Timed Lossy Channel Systems*

Parosh Aziz Abdulla[1], Mohamed Faouzi Atig[1], and Jonathan Cederberg[1]

1 Uppsala University, Sweden

Abstract

Lossy channel systems are a classical model with applications ranging from the modeling of communication protocols to programs running on weak memory models. All existing work assume that messages traveling inside the channels are picked from a finite alphabet. In this paper, we extend the model by assuming that each message is equipped with a clock representing the age of the message, thus obtaining the model of *Timed Lossy Channel Systems (TLCS)*. The main contribution of the paper is to show that the control state reachability problem is decidable for TLCS.

1998 ACM Subject Classification D.2.4 Software/Program Verification

Keywords and phrases Lossy channel systems, timed automata, model checking

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2012.374

1 Introduction

During the last two decades there has been a large amount of work devoted to the verification of *discrete* program models that have *infinite* state spaces such as Petri nets, pushdown systems, counter automata, and channel machines. In particular *lossy channel systems* have been studied extensively as a model of communication protocols. Such protocols are designed to work correctly even in the case where the underlying medium is unreliable in the sense that it can lose messages [8]. Recently, lossy channel systems have been proposed as a fundamental tool for describing programs running on *weak memories* [10, 6] since they are able to capture the behaviors of classical models such as TSO and PSO. In parallel, *timed automata* [9, 15, 14] are the most widely used model for the analysis of systems with *timed* behaviors. Several works have augmented discrete infinite-state models with timed behaviors. For instance, many different formalisms have been proposed for extending Petri nets with clocks and timed constraints, leading to various definitions of *Timed Petri Nets* (e.g., [12, 5]). Also, several works [4, 13, 11, 17, 18, 19, 22] consider timed pushdown automata. In this paper, we consider *(Dense-)Timed Lossy Channel Systems* (or *TLCS* for short). A TLCS combines the classical models of lossy channel systems and timed automata. More precisely, a TLCS consists of finite number of processes. The processes operate on finite set of real-valued clocks, together with a finite number of lossy channels each of which behaves as an unbounded FIFO buffer. Each message traveling inside a channel is equipped with a real-valued clock representing its "age". Processes can *send* messages to the channels in which case the message is appended to the end of the channel. A *receive* operation may only take place if the message at the head of the channel is of the correct type and only if its age lies in a pre-defined interval associated with the transition. In a similar manner to timed automata, a transition may be conditioned by the values of the clocks. In a *timed transition*,

* Supported by the Swedish Research Council within the UPMARC Linnaeus centre of Excellence

32nd Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012).
Editors: D. D'Souza, J. Radhakrishnan, and K. Telikepalli; pp. 374–386
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the clock values and the ages of all the messages inside the channels are increased uniformly (by the same real number). Finally, any message inside a channel may non-deterministically be lost (deleted from the channel). The TLCS model thus subsumes both the models of lossy channel systems and timed automata. More precisely, we obtain the former if we prevent the TLCS from using the timed information (all the timing constraints are trivially valid); and obtain the latter if we prevent the TLCS from using the channels (no symbols are sent or received from the channels). Notice that a TLCS induces a system that is infinite in two dimensions, namely it has channels containing unbounded numbers of messages, and each message is equipped with a real-valued clock.

In this paper, we show decidability of the control state reachability problem for TLCS. We show the decidability result through a novel reduction formulated in two steps. First, we introduce a new model called *Dynamic Lossy Channel Systems (DLCS)* which is a generalization of (untimed) LCS. More precisely, a DLCS contains, in addition to a (fixed) finite set of lossy channels, a *dynamic* part that contains an *a priori* unbounded number of channels. The dynamic part behaves as a *second-order* lossy channel, i.e., a "lossy channel of lossy channels". We show that each DLCS induces a transition system that is *well quasi-ordered* in the sense of [7, 1], and thus the control state reachability problem is decidable for DLCS. In the second step, we reduce the control state reachability problem for TLCS to the the control state reachability problem for DLCS and thus prove the decidability of the former.

The complexity of the reachability problem for TLCS is not primitive recursive as it is not primitive recursive already for untimed LCS [16].

2 Preliminaries

Notation

We use $\mathbb{N}$ and $\mathbb{R}^{\geq 0}$ to denote the sets of natural numbers resp. non-negative reals. For a real number $r \in \mathbb{R}^{\geq 0}$, we define $\mathsf{Int}(r)$ as the greatest $n \in \mathbb{N}$ such that $n \leq r$, and $\mathsf{Frac}(r)$ as $r - \mathsf{Int}(r)$. We call $\mathsf{Int}(r)$ the *integer part* and $\mathsf{Frac}(r)$ the *fractional part* of r respectively. An open interval is written as (i, j) where $i \in \mathbb{N}$ and $j \in \mathbb{N} \cup \{\infty\}$. Intervals can also be closed in one or both directions, e.g. $[i, j]$ is closed in both directions and $[i, j)$ is closed to the left and open to the right. We denote the set of all intervals by $\mathcal{I}$. For $n \in \mathbb{N}$, we define the set $[n]^0 := \{0, 1, \ldots, n\}$, and define $[n]^1 := \{1, 2, \ldots, n\}$. For sets A and B, we use $h : A \to B$ to denote that h is a total function from A to B, and use $h[a \mapsto b]$ to denote the function h' where $h'(a) = b$ and $h'(a') = h(a')$ if $a' \neq a$. We use $(A \to B)$ to denote the set of total functions from A to B. We say that a function $f : \mathbb{N} \to \mathbb{N}$ is *strictly increasing* if whenever $i < j$ we also have $f(i) < f(j)$. We use A^* to denote the set of finite words over A. For words $w_1, w_2 \in A^*$, we use $w_1 \cdot w_2$ to denote the concatenation of w_1 and w_2. We use ϵ to denote the empty word. For a word $w = a_1 \cdots a_n$, we use $w[i]$ to denote the ith symbol a_i in w, and we will write $a \in w$ if $a = w[i]$ for some $i : 1 \leq i \leq n$. We will use a similar notation for tuples. We recall the classical subword ordering $\sqsubseteq$ on the set A^* of words, where $a_1 \ldots a_m \sqsubseteq a'_1 \cdots a'_n$ if there is a strictly increasing injection $g : [m]^1 \to [n]^1$ such that $a_i = a'_{g(i)}$. To simplify the notation, we write $\omega \in (A^*)^*$ as $\langle w_1 \rangle \cdots \langle w_n \rangle$ where $w_1, \cdots, w_n$ are words in A^*. We extend the ordering $\sqsubseteq$ to $(A^*)^*$ in such a way that $\omega = \langle w_1 \rangle \cdots \langle w_n \rangle \sqsubseteq \langle w'_1 \rangle \cdots \langle w'_n \rangle = \omega'$ if there is a strictly increasing injection $h : [m]^1 \to [n]^1$ where $w_i \sqsubseteq w'_{h(i)}$.

