Verification of Buffered Dynamic Register Automata *

Parosh Aziz Abdulla[1], Mohamed Faouzi Atig[1] Ahmet Kara[2], and Othmane Rezine[1]

[1] Uppsala University, Sweden
{parosh, mohamed_faouzi.atig, othmane.rezine}@it.uu.se,
[2] TU Dortmund University, Germany
ahmet.kara@cs.tu-dortmund.de

Abstract. We consider the verification problem for Communicating Register Automata (BDRA) which extend classical register automata by process creation. In this setting, each process is equipped with a mailbox (i.e., a channel) in which received messages can be stored. Moreover, each process has a finite number of registers in which IDs of other processes can be stored. A process can send messages to the mailbox of the processes whose IDs are stored in its registers and can send them the content of its registers. The state reachability problem asks whether a BDRA reaches a configuration where at least one process is in an error state. In this paper, we study the decidability of the reachability problem for different kind of channels and we provide a complete characterisation of the (un)decidable subclasses in this generalised setting.

Keywords: Formal Verification, Distributed Systems

1 Introduction

Register automata [14] were introduced as a reasonable extension of finite automata to deal with languages over infinite alphabets. The expressiveness and computational properties of different versions of this model are intensively studied (see e.g. [17, 18, 4, 19, 12]). A register automaton is a finite state automaton equipped with a finite number of registers in which symbols from an infinite domain can be stored for later comparison. There are many papers investigating the strong relationship between logics on structures over infinite alphabets and register automata [8, 10, 15, 13].

In [6, 5] register automata with process creation are proposed to describe the behavior of parallel processes. In this approach registers are used to store the IDs of other processes in the network. Every process can spawn new processes and communicate asynchronously with processes whose IDs are stored in its registers through unbounded channels. This extended register automata model is used as an (implementation) model for ad-hoc networks [6] and dynamic message sequence charts [5].

In [3], we studied the *state reachability problem* for *Dynamic Register Automata* (DRA) which is basically the automata model in [6, 5] adapted to rendezvous-based

* Supported by the Uppsala Programming for Multicore Architectures Research Center (UPMARC) and the Programming Platform for Future Wireless Sensor Networks Project (PROFUN). The third author acknowledges the financial support by the German DFG under grant SCHW 678/4-2.

communication and equipped with a reset transition for deleting register contents. Given a DRA and an (error) state q_{err} the sate reachability problem asks whether the network induced by the DRA reaches a configuration where at least one process is in state q_{err}. The reachability problem for DRA is in general not decidable. Searching for decidable sub-classes and inspired by recent investigations on ad-hoc networks [7, 1], we set several restrictions on the configuration graphs induced by DRA and considered *degenerative* DRA, i.e. DRA which are able to reset registers nondeterministically.

In this paper we consider *Buffered Dynamic Register Automata* (BDRA) which, compared to the model we studied in [3], is closer to the original model in [6, 5] in terms of communication. Besides finitely many registers, a BDRA is equipped with an (un)bounded FIFO buffer. A process described by a BDRA can create new processes and send messages to the buffers of other processes whose IDs are stored in its registers. An exchanged message can contain a symbol from a finite alphabet along with a process ID (from one of the process registers for instance). Moreover, the process can read messages from its own buffer and store incoming IDs in its own registers. Thus, the number of processes involved in the network induced by a BDRA and the communication topology of the network are not fixed a priori but change dynamically during the run of the system. Note also, that message sending and message receiving occur asynchronously. Finally, processes may execute a disconnect action, which will detach them from the whole network. As a result of this action, the content of the process registers and the process buffer are deleted.

We investigate the decidability borders of the state reachability problem for both BDRA and *lossy* BDRA, a sub-class of BDRA in which any process in the network can non deterministically disconnect itself. We show first that, in terms of reachable states, every BDRA is equivalent to its lossy counterpart.

Note that in order to simulate rendezvous communication through buffered systems, acknowledgement messages from receiver to sender are needed. This requires the existence of communication cycles in the graph of the network, which in turn makes the state reachability problem undecidable. In fact, we show that the reachability problem for (lossy) BDRA is undecidable even in the case where only configurations of which the graph of the network contains at most one edge are allowed.

We consider therefor a new restriction on (lossy) BDRA that would diminish the power of the model coming from the buffer: bounding the process buffers. We show that the problem remains undecidable for this case, even if the buffer is set to contain at most one message. The undecidability result still holds when only acyclic configurations are allowed and even if we bound the simple paths of the communication graph.

Finally, we concentrate on *strongly bounded* BDRA with bounded buffers. A BDRA is called strongly bounded if the only configurations allowed are those in which the simple paths of the underlying undirected graph of the network is bounded by some constant. While the reachability problem for strongly bounded BDRA with bounded buffer is still undecidable, we get decidability when we consider lossy BDRA. The proof comes from a non-trivial instantiation of the well-structured transition system framework. It is worth mentioning here that, due to the channel semantics we considered in our model (non-lossy FIFO), messages with IDs can not be dropped. Therefor, there is no trivial reduction from strongly bounded lossy BDRA with bounded buffers to strongly bounded degenerative DRA considered in [3]. Furthermore, the definitions of the graph encoding of configurations and the well-quasi ordering needed in order to instantiate the framework of well-structured transition systems to show decidability for strongly bounded lossy BDRA with bounded buffer are different from the ones used to prove the decidability of strongly bounded degenerative DRA in [3] and more involved.

Related work. For related work concerning register automata and wireless Ad-Hoc networks, we refer the reader to the related work section in [3]. In the following, we mainly compare our work with [3].

The main difference between the two works is the communication modality. In [3], the communication is done via rendezvous, while in our work, we consider asynchronous communication through the use of buffers. Both models are Turing powerful. Also, as it is not obvious how reset transitions can be simulated by disconnect transitions, there is no simple reduction of the reachability problem from one model to the other. Moreover, our model allows a more fine-grained analysis since we can reason about the acyclicity of the communication graph while the rendezvous communication requires the synchronisation of sender and receiver and consequently the creation of an implicit cycle in the communication graph.

We show that our general undecidability result and the undecidability result for BDRA with bounded buffer hold even in the case where the communication graph is acyclic and all its simple paths are bounded. Our undecidability proofs are more involved and complicated than in the case of DRA [3] due to the acyclicity restriction of the communication graph. We show also the decidability of the reachability problem for strongly bounded BDRA when the underlying undirected graphs of the network are acyclic. This case was not considered in [3]. Finally, the graph encoding used for the configurations and the well-quasi ordering for strongly bounded lossy BDRA are different and more involved than the ones used in the case of strongly bounded degenerative DRA in [3].

2 Preliminaries

Let A and B be two sets. We use $|A|$ to denote the cardinality of A ($|A| = \omega$ if A is infinite). Let $\mathbb{N}$ be the set of natural numbers. For a partial function $g : A \rightharpoonup B$ and $a \in A$, we write $g(a) = \perp$ if g is undefined on a. We use $\perp_A$ to denote the partial function which is undefined on all elements of A, i.e. $\perp_A(a) = \perp$ for all $a \in A$. Given a (partial) function $f : A \rightharpoonup B$, $a \in A$ and $b \in B$, we denote by $f[a \leftarrow b]$ the function f' defined by $f'(a) = b$ and $f'(a') = f(a')$ for all $a' \in A$ with $a \neq a'$.

