Verification of 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 Dynamic Register Automata (Dra). Dra extend classical register automata by process creation. In this setting, each process is equipped with a finite set of registers in which the process IDs of other processes can be stored. A process can communicate with processes whose IDs are stored in its registers and can send them the content of its registers. The state reachability problem asks whether a Dra reaches a configuration where at least one process is in an error state. We first show that this problem is in general undecidable. This result holds even when we restrict the analysis to configurations where the maximal length of the simple paths in their underlying (un)directed communication graphs are bounded by some constant. Then we introduce the model of *degenerative* Dra which allows non-deterministic reset of the registers. We prove that for every given Dra, its corresponding degenerative one has the same set of reachable states. While the state reachability of a degenerative Dra remains undecidable, we show that the problem becomes decidable with nonprimitive recursive complexity when we restrict the analysis to strongly bounded configurations, i. e. configurations whose underlying undirected graphs have bounded simple paths. Finally, we consider the class of strongly safe Dra, where all the reachable configurations are assumed to be strongly bounded. We show that for strongly safe Dra, the state reachability problem becomes decidable.

1998 ACM Subject Classification D.2.4 Software/Program Verification

Keywords and phrases Verification, Reachability problem, Register automata

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2014.653

1 Introduction

Register automata are a well-known computational model for languages over infinite alphabets (e.g. [20, 23, 24]). A register automaton is a finite state automaton equipped with a finite set of registers which can store data for later comparison. The expressive power and algorithmic properties of this model are well-studied (see e.g., [6, 23, 24, 28]). In addition, several works consider the relationship between different classes of register automata and logics for data words and trees (see e.g., [14, 15, 21, 19]).

Recently, register automata have been extended with dynamic creation of processes [8, 7]. In this setting, the behaviour of each process is described by a register automaton. Each process has a unique identifier (ID). The registers of each process are used to store the IDs of other process. The IDs stored in the registers of a process p correspond to the processes

* 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.

34th Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2014).
Editors: Venkatesh Raman and S. P. Suresh; pp. 653–665

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

known by p. Each process can perform two types of actions: (i) creating a new process and (ii) exchanging messages and IDs with other processes. The class of extended register automata can be used as: (1) a model of programs with process creation where the network topology and the number of involved processes are not known in advance but change dynamically [8], and (2) an implementation model for Dynamic Message Sequence Charts [8, 7].

In this paper, we consider the verification problem for Dynamic Register Automata (Dra), where the communication between processes is *synchronous* (i. e., *rendezvous* based)[1]. The synchronous communication involves two processes: *sender* and *receiver*. Besides creating new processes, each process can send a message from a finite alphabet or an ID from one of its registers (or its own ID). The receiver process can synchronize over the sent message or store the incoming ID in its own registers. Thus, the system may create an unbounded number of processes, and the communication topology can change dynamically.

As argued in [10, 11], the *state reachability problem* or the *coverability problem* are adequate for capturing several interesting properties that arise in communicating systems (e.g., Ad-Hoc networks). The problem consists in checking whether the system can start from a given initial configuration and evolve to reach a configuration in which at least one of the processes is in a given error state. To the best of our knowledge, this is the first work addressing the control state reachability problem for a class of dynamic register automata.

In this paper, we first show that the state reachability problem is undecidable even in the case where each process is equipped with only one register. Then, an important task is to identify subclasses of Dra for which algorithmic verification is possible. Inspired by some recent works on the verification of Ad-Hoc networks [10, 1], we consider a restricted version of the verification problem where we restrict the analysis to only *bounded* configurations, in which the maximum length of directed simple paths in the induced communication graph is bounded by a given natural number k. The communication graph represents the connectivity of the network induced by a Dra. In this graph each process is represented by a node and there is an edge from a node u to a node v if the process corresponding to u knows the process corresponding to v. It turns out that the verification problem remains undecidable for bounded Dra with at least two registers. Moreover, this undecidability holds even if we restrict the analysis to *strongly bounded* configurations, in which we require that the maximum length of simple paths in the *undirected* communication graph (i. e., regardless of the direction of the edges) is bounded (unlike the case of Ad-hoc networks [10, 11, 13, 1]).

Then, we introduce the model of *degenerative* Dra, a Dra in which any register can be reset in non-deterministic way. Degenerative Dra can be used to model unexpected loss of communication links in mobile Ad-hoc networks. Given a Dra, we associate a degenerative counterpart by allowing reset transitions at every state and for every register of the Dra. We show that the degenerative counterpart of a Dra represents an over-approximation of the original Dra in terms of reachable states. We prove that the approximation is exact by showing that the degenerative Dra does not expose more states than its non-degenerative counterpart. This implies that the reachability problem for degenerative Dra is also undecidable. Therefore, we consider the subclass of strongly bounded degenerative Dra. We show that degenerative Dra is a (*strict*) over-approximation of its non-degenerative counterpart (in terms of reachable states) when both are restricted to strongly bounded communication graphs. We also show that the state reachability problem for the class of strongly bounded degenerative Dra is decidable. The decidability proof is carried out by defining a symbolic backward reachability analysis based on a non-trivial instantiation of the

[1] Observe that in [7, 8], processes of Dra communicate asynchronously via (bounded) FIFO channels.

framework of well structured transition systems [2, 16]. Furthermore, we show that state reachability for the class of strongly bounded degenerative DRA is nonprimitive recursive by a reduction from reachability for lossy counter machines [25]. Hence, the class of strongly bounded degenerative DRA represents a good candidate for a decidable subclass of DRA.

We point out that bounded DRA with only one register is in fact strongly bounded. Thus, the state reachability problem for bounded degenerative 1-register-DRA is also decidable.

Finally, we introduce (strongly) safe DRA where we assume that all the reachable configurations are (strongly) bounded. We show that the state reachability problem for strongly safe DRA becomes decidable while the undecidability still holds for safe DRA.

Related work. Communicating finite state machines [9] are a well-known computational model for distributed systems where processes communicate through unbounded channels. They serve, for instance, as an implementation model for Message Sequence Charts with finitely many processes [5, 4, 18]. Several works address the verification problem, in particular the state reachability problem, of different classes of this model [3, 22, 17]. However, in contrary to our model, in most of these settings a *fixed* number of processes is considered which restricts their applicability for dynamic systems.