Transition Systems

A *transition system* is a pair $\mathcal{S} = \langle \Gamma, \longrightarrow \rangle$ where Γ is the set of configurations, and $\longrightarrow \subseteq \Gamma \times \Gamma$ is a binary relation on the set of configurations. As usual, we write $\gamma_1 \longrightarrow \gamma_2$ instead of $\langle \gamma_1, \gamma_2 \rangle \in \longrightarrow$. We use $\xrightarrow{*}$ to denote the reflexive transitive closure of $\longrightarrow$. For a set $\Gamma' \subseteq \Gamma$ of configurations, we define the set $Pre(\Gamma') := \{\gamma | \exists \gamma' \in \Gamma'. \gamma \longrightarrow \gamma'\}$. Sometimes, we equip the set Γ with an ordering $\trianglelefteq$ and write the transition system as a triple $\langle \Gamma, \longrightarrow, \trianglelefteq \rangle$. We say that $\mathcal{S}$ is *monotone* (wrt. $\trianglelefteq$) if whenever $\gamma_1 \longrightarrow \gamma_2$ and $\gamma_1 \trianglelefteq \gamma_3$ then $\gamma_2 \xrightarrow{*} \gamma_4$ for some γ_4 with $\gamma_3 \trianglelefteq \gamma_4$. We say that $\trianglelefteq$ is a *well quasi-ordering* (*wqo* for short), if, for all sequences $\gamma_0, \gamma_1, \gamma_2, \ldots$, there are $i < j$ with $\gamma_i \trianglelefteq \gamma_j$. A set $U \subseteq \Gamma$ is *upward closed* if whenever $\gamma_1 \in U$ and $\gamma_1 \trianglelefteq \gamma_2$ then $\gamma_2 \in U$. The upward closure of a set $\Gamma' \subseteq \Gamma$ is defined by $\Gamma' \uparrow := \{\gamma \in \Gamma | \exists d \in \Gamma'. d \trianglelefteq \gamma\}$. For sets $\Gamma'_1 \subseteq \Gamma'_2 \subseteq \Gamma$, we say that Γ'_1 is a *minor* of Γ'_2 if (i) for each $\gamma_2 \in \Gamma'_2$ there is a $\gamma_1 \in \Gamma'_1$ such that $\gamma_1 \trianglelefteq \gamma_2$, and (ii) $\gamma_1 \trianglelefteq \gamma_2$ implies $\gamma_1 = \gamma_2$ for all $\gamma_1, \gamma_2 \in \Gamma'_1$. If $\trianglelefteq$ is a wqo, then each minor is finite. However, in general, a set may have several different minors. In the applications of this paper, each set Γ' has a unique minor, denoted $\min(\Gamma')$. An instance of the *coverability problem* consists of two configurations γ_1 and γ_2. The task is to check whether $\gamma_1 \xrightarrow{*} \gamma_2\uparrow$. A transition system $\langle \Gamma, \longrightarrow, \trianglelefteq \rangle$ is said to be *well quasi-ordered* if the following conditions are satisfied: (i) $\trianglelefteq$ is computable, i.e., for given configurations γ, γ', we can check whether $\gamma_1 \trianglelefteq \gamma'$, (ii) $\trianglelefteq$ is a wqo, (iii) $\longrightarrow$ is monotone wrt. $\trianglelefteq$, (iv) for a configuration γ, we can compute the (finite) set $\min(Pre(\{\gamma\}\uparrow))$. Notice that, since the transition relation is monotone with respect to $\trianglelefteq$, it follows that the set $Pre(\{\gamma\}\uparrow)$ is upward closed. The classical framework of well quasi-ordered transition systems [7, 1] provides the following sufficient conditions for decidability of the coverability problem.

▶ **Theorem 1.** *The coverability problem is decidable for well quasi-ordered transition systems.*

3 Timed Lossy Channel Systems

In this section, we introduce TLCS, define their operational semantics, and present the reachability problem. Furthermore, we show that it is sufficient to consider a class of "normalized" TLCS where initial ages of messages and values assigned to clocks are always 0.

A TLCS has three parts, a control part, a finite set of clocks, and a finite set of channels. The control part is a finite-state labeled transition system, where the labels are either clock operations or channel operations. The control part can be used to model the total behavior of a number of processes that communicate through the channels. The clocks assume real values, while the channels are unbounded lossy FIFO buffers.

Model

A *Timed Lossy Channel System* (*TLCS* for short) is a tuple $\mathcal{T} = \langle S, s_{init}, C, M, X, \Delta \rangle$, where S is a finite set of (control) states, $s_{init} \in S$ is the initial control state, C is a finite set of channels, M is a finite set of messages, X is a finite set of clocks, and Δ is a finite set of transitions. A transition $t \in \Delta$ is a triple $\langle s_1, op, s_2 \rangle$ where $s_1, s_2 \in S$ are states and op is an operation of one of the following forms:

1. nop is an empty operation that does not check or update the clock values or the channel contents.

2. $c!(m \in I)$ appends a new message $m \in M$ to the end of the channel $c \in C$. The initial age of the new message is selected non-deterministically from $I \in \mathcal{I}$.
3. $c?(m \in I)$ removes (receives) the message at the head of the channel $c \in C$ provided that this message is $m \in M$ and that its age lies in $I \in \mathcal{I}$.
4. $x \in I$ checks whether the value of $x \in X$ belongs to the interval $I \in \mathcal{I}$.
5. $x \leftarrow I$ assigns non-deterministically a value to $x \in X$ from $I \in \mathcal{I}$.

Configurations

A configuration γ of $\mathcal{T}$ is a triple $\langle s, \mathtt{X}, \nu \rangle$, where $s \in S$ is a control state, $\mathtt{X} \in \left(X \to \mathbb{R}^{\geq 0}\right)$ defines the clock values (assigns a real number to each clock), and $\nu \in \left(C \to (M \times \mathbb{R}^{\geq 0})^*\right)$ defines the content of each channel (the content of a channel is represented by a word, where each message is represented by a pair containing its name and its age).