Let Σ be an alphabet. We denote by Σ^* (resp. Σ^+) the set of all finite words (resp. finite non-empty words) over Σ, and by ε the empty word. Let w be a word over Σ. The length of w is denoted by $|w|$; we assume that $|\varepsilon| = 0$. For every $j : 1 \leq j \leq |w|$, we use $w(j)$ to denote the j^{th} letter of w. For every letter $a \in \Sigma$, we use $a \in w$ to denote that there is an index j such that $1 \leq j \leq |w|$ and $w(j) = a$. Let $A = \langle Q_A, q_A^0, \delta_A, F_A \rangle$ be a finite state automaton over the alphabet Σ, with Q_A being the set of *control states*, q_A^0 the *initial state*, $\delta_A \subseteq Q_A \times \Sigma \times Q_A$ the *transition relation*, and F_A the set of *accepting states*. We use $L(A)$ to denote the regular language accepted by A.

A *transducer* T over the alphabet Σ is a tuple $\left\langle Q_T, q_T^0, \delta_T, F_T \right\rangle$ where Q_T is the set of control states, q_T^0 is the initial state, $\delta_T \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\}) \times Q$ is the transition relation and F_T is the set of accepting states. A transducer T induces a binary relation $\mathtt{Rel}_T$ over Σ^* where two words $w_1, w_2 \in \Sigma^*$ are in relation $(w_1 \, \mathtt{Rel}_T \, w_2)$ if T outputs w_2 when accepting w_1. If $(w_1 \, \mathtt{Rel}_T \, w_2)$ we say that w_2 is a *transduction* of w_1 by T. Given a word $w \in \Sigma^*$, we use $T(w) := \{w' \in \Sigma^* | \, w \, \mathtt{Rel}_T \, w'\}$ to denote the set of all possible transductions of the word w by T. We define the transduction of a language $L \subseteq \Sigma^*$ as $T(L) := \{w' \in \Sigma^* | \, \exists w \in L, w \, \mathtt{Rel}_T \, w'\}$. By induction, we define the i^{th} transduction of L as follows: $T^0(L) := L$ and $T^{i+1}(L) := T\left(T^i(L)\right)$.

Given two finite state automaton A and B and a transducer T, all over the same alphabet Σ, the *transduction problem* TRANSD consists in checking whether there exits $i \in \mathbb{N}$ such that $T^i(L(A)) \cap L(B) \neq \varnothing$.

We define a *directed labeled graph* (or simply *graph*) G as a tuple $\langle V, L_v, L_e, l, E \rangle$ composed of a finite set of vertices V, a set of vertex labels L_v, a set of edge labels L_e, the vertex labeling function $l : V \to L_v$ and the set of labeled edges $E \subseteq V \times L_e \times V$. A *path* in G is a finite sequence of vertices $\pi = v_1 v_2 \ldots v_k$, $k \geq 1$, where, for every $i : 1 \leq i < k$, there is an $a \in L_e$ such that $\langle v_i, a, v_{i+1} \rangle \in E$. The path is a cycle if $v_1 = v_k$ and $k \geq 2$. The path π is *simple* if all vertices in π are distinct, i.e. $v_i \neq v_j$ for all $i, j : 1 \leq i < j \leq k$. We define $\mathtt{length}(\pi) := k - 1$. The largest k such that there is a simple path π in G with $\mathtt{length}(\pi) = k$ is called the *diameter* of G, and is denoted by $\varnothing(G)$.

We define a *transition system* $\mathcal{T}$ as a triple $\langle C, C_{init}, \longrightarrow \rangle$, where C is a set of *configurations*, $C_{init} \subseteq C$ is a set of *initial* configurations, and $\longrightarrow \subseteq C \times C$ is a *transition relation*. We write $c_1 \longrightarrow c_2$ if $\langle c_1, c_2 \rangle \in \longrightarrow$ and $\longrightarrow^*$ to denote the reflexive transitive closure of $\longrightarrow$. A configuration $c \in C$ is *reachable* in $\mathcal{T}$ if there is some $c_{init} \in C_{init}$ such that $c_{init} \longrightarrow^* c$.

3 Buffered Dynamic Register Automata

A network induced by a *Buffered Dynamic Register Automaton* (BDRA or buffered DRA) consists of a set of processes. Each process has a unique ID and is modelled as a finite-state system equipped with a mailbox and a finite number of registers. The mailbox is the recipient of all messages addressed to that process. In this paper, we assume that the mailbox is described by a (bounded) perfect FIFO buffer. The finitely many registers of a process are used to store the IDs of other processes in the network. A process is allowed to send a message (together with a possible content of one of its registers) to the mailbox of another process only if the ID of the receiver is stored in one of its registers. A process can receive a message from its mailbox and store a received ID in one of its registers. Finally, a process can also create (or spawn) a new process, allowing the number of processes in the network to increase over time.

In the following, we describe the syntax and semantics of BDRA. We introduce its subclass of *Lossy* BDRA where any process can be disconnected from the network in a non-deterministic way. Finally, we define the state reachability problem

Syntax. A BDRA D is a tuple $\langle Q, q_0, M, X, \delta \rangle$ where Q is a finite set of control states, $q_0 \in Q$ is the initial state, M is a finite set of messages, $X = \{x_1, \ldots, x_n\}$ is a finite set of registers, and δ is a set of transitions, each of the form $\langle q_1, \texttt{action}, q_2\rangle$ where $q_1, q_2 \in Q$ are control states and `action` is of one of the following forms:

(i) τ which corresponds to a local action,
(ii) $x \leftarrowtail \texttt{create}(q)$ where $x \in X$ and $q \in Q$, which creates a new process with a fresh ID in state q, and stores this fresh ID in the register x of the creating process,
(iii) $m(v)\,!\,y$ where $m \in M$, $v \in X \cup \{\perp, \texttt{self}\}$, and $y \in X$. This transition sends the message m together with the value pointed by v to the mailbox of the process whose ID is stored in register y. Observe that the value pointed by v is either the content of register v if $v \in X$ and v is assigned to a process ID, the ID of the sending process if $v = \texttt{self}$ or the null value otherwise. This transition can not be performed if the register y is undefined (i.e. containing the value $\perp$).
(iv) $m?x$ where $m \in M$ and $x \in X$, receives a message of the form $m(id)$ and stores id in its register x if id is a process ID or deletes the content of x otherwise.
(v) `disconnect` which disconnects the process from the network by disabling any possible future communication with any other process. This is done by (1) reseting (to $\perp$) all registers belonging to the disconnecting process or containing its ID, (2) emptying the buffer of the process, and (3) resetting the ID field in all the messages containing the ID of the process to undefined.

A BDRA D is said to be *lossy* if for every state $q \in Q$ the transition $\langle q, \texttt{disconnect}\,\langle x\rangle\,, q\rangle$ is contained in δ (i.e. any process can be disconnected from the network at any time). Given a BDRA $D = \langle Q, q_0, M, X, \delta\rangle$, we define its lossy counterpart $\texttt{Lossy}\,(D)$ as the tuple $\langle Q, q_0, M, X, \delta'\rangle$ with $\delta' = \delta \cup \{\langle q, \texttt{disconnect}, q\rangle \mid q \in Q\}$.

