. Well-abstracted transition systems: Application to FIFO automata. *Information and Computation*, 181(1):1–31, 2003.

[GMK04] Blaise Genest, Anca Muscholl, and Dietrich Kuske. A Kleene theorem for a class of communicating automata with effective algorithms. In *Proc. of DTL'04*, volume 3340 of *LNCS*, pages 30–48. Springer, 2004.

[KP05] Pavel Krčál and Radek Pelánek. On sampled semantics of timed systems. In *Proc. of FSTTCS'05*, volume 3821 of *LNCS*, pages 310–321. Springer, 2005.

[KY06] Pavel Krcal and Wang Yi. Communicating timed automata. Technical Report 2006-008, Uppsala University, 2006.

[Pac82] Jan K. Pachl. Reachability problems for communicating finite state machines. Technical Report CS-82-12, Department of Computer Science, University of Waterloo, 1982.

[Pac03] Jan K. Pachl. Reachability problems for communicating finite state machines. *ArXiv Computer Science e-prints*, arXiv:cs/0306121, 2003.

[PP92] Wuxu Peng and S. Purushothaman Iyer. Analysis of a class of communicating finite state machines. *Acta Informatica*, 29(6/7):422–499, 1992.