Transition Relation

We define a transition relation on configurations $\longrightarrow_{\mathcal{T}} := \xrightarrow{D}_{\mathcal{T}} \cup \xrightarrow{T}_{\mathcal{T}} \cup \xrightarrow{\mathcal{L}}_{\mathcal{T}}$ as the union of a discrete transition relation $\xrightarrow{D}_{\mathcal{T}}$, a timed transition relation $\xrightarrow{T}_{\mathcal{T}}$, and a lossy transition relation $\xrightarrow{\mathcal{L}}_{\mathcal{T}}$.

We define the discrete transition relation as the union $\xrightarrow{D}_{\mathcal{T}} := \bigcup_{t \in \Delta} \xrightarrow{t}_{\mathcal{T}}$ of the transition relations induced by all transitions in Δ. For configurations $\gamma_1 = \langle s_1, \mathtt{X}_1, \nu_1 \rangle$, $\gamma_2 = \langle s_2, \mathtt{X}_2, \nu_2 \rangle$, and a transition $t = \langle s_1, op, s_2 \rangle \in \Delta$, we have $\gamma_1 \xrightarrow{t} \gamma_2$ if one of the following conditions holds:

1. $op = \mathsf{nop}$, $\mathtt{X}_2 = \mathtt{X}_1$, and $\nu_2 = \nu_1$. The empty operation does not affect the clock values or the channel contents.
2. $op = c!(m \in I)$, $\mathtt{X}_2 = \mathtt{X}_1$, $\nu_2 = \nu_1[c \mapsto (m, \delta) \cdot \nu_1(c)]$, and $\delta \in I$. The transition appends a new message to the end of the channel c with name m, and with an age that belongs to the interval I.
3. $op = c?(m \in I)$, $\mathtt{X}_2 = \mathtt{X}_1$, $\nu_1 = \nu_2[c \mapsto \nu_2(c) \cdot (m, \delta)]$, and $\delta \in I$. The transition removes the message at the head of the channel c provided that its name is m, and that its age is in the interval I.
4. $op = x \in I$, $\mathtt{X}_1(x) \in I$, $\mathtt{X}_2 = \mathtt{X}_1$, and $\nu_2 = \nu_1$. The transition is enabled only if the value of x belongs to I. The clock values and the channel contents are not affected.
5. $op = x \leftarrow I$, $\mathtt{X}_2 = \mathtt{X}_1[x \mapsto \delta]$, $\delta \in I$, and $\nu_2 = \nu_1$. The transition assigns a new value (belonging to I) to the clock x.

Notice that in all five cases the control state changes from s_1 to s_2.

The timed transition relation models the passage of time, in the sense that the values of all clocks and the ages of all messages inside the channels are uniformly increased by (the same) real number. For configurations $\gamma_1 = \langle s, \mathtt{X}_1, \nu_1 \rangle$, $\gamma_2 = \langle s, \mathtt{X}_2, \nu_2 \rangle$, and a real number $\delta \in \mathbb{R}^{\geq 0}$, the relation $\gamma_1 \xrightarrow{\delta}_{\mathcal{T}} \gamma_2$ holds if the following two conditions hold: (i) $\mathtt{X}_2(x) = \mathtt{X}_1(x) + \delta$ for all $x \in X$, and (ii) for every $c \in C$, if $\nu_1(c)$ is of the form $(m_1, \delta_1) \cdots (m_n, \delta_n)$ then ν_2 is of the form $(m_1, \delta_1 + \delta) \cdots (m_n, \delta_n + \delta)$. We write $\gamma_1 \xrightarrow{T}_{\mathcal{T}} \gamma_2$ to denote that $\gamma_1 \xrightarrow{\delta}_{\mathcal{T}} \gamma_2$ for some $\delta \in \mathbb{R}^{\geq 0}$.

Finally the lossy transition relation allows messages to be lost from the channels at any time. Formally, if $\gamma_1 = \langle s, \mathtt{X}, \nu_1 \rangle$ and $\gamma_2 = \langle s, \mathtt{X}, \nu_2 \rangle$, the relation $\gamma_1 \xrightarrow{\mathcal{L}}_{\mathcal{T}} \gamma_2$ holds if $\nu_2(c) \sqsubseteq \nu_1(c)$ for all $c \in C$.

Reachability

The initial configuration of a TLCS $\mathcal{T}$ is defined by $\gamma_{init} := \langle s_{init}, \mathtt{X}_{init}, \nu_{init} \rangle$ where $\mathtt{X}_{init}(x) = 0$ for all $x \in X$, and $\nu_{init}(c) = \epsilon$ for all $c \in C$. In other words, $\mathcal{T}$ is initiated from a configuration where it is in its initial control state, where all the clocks have a value equal to 0, and where all the channels are empty. A control state $s \in S$ is said to be *reachable* if $\gamma_{init} \xrightarrow{*}_{\mathcal{T}} \langle s, \mathtt{X}, \nu \rangle$ for some $\mathtt{X}$ and ν. An instance of the reachability problem consists of an TLCS $\mathcal{T} = \langle S, s_{init}, C, M, X, \Delta \rangle$ and a control state $s \in S$. The task is to check whether s is reachable.

Normalization

A TLCS $\mathcal{T} = \langle S, s_{init}, C, M, X, \Delta \rangle$ such that $I = [0, 0]$ for all $\langle s_1, c!(m \in I), s_2 \rangle \in \Delta$ is said to be *message-normalized*. We say that $\mathcal{T}$ is *clock-normalized* if whenever $\langle s_1, x \leftarrow I, s_2 \rangle \in \Delta$ then $I = [0, 0]$. Finally, $\mathcal{T}$ is *normalized* if it is both clock- and message-normalized. The following two lemmas show that the reachability problem for general TLCS can be reduced to that for normalized TLCS. Therefore, in the rest of the paper, we assume that all TLCS are normalized.