Configuration. We use $\mathcal{P}$ to denote the domain of all possible process IDs. In the following, we sometimes refer to a process by its ID. We define a configuration c of a BDRA $D = \langle Q, q_0, M, X, \delta\rangle$ as a tuple $\langle \texttt{procs}, \texttt{s}, \texttt{r}, \texttt{ch}\rangle$, where $\texttt{procs} \subseteq \mathcal{P}$ is a finite set of processes, $\texttt{s} : \mathcal{P} \rightharpoonup Q$ is a partial function that associates each process $p \in \texttt{procs}$ with its current state, $\texttt{r} : \mathcal{P} \rightharpoonup \{X \rightharpoonup \texttt{procs}\}$ is a partial function that maps every process $p \in \texttt{procs}$ to its register contents and $\texttt{ch} : \mathcal{P} \rightharpoonup (M \times (\mathcal{P} \cup \{\perp\}))^*$ maps each process $p \in \texttt{procs}$ to the content of its channel. We use $\texttt{msg}\,(m\,(id)) = m$ (respectively $\texttt{Id}\,(m\,(id)) = id$) to denote the message part (respectively the ID part) of a message tuple $m\,(id)$. For two processes $p_1, p_2 \in \texttt{procs}$ and $x \in X$, $\texttt{r}\,(p_1)\,(x) = p_2$ means that register x of p_1 contains the ID of p_2. If $\texttt{r}\,(p_1)\,(x)$ is not defined then register x of p_1 is empty. We use $q \in c$ to denote that there is a process $p \in \texttt{procs}$ such that $\texttt{s}\,(p) = q$. The set of all possible configurations of D is denoted by $C(D)$. A configuration $c = \langle \texttt{procs}, \texttt{s}, \texttt{r}, \texttt{ch}\rangle \in C(D)$ is said to be *initial* if it contains exactly one process (i.e., $\texttt{procs} = \{p\}$ for some $p \in \mathcal{P}$) in the initial state ($\texttt{s}\,(p) = q_0$), whose registers are empty ($\texttt{r}\,(p)\,(x) = \perp, \forall x \in X$) and whose mailbox is empty ($\texttt{ch}\,(p) = \varepsilon$). The set of initial configurations is denoted by $C_{init}(D)$.

Graph Representation. Let $c = \langle \texttt{procs}, \texttt{s}, \texttt{r}, \texttt{ch}\rangle$ be a configuration. We propose a graph encoding for c in order to show the communication possibilities between processes. In this encoding, every process is represented by a vertex labeled with its state. Moreover, if register $x \in X$ of a process p_1 contains the ID of another process p_2, then there is an edge from the vertex representing p_1 to the vertex representing p_2 and the edge is labeled with x. Furthermore, we add edges that represent the potential connectivity between processes that comes from the message-ID tuples contained in the process mailboxes, and we label them with the symbol $-$. Formally, the *encoding* of the configuration c is defined as the graph $\mathtt{enc}(c) := \langle \mathtt{procs}, Q, X \cup \{-\}, \mathtt{s}, E \rangle$ with $E = \{\langle p, x, p' \rangle \mid \mathtt{r}(p)(x) = p' \neq \bot\} \cup \{\langle p, -, p' \rangle \mid m(p') \in \mathtt{ch}(p) \text{ and } p' \neq \bot\}$.

Operational Semantics. We define a transition relation $\longrightarrow_D$ on the set of configurations $C(D)$ of D. Let $c = \langle \mathtt{procs}, \mathtt{s}, \mathtt{r}, \mathtt{ch} \rangle, c' = \langle \mathtt{procs}', \mathtt{s}', \mathtt{r}', \mathtt{ch}' \rangle \in C(D)$ be two configurations. We have $c \longrightarrow_D c'$ if one of the following conditions holds:

`Local:` There is a transition $\langle q_1, \tau, q_2 \rangle \in \delta$ and a process $p \in \mathtt{procs}$ such that i) $\mathtt{s}(p) = q_1$, ii) $\mathtt{s}' = \mathtt{s}[p \leftarrow q_2]$, and iii) $\mathtt{procs}' = \mathtt{procs}$, $\mathtt{r}' = \mathtt{r}$ and $\mathtt{ch}' = \mathtt{ch}$, i.e. processes, registers and mailboxes are left unchanged. A local transition changes the state of at most one process.

`Process Creation:` There is a transition $\langle q_1, x \leftarrowtail \mathtt{create}(q), q_2 \rangle \in \delta$ and a process $p \in \mathtt{procs}$ such that: i) $\mathtt{s}(p) = q_1$, ii) $\mathtt{procs}' = \mathtt{procs} \cup \{p'\}$ for some process $p' \notin \mathtt{procs}$, i.e. a new process p' is created, iii) $\mathtt{s}' = \mathtt{s}[p \leftarrow q_2][p' \leftarrow q]$, i.e. process p' is spawned in state q, while the new state of process p is q_2, iv) $\mathtt{r}' = \mathtt{r}[p \leftarrow \mathtt{r}(p)[x \leftarrow p']]$, i.e. register x of process p is assigned the ID of the new process p', and v) $\mathtt{ch}'[p' \leftarrow \varepsilon]$, i.e. the new process p' gets an empty buffer.

`Message Sending:` There are two distinct processes $p, p' \in \mathtt{procs}$ and a transition $\langle q_1, m(v)\,!\,y, q_2 \rangle \in \delta$ such that: (i) $\mathtt{s}(p) = q_1$ and $\mathtt{s}' = \mathtt{s}[p \leftarrow q_2]$, (ii) $\mathtt{r}(p)(y) = p'$, i.e. register y of p contains the ID of p', (iii) $\mathtt{procs}' = \mathtt{procs}$ and $\mathtt{r}' = \mathtt{r}$, and (iv) $\mathtt{ch}' = \mathtt{ch}[p' \leftarrow \mathtt{ch}(p') \cdot m(id)]$, where $id = \mathtt{r}(p)(v)$ if $v \in X$, $id = p$ if $v = \mathtt{self}$ and $id = \bot$ otherwise. Observe that this transition can not be performed when y is undefined since there is no process $p' \in \mathtt{procs}$ such that $\mathtt{r}(p)(y) = p'$.

`Message Receiving:` There is a process $p \in \mathtt{procs}$ and a transition $\langle q_1, m?x, q_2 \rangle \in \delta$ such that: (i) $\mathtt{s}(p) = q_1$ and $\mathtt{s}' = \mathtt{s}[p \leftarrow q_2]$, (ii) $\mathtt{ch} = \mathtt{ch}'[p \leftarrow m(id) \cdot \mathtt{ch}'(p)]$, i.e. channel $\mathtt{ch}(p)$ of process p in configuration c contains a message of the form $m(id)$ that will be read, (iii) $\mathtt{r}' = \mathtt{r}[p \leftarrow \mathtt{r}(p)[x \leftarrow id]]$, and (iv) $\mathtt{procs}' = \mathtt{procs}$. Note that id can be empty ($id = \bot$) or contain the ID of another process $p'' \in \mathtt{procs}$.