Communicating finite state machines are also used as a formal model for wireless Ad-Hoc networks [26, 27, 10, 11, 13, 1]. Every process in an Ad-Hoc network can perform local, (selective) broadcast and receive actions. While processes in a DRA perform 1-to-1 communications, broadcast actions in Ad-Hoc networks involve multiple processes. By performing a broadcast action a process sends a message to all its neighbour processes (whose number is not bounded *a priori*). An important question in the realm of Ad-Hoc networks is the state reachability problem, parametrized by the number of involved processes and by the network topology: is there a number of processes and a network topology such that after a finite number of transitions one process reaches a special state? Even though [26] and [10] consider models where the topology of the network can change, the processes cannot perform process creation, thus, the number of interacting processes is (arbitrary but) fixed.

Broadcast networks of register automata are introduced in [12]. The model is similar to DRA in the sense that the automata are equipped with a finite set of registers which can store some data. Besides this fact, the model of [12] does not support process creation and exchanging process ID does not affect the network topology.

2 Preliminaries

Let $\mathbb{N}$ denote the set of natural numbers. Let A and B be two sets. We use $|A|$ to denote the cardinality of A ($|A| = \omega$ if A is infinite). 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 every $a \in A$. 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 $a' \neq a$.

A *transition system* $\mathcal{T}$ is a triple $\langle C, C_{init}, \longrightarrow \rangle$, where C is a set of *configurations*, $C_{init} \subseteq C$ is an *initial* set of configurations, and $\longrightarrow \subseteq C \times C$ is a *transition relation*. We write $c_1 \longrightarrow c_2$ when $\langle c_1, c_2 \rangle \in \longrightarrow$ and $\longrightarrow^*$ to denote the reflexive transitive closure of $\longrightarrow$. For every $i \in \mathbb{N}$, we use $\longrightarrow^i$ to denote the i-times composition of $\longrightarrow$. A configuration $c \in C$ is said *reachable* in $\mathcal{T}$ if there is $c_{init} \in C_{init}$ such that $c_{init} \longrightarrow^* c$.

A *directed labeled graph* (or simply *graph*) G is a tuple $\langle V, \Sigma_v, \Sigma_e, \lambda, E \rangle$ where V is a finite set of vertices, Σ_v is a set of vertex labels, Σ_e is a set of edge labels, $\lambda : V \to \Sigma_v$ is the vertex

labeling function, and $E \subseteq V \times \Sigma_e \times V$ is the set of edges. A *path* in G is a finite sequence of vertices $\pi = v_1 v_2 \dots v_k, k \geq 1$, where, for every $i : 1 \leq i < k$, there is an $a \in \Sigma_e$ such that $\langle v_i, a, v_{i+1} \rangle \in E$. We say that π is *simple* if all vertices in π are different, i. e. $v_i \neq v_j$ for all $i, j : 1 \leq i < j \leq k$, and we define $\texttt{length}(\pi) := k - 1$. We define the diameter of G, denoted by $\oslash(G)$, to be the largest k such that there is a simple path π in G with $\texttt{length}(\pi) = k$.

3 Dynamic Register Automata

A *Dynamic Register Automaton* DRA consists of a set of processes that exchange messages and create new processes. Each process is modelled as a finite state automaton equipped with a finite set of registers. A register may contain the identifier (ID) of another process. A process can perform a *local* action that changes its current state. It can also create (or spawn) a new process, allowing the number of processes to increase over time. Communication is allowed between two processes given that the sender has the ID of the receiver in one of its registers. A process can send a message from a finite alphabet, its own ID as well as the content of one of its registers. Below, we describe the syntax of DRA and introduce the subclass of *degenerative* DRA where any register can be reset in a non-deterministic way. Then, we define the operational semantics of a DRA, and its state reachability problem.

Definition. A DRA 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, $X = \{x_1, \dots, x_n\}$ 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 $\texttt{action}$ is of one of the following forms: (i) τ (local action), (ii) $x \leftharpoondown \texttt{create}(q, y)$ where $x, y \in X$ and $q \in Q$, creates a new process with a fresh ID in state q, stores the ID of the new process in register x of the creator process, and stores the ID of the creating process in register y of the new process, (iii) $x!\langle m \rangle$ where $x \in X, m \in M$, sends message m to the process whose ID is stored in register x, (iv) $x!\langle y \rangle$ where $x \in X, y \in X \cup \{\texttt{self}\}$, sends either the ID contained in register y or the ID of the process itself ($\texttt{self}$) to the process whose ID is stored in x, (v) $x?\langle m \rangle$ where $x \in X, m \in M$ (selective message reception), receives message m from the process whose ID is stored in register x, (vi) $\star?\langle m \rangle$ where $m \in M$ (nonselective message reception), receives message m from some other process, (vii) $x?\langle y \rangle$ where $x \in X, y \in X$ (selective ID reception), receives an ID to be stored in register y from a process whose ID is stored in x, (viii) $\star?\langle y \rangle$ where $y \in X$ (nonselective ID reception), receives an ID to be stored in register y from some other process, and (ix) $\texttt{reset}\langle x \rangle$ where $x \in X$, resets register x so that it becomes undefined.

The DRA D is *degenerative* if for every state $q \in Q$ and register $x \in X$, $\langle q, \texttt{reset}\langle x \rangle, q \rangle \in \delta$. Given a DRA $D = \langle Q, q_0, M, X, \delta \rangle$, we define its degenerative counterpart DRA $\texttt{Deg}(D)$ by the tuple $\langle Q, q_0, M, X, \delta' \rangle$ with $\delta' = \delta \cup \{\langle q, \texttt{reset}\langle x \rangle, q \rangle \mid q \in Q, x \in X\}$.