▶ **Lemma 2.** *The reachability problem for TLCS can be reduced to the reachability problem for message-normalized TLCS.*

▶ **Lemma 3.** *The reachability problem for TLCS can be reduced to the reachability problem for clock-normalized TLCS.*

4 Dynamic Lossy Channel Systems

In this section, we introduce the model of Dynamic Lossy Channel Systems (*DLCS* for short). The model is a generalization of lossy channel systems [8] in the sense that it contains a second-order channel (a "channel of channels"). A DLCS consists of three parts: a control part, a static part, and a dynamic part. The control part is a finite-state labeled transition system. The static part consists of a finite set of (static) channels, each of which contains a sequence of messages from a finite alphabet. The dynamic part contains a (possibly unbounded) sequence of (dynamic) channels over the same alphabet. Each transition of the control part may be labeled by an operation on the static or dynamic channels. In the former case, the operation may remove a message from the head of a static channel or insert a message at its end (as in the case of lossy channels). In the latter case, the operation may copy the content of a static channel and append it (as a new channel) to the end of the sequence of dynamic channels (thus creating a new channel at the leftmost position of the dynamic part), or copy the content of the rightmost dynamic channel (the one at the head of the sequence of channels) to a static channel and then delete this dynamic channel. Furthermore, messages inside any channel can be lost (deleted) non-deterministically, and also any (whole) dynamic channel may be lost non-deterministically. The static channels are static (they can cannot be created, deleted, or lost). Notice that all the channels in the system are unbounded and that there is no bound on the number of dynamic channels that may be created during a run of the system.

Model

A *DLCS* is a tuple $\mathcal{D} = \langle S, s_{init}, C, \Sigma, \Delta \rangle$ where S is a finite set of (control) states, $s_{init} \in S$ is the initial control state, C is a finite set of channels names, Σ is the channel alphabet, and Δ is a finite set of transitions. A transition $t \in \Delta$ is a triple $\langle s_1, op, s_2\rangle$ where $s_1, s_2 \in S$ are states and op is an operation of one of the following forms:

1. nop is an empty operation that does not check or update the channels,
2. $c!m$ appends the message $m \in \Sigma$ to the end of the static channel $c \in C$,
3. $c?m$ removes the message $m \in \Sigma$ from the head of the static channel $c \in C$,
4. $send\_channel(c)$ makes a copy of the content of the static channel c to a new dynamic channel, and appends the new channel to the end of the sequence of dynamic channels.
5. $receive\_channel(c)$ copies the content of the rightmost dynamic channel to the static channel $c \in C$ and then removes this dynamic channel from the sequence of channels.

Configurations

A configuration d of $\mathcal{D}$ is a triple $\langle s, \nu, \omega\rangle$, where $s \in S$ is a control state, $\nu \in (C \to \Sigma^*)$ is a function that represents the content of the set of static channels C, and $\omega \in (\Sigma^*)^*$ is the content of the sequence of dynamic channels, also called the dynamic part of $\mathcal{D}$.

For configurations $d_1 = \langle s_1, \nu_1, \omega_1\rangle$, $d_2 = \langle s_2, \nu_2, \omega_2\rangle$, we say that $d_1 \sqsubseteq d_2$ if $s_1 = s_2$, $\nu_1(c) \sqsubseteq \nu_2(c)$ for all $c \in C$, and $\omega_1 \sqsubseteq \omega_2$ (recall the definition of $\sqsubseteq$ from Section 2). Intuitively, we derive d_1 from d_2 by deleting messages from the channels (both static and dynamic) and by removing dynamic channels.

Transition Relation

We define the transition relation as the set $\longrightarrow_{\mathcal{D}} := \left(\bigcup_{t\in\Delta} \xrightarrow{t}_{\mathcal{D}}\right) \cup \xrightarrow{\mathcal{L}}_{\mathcal{D}}$ where $\bigcup_{t\in\Delta} \xrightarrow{t}_{\mathcal{D}}$ is the union of transition relations induced by all transitions in Δ, and $d_1 \xrightarrow{\mathcal{L}}_{\mathcal{D}} d_2$ whenever $d_2 \sqsubseteq d_1$. The relation $\xrightarrow{\mathcal{L}}_{\mathcal{D}}$ models the loss of messages and dynamic channels. For configurations $d_1 = \langle s_1, \nu_1, \omega_1\rangle$, $d_2 = \langle s_2, \nu_2, \omega_2\rangle$, and a transition $t = \langle s_1, op, s_2\rangle \in \Delta$, we have $d_1 \xrightarrow{t}_{\mathcal{D}} d_2$ if one of the following conditions holds:

1. $op =$ nop, $\nu_1 = \nu_2$, and $\omega_1 = \omega_2$.
2. $c!m$, $\nu_2 = \nu_1[c \mapsto m \cdot \nu_1(c)]$, and $\omega_2 = \omega_1$. The message m is appended to the end of the channel c.
3. $c?m$, $\nu_1 = \nu_2[c \mapsto \nu_2(c) \cdot m]$, and $\omega_2 = \omega_1$. The message m is received (deleted) from the head of the channel c.
4. $send\_channel(c)$, $\nu_1 = \nu_2$, and $\omega_2 = \langle \nu_1(c)\rangle \cdot \omega_1$. A copy of the content of the static channel c is appended (as a new channel) to the end of the dynamic part of $\mathcal{D}$.
5. $receive\_channel(c)$, $\nu_2 = \nu_1[c \mapsto w]$, and $\omega_1 = \omega_2 \cdot \langle w\rangle$. The content of the right-most dynamic channel is copied to the static channel $c \in C$. The right-most dynamic channel is then removed.

Reachability

The initial configuration of an DLCS $\mathcal{D}$ is defined by $d_{init} := \langle s_{init}, \nu_{init}, \omega_{init}\rangle$ where $\nu_{init}(c) = \epsilon$ for all $c \in C$, and $\omega_{init} = \epsilon$. In other words, $\mathcal{D}$ is initiated from a configuration where it is in its initial control state, all the static channels are empty, and the sequence of dynamic channels is empty (no channel has yet been appended). We define the control state reachability problem (or simply the reachability problem in the sequel) in a similar manner to the case of TLCS (cf. Section 3). Notice that the checking of reachability of a control state s can translated to the coverability problem $d_{init} \xrightarrow{*}_{\mathcal{D}} \langle s, \nu_{init}, \omega_{init}\rangle\uparrow$.

▶ **Lemma 4.** *Any transition system $\langle \Gamma, \longrightarrow, \sqsubseteq \rangle$ induced by a DLCS is well quasi-ordered.*

Proof. We prove the lemma by showing that each of the four conditions in the definition of well quasi-ordered transition systems given in Section 2 holds.