`Process Disconnection:` There is a transition $\langle q_1, \mathtt{disconnect}, q_2 \rangle \in \delta$ and a process $p \in \mathtt{procs}$ such that: (i) $\mathtt{s}(p) = q_1$ and $\mathtt{s}' = \mathtt{s}[p \leftarrow q_2]$, (ii) $\mathtt{procs}' = \mathtt{procs}$, (iii) $\mathtt{r}'(p) = \bot_X$ i.e. all registers of process p are reset, (iv) for every other process $p' \in \mathtt{procs}$ we have, for every register $x \in X$, either $\mathtt{r}(p')(x) \neq p$ and the value of the register is preserved ($\mathtt{r}'(p')(x) = \mathtt{r}(p')(x)$), or $\mathtt{r}'(p')(x) = p$ and the register is reset ($\mathtt{r}'(p')(x) = \bot$), (v) $\mathtt{ch}'(p) = \varepsilon$, i.e. the channel of process p is emptied, and (vi) for every other process $p' \in \mathtt{procs}$, every message of the form $m(p)$ in $\mathtt{ch}(p')$ is replaced by $m(\bot)$.

For $c, c' \in C(D)$, we use $c \xrightarrow{\mathtt{action}} c'$ to denote that c' can be obtained from c by the execution of a transition $\langle q_1, \mathtt{action}, q_2 \rangle \in \delta_D$.

State Reachability. We use $\mathcal{T}(D)$ to denote the transition system defined by the triple $\langle C(D), C_{init}(D), \longrightarrow_D \rangle$. We say that a state $\mathtt{target} \in Q$ is reachable in $\mathcal{T}(D)$ if there exists a reachable configuration $c = \langle \mathtt{procs}, \mathtt{s}, \mathtt{r}, \mathtt{ch} \rangle$ with $p \in \mathtt{procs}$ and

$\texttt{s}(p) = \texttt{target}$. The problem of checking whether the state $\texttt{target}$ is reachable or not is the *state reachability problem*. We use $\texttt{Reach}(D, \texttt{target})$ to denote the state reachability of $\texttt{target}$ in D.

Any lossy BDRA is an over-approximation of its non-lossy counterparts in terms of reachable states. Lemma 1 states that this approximation is exact.

Lemma 1. *Let D be a* BDRA. *Then, D and $\texttt{Lossy}(D)$ reach the same set of control states.*

The idea of the proof is that a buffered BDRA D can simulate any run of its lossy counterpart $\texttt{Lossy}(D)$ by ignoring any of its disconnecting processes. More precisely, the simulation is done by letting the network of D follow each step of the run of the $\texttt{Lossy}(D)$ besides the process disconnecting transitions that are not present in D.

4 BDRA State Reachability is Undecidable

We give in this section a proof to the following theorem:

Theorem 1. *Given a (lossy)* BDRA $D = \langle Q, q_0, M, X, \delta \rangle$ *and a control state $\texttt{target} \in Q$, the problem $\texttt{Reach}(D, \texttt{target})$ is undecidable.*

Observe that the reduction used in this proof generates configurations of which graph encodings contain at most one edge.

Proof sketch. The proof is carried out by a reduction from the TRANSD problem introduced in Section 2 which has been proven to be undecidable in [1]. Given two finite state automata A and B and a transducer T, we first define a BDRA D that we use in order to build a *transduction chain*. A transduction chain is a chain of processes $p_0, p_1, \ldots, p_m$, $m \geq 1$, where the first process p_0 simulates automaton A, the last process p_m simulates automaton B and all processes in between (i.e. $p_2, \ldots, p_{m-1}$) simulate transducer T. Note that the length of the chain should be as big as desired. We show in the rest of this section how to reduce TRANSD problem to $\texttt{Reach}(D, \texttt{target})$ for some control state $\texttt{target}$ that we will define. Note that the graph representations of the configurations generated by D contain no cycles and that their simple paths are bounded by 1 (Figure 1 shows the configurations used during the simulation).

Reduction. Let $A = \langle Q_A, q_A^0, \delta_A, F_A \rangle$ and $B = \langle Q_B, q_B^0, \delta_B, F_B \rangle$ be two finite state automaton and $T = \langle Q_T, q_T^0, \delta_T, F_T \rangle$ be a transducer, all over the same alphabet Σ. We construct the BDRA $D = \langle Q, q_0, M, X, \delta \rangle$ as follows. Process p_0 of the initial configuration (c_0 in Fig. 1) is in the initial state q_0. Process p_0 starts the simulation by creating a new process p_1 and moves to a state q_A^0. The new process p_1 is spawned in state q_{temp}^0 and its ID is saved into register x of p_0 (c_1 in Fig. 1). Simulation of automaton A by process p_0 can now start: p_0 sends all letters generated by the traversal of automaton A to the channel of the created process p_1 (c_2 in Fig. 1). If p_0 reaches an accepting state of A, it chooses non-deterministically to either send an accepting symbol to p_1 or to keep traversing A. If p_0 decides to send the accepting symbol, then it stops traversing A and disconnects itself from the rest of the network (c_3 and then c_4 in Fig. 1). In state q^0_{temp}, the spawned process p_1 makes the non-deterministic choice of either simulating automaton B or simulating transducer T. It does so by moving either to state q^0_B or to state q^0_T. If it moves to state q^0_B (c_5 in Fig. 1), it will simulate automaton B by reading from its channel the word sent by p_0 and simultaneously traversing automaton B. When reading the acceptance symbol, process p_1 checks if it reached an accepting state of B. If it is the case, the process moves to state `target`. If not, it moves to an error state q_{error}. If instead process p_1 made the initial choice of simulating transducer T (c_4 in Fig. 1), then it creates a new process p_2 and moves to state q^0_T. From state q^0_T, process p_1 will simulate transducer T by reading the input letters from its channel, traversing T and sending the output letters to the channel of the next process, here p_2. When reading the acceptance symbol from its channel, process p_1 checks if it reached an accepting state in T. If it is the case, it sends the acceptance symbol to the next process p_2, stops simulating T and disconnects itself from the rest of the network. If not, it moves to an error state q_{error}. The newly created process p_2 is spawned in state q^0_{temp} from which, again, the choice between simulating B or T will be non-deterministically made.

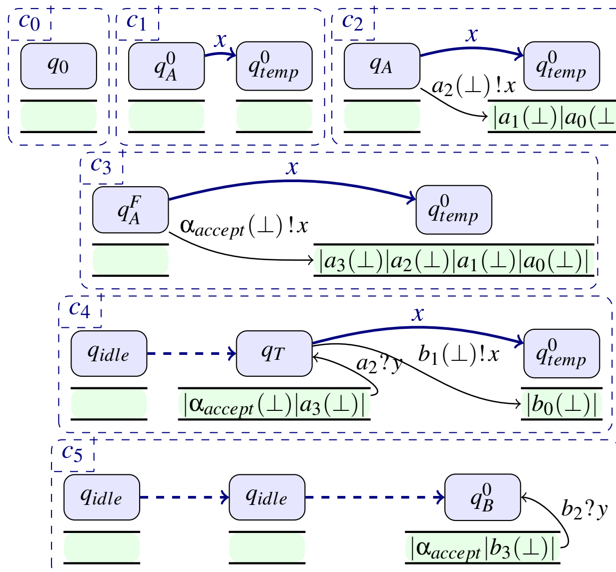

Fig. 1. TRANSD encoding into BDRA.