Configuration. We use $\mathcal{P}$ to denote the domain of all possible process IDs. Let $D = \langle Q, q_0, M, X, \delta \rangle$ be a DRA. We define a configuration c as the tuple $\langle \texttt{procs}, \texttt{s}, \texttt{r} \rangle$, where $\texttt{procs} \subseteq \mathcal{P}$ is a finite set of processes, $\texttt{s} : \mathcal{P} \rightharpoonup Q$ maps each process $p \in \texttt{procs}$ to its current state and $\texttt{r} : \mathcal{P} \rightharpoonup \{X \rightharpoonup \texttt{procs}\}$ is a partial function that maps every process $p \in \texttt{procs}$ to its registers contents. 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 exists 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 $c = \langle \texttt{procs}, \texttt{s}, \texttt{r} \rangle \in C(D)$ is said to be *initial* if it contains exactly one process (i. e., $\texttt{procs} = \{p\}$ for some $p \in \mathcal{P}$),

which is in the initial state ($\mathtt{s}(p) = q_0$) and whose registers are empty ($\mathtt{r}(p) = \bot_X$). The set of initial configurations is denoted by $C_{init}(D)$.

Encoding of Configurations. The encoding of a configuration c is a graph $\mathtt{enc}(c)$ that models its register mappings. Every process in the encoding is represented by a vertex labeled with the state of the process. Furthermore, there is an edge from vertex u to vertex v labeled with $x \in X$ if the process corresponding to u has the ID of the process corresponding to v in its register x. Formally, the encoding of a configuration $c = \langle \mathtt{procs}, \mathtt{s}, \mathtt{r} \rangle$ is defined as the graph $\mathtt{enc}(c) := \langle \mathtt{procs}, Q, X, \mathtt{s}, E = \{\langle p, x, p' \rangle | \, \mathtt{r}(p)(x) = p'\} \rangle$.

Transition Relation. We define a transition relation $\rightarrow_D$ on the set $C(D)$ of configurations of the DRA D. Given two configurations $c = \langle \mathtt{procs}, \mathtt{s}, \mathtt{r} \rangle, c' = \langle \mathtt{procs}', \mathtt{s}', \mathtt{r}' \rangle \in C(D)$, we have $c \rightarrow_D c'$ 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) $\mathtt{procs}' = \mathtt{procs}$ and $\mathtt{r}' = \mathtt{r}$, i. e., the processes and registers are left unchanged, (ii) $\mathtt{s}(p) = q_1$, and (iii) $\mathtt{s}' = \mathtt{s}[p \leftarrow q_2]$. A local transition changes the state of (at most) one process.

`Create` There is a transition $\langle q_1, x \leftarrowtail \mathtt{create}\langle q, y \rangle, q_2 \rangle \in \delta$ and a process $p \in \mathtt{procs}$ such that (i) $\mathtt{s}(p) = q_1$, i. e., p is in state 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, and (iv) $\mathtt{r}' = \mathtt{r}[p \leftarrow \mathtt{r}(p)[x \leftarrow p']][p' \leftarrow \bot_X[y \leftarrow p]]$, i. e., register x of process p is assigned the ID of the new process p' and register y of process p' is assigned the ID of process p.

`Selective message sending` There are two different processes $p, p' \in \mathtt{procs}$ and two transitions $\langle q_1, x!\langle m \rangle, q_2 \rangle, \langle q_3, y?\langle m \rangle, q_4 \rangle \in \delta$ such that (i) $\mathtt{s}(p) = q_1$ and $\mathtt{s}(p') = q_3$, i. e., p and p' are in states q_1 and q_3, respectively, (ii) $\mathtt{r}(p)(x) = p'$ and $\mathtt{r}(p')(y) = p$, i. e., the sender p has the ID of p' in its register x and the receiver p' has the ID of p in its register y, (iii) $\mathtt{s}' = \mathtt{s}[p \leftarrow q_2][p' \leftarrow q_4]$, i. e., the states of both processes p and p' are updated simultaneously, and (iv) $\mathtt{r}' = \mathtt{r}$, i. e., the registers are unchanged.

`Selective ID sending` There are two different processes $p, p' \in \mathtt{procs}$ and two transitions $\langle q_1, x!\langle z_1 \rangle, q_2 \rangle, \langle q_3, y?\langle z_2 \rangle, q_4 \rangle \in \delta$ such that (i) $\mathtt{s}(p) = q_1$ and $\mathtt{s}(p') = q_3$, (ii) $\mathtt{r}(p)(x) = p'$ and $\mathtt{r}(p')(y) = p$, (iii) $\mathtt{s}' = \mathtt{s}[p \leftarrow q_2][p' \leftarrow q_4]$, (iv) either $z_1 = \mathtt{self}$ or there exist $p'' \in \mathtt{procs}$ such that $\mathtt{r}(p)(z_1) = p''$, i. e., the ID to be sent should be the ID of some process, and (v) $\mathtt{r}' = \mathtt{r}[p' \leftarrow \mathtt{r}(p')[z_2 \leftarrow p]]$ if $z_1 = \mathtt{self}$ or $\mathtt{r}' = \mathtt{r}[p' \leftarrow \mathtt{r}(p')[z_2 \leftarrow p'']]$ otherwise, i. e., register z_2 of p' is updated with what it receives from p.

`Register resetting` There is a transition $\langle q_1, \mathtt{reset}\langle x \rangle, q_2 \rangle \in \delta$ and a process $p \in \mathtt{procs}$ such that (i) $\mathtt{s}(p) = q_1$ and $\mathtt{s}' = \mathtt{s}[p \leftarrow q_2]$, and (ii) $\mathtt{r}' = \mathtt{r}[p \leftarrow \mathtt{r}(p)[x \leftarrow \bot]]$, i. e., register x of process p' is reset.

The only difference between `Nonselective message sending` and `Nonselective ID sending` and their selective counterparts is that the receiver does not need to know the sender, i. e., the ID of the sending process does not have to be in the registers of the receiver.

For a DRA $D = \langle Q, q_0, M, X, \delta \rangle$, we use $\xrightarrow{\mathtt{reset}}_D \subseteq C(D) \times C(D)$ to denote the set of transitions induced by the set of `Register resetting` transitions in δ of the form $\langle q, \mathtt{reset}\langle x \rangle, q \rangle$ with $q \in Q$ and $x \in X$.