1. The ordering defined is clearly computable.
2. Since any finite set is well quasi-ordered and also tuples and words over well quasi-ordered sets are well quasi-ordered [21], the ordering $\sqsubseteq$ as defined on configurations is a well quasi-ordering.
3. Assume $d_1 \longrightarrow d_2$ and $d_1 \sqsubseteq d_3$. From the definition of $\longrightarrow$, we get that $d_3 \xrightarrow{\mathcal{L}} d_1$, and by transitivity we immediately get $d_3 \xrightarrow{*} d_2$. Thus, $\longrightarrow$ is monotone wrt. $\sqsubseteq$.
4. Assume a configuration $d = \langle s, \nu, \omega \rangle$. We define $\min(Pre(\{d\}\uparrow)) :=$ $\min\left(\bigcup_{t \in \Delta} \min(Pre(t)(\{d\}\uparrow)) \cup \{d\}\right)$, where $Pre(t)(\{d\}\uparrow) = \{d_1 \mid \exists d_2 \in \{d\}\uparrow \,.\, d_1 \xrightarrow{t}_{\mathcal{D}} d_2\}$ is the predecessor relation wrt. the transition $t \in \Delta$. Consider a transition $t = \langle s_1, op, s_2 \rangle \in \Delta$. We define $\min(Pre(t)(\{d\}\uparrow))$ as a set A with the following properties. If $s \neq s_2$ then $A := \emptyset$. Otherwise, we have:
   - If $op = \mathsf{nop}$ then $A = \{\langle s_1, \nu, \omega \rangle\}$.
   - If $op = c!m$ and $\nu(c)$ is of the form $m \cdot w$ then $A := \{\langle s_1, \nu[c \mapsto w], \omega \rangle\}$.
   - If $op = c!m$, $\nu(c)$ is of the form $m' \cdot w$, and $m' \neq m$, then $A := \{\langle s_1, \nu, \omega \rangle\}$.
   - If $op = c?m$ then $A := \{\langle s_1, \nu[c \mapsto w \cdot m], \omega \rangle\}$.
   - If $op = send\_channel(c)$ and ω is of the form $\langle w \rangle \cdot \omega'$ then $A := \min(\{\langle s_1, \nu[c \mapsto w'], \omega \rangle \mid (\nu(c) \sqsubseteq w') \wedge (w \sqsubseteq w')\} \cup \{\langle s_1, \nu, \omega \rangle\})$
   - If $op = send\_channel(c)$ and $\omega = \epsilon$ then $A := \{\langle s_1, \nu, \omega \rangle\}$.
   - If $op = receive\_channel(c)$ then $A := \{\langle s_1, \nu[c \mapsto \epsilon], \omega \cdot \langle \nu(c) \rangle \rangle\}$.

◀

From this and Theorem 1 we get the following theorem.

▶ **Theorem 5.** *The reachability problem is decidable for DLCS.*

5 From TLCS to DLCS

In this section, we show how we can encode a TLCS by a DLCS such that we preserve control state reachability. This enables us to extend decidability of the reachability problem from DLCS to TLCS.

▶ **Theorem 6.** *The reachability problem is decidable for TLCS.*

Given an instance of the reachability problem, defined by a TLCS $\mathcal{T} = \langle S, s_{init}, C, M, X, \Delta \rangle$ and a control state $s \in S$, we construct an equivalent instance of the reachability problem, defined by a DLCS $\mathcal{D} = \left\langle S^{\mathcal{D}}, s^{\mathcal{D}}_{init}, C^{\mathcal{D}}, \Sigma^{\mathcal{D}}, \Delta^{\mathcal{D}} \right\rangle$ (that we derive from $\mathcal{T}$) and the (same) control state s (as we shall see, all control states in S belong also to $S^{\mathcal{D}}$). The idea of the proof is inspired in parts by the region construction for timed automata [9]. A major difficulty in our case is the fact that we have unboundedly many ages to keep track of, and the fact that we also have to keep track of the ordering of an unbounded number of messages inside the channels. We will describe the ingredients of the encoding (the derivation of $\mathcal{D}$ from $\mathcal{T}$) step by step. First, we will introduce the set $C^{\mathcal{D}}$ of channels and the alphabet $\Sigma^{\mathcal{D}}$ for such channels, then we will define the encoding into a configuration of $\mathcal{D}$ of a configuration of $\mathcal{T}$. We will then define a set of meta-transitions, to aid us in the final task of this section, namely presenting how to simulate a run of $\mathcal{T}$ using our encoding $\mathcal{D}$.

Below, let k_{max} be the largest integer that occurs in the definition of any interval in Δ.

$\Sigma^{\mathcal{D}}$ and $C^{\mathcal{D}}$

As in the case of timed automata, we conclude that it is not meaningful to keep track of exact values of clocks and exact ages of messages beyond k_{max}. Each message in m with age r traveling inside a channel c in $\mathcal{T}$ will be encoded by a pair $\langle\langle c, m\rangle, j\rangle$ in $\mathcal{D}$ where $j = \mathsf{Int}(r)$ if $r \leq k_{max}$ and $j = \infty$ if $r > k_{max}$. The message m thus belongs to the set $\Sigma_m := (C \times M) \times \left([k_{max}]^0 \cup \{\infty\}\right)$. We will use three types of channels in $\mathcal{D}$ to store messages. First, we use a static channel c_0 to store messages whose ages are $\leq k_{max}$ and whose fractional parts are zero. Second, we use the dynamic part to store messages whose values are $\leq k_{max}$ and whose fractional parts are strictly positive. Messages stored in the same dynamic channels encode messages in $\mathcal{T}$ that have identical fractional parts. The fractional parts of messages inside different dynamic channels have increasing fractional parts as we move from left to right. Finally, we use a static channel c_∞ to store messages whose ages are $> k_{max}$.

We will also encode the clocks of $\mathcal{T}$ as messages in the channels of $\mathcal{D}$. To that end we define $\Sigma_x := X \times \left([k_{max}]^0 \cup \{\infty\}\right)$. A clock x will then be represented by a pair $\langle x, j\rangle$ that will be interpreted in a similar manner as above. Throughout the simulation, we will satisfy the invariant that at most one copy of each clock x will be present inside the channels of $\mathcal{D}$. For messages from the set $\Sigma_m \cup \Sigma_x$, we refer to the second component of the tuple as the *age* of the message.

Finally, for technical reasons, we will use a special *sentinel* message $\#$ and a *temporary channel* c_{tmp}. In summary we define $\Sigma^{\mathcal{D}} := \Sigma_m \cup \Sigma_x \cup \{\#\}$, and define $C^{\mathcal{D}} := \{c_0, c_\infty, c_{tmp}\}$.