5 Bounded Buffer BDRA

We saw in the previous section that the undecidability result holds for configurations of which the graph encodings contain at most one single edge. This means that bounding the simple paths in the graph representation of the configurations or not allowing cycles will not bring decidability to the reachability problem. We consider therefor a rather different direction: bounding the channels, i.e. we only consider runs of the BDRA where the communication buffers are under a certain bound $l \in \mathbb{N}$. In the following, we first formally define the state reachability problem with this new restriction. Then we show that the problem is still undecidable even if buffers are bounded by 1.

Bounded Buffer State Reachability. Let $l \geq 1$, D be a BDRA and $\mathcal{T}(D) = \langle C(D), C_{init}(D), \longrightarrow_D \rangle$ be its corresponding transition system. We define the l-

bounded buffer transition system associated with D as the tuple $\mathcal{T}^{\mathtt{buf}\leq l}(D) = \left\langle C^{\mathtt{buf}\leq l}(D), C^{\mathtt{buf}\leq l}_{init}(D), \longrightarrow^{\mathtt{buf}\leq l}_{D} \right\rangle$ where i) $C^{\mathtt{buf}\leq l}(D) \subseteq C(D)$ is the set of all possible configurations $c = \langle \mathtt{procs}, \mathtt{s}, \mathtt{r}, \mathtt{ch} \rangle$ of which channels are bounded by l (i.e. $|\mathtt{ch}(p)| \leq l$ for every $p \in \mathtt{procs}$), ii) $C^{\mathtt{buf}\leq l}_{init}(D) = C_{init}(D)$ is the set of initial configurations, and finally iii) $\longrightarrow^{\mathtt{buf}\leq l}_{D} \subseteq \longrightarrow_{D} \cap \left(C^{\mathtt{buf}\leq l}(D) \times C^{\mathtt{buf}\leq l}(D)\right)$ is the transition relation. We say that a state $\mathtt{target} \in Q$ is reachable in $\mathcal{T}^{\mathtt{buf}\leq l}(D)$ if there is a reachable configuration $c = \langle \mathtt{procs}, \mathtt{s}, \mathtt{r}, \mathtt{ch} \rangle$ in $\mathcal{T}^{\mathtt{buf}\leq l}(D)$ and a process $p \in \mathtt{procs}$ with $\mathtt{s}(p) = \mathtt{target}$. Checking whether $\mathtt{target}$ is reachable or not in $\mathcal{T}^{\mathtt{buf}\leq l}(D)$ is the *l-bounded buffer state reachability* problem that we denote hereafter by $\mathtt{BufReach}(D, \mathtt{target}, l)$.

Theorem 2. *Given a* BDRA *D and a state $\mathtt{target} \in Q$, the 1-bounded buffer state reachability problem $\mathtt{BufReach}(D, \mathtt{target}, 1)$ is undecidable.*

This result holds even if we forbid cycles and if we impose a bound on the length of the simple paths in the graph encoding of the configurations.

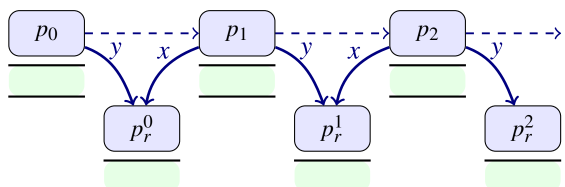

Fig. 2. Transduction chain $(p_0, p_1, p_2, \ldots)$.

Proof sketch. The proof is carried out by a reduction from the TRANSD problem. More precisely, given two finite state automata A and B and a transducer T, we define a 1-bounded buffer BDRA D that we use in order to build a transduction chain $p_0, p_1, \ldots, p_m$ of arbitrary length $m+1$. Since we dispose of channels of bounded size, we cannot transmit a word in one chunk and then transmit its transduction. Instead, we adopt a *continuous* communication flow, i.e. words are transmitted symbol by symbol and the transduction operates at the level of a symbol. One simple way to build the transduction chain consists in letting process p_0 of the initial configuration create a new process p_1, then letting p_1 create the next process p_2, and so on until some process p_m decides non deterministically to stop the chain construction. We obtain then a chain of length $m+1$. Although this approach fulfils our goal of building a transduction chain, configurations corresponding to these chains do not have a bound on the simple paths of their graph encoding. We consider therefor a more intricate chain building method that generates configurations for which the simple paths of their graph encoding is bounded (by two). The shape of the generated chain is shown in figure 2. Building such a transduction chain represents the first part of the proof. The second part of the proof consists in showing how the communication is carried through the chain. The idea here is to let processes p_r^i, $0 \leq i \leq m$, play the role of *relays* between each pair of consecutive processes (p_i, p_{i+1}) of the chain. They do so by by making use of the *boundedness of the buffer* and ensuring that messages they receive from p_i and p_{i+1} match.

6 Strongly Bounded BDRA with Bounded Buffer

Attempts to get decidability for the state reachability problem by bounding the size of the channels or by bounding the simple paths of the graph encoding of the configurations were vain. We consider therefor another direction, which consists, together with bounding the channels, in bounding the simple paths of the *underlying* undirected graph of the encoding of the configurations. By underlying undirected graph, we mean the undirected graph that we obtain after removing the direction and the labels from the edges. Formally, we define an undirected graph as a tuple $\langle V, L_v, l, E\rangle$, where V is a finite set of vertices, L_v is a set of vertex labels, $l : V \mapsto L_v$ is a vertex labeling function and $E \subseteq \{\{v,u\} \mid v,u \in V\}$ is a set of edges. Let $G = \langle V, L_v, L_e, l, E\rangle$ be a labeled directed graph. We use $\texttt{closure}(G) := \langle V, L_v, l, F\rangle$ to denote its underlying undirected graph with $F := \{\{u,v\} \mid \langle u,e,v\rangle \in E\}$. We extend in a straightforward manner the definition of diameter to undirected graphs.

In the following, we first define the transition system where only configurations that are bounded in their buffer size and in their undirected graph are allowed. Then, we give the undecidability result for this subclass.

Strongly Bounded State Reachability with Bounded Buffer. Let $k \geq 1$ and $l \geq 1$, $D = \left\langle Q_D, q_D^0, \delta_D, F_D\right\rangle$ be a BDRA and $\mathcal{T}(D) = \langle C(D), C_{init}(D), \longrightarrow_D\rangle$ its corresponding transition system. We define the *l-bounded buffer*, *k-strongly bounded* (l,k-strong) transition system associated to D as the tuple $\mathcal{T}^{(l,k)}(D) = \left\langle C^{(l,k)}(D), C_{init}^{(l,k)}(D), \longrightarrow_D^{(l,k)}\right\rangle$ composed of the set of (l,k)-strong configurations $C^{(l,k)}(D) := C^{\texttt{buf}\leq l}(D) \cap \{c \in C(D) \mid \varnothing(\texttt{closure}(\texttt{enc}(c))) \leq k\}$, the set of initial configurations $C_{init}^{(l,k)}(D) := C_{init}(D)$ and the transition relation $\longrightarrow_D^{(l,k)} := \longrightarrow_D \cap \left(C^{(l,k)}(D) \times C^{(l,k)}(D)\right)$. Given a control state $\texttt{target} \in Q_D$, checking whether there is a reachable configuration $c = \langle \texttt{procs}, \texttt{s}, \texttt{r}, \texttt{ch}\rangle$ in the transition system $\mathcal{T}^{(l,k)}(D)$ such that there is a process $p \in \texttt{procs}$ with $\texttt{s}(p) = \texttt{target}$ is the (l,k)-strong state reachability problem and is denoted by $\texttt{StrongReach}(D, \texttt{target}, k, l)$.