State Reachability. Let $\mathcal{T}(D)$ denote the transition system defined by the triple $\langle C(D), C_{init}(D), \longrightarrow_D \rangle$. Let $\mathtt{target} \in Q$ be a state of D. The state `target` is said to be reachable if there exists a reachable configuration that has a process in state `target`. The state

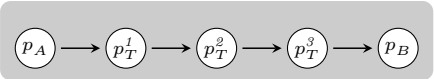

Figure 1 Transduction chain.

reachability problem consists in checking whether the state `target` is reachable or not. We use $\texttt{StateReach}(D, \texttt{target})$ to denote the state reachability problem for D and `target`.

It is obvious that any degenerative Dra is an over-approximation of its non-degenerative counterparts in terms of reachable states. Lemma 1 states that this approximation is exact.

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

4 State Reachability for (Degenerative) Dra

In the following, we show that the state reachability for (degenerative) Dra with at least one register is undecidable.

▶ **Theorem 2.** *Given a (degenerative)* Dra *$D = \langle Q, q_0, M, X, \delta\rangle$ and a state $q_f \in Q$, $\texttt{StateReach}(D, q_f)$ is undecidable. This undecidability holds even in the case where $|X| = 1$.*

The proof proceeds by reduction from the `Transd` problem defined below.

The `Transd` Problem. A *transducer* T is a tuple $\langle Q, q_{init}, \Sigma, \delta, F\rangle$ where Q is a finite set of *states*, q_{init} is the *initial* state, Σ is a finite *alphabet*, $\delta \subseteq Q \times \Sigma \times \Sigma \times Q$ is the *transducer transition relation*, and F is the set of *accepting* states. Every transition $t \in \delta$ gets as input some symbol $a \in \Sigma$ and outputs another symbol $b \in \Sigma$. The transducer transition relation δ induces on Σ^* a binary relation *Rel*, where $w\,Rel\,w'$ if w' is the output of T when accepting w. Given a word $w \in \Sigma^*$, let $T(w) := \{v \in \Sigma^* |\ w\,Rel\,v\}$ denote the set of any possible transduction of w by T. We extend the notion of transduction to a language $L \subseteq \Sigma^*$ by defining $T(L) := \bigcup_{w\in L} T(w)$. In an iterative way, we define for $i \in \mathbb{N}$ the i^{th} transduction of L as $T^0(L) := L$ and $T^{i+1}(L) := T(T^i(L))$. Given a finite state automaton A over the alphabet Σ, we denote by $L(A)$ the regular language accepted by A. An instance of the problem `Transd` consists of two finite state automata A and B, and a transducer T, all over the same alphabet Σ. In `Transd` it is checked whether there is a natural number $i \in \mathbb{N}$ such that $T^i(L(A)) \cap L(B) \neq \emptyset$. The problem `Transd` is known to be undecidable [1].

A Sketched proof of Thm. 2. Given an instance of `Transd`, i. e. two automata A and B and a transducer T over the same alphabet, the encoding of `Transd` into the state reachability problem of Dra consists of constructing a *transduction chain*, where the first element of the chain is a process p_A encoding A, the last one is a process p_B encoding B and all intermediate elements are processes p_T^i encoding T (Figure 1). The simulation of the transduction works as follows: The first process p_A sends a word $w \in \Sigma^*$ symbol by symbol to its successor in the chain. If w is a word accepted by A, p_A sends a special acceptance symbol to its successor. Meanwhile, each intermediate process simulating T sends for every incoming symbol from Σ a corresponding output symbol to its successor. If it gets the acceptance symbol it checks whether the so far received word is accepted by T. If it is the case, it transmits the acceptance symbol to the next process. At the reception of the acceptance symbol, the last process p_B in the chain checks whether the received word is accepted by B. If it is the case, it moves to the state q_f, if not, it moves to an error (deadlock) state. Note that if there are no

intermediate processes simulating T, process p_A sends the symbols directly to p_B. It can be shown by induction that there exists an $i \geq 0$ with $T^i(L(A)) \cap L(B) \neq \emptyset$ if and only if a transduction chain of length $i+2$ which reaches q_f can be constructed. Note that processes in the chain do not need more than one register and that a correct transduction chain can also be constructed by a degenerative Dra.

5 Bounded (Degenerative) Dra

The reduction from `Transd` to the state reachability for (degenerative) Dra relies on the fact that the transduction chain can be made as long as desired, allowing for $i \in \mathbb{N}$ in $T^i(L(A))$ to be as large as needed. One way to break the transducer chain proof would be to bound the diameter of the configuration encodings. In the following we show that this condition is still not sufficient. Let us first define a transition system where only configurations with bounded diameter are allowed. Let k be a natural number, D a Dra and $\mathcal{T}(D) = \langle C(D), C_{init}(D), \longrightarrow_D \rangle$ its corresponding transition system. We say that a configuration $c \in C(D)$ is k-bounded if the diameter of its encoding is bounded by k, i.e $\oslash(\mathtt{enc}(c)) \leq k$. Given a set $B \subseteq C(D)$ of configurations, we use $(B\Box k)$ to denote the set of k-bounded configurations in B. The restriction of $\longrightarrow_D$ to the set $C(D)\Box k$ of k-bounded configurations is denoted by $\longrightarrow_D^{\Box k} := \longrightarrow_D \cap ((C(D)\Box k) \times (C(D)\Box k))$. We use $\mathcal{T}^{\Box k}(D)$ to denote the resulting transition system defined by $\langle (C(D)\Box k), (C_{init}(D)\Box k), \longrightarrow_D^{\Box k} \rangle$. Given a state $\mathtt{target} \in Q$, the *k-bounded state reachability problem* consists in checking whether a configuration c with $\mathtt{target} \in c$ is reachable in $\mathcal{T}^{\Box k}(D)$. We use $\mathtt{BoundedStateReach}(D, \mathtt{target}, k)$ to denote the k-bounded state reachability problem. We prove the following result:

▶ **Theorem 3.** *Given a natural number $k \in \mathbb{N}$, a (degenerative)* Dra *$D = \langle Q, q_0, M, X, \delta \rangle$ and a state $q_f \in Q$, $\mathtt{BoundedStateReach}(D, q_f, k)$ is undecidable. This undecidability still holds even if $k = 2$ and $|X| = 2$.*

The proof can be done by a reduction from the `Transd` problem. Observe that there is no straightforward reduction from Thm. 3 to Thm. 2 and vice-versa.