Encoding of Configurations

We show how to abstract (encode) configurations of $\mathcal{T}$ by configurations of $\mathcal{D}$. For each configuration in $\mathcal{T}$ we will define a set $\alpha(\gamma)$ of configurations in $\mathcal{D}$. In our simulation, all these configurations will have equivalent behaviors and any one of them may be chosen to represent γ. The abstraction relies crucially on a property satisfied by all configurations that arise in a run of $\mathcal{T}$. More precisely, since $\mathcal{T}$ is normalized (cf. Section 3), the ages of messages inside any channel are sorted (if $\langle m_1, r_1\rangle$ is in on the left of $\langle m_2, r_2\rangle$ then $r_1 \leq r_2$). Furthermore, the ordering in which the messages occur inside the channel reflects the ordering in which they were sent to the channel (in particular, this holds even if $r_1 = r_2$).

We present the encoding in several steps. First, we define some operations on words $w \in \left(((C \times M) \cup X) \times \mathbb{R}^{\geq 0}\right)^*$. Let $r \in [0, 1)$ and $u = \langle\sigma'_1, a'_1\rangle \cdots \langle\sigma'_n, a'_n\rangle$ be the longest subword of w such that $\mathsf{Frac}(a'_i) = r$ for all i. We define the *fractional projection* of w with respect to r, written $w|_r$, as the word $\langle\sigma'_1, \mathsf{Int}(a'_1)\rangle \cdots \langle\sigma'_n, \mathsf{Int}(a'_n)\rangle$. In other words, $w|_r$ is obtained by (i) constructing the subword of w that consists of only pairs where the fractional part of the age is equal to r, and (ii) removing r from the age of each message in the sequence.

Consider a configuration $\gamma = \langle s, \nu, \omega\rangle$. We will partition the messages and the clocks depending on whether their ages exceed k_{max} or not. For a channel $c \in C$ such that $\nu(c) = (m_1, a_1)(m_2, a_2) \cdots (m_n, a_n)$, let k be the greatest i such that $a_i \leq k_{max}$. For ease of notation, we define the two words $c^{\leq k_{max}} := \langle\langle c, m_1\rangle, a_1\rangle \cdots \langle\langle c, m_k\rangle, a_k\rangle$ and $c^{>k_{max}} := \langle\langle c, m_{k+1}\rangle, \infty\rangle \cdots \langle\langle c, m_n\rangle, \infty\rangle$. Similarly we let $x^{\leq k_{max}} = \langle x_1, \mathtt{X}(x_1)\rangle \cdots \langle x_k, \mathtt{X}(x_k)\rangle$ where $x_1 \cdots x_k$ is an arbitrary enumeration of all $x \in X$ such that $\mathtt{X}(x) \leq k_{max}$. In the same manner, we define $x^{>k_{max}}$ as a word $\langle x_{k+1}, \infty\rangle \cdots \langle x_n, \infty\rangle$ where $x_{k+1} \cdots x_n$ is an arbitrary enumeration of all $x \in X$ such that $\mathtt{X}(x) > k_{max}$. Let $c_1, c_2, \ldots, c_l$ be an enumeration of C. We define $u := (c_1^{\leq k_{max}} \cdot c_2^{\leq k_{max}} \cdots c_l^{\leq k_{max}} \cdot x^{\leq k_{max}})$, i.e., u is the concatenation of the parts of all the channels that has not exceeded k_{max}, and clocks that has not exceeded k_{max}. Finally, let $r_1 < r_2 \ldots < r_j$ be all strictly positive fractional parts occurring in some $c_i^{\leq k_{max}}$ or in $x^{\leq k_{max}}$.

Now we can define the abstraction of γ, written $\alpha(\gamma)$, as the set of all $d = \langle q, \nu, \omega \rangle$ where

- $q = s$
- ν is the function such that $\nu(c_0) = (u)|_0$, $\nu(c_\infty) = c_1^{>k_{max}} \cdot c_2^{>k_{max}} \cdots c_l^{>k_{max}} \cdot x^{>k_{max}}$ and $\nu(c_{tmp}) = \epsilon$.
- $\omega = \langle (u)|_{r_1} \rangle \cdots \langle (u)|_{r_j} \rangle$.

In other words: (i) the abstraction preserves the control state, (ii) all messages and clocks that are $\leq k_{max}$ and have zero fractional parts, are put in c_0, where the relative order of elements in the same channel is preserved, (iii) all messages and clocks that are $> k_{max}$ are put in c_∞, again with relative order preserved, and (iv) the dynamic channel vector is constructed by building a word for each positive fractional part, and order them by these fractional parts.

Intuitively, the abstraction preserves the following invariants:

- Any message or clock with an age not greater than k_{max} is translated into a message consisting of the same message or clock, and and its original age with the fractional part stripped.
- Any two messages, a message and a clock, or two clocks, with age less than or equal to k_{max} will end up in the same channel in the abstracted system if and only if they have the same fractional part of their age in $\mathcal{T}$. For pairs of messages from the same channel in $\mathcal{T}$, their relative order in the channel in $\mathcal{D}$ will be the same as in $\mathcal{T}$.
- For any two messages, a message and a clock, or two clocks, with age less than or equal to k_{max}, the one with the greater fractional part will end up to the right of one with the smaller fractional part.
- Any two messages with an age greater than k_{max} will end up in the c_∞, with their relative order preserved.

Meta-Transitions

We start by defining some meta-transitions for the DLCS, allowing us to more compactly describe the simulation. Due to space restrictions, we only provide an overview of the construction here, for more details see [2]. Each meta-transition consists of a finite set of ordinary DLCS transitions, possibly containing loops and passing through a number of temporary states. Note that even though the meta transitions might cause an execution of our system to block because of picking the wrong branch in some nondeterministic choice, this is not a problem since we are only interested in the study of safety properties. The meta-transitions are defined as follows:

- $\mathsf{empty}(c)$: empties the channel c, by receiving all possible messages.
- $\mathsf{copy}(c_1, c_2)$: copies the content of channel c_1 into channel c_2, overwriting any previous content, while c_1 remains unchanged.
- $\mathsf{filter}(c, \Sigma)$: filters the channel c, such that only elements from Σ remain.
- $\mathsf{map}(c, f)$, acts on the channel c by replacing each message σ with $f(\sigma)$.
- $\mathsf{HasElementsFrom}(c, \Sigma)$: enforces that there is at least one element in the channel c from the set Σ. If this is not the case, the simulation blocks. $\mathsf{HasElementsFrom}(\omega, \Sigma)$ performs the same operation on the set of dynamic channels rather than on a static channel c.