Theorem 3. *Given $l \geq 1$, $k \geq 4$, a* BDRA $D = \left\langle Q_D, q_D^0, \delta_D, F_D\right\rangle$ *and a control state* $\texttt{target} \in Q_D$*,* $\texttt{StrongReach}(D, \texttt{target}, l, k)$ *is undecidable.*

Proof Idea. The proof proceeds by a reduction from *Minsky's two counter machines* to the $(1,4)$-strong state reachability problem. A counter is simulated by a process to which a set of processes are attached. The value of the counter is defined by the number of such processes. In order to test if a counter is equal to zero, we make use of the fact that configurations are strongly bounded, i.e. transition to a configuration for which a simple path is over the bound is forbidden.

7 Lossy Strongly Bounded BDRA with Bounded Buffer

In this section, we show that the bounded buffer, strongly bounded state reachability problem becomes decidable if we consider lossy BDRA. To that purpose, we start by providing a more precise graph encoding of configurations where mailboxes are bounded by some $l \in \mathbb{N}$. Then, we state our result and dedicate the rest of this section to prove it.

Graph Representations for Bounded Buffered Configurations. Let $l \geq 1$ be natural number and $c = \langle \texttt{procs}, \texttt{s}, \texttt{r}, \texttt{ch} \rangle$ be a configuration with l-bounded buffers, i.e. $|\texttt{ch}(p)| \leq l$ for every process $p \in \texttt{procs}$. We propose a new graph encoding $\texttt{extenc}(c)$ of configuration c in the form of an extension of the previous encoding $\texttt{enc}(p)$. The new encoding takes into account the presence (and absence) of every message contained in the mailboxes. Besides representing processes as vertices and register contents as edges, we encode mailboxes as follows. Let $p \in \texttt{procs}$ be a process. Each message $m(id) \in \texttt{ch}(p)$ of index j, $1 \leq j \leq |\texttt{ch}(p)| \leq l$, is encoded with i) a vertex v_p^j labeled with (m, j), ii) an edge going from the vertex representing p to v_p^j and iii) an edge going from v_p^j to the vertex representing process p', if $id = p' \neq \perp$. Furthermore, we encode every empty message place holder of index j, $|\texttt{ch}(p)| < j \leq l$ with i) a vertex v_p^j labeled by (ε, j) and ii) an edge going from the vertex representing p to v_p^j. Formally, the extended encoding is defined by $\texttt{extenc}(c) := \langle \texttt{procs} \cup \left\{ v_p^j | \, p \in \texttt{procs}, 1 \leq j \leq l \right\}, Q \cup \{(m, j) | \, m \in (M \cup \{\varepsilon\}), 1 \leq j \leq l\}, X \cup \{-\}, L_c, E_c \rangle$ where the vertex labeling function is given by, $L_c(p) = \texttt{s}(p)$ for every process $p \in \texttt{procs}$, $L_c(v_p^j) = (\texttt{msg}(\texttt{ch}(p)(j)), j)$ for every $j : 1 \leq j \leq |\texttt{ch}(p)|$ and $L_c(v_p^j) = (\varepsilon, j)$ for every $j : |\texttt{ch}(p)| < j \leq l$, and the set of edges is given by $E_c = \{\langle p, x, p' \rangle | \, \texttt{r}(p)(x) = p'\} \cup \left\{ \left\langle p, -, v_p^j \right\rangle \right\} \cup \left\{ \left\langle v_p^j, -, p' \right\rangle | \, \exists m \in M, \texttt{ch}(p)(j) = m(p') \right\}$.

Observe that if the diameter of the closure of the graph encoding of some bounded buffer configuration is bounded by k, i.e. $\varnothing(\texttt{closure}(\texttt{enc}(c))) \leq k$), then the diameter of the closure of the extended graph encoding of the same configuration is bounded by at most $2 * k$, i.e. $\varnothing(\texttt{closure}(\texttt{extenc}(c))) \leq 2 * k$.

The rest of this section is devoted to the proof of the following theorem.

Theorem 4. *The strongly bounded state reachability problem for lossy* BDRA *with bounded buffers is decidable.*

We show the decidability of the strongly bounded state reachability problem for lossy BDRA with bounded buffers by a non-trivial instantiation of the framework of Well-Structured Transition Systems (WSTS) [2, 11]. We proceed to that end in several steps. First, we list the three main ingredients required in order to instantiate the WSTS framework on any transition system $\mathcal{T} = \langle C, C_{init}, \longrightarrow \rangle$. Then, we introduce the notion of *coverability* and show how to reduce the state reachability problem to it. Finally, we show the applicability of the main ingredients to our problem.

Ordering, Predecessors and Monotonicity. We present in this paragraph the three main components required for the instantiation of the WSTS framework.

Well-Quasi Ordering: First, we need to define a *Well-Quasi Ordering* (WQO) over the set of configurations C, i.e. a reflexive and transitive binary relation $\preceq$ over C such that, for every infinite sequence of configurations $(c_k)_{k \geq 0}$, there exist $i, j \in \mathbb{N}$ such that $i < j$ and $c_i \preceq c_j$. Let $U \subseteq C$ be a set of configurations. Using the notion of well-quasi ordering, we can define the following notions:

- The *upward closure* of U is the set $U\!\uparrow := \{c \in C | \, \exists c' \in U$ with $c' \preceq c\}$.
- U is *upward closed* if $U = U\!\uparrow$.

It has been shown that every upward closed set U can be characterised by a finite *minor set* $M \subseteq U$ such that i) for every $c' \in U$ there is $c \in M$ with $c \preceq c'$, and ii) if $c, c' \in M$ and $c \preceq c'$ then $c = c'$. We use $\mathtt{min}$ to denote the function which for a given upward closed set U returns one minor set of U.

Computing the $\mathtt{minpre}$: We use $\mathtt{Pre}(U) := \{c \mid \exists c_1 \in U, c \longrightarrow c_1\}$ to denote the set of predecessors of U. Given a configuration c, we denote by $\mathtt{minpre}(c)$ the set $\mathtt{min}(\mathtt{Pre}(\{c\}\uparrow) \cup \{c\}\uparrow)$. Providing an algorithm that computes a finite $\mathtt{minpre}(c)$ represents the second ingredient.

Monotonicity: The third ingredient consists in showing that the transition relation $\longrightarrow$ is monotonic with regard to the ordering $\preceq$, i.e. for every three configurations $c_1, c_2, c_3 \in C$ such that $c_1 \preceq c_2$ and $c_1 \longrightarrow c_3$ there should be a configuration $c_4 \in C$ such that $c_3 \preceq c_4$ and $c_2 \longrightarrow c_4$.