6 Strongly Bounded (Degenerative) Dra

As we have seen, bounding the diameter of the configuration encoding is insufficient to get decidability of the state reachability problem. Therefore, we consider a new constraint on the graph encoding of the configurations. The new constraint consists in restricting the set of configurations such that the diameter of their graph encodings is bounded by some natural number k, this time regardless of the direction of the edges in the graph. In order to formally specify the new constraint, let us introduce the class of label-free undirected graphs.

Label-free Undirected Graph. A *label-free undirected graph* G is a graph whose edges have no labels and no direction, i. e. G is a tuple $\langle V, \Sigma_v, \lambda, E \rangle$ where V is a finite set of vertices, Σ_v is a finite set of vertex labels, $\lambda : V \to \Sigma_v$ is a vertex labeling function and $E \subseteq \{\{u, v\} |\ u, v \in V\}$ is a set of unlabeled and undirected edges. Notions of simple path and diameter of a graph are extended in the natural way to label-free undirected graphs. Given a (directed) graph $G = \langle V, \Sigma_v, \Sigma_e, \lambda, E \rangle$, we use $\mathtt{closure}(G) := \langle V, \Sigma_v, \lambda, F \rangle$ to denote the undirected graph obtained from G by removing directions and labels from its edges, i. e. $F := \{\{u, v\} |\ \langle u, a, v \rangle \in E\}$.

Strongly Bounded Configurations. Let k be a natural number, $D = \langle Q, q_0, M, X, \delta\rangle$ a DRA and $\mathcal{T}(D) = \langle C(D), C_{init}(D), \longrightarrow_D\rangle$ the transition system induced by D. Let c be a configuration in $C(D)$. We say that c is k-strongly bounded if $\circledcirc(\texttt{closure}(\texttt{enc}(c))) \leq k$. Given $B \subseteq C(D)$, we use $(B\Diamond k)$ to denote the set of k-strongly bounded configurations in B, i. e. $(B\Diamond k) := \{c \in B \mid \circledcirc(\texttt{closure}(\texttt{enc}(c))) \leq k\}$. We consider the transition relation $\longrightarrow_D^{\Diamond k}$ defined on $(C(D)\Diamond k)$ by $\longrightarrow_D^{\Diamond k} := \longrightarrow_D \cap((C(D)\Diamond k) \times (C(D)\Diamond k))$. We define the transition system $\mathcal{T}^{\Diamond k}(D) := \left\langle (C(D)\Diamond k), (C_{init}(D)\Diamond k), \longrightarrow_D^{\Diamond k}\right\rangle$. Given a state $\texttt{target} \in Q$, the *k-strongly bounded state reachability problem* consists in checking whether a configuration c with $\texttt{target} \in c$ is reachable in $\mathcal{T}^{\Diamond k}(D)$. We use $\texttt{StrongBoundStateReach}(D, \texttt{target}, k)$ to denote the k-strongly bounded state reachability problem.

▶ **Theorem 4.** *Given $k \in \mathbb{N}$, a* DRA *$D = \langle Q, q_0, M, X, \delta\rangle$ and a state $\texttt{target} \in Q$, $\texttt{StrongBoundStateReach}(D, \texttt{target}, k)$ is undecidable. This undecidability still holds even if $k = 4$ and $|X| = 2$.*

The proof of Thm. 4 can be established by a reduction from the reachability problem for Minsky's 2-counter machines.

▶ **Theorem 5.** *Given $k \in \mathbb{N}$, a degenerative* DRA *$D = \langle Q, q_0, M, X, \delta\rangle$ and a state $\texttt{target} \in Q$, $\texttt{StrongBoundStateReach}(D, \texttt{target}, k)$ is decidable and nonprimitive recursive.*

The decidability of the strongly bounded state reachability problem for degenerative DRA is established by a non-trivial instantiation of the framework of well-quasi-ordered systems [2, 16] (See Section 8). The nonprimitive recursive lower bound is carried out through a reduction from the reachability problem for *Lossy Counter Machines* [25].

Furthermore, it is clear that the set of k-strongly bounded reachable states by a DRA D is a (strict) subset of the set of k-strongly bounded reachable states by its degenerative DRA counterpart $\texttt{Deg}(D)$. Moreover, the set of k-strongly bounded reachable states by the degenerative DRA $\texttt{Deg}(D)$ is a subset of the set of reachable states by D. Thus, the strongly bounded reachability problem for $\texttt{Deg}(D)$ is a good under-approximation of the state reachability problem for D. This relation[2] between the strongly bounded reachability problems for a DRA D and its corresponding degenerative one $\texttt{Deg}(D)$ is given by the following observation:

▶ Observation 1. Let $k \in \mathbb{N}$ be a natural number, D a DRA, and $\texttt{target}$ a state of D. If $\texttt{target}$ is reachable in $\mathcal{T}^{\Diamond k}(D)$ then it is reachable in $\mathcal{T}^{\Diamond k}(\texttt{Deg}(D))$. Furthermore, if $\texttt{target}$ is reachable in $\mathcal{T}^{\Diamond k}(\texttt{Deg}(D))$ then there is $k' \geq k$ such that $\texttt{target}$ is reachable in $\mathcal{T}^{\Diamond k'}(D)$.