- HasNoElementsFrom(c, Σ), enforces that there no element in the channel c from the set Σ. If this is not the case, the simulation blocks. HasNoElementsFrom(ω, Σ) is defined analogously.
- ReceiveFromSet(c, m, Σ) receives (deletes) the message m from c but only if the following condition holds. Search for the first (rightmost) occurrence of a message $m' \in \Sigma$ in c. If $m' = m$ then it is deleted. If $m' \neq m$ or c does not contain any messages from Σ, the simulation blocks. ReceiveFromSet(ω, m, Σ) is defined analogously for the dynamic part, namely the search is carried out through all the channels from right to left. For a given channel, we search from right to left.

Simulation of Discrete Transitions

Each transition $t = \langle s_1, op, s_2 \rangle \in \Delta$ is simulated using a set of transitions in $\Delta^{\mathcal{D}}$ as follows:

- If $t = \mathsf{nop}$, we let $\langle s_1, \mathsf{nop}, s_2 \rangle \in \Delta^{\mathcal{D}}$.
- If $t = c!m$, we let $\langle s_1, c_0!\langle \langle c, m \rangle, 0 \rangle, s_2 \rangle \in \Delta^{\mathcal{D}}$. In other words, we send the message m, tagged with the identity of the channel, to c_0. This reflects the fact that initial ages of messages are set to 0 (since $\mathcal{T}$ is normalized).
- If $t = c?m \in I$. This is the most complicated case. We need to search the dynamic channels and also the static channels c_0 and c_∞ in $\mathcal{D}$ in order to find the message corresponding to the rightmost message in c. If this message is m then we delete it, otherwise we block the simulation. This is carried out in two steps, namely (i) *guessing*: we non-deterministically "guess" the age of the message, and (ii) *checking*: for the given guess, we check that there are no other messages in channel c that are older than the current one. Concretely, in the *guessing* step we assume that the message has an age which is either (i) $k \in [k_{max}]^0$ for some integer $k \in I$, or (ii) in the interval $(k, k+1)$ for some $k \in [k_{max} - 1]^0$ where $(k, k+1) \subseteq I$, or (iii) in the interval (k_{max}, ∞) if $(k_{max}, \infty) \subseteq I$. Let $\Sigma_1 = ((\{c\} \times M) \times \{\infty\})$, $\Sigma_2 = ((\{c\} \times M) \times \{\ell \,|\, k \leq \ell \leq k_{max}\})$, $\Sigma_3 = ((\{c\} \times M) \times \{\ell \,|\, k < \ell \leq k_{max}\})$ and $\Sigma_4 = ((\{c\} \times M) \times \{k\})$. The *checking* step is carried out depending on the guessed age of the message as follows.
  - Guess $k \in [k_{max}]^0$. We use the operations (i) HasNoElementsFrom(c_∞, Σ_1), (ii) HasNoElementsFrom(ω, Σ_2), and (iii) HasNoElementsFrom(c_0, Σ_3), to ensure that c does not contain any message older than m. Then, use ReceiveFromSet(c_0, m, Σ_4) to try to receive m.
  - Guess $(k, k+1)$ for some $k \in [k_{max} - 1]^0$. We use (i) HasNoElementsFrom(c_∞, Σ_1), (ii) HasNoElementsFrom(ω, Σ_3), and (iii) HasNoElementsFrom(c_0, Σ_3) to ensure that c does not contain any message older than m. Then, use ReceiveFromSet(ω, m, Σ_4) to try to receive m.
  - Guess (k_{max}, ∞). Use ReceiveFromSet(c_∞, m, Σ_1) to try to receive m.
- If $t = x \in I$ then we guess the value of x according to one of the three forms described in the previous case. Since we satisfy the invariant that there is at most one message representing x in the channels of $\mathcal{D}$, the simulation is simpler in this case. More precisely, if we guess the age of x to be k for some $k \in [k_{max}]^0$ then we use HasElementsFrom$(c_0, \{\langle x, k \rangle\})$. If we guess $(k, k+1)$ for some $k \in [k_{max} - 1]^0$ then we use HasElementsFrom$(\omega, \{\langle x, k \rangle\})$. Finally, if we guess (k_{max}, ∞) then we use HasElementsFrom$(c_\infty, \{\langle x, \infty \rangle\})$.
- If $t = x \leftarrow 0$, we simply remove the message representing x from the channels of $\mathcal{D}$, and then send it again with age 0 to c_0. Concretely, we non-deterministically use ReceiveFromSet$(c_0, \langle x, i \rangle, (\{x\} \times [k_{max}]^0))$, ReceiveFromSet$(\omega, (x, i), (\{x\} \times [k_{max}]^0))$, or ReceiveFromSet$(c_\infty, \langle x, \infty \rangle, \{\langle x, \infty \rangle\})$ where $i \in [k_{max}]^0$. After that, we know that we

have no message representing x in the channels of $\mathcal{D}$ anymore, so we add an operation $c_0!\langle x, 0\rangle$ to send $\langle x, 0\rangle$ to c_0. The clock has now been reset.

Simulating Timed Transitions

We show how to simulate timed transitions of the form $\langle s, \mathtt{X}, \nu\rangle \xrightarrow{\delta}_{\mathcal{T}} \langle s, \mathtt{X}', \nu'\rangle$ for some $\delta > 0$. We distinguish between two cases, namely (i) there is at least one message or clock with value ($\leq k_{max}$) and a zero fractional part (i.e., $c_0 \neq \epsilon$), and (ii) that no such message or clock exists (i.e., $c_0 = \epsilon$):