Coverability. Given a configuration $c_{\mathtt{target}} \in C$, the *coverability problem* asks whether there is a configuration $c' \succcurlyeq c_{\mathtt{target}}$ reachable in $\mathcal{T}$. Given that the three ingredients are provided, the following conditions are sufficient for the decidability of this problem: i) For every $c \in C$, we can check whether $\{c\}\uparrow \cap C_{init} \neq \emptyset$, and ii) for every two configurations c_1 and c_2, it is decidable whether $c_1 \preceq c_2$. The solution for the coverability problem of WSTS suggested in [2, 11] is based on a backward analysis approach. It is shown that starting from $U_0 := \{c_{\mathtt{target}}\}$, the sequence $(U_i)_{i \geq 0}$ with $U_{i+1} := \mathtt{min}(\mathtt{Pre}(U_i)\uparrow \cup U_i\uparrow)$, for $i \geq 0$ reaches a fix-point and is computable.

In the following, we instantiate the framework of WSTS to show the decidability of the state reachability problem for strongly bounded lossy BDRA with bounded buffer, but first we need to introduce some notations.

Let l and k be two natural numbers, $D = \langle Q, q_0, M, X, \delta \rangle$ a lossy BDRA, $\mathtt{target} \in Q$ a target state and $C = C^{(l,k)}(D)$. We introduce the *disconnect prefix* transition relation $\dashrightarrow := \xrightarrow{\mathtt{disconnect}}{}^{*}_{D} \circ \longrightarrow^{(l,k)}_{D}$. Note that the reflexive transitive closures of $\dashrightarrow$ and $\longrightarrow^{(l,k)}_{D}$ are identical. Thus, the state reachability of $\mathtt{target}$ in $\left\langle C, C_{init}, \longrightarrow^{(l,k)}_{D} \right\rangle$ is equivalent to its corresponding problem in $\langle C, C_{init}, \dashrightarrow \rangle$. Next, we will prove the decidability of the latter problem.

We will show that $\langle C, C_{init}, \dashrightarrow \rangle$ is a well-structured transition system. Let $C_{\mathtt{target}}$ denote the set of all configurations of the form $\langle \{p\}, \mathtt{s}, \mathtt{r}, \mathtt{ch} \rangle$, composed of a single process in state $\mathtt{target}$ ($\mathtt{s}(p) = \mathtt{target}$), whose registers are empty, and whose channel contains any (finitely many) possible word $w \in (M \times \{\perp\})^*$ of length $|w| \leq l$. We will define the well-quasi ordering on C in such a way that the upward closure of C_{target} consists of all configurations $c \in C$ with $\mathtt{target} \in c$. It becomes then clear that the coverability of any configuration $c \in C_{\mathtt{target}}$ in $\langle C, C_{init}, \dashrightarrow \rangle$ is equivalent to the reachability of $\mathtt{target}$ in the same transition system. We define in the next paragraph an ordering on the set of configurations C and show that it is a well-quasi ordering.

A well-quasi order on configurations. We define in this paragraph a well-quasi ordering on the set of configurations C. The ordering is defined by using the notion of *induced sub-graph* embedding $\sqsubseteq_{ind}$ on directed graphs defined as follows. Let $G_1 = \langle V_1, L_v, L_e, l_1, E_1 \rangle$ and $G_2 = \langle V_2, L_v, L_e, l_2, E_2 \rangle$ be two directed graphs. We say that G_1 is an induced sub-graph of G_2, and we write $G_1 \sqsubseteq_{ind} G_2$, if there is an injective mapping $t : V_1 \to V_2$ such that i) for all $v \in V_1$ we have $l_1(v) = l_2(t(v))$, and ii) for all $v, u \in V_1$ and $a \in L_e$ we have $\langle v, a, u\rangle \in E_1 \Leftrightarrow \langle t(v), a, t(u)\rangle \in E_2$. The induced sub graph relation on undirected graphs is defined in a similar manner.

Let $c_1 = \langle \mathtt{procs}_1, \mathtt{s}_1, \mathtt{r}_1, \mathtt{ch}_1\rangle$ and $c_2 = \langle \mathtt{procs}_2, \mathtt{s}_2, \mathtt{r}_2, \mathtt{ch}_2\rangle$ be two configurations from C. We define the ordering $\preceq$ on configurations by $c_1 \preceq c_2$ if $\mathtt{extenc}(c_1) \sqsubseteq_{ind} \mathtt{extenc}(c_2)$. We can show that, if $c_1 \preceq c_2$, then there should be an injective mapping $t : \mathtt{procs}_1 \mapsto \mathtt{procs}_2$ such that i) $\mathtt{s}_2(t(p)) = \mathtt{s}_1(p)$ for every $p \in \mathtt{procs}_1$, ii) $\mathtt{r}_2(t(p))(x) = t(p') \Leftrightarrow \mathtt{r}_1(p)(x) = p'$ for every $p, p' \in \mathtt{procs}_1$ and every $x \in X$, and iii) for every $p \in \mathtt{procs}_1$ with $\mathtt{ch}_1(p) = m_1(id_1)\ldots m_n(id_n)$ for some n we have $\mathtt{ch}_2(t(p)) = m_1(id'_1)\ldots m_n(id'_n)$ and for every $i : 1 \le i \le n$, if $id_i = p_i \in \mathtt{procs}_1$ then $id'_i = t(p_i)$, otherwise $id_i = id'_i = \perp$.

Based on a result by Ding in [9] and using the fact that the underlying undirected graph of the configuration encoding is bounded, we can show the following lemma:

Lemma 2. *The relation $\preceq$ is a well-quasi ordering on C.*

Monotonicity. Let $c_1, c_2, c_3 \in C$ be three configurations such that: $c_1 \preceq c_2$ and $c_1 \dashrightarrow c_3$. The goal here is to find a fourth configuration $c_4 \in C$ such that $c_3 \preceq c_4$ and $c_2 \dashrightarrow c_4$. This can be achieved by disconnecting as many processes as necessary in c_2 in order to obtain a configuration c_{sub} equal to c_1 modulo disconnected processes. From there, we let c_{sub} take the same transition as the one taken by c_1 to get to c_3 and we obtain a configuration c_4 such that $c_3 \preceq c_4$, $c_2 \xrightarrow{\mathtt{disconnect}}{}^{*}_{D} c_{sub}$ and $c_{sub} \dashrightarrow c_4$, thus $c_2 \dashrightarrow c_4$.

Lemma 3. *The transition relation $\dashrightarrow$ is monotonic w.r.t. $\preceq$.*

Summary of the* WSTS *Instantiation. The first sufficient condition for the decidability of the coverability problem, namely checking whether the upward closed set $\{c\}\uparrow$ of some configuration c contains an initial configuration, is trivial (we check that c contains one process only, that the process is in state q_{init} and that its registers and channels are empty). The second sufficient condition is also trivial (checking whether $c_1 \preceq c_2$ amount to checking graph embedding, which is decidable). The first ingredient needed in order to use the WSTS transition system has been provided with Lemma 2, which states that the induced sub-graph relation on the extended graph encoding of the configurations is a well-quasi ordering. The second ingredient, i.e. computability of the $\mathtt{minpre}$, is given by the following lemma.

Lemma 4. *Given a configuration $c \in C$, we can effectively compute* $\mathtt{minpre}(c)$.

The third ingredient, i.e. the monotonicity of the ordering wrt. the transition relation, is given by lemma 3.

Thus, Lemma 4, Lemma 2 and Lemma 3 show that the coverability of the finite set C_{target} is decidable. Hence, the state reachability problem for strongly bounded lossy BDRA with bounded buffers is decidable. $\square$

8 Acyclic Strongly Bounded BDRA with Bounded Buffer

In the following we show the decidability of the reachability problem for strongly bounded BDRA when the *underlying* undirected graph of configurations is acyclic.