The decidability of bounded degenerative DRA with one register (see Corollary 7) can be inferred from Theorem 5 and the following lemma:

▶ **Lemma 6.** *Any k-bounded configuration of a* DRA *with one register is $2k$-strongly bounded.*

▶ **Corollary 7.** *Given a natural number $k \in \mathbb{N}$, a degenerative* DRA *$D = \langle Q, q_0, M, X, \delta\rangle$ with $|X| = 1$ and a state $\texttt{target} \in Q$, $\texttt{BoundedStateReach}(D, \texttt{target}, k)$ is decidable.*

7 (Strongly) Safe Dra

A k-strongly bounded DRA forbids transitions to configurations that are not k-strongly bounded. This allows to simulate zero tests of the Minsky's 2-counter machine in the proof

[2] Observe that this relation holds also for the (bounded) reachability problem.

of Thm. 4. Therefore, we introduce k-(strongly) safe Dra, with $k \in \mathbb{N}$, which is a Dra where we assume that all reachable configurations are k-(strongly) bounded. Formally, let D be a Dra and $\mathcal{T}(D)$ its induced transition system. The Dra D is said to be k-(strongly) safe iff every reachable configuration in $\mathcal{T}(D)$ is k-(strongly) bounded. We can state:

▶ Observation 2. If D is a k-strongly safe Dra then $\mathtt{Deg}(D)$ is a k-strongly bounded Dra.

As an immediate consequence of Lemma 1, Observation 2 and Theorem 5, we infer:

▶ **Corollary 8.** *Given a k-strongly safe* Dra $D = \langle Q, q_0, M, X, \delta\rangle$ *and a state $q_f \in Q$, $\mathtt{StateReach}(D, q_f)$ is decidable.*

However, the state reachability problem is still undecidable for k-safe (degenerative) Dra.

▶ **Theorem 9.** *Given a k-safe (degenerative)* Dra $D = \langle Q, q_0, M, X, \delta\rangle$ *and a state $q_f \in Q$, $\mathtt{StateReach}(D, q_f)$ is undecidable.*

8 Strongly Bounded Degenerative Dra: Proof of Theorem 5

This section is devoted to the decidability proof of Theorem 5 by making use of the framework of *Well-Structured Transition Systems* (Wsts) [2, 16].

We briefly recall the framework of Wsts. Let C be a (possibly infinite) set and $\preccurlyeq$ be a *well-quasi order* on C. Recall that a well-quasi order on C is a binary relation over C that is reflexive and transitive and for every infinite sequence $(a_i)_{i\geq 0}$ of elements in C there exist $i, j \in \mathbb{N}$ such that $i < j$ and $a_i \preccurlyeq a_j$. A set $U \subseteq C$ is called *upward closed* if for every $a \in U$ and $b \in C$ with $a \preccurlyeq b$ we have $b \in U$. The upward closure of some set $U \subseteq C$ is defined as $U{\uparrow} := \{b \in C \mid \exists a \in U \text{ with } a \preccurlyeq b\}$. It is known that every upward closed set U can be characterised by a finite *minor set* $M \subseteq U$ such that (i) for every $a \in U$ there is $b \in M$ such that $b \preccurlyeq a$, and (ii) if $a, b \in M$ and $a \preccurlyeq b$ then $a = b$. We use $\mathtt{min}$ to denote the function which for a given upward closed set U returns one minor set of U.

Let $\mathcal{T} = \langle C, C_{init}, \rightsquigarrow\rangle$ be a transition system and $\preccurlyeq$ be a well-quasi ordering on C. For a subset $U \subseteq C$ of configurations we define the set of predecessors of U as $\mathtt{Pre}(U) := \{c \in C \mid \exists c_1 \in U, c \rightsquigarrow c_1\}$. For a configuration c we denote the set $\mathtt{min}(\mathtt{Pre}(\{c\}{\uparrow}) \cup \{c\}{\uparrow})$ as $\mathtt{minpre}(c)$. $\mathcal{T}$ is called *well-structured* if $\rightsquigarrow$ is *monotonic* wrt. $\preccurlyeq$, i. e. given three configurations $c_1, c_2, c_3 \in C$, if $c_1 \rightsquigarrow c_2$ and $c_1 \preccurlyeq c_3$ then there exists a fourth configuration $c_4 \in C$ such that $c_3 \rightsquigarrow c_4$ and $c_2 \preccurlyeq c_4$.

Given a configuration $c_{\mathtt{target}} \in C$, the *coverability problem* asks whether there is a configuration $c' \succcurlyeq c_{\mathtt{target}}$ reachable in $\mathcal{T}$. For the decidability of this problem the following conditions are sufficient: (i) For every two configurations c_1 and c_2 it is decidable whether $c_1 \preccurlyeq c_2$, (ii) for every $c \in C$, we can check whether $\{c\}{\uparrow} \cap C_{init} \neq \emptyset$, and (iii) for every $c \in C$, the set $\mathtt{minpre}(c)$ is finite and computable.

The solution for the coverability problem of Wsts suggested in [2, 16] 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 degenerative Dra, but first we need to introduce some notations.

Let k be a natural number, $D = \langle Q, q_0, M, X, \delta\rangle$ a degenerative Dra and $\mathtt{target} \in Q$ a target state. Let $C_{init} = (C_{\mathtt{init}}(D)\Diamond k)$ and $C = (C(D)\Diamond k)$. We use $\mathcal{T}^{\Diamond k}(D) = \left\langle C, C_{init}, \longrightarrow_D^{\Diamond k}\right\rangle$ to denote the corresponding k-strongly bounded transition system of D. We introduce the *reset prefix* transition relation $\rightsquigarrow := \xrightarrow{\mathtt{reset}}_D^{*} \circ \longrightarrow_D^{\Diamond k}$. Note that the

reflexive transitive closures of $\rightsquigarrow$ and $\longrightarrow_D^{\Diamond k}$ are identical. Thus, the state reachability of $\texttt{target}$ in $\left\langle C, C_{init}, \longrightarrow_D^{\Diamond k} \right\rangle$ is equivalent to its corresponding problem in $\langle C, C_{init}, \rightsquigarrow \rangle$. Next, we will prove the decidability of the latter problem.

We will show that $\langle C, C_{init}, \rightsquigarrow \rangle$ is a well-structured transition system. Let $c_{target} = \langle \{p\}, \mathtt{s}, \mathtt{r} \rangle$ be a configuration composed of a single process in state $\texttt{target}$ ($\mathtt{s}(p) = \texttt{target}$) whose registers are empty ($\mathtt{r}(p) = \bot_X$). 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 $\texttt{target} \in c$. Then, it is clear that the coverability of $c_{\texttt{target}}$ in $\langle C, C_{init}, \rightsquigarrow \rangle$ is equivalent to the reachability of $\texttt{target}$ in the same transition system.