- In the first case, we can let time pass by a sufficiently small real number, such that no clock with a positive fractional part before the transition reaches the next integer value after the transition. The contents of c_0 will be divided between messages that will be transferred to c_∞ (representing message ages and clocks values equal to k_{max}); and messages that will be placed in a new channel at the leftmost position in the dynamic part (representing message ages and clock values $< k_{max}$). Concretely, we perform the following steps: (i) we use $\mathsf{copy}(c_0, c_{tmp})$ to copy the contents of c_0 to the temporary channel c_{tmp}. (ii) we use $\mathsf{filter}(c_{tmp}, \Sigma_1)$ where $\Sigma_1 = (X \times \{k_{max}\}) \cup ((C \times M) \times \{k_{max}\})$ to only keep messages with ages equal to k_{max} in c_{tmp}. (iii) We send the messages of c_{tmp} one after one to c_∞, changing the second component from k_{max} to ∞ for each message. (iv) We use $\mathsf{filter}(c_0, \Sigma_2)$ where $\Sigma_2 = \left(X \times [k_{max} - 1]^0\right) \cup \left((C \times M) \times [k_{max} - 1]^0\right)$ to only keep messages with ages $< k_{max}$ in c_0. (v) We send the content of c_0 to the dynamic part using $\mathit{send\_channel}(c_0)$. (vi) We use $\mathsf{empty}(c_0)$ to empty c_0.
- In the second case, we let time pass by exactly the amount needed to make the clock values and the message ages in the rightmost dynamic channel equal to the next integer. Let $f \in ((\Sigma_m \cup \Sigma_x) \to (\Sigma_m \cup \Sigma_x))$ be a function such that $f(\langle\langle c, m\rangle, i\rangle) = \langle\langle c, m\rangle, i+1\rangle$ and $f(\langle x, i\rangle) = \langle x, i+1\rangle$ for any $c \in C$, $m \in M$, $x \in X$, and $i \in [k_{max} - 1]^0$. We use $\mathit{receive\_channel}(c_0)$ to move the contents of the rightmost dynamic channel to c_0. Then, we use $\mathsf{map}(c_0, f)$ to increase the integer parts of ages of clocks and messages by one.

Simulating Lossy Transitions

Since we have lossiness in $\mathcal{D}$, the simulation is immediate.

6 Conclusions, Discussion, and Future Work

We have shown the decidability of the reachability problem for TLCS, a model that extends both lossy channel systems and timed automata. To this end, we have introduced a new model, namely DLCS that operates on second-order lossy channels. We believe that DLCS are interesting in their own right. In fact, we can define higher-order LCS that contain "nested channels of channels" of arbitrary depth, in a similar manner to higher-order push-down automata [20]. It is straightforward to extend the method we present in this paper to show that transition systems induced by higher-order LCS are also well quasi-ordered and hence their reachability problem is decidable. To simplify the presentation (and since it suffices for our purposes) we have chosen to present the proof only for the case where the hierarchy is restricted to two levels (i.e., DLCS).

The proof techniques we provide in this paper are entirely different from the ones earlier presented for other timed models. For instance, decidability of the reachability (coverability) problem for timed Petri nets [5] is achieved by directly proving that the induced transition system is well quasi-ordered. In particular, in contrast to our method, the proof does not rely on a translation to an untimed model. On the other hand, the proof for timed pushdown systems [3] reduces the problem to the underlying untimed model, i.e., (untimed) pushdown automata. Although, we here provide a reduction to an untimed model, the target model is more powerful than the original one (DLCS vs. plain LCS). Indeed, we believe that a translation from TLCS to plain LCS that preserves reachability properties is not possible.

As future work, we will consider probabilistic and game extensions of the current model.

References

1. P. A. Abdulla. Well (and better) quasi-ordered transition systems. *The Bulletin of Symbolic Logic*, 16(4):457–515, 2010.
2. P. A. Abdulla, M. F. Atig, and J. Cederberg. Timed lossy channel systems. Technical Report 2012-031, Department of Information Technology, Uppsala University, October 2012.
3. P. A. Abdulla, M. F. Atig, and J. Stenman. Dense-timed pushdown automata. In *LICS*. IEEE Computer Society, 2012.
4. P. A. Abdulla, M. F. Atig, and J. Stenman. The minimal cost reachability problem in priced timed pushdown systems. In *LATA*, volume 7183 of *LNCS*, 2012.
5. P. A. Abdulla and A. Nylén. Timed Petri nets and BQOs. In *ICATPN*, 2001.
6. Parosh Aziz Abdulla, Mohamed Faouzi Atig, Yu-Fang Chen, Carl Leonardsson, and Ahmed Rezine. Counter-example guided fence insertion under tso. In *TACAS*, 2012.
7. Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *LICS*, pages 313–321, 1996.
8. Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. In *LICS*, pages 160–170. IEEE Computer Society, 1993.
9. R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
10. M. F. Atig, A. Bouajjani, S. Burckhardt, and M. Musuvathi. On the verification problem for weak memory models. In *POPL*, pages 7–18. ACM, 2010.
11. M. Benerecetti, S. Minopoli, and A. Peron. Analysis of timed büchi state machines. In *TIME*, pages 61–68. IEEE Computer Society, 2010.
12. B. Bérard, F. Cassez, S. Haddad, O. Roux, and D. Lime. Comparison of different semantics for time Petri nets. In *ATVA 2005*, 2005.
13. A. Bouajjani, R. Echahed, and R. Robbana. On the automatic verification of systems with continuous variables and unbounded discrete data structures. In *Hybrid Systems*, LNCS 999, pages 64–85. Springer, 1994.
14. P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In *FSTTCS*, LNCS 3328, pages 148–160. Springer, 2004.
15. P. Bouyer and F. Laroussinie. Model checking timed automata. In Stephan Merz and Nicolas Navet, editors, *Modeling and Verification of Real-Time Systems*, pages 111–140. ISTE Ltd. – John Wiley & Sons, Ltd., January 2008.
16. Pierre Chambart and Philippe Schnoebelen. The ordinal recursive complexity of lossy channel systems. In *LICS*, pages 205–216. IEEE Computer Society Press, 2008.
17. Z. Dang. Pushdown timed automata: a binary reachability characterization and safety verification. *Theor. Comput. Sci.*, 302(1-3):93–121, 2003.
18. Z. Dang, T. Bultan, O. H. Ibarra, and R. A. Kemmerer. Past pushdown timed automata and safety verification. *Theor. Comput. Sci.*, 313(1):57–71, 2004.
19. M. Emmi and R. Majumdar. Decision problems for the verification of real-time software. In *HSCC*, LNCS 3927, pages 200–211. Springer, 2006.
20. M. Hague and L. Ong. Symbolic backwards-reachability analysis for higher-order pushdown systems. *Logical Methods in Computer Science*, 4(4), 2008.

21 G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc. (3)*, 2(7):326–336, 1952.
22 A. Trivedi and D. Wojtczak. Recursive timed automata. In *ATVA*, pages 306–324, 2010.