Acyclic Strongly Bounded State Reachability with Bounded Buffer. Let $k,l \geq 1$, $D = \left\langle Q_D, q_D^0, \delta_D, F_D \right\rangle$ be a BDRA and $\mathcal{T}(D) = \langle C(D), C_{init}(D), \longrightarrow_D \rangle$ its corresponding transition system. We define the *l-bounded buffer*, *k-strongly bounded* acyclic ((l,k)-strong acyclic) transition system associated to D as the tuple $\mathcal{T}^{a(l,k)}(D) = \left\langle C^{a(l,k)}(D), C_{init}^{a(l,k)}(D), \longrightarrow_D^{a(l,k)} \right\rangle$ composed of the set of (l,k)-strong acyclic configurations $C^{a(l,k)}(D) := \left\{ c \in C^{(l,k)} \mid \texttt{closure}(\texttt{enc}(c)) \text{ is acyclic} \right\}$, the set of initial configurations $C_{init}^{a(l,k)}(D) := C_{init}(D)$ and the transition relation $\longrightarrow_D^{a(l,k)} := \longrightarrow_D \cap \left(C^{a(l,k)}(D) \times C^{a(l,k)}(D) \right)$. Let $\texttt{target} \in Q_D$ be a control state. Checking whether there is reachable configuration $c = \langle \texttt{procs}, \texttt{s}, \texttt{r}, \texttt{ch} \rangle$ in the transition system $\mathcal{T}^{a(l,k)}(D)$ such that there is a process $p \in \texttt{procs}$ with $\texttt{s}(p) = \texttt{target}$ is the (l,k)-strong acyclic state reachability problem and is denoted by $\texttt{AStrongReach}(D, \texttt{target}, l, k)$.

Theorem 5. *Given $l \geq 1$, $k \geq 1$, a buffered* DRA $D = \left\langle Q_D, q_D^0, \delta_D, F_D \right\rangle$ *and a control state* $\texttt{target} \in Q_D$, $\texttt{AStrongReach}(D, \texttt{target}, l, k)$ *is decidable.*

Proof sketch. The proof of Theorem 5 is based on the simple observation that the processes cannot exchange IDS, otherwise there will be a creation of a cycle in the *underlying* undirected graph configuration encodings. Hence, each process can only receive plain messages (without an ID) from its creator. Furthermore, at any time each process can send plain messages to a finite number of other processes (i.e. bounded by the number of registers). Since simple paths are also bounded, we have that the graph representation of any reachable configuration is a disjoint union of finite trees. Furthermore, any two processes in two different disjoint trees can never communicate with each other. This implies that the acyclic strongly bounded state reachability problem with bounded buffer can be reduced to the standard reachability problem for finite-state systems. Such a finite-state system keeps track of at most one tree in which each node corresponds to a process and its channel. When a tree is split into a finite number of subtrees due to the creation of new processes or disconnect operations, the finite-state system can decide in non-deterministic manner, to follow one of these sub-trees.

9 Conclusion

In this paper, we studied the state reachability problem for the class of buffered DRA. This work is a continuation of [3] where the analysis was carried for DRA with rendezvous communication. The problem is undecidable even if we bound the simple paths and even if we forbid cycles in the communication graph of the network. Our goal was to investigate sub-classes where state reachability becomes decidable. To that end, we considered different directions including bounding the size of the buffers, bounding the simple paths of the underlying (un)directed communication graph of the network and / or disallowing cycles in the network. It turned out that many of these restrictions, even combined, were not sufficient. However, we proved that the problem becomes decidable in two particular and interesting cases. In the first one, we considered the class of *lossy* buffered DRA, in which processes are allowed to disconnect themselves from the network in a non-deterministic fashion, where we bounded both the size of the buffers and the simple paths of the undirected communication graph. The proof was obtained through a non-trivial instantiation of well-structured transition systems. In the second case, we showed that the number of possible shapes of the network is finite if we bound the size of the buffers, disallow cycles and bound the simple paths in the communication graph. As future work, we think that it is worth checking whether decidability can be obtained for more general classes by considering other channel semantics, such as the unordered and the lossy ones. We also think that an important line of work would be to study the link between register automata and π-calculus and to study the relation between our results using the DRA formalism and the work of Meyer [16] in which π-calculus has been used.

References

1. Abdula, P.A., Atig, M.F., Rezine, O.: Verification of directed acyclic ad hoc networks. In: FMOODS/FORTE. pp. 193–208 (2013)
2. Abdulla, P., Cerans, K., Jonsson, B., Tsay, Y.: General decidability theorems for infinite-state systems. In: LICS'96. pp. 313–321. IEEE Computer Society (1996)
3. Abdulla, P.A., Atig, M.F., Kara, A., Rezine, O.: Verification of dynamic register automata. In: FSTTCS 2014. pp. 653–665 (2014)
4. Benedikt, M., Ley, C., Puppis, G.: Automata vs. logics on data words. In: CSL. LNCS, vol. 6247, pp. 110–124. Springer (2010)
5. Bollig, B., Cyriac, A., Hélouët, L., Kara, A., Schwentick, T.: Dynamic communicating automata and branching high-level MSCs. In: LATA. LNCS, vol. 7810. Springer (2013)
6. Bollig, B., Hélouët, L.: Realizability of dynamic MSC languages. In: CSR. LNCS, vol. 6072. Springer (2010)
7. Delzanno, G., Sangnier, A., Zavattaro, G.: Parameterized verification of ad hoc networks. In: CONCUR'10. LNCS, vol. 6269. Springer (2010)
8. Demri, S., Lazic, R.: LTL with the freeze quantifier and register automata. In: LICS. pp. 17–26. IEEE Computer Society (2006)
9. Ding, G.: Subgraphs and well quasi ordering. J. of Graph Theory 16(5), 489–502 (1992)
10. Figueira, D.: Alternating register automata on finite words and trees. Logical Methods in Computer Science 8(1) (2012)
11. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theor. Comput. Sci. 256(1-2), 63–92 (2001)
12. Grigore, R., Distefano, D., Petersen, R.L., Tzevelekos, N.: Runtime verification based on register automata. In: TACAS 2013. pp. 260–276 (2013)
13. Jurdzinski, M., Lazic, R.: Alternation-free modal mu-calculus for data trees. In: LICS. pp. 131–140. IEEE Computer Society (2007)
14. Kaminski, M., Francez, N.: Finite-memory automata. Theor. Comput. Sci. 134(2), 329–363 (1994)
15. Lazic, R.: Safely freezing LTL. In: FSTTCS. LNCS, vol. 4337, pp. 381–392. Springer (2006)
16. Meyer, R.: On boundedness in depth in the pi-calculus. In: IFIP TCS. pp. 477–489 (2008)
17. Neven, F., Schwentick, T., Vianu, V.: Finite state machines for strings over infinite alphabets. ACM Trans. Comput. Log. 5(3), 403–435 (2004)
18. Sakamoto, H., Ikeda, D.: Intractability of decision problems for finite-memory automata. Theor. Comput. Sci. 231(2), 297–308 (2000)
19. Tzevelekos, N.: Fresh-register automata. In: POPL. ACM (2011)