In section 8.1, we define the well-quasi ordering $\preccurlyeq$ (Lemma 11) on C such that for every $c_1, c_2 \in C$ it is decidable whether $c_1 \preccurlyeq c_2$. The monotonicity of $\rightsquigarrow$ with respect to $\preccurlyeq$ is shown in section 8.2 (Lemma 12). The second sufficient condition for the decidability of the coverability problem, namely checking whether the upward closure of a configuration c contains an initial configuration, is trivial (we check whether c is an initial configuration). The last sufficient condition is shown by the following lemma:

▶ **Lemma 10.** *Given a configuration $c \in C$, we can effectively compute $\texttt{minpre}(c)$.*

Lemma 10, Lemma 11 and Lemma 12 show that coverability of c_{target} is decidable. Hence, the state reachability problem for strongly bounded degenerative Dra is decidable.

8.1 A well-quasi order on configurations

In this section, we define a well-quasi ordering $\preccurlyeq$ over the set C of configurations. Let us first introduce the notion of *subgraph* embedding. We use $\sqsubseteq_{sub}$ to denote the *subgraph* relation defined on graphs as follows: $\langle V_1, \Sigma_v, \Sigma_e, \lambda_1, E_1 \rangle \sqsubseteq_{sub} \langle V_2, \Sigma_v, \Sigma_e, \lambda_2, E_2 \rangle$ if there exists an injective mapping $t : V_1 \to V_2$ that is label and edge preserving, i. e. $\forall v, u \in V$ and $\forall a \in \Sigma_e$ we have $\lambda_1(v) = \lambda_2(t(v))$ and $\langle v, a, u \rangle \in E_1 \Rightarrow \langle t(v), a, t(u) \rangle \in E_2$. The subgraph relation over undirected (label-free) graphs are defined in a similar manner. We define the ordering $\preccurlyeq$ over the set of configurations as follows: Given two configurations $c_1 = \langle \texttt{procs}_1, \mathtt{s}_1, \mathtt{r}_1 \rangle$ and $c_2 = \langle \texttt{procs}_2, \mathtt{s}_2, \mathtt{r}_2 \rangle$, $c_1 \preccurlyeq c_2$ holds if $\texttt{enc}(c_1) \sqsubseteq_{sub} \texttt{enc}(c_2)$. Note that $c_1 \preccurlyeq c_2$ is equivalent to say that there exists an injective mapping $g : \texttt{procs}_1 \to \texttt{procs}_2$, such that (i) for every $p \in \texttt{procs}$, $\mathtt{s}_1(p) = \mathtt{s}_2(g(p))$ (ii) for every $p_1, p_2 \in \texttt{procs}_1$ and every $x \in X$, if $\mathtt{r}_1(p_1)(x) = p_2$ then $\mathtt{r}_2(g(p_1))(x) = g(p_2)$. It is easy to see that for two configurations c_1, c_2, we can check whether $c_1 \preccurlyeq c_2$.

▶ **Lemma 11.** *The relation $\preccurlyeq$ is a well-quasi ordering on C.*

8.2 Monotonicity

Let $c_1, c_2, c_3 \in C$ be three configurations such that $c_1 \preccurlyeq c_3$, i. e. the encoding of c_1 can be embedded in the encoding of c_3, and $c_1 \rightsquigarrow c_2$, i. e. there exist $c_1' \in C$ and $r \in \mathbb{N}$ such that $c_1 \xrightarrow{\texttt{reset}}_D^r c_1'$ and $c_1' \longrightarrow_D^{\Diamond k} c_2$. In order to prove $\rightsquigarrow$ monotonicity wrt. $\preccurlyeq$, we need to prove that there exists a fourth configuration $c_4 \in C$ such that $c_3 \rightsquigarrow c_4$ and $c_2 \preccurlyeq c_4$. To that end, we proceed by isolating the *sub configuration* c_{sub} induced by the embedding of c_1 into c_3 (see Figure 2). After a certain number r' (3 in Figure 2) of reset transitions $\xrightarrow{\texttt{reset}}_D$, one can obtain from c_3 a configuration c_3° composed of the disjoint union of the sub configuration c_{sub} and a set of *isolated* processes, i. e. processes whose registers are empty. As a consequence, diameters of c_3° and c_1 are equal. Furthermore, since c_{sub} is an embedding of c_1 into c_3°, and since $\oslash(\texttt{closure}(\texttt{enc}(c_3^\circ))) = \oslash(\texttt{closure}(\texttt{enc}(c_1)))$, c_3° can perform the

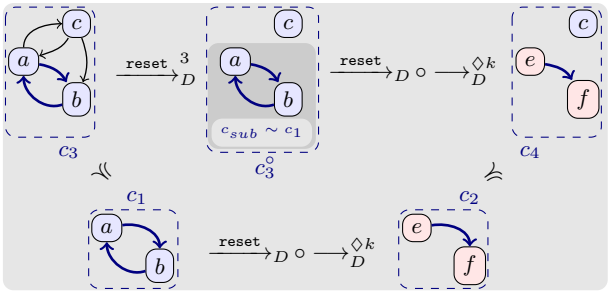

Figure 2 Monotonicity and reset transitions.

same transition as c_1 did in order to get to c_2 without violating the bound k. Thus, after two consecutive transitions whose composition ($\xrightarrow{\texttt{reset}}{}^{r'}_{D} \circ \xrightarrow{\texttt{reset}}{}^{r}_{D} \circ \longrightarrow^{\Diamond k}_{D} = \xrightarrow{\texttt{reset}}{}^{r'+r}_{D} \circ \longrightarrow^{\Diamond k}_{D}$) is a $\rightsquigarrow$ -transition, c_3 can reach a configuration where c_2 can be embedded.

▶ **Lemma 12.** *The transition relation $\rightsquigarrow$ is monotonic w.r.t. $\preccurlyeq$.*

9 Conclusion and Future Directions

We have presented the first work addressing the state reachability problem for Dra. We have shown that this problem is undecidable and that this undecidability holds even if we restrict the analysis to the case where transitions are only allowed between (strongly) bounded configurations (i. e., simple paths of the underlying (undirected) graph are bounded by some constant), unlike the case of Ad-hoc networks [10, 11, 13, 1]. Our main goal was to identify subclasses of Dra for which the reachability problem is decidable. To that end, we have introduced degenerative Dra for which any register can be reset in a non-deterministic manner. We have shown that the sets of reachable states of a Dra and its degenerative counterpart are identical. Moreover, we have shown that the reachability problem for degenerative Dra becomes decidable but nonprimitive recursive when we restrict the analysis to strongly bounded configurations. Furthermore, we have considered (strongly) safe Dra where we assume that all reachable configurations are (strongly) bounded. We have shown that the state reachability problem is decidable for strongly safe Dra.

To the best of our knowledge these are the first results concerning the verification of dynamic register automata. While the communication in Dra is rendezvous based, the automata models considered in [8] and [7] use asynchronous communication through unbounded channels. It is well-known that, even for finitely many processes communicating through unbounded perfect FIFO channels, most of the interesting verification questions are undecidable [9]. A possible direction of further research would be to investigate whether our decidability result carries over to the case of asynchronous communication through *"well-structured"* channels (e.g., bounded, lossy, unordered).

Acknowledgements. Authors would like to thank Benedikt Bollig, Thomas Schwentick and Thomas Zeume for their careful reading of the paper and their helpful comments.

References

1 Parosh Aziz Abdula, Mohamed Faouzi Atig, and Othmane Rezine. Verification of directed acyclic ad hoc networks. In *FMOODS/FORTE*, pages 193–208, 2013.

2 P. A. Abdulla, K. Cerans, B. Jonsson, and Y. K. Tsay. General decidability theorems for infinite-state systems. In *LICS'96*, pages 313–321. IEEE Computer Society, 1996.

3 Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. *Inf. Comput.*, 127(2):91–101, 1996.

4 Bharat Adsul, Madhavan Mukund, K. Narayan Kumar, and Vasumathi Narayanan. Causal closure for MSC languages. In *FSTTCS*, volume 3821 of *LNCS*, pages 335–347. Springer, 2005.

5 Rajeev Alur, Kousha Etessami, and Mihalis Yannakakis. Realizability and verification of MSC graphs. *Theor. Comput. Sci.*, 331(1):97–114, 2005.

6 Michael Benedikt, Clemens Ley, and Gabriele Puppis. Automata vs. logics on data words. In *CSL*, volume 6247 of *LNCS*, pages 110–124. Springer, 2010.

7 Benedikt Bollig, Aiswarya Cyriac, Loïc Hélouët, Ahmet Kara, and Thomas Schwentick. Dynamic communicating automata and branching high-level MSCs. In *LATA*, volume 7810 of *LNCS*. Springer, 2013.

8 Benedikt Bollig and Loïc Hélouët. Realizability of dynamic MSC languages. In *CSR*, volume 6072 of *LNCS*. Springer, 2010.

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

10 G. Delzanno, A. Sangnier, and G. Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR'10*, volume 6269 of *LNCS*. Springer, 2010.

11 G. Delzanno, A. Sangnier, and G. Zavattaro. On the power of cliques in the parameterized verification of ad hoc networks. In *FoSSaCS'11*, volume 6604 of *LNCS*, pages 441–455. Springer, 2011.

12 Giorgio Delzanno, Arnaud Sangnier, and Riccardo Traverso. Parameterized verification of broadcast networks of register automata. In *RP*, volume 8169 of *LNCS*. Springer, 2013.

13 Giorgio Delzanno, Arnaud Sangnier, Riccardo Traverso, and Gianluigi Zavattaro. On the complexity of parameterized reachability in reconfigurable broadcast networks. In *FSTTCS*, volume 18 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

14 Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. In *LICS*, pages 17–26. IEEE Computer Society, 2006.

15 D. Figueira. Alternating register automata on finite words and trees. *Logical Methods in Computer Science*, 8(1), 2012.

16 A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.

17 Blaise Genest, Dietrich Kuske, and Anca Muscholl. A kleene theorem and model checking algorithms for existentially bounded communicating automata. *Inf. Comput.*, 204(6):920–956, 2006.

18 Jesper G. Henriksen, Madhavan Mukund, K. Narayan Kumar, Milind A. Sohoni, and P. S. Thiagarajan. A theory of regular MSC languages. *Inf. Comput.*, 202(1):1–38, 2005.

19 M. Jurdzinski and R. Lazic. Alternation-free modal mu-calculus for data trees. In *LICS*, pages 131–140. IEEE Computer Society, 2007.

20 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

21 Ranko Lazic. Safely freezing LTL. In *FSTTCS*, volume 4337 of *LNCS*, pages 381–392. Springer, 2006.

22 Anca Muscholl and Doron Peled. Message sequence graphs and decision problems on Mazurkiewicz traces. In *MFCS*, volume 1672 of *LNCS*, pages 81–91. Springer, 1999.

23 Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004.

24 Hiroshi Sakamoto and Daisuke Ikeda. Intractability of decision problems for finite-memory automata. *Theor. Comput. Sci.*, 231(2):297–308, 2000.

25 P. Schnoebelen. Revisiting ackermann-hardness for lossy counter machines and reset petri nets. In *MFCS*, volume 6281 of *LNCS*, pages 616–628. Springer, 2010.

26 Anu Singh, C. R. Ramakrishnan, and Scott A. Smolka. Query-based model checking of ad hoc network protocols. In *CONCUR*, volume 5710 of *LNCS*, pages 603–619. Springer, 2009.

27 Anu Singh, C. R. Ramakrishnan, and Scott A. Smolka. A process calculus for mobile ad hoc networks. *Sci. Comput. Program.*, 75(6):440–469, 2010.

28 Nikos Tzevelekos. Fresh-register automata. In *POPL*. ACM, 2011.