

# Regular Model Checking for Systems with Effectively Regular Reachability Relation

Javier Esparza 

Technical University of Munich, Germany

Valentin Krasotin 

Technical University of Munich, Germany

---

## Abstract

Regular model checking is a well-established technique for the verification of *regular transition systems* (RTS): transition systems whose initial configurations and transition relation can be effectively encoded as regular languages. In 2008, To and Libkin studied RTSs in which the reachability relation (the reflexive and transitive closure of the transition relation) is also effectively regular, and showed that the recurrent reachability problem (whether a regular set  $L$  of configurations is reached infinitely often) is polynomial in the size of RTS and the transducer for the reachability relation. We extend the work of To and Libkin by studying the decidability and complexity of verifying *almost-sure* reachability and recurrent reachability – that is, whether  $L$  is reachable or recurrently reachable with probability 1. We then apply our results to the more common case in which only a regular overapproximation of the reachability relation is available. In particular, we extend recent complexity results on verifying safety using *regular abstraction frameworks* – a technique recently introduced by Czerner, the authors, and Welzel-Mohr – to liveness and almost-sure properties.

**2012 ACM Subject Classification** Theory of computation → Regular languages; Theory of computation → Problems, reductions and completeness; Theory of computation → Verification by model checking

**Keywords and phrases** Regular model checking, abstraction, inductive invariants

**Digital Object Identifier** 10.4230/LIPIcs.MFCS.2025.45

**Related Version** *Full Version*: <https://arxiv.org/abs/2506.18833> [30]

**Acknowledgements** We thank Anthony Widjaja Lin and Pascal Bergsträßer for very insightful discussions, and the anonymous reviewers for helpful comments.

## 1 Introduction

Regular model checking is a well-established technique for the verification of infinite-state systems whose configurations can be represented as finite words over a suitable alphabet. It applies to *regular transition systems* (RTS): systems whose set of initial configurations and transition relation are both regular, and presented as a finite automaton and a finite-state transducer, respectively [14, 8, 9, 1, 2]. The main application of regular model checking is the verification of parameterized systems in which an arbitrarily long array or ring of finite-state processes interact [10]. Examples of these systems include mutual exclusion algorithms, cache coherence protocols, communication protocols, consensus algorithms, and others.

RTSs are very general; in particular, it is easy to encode Turing machines as regular transition systems, which makes all interesting analysis problems for RTSs undecidable. In [43, 44], To and Libkin observed that in some classes of RTSs, like pushdown systems and ground-term-rewriting systems, the reachability relation (i.e., the reflexive and transitive closure of the transition relation) is regular and one can effectively compute a transducer recognizing it. They showed that in this case, the *reachability* and *recurrent reachability* problems become decidable and solvable in polynomial time in the size of the transducer



© Javier Esparza and Valentin Krasotin;

licensed under Creative Commons License CC-BY 4.0

50th International Symposium on Mathematical Foundations of Computer Science (MFCS 2025).

Editors: Paweł Gawrychowski, Filip Mazowiecki, and Michał Skrzypczak; Article No. 45; pp. 45:1–45:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

for the reachability relation. These problems ask, given a RTS  $\mathcal{S}$  and a regular set  $L$  of configurations, whether some run of  $\mathcal{S}$  – that is, some run starting at some initial configuration of  $\mathcal{S}$  – visits  $L$  at least once (reachability) or infinitely often (recurrent reachability). In the notation of CTL\*, they correspond to deciding the formulas  $\mathbf{EF} L$  and  $\mathbf{EGF} L$ , respectively. Many other problems can be reduced to reachability or recurrent reachability

In this paper we extend the work of To and Libkin in two different directions. Both of them are motivated by our interest in applying regular model checking to the verification of liveness properties of *randomized* distributed systems with an arbitrary number of processes, like many algorithms for distributed consensus or self-stabilization.

**Verification of almost-sure properties.** Proving liveness properties for randomized distributed systems amounts to showing that, under a class of adversarial stochastic schedulers selecting the process making the next move, something happens *almost surely* (a.s.), or, in other words, that for every scheduler in the class and every initial configuration, the runs of the system starting at that configuration and satisfying a given property have probability one [38, 40]. This raises the question whether a.s. reachability and a.s. recurrent reachability are decidable when the reachability relation is effectively regular assuming, as in [44], that a transducer for the reachability relation is part of the input. We also study whether reachability and recurrent reachability holds for all possible schedulers, that is, whether  $L$  is visited at least once or infinitely often by *every* run starting at any initial configuration. These properties – which we call sure reachability and sure recurrent reachability and are expressed by the formulas  $\mathbf{AF} L$  and  $\mathbf{AGF} L$  in CTL\* – hold for a fair number of probabilistic systems, as observed in [40]. We also study the complexity of some related properties, like a.s. termination or deadlock-freedom, which are important in applications.

Most algorithms for distributed consensus or self-stabilization are designed to work for an arbitrary but fixed number of processes, without dynamic process creation or deletion. In these systems the successors of a configuration with  $k$  processes also have  $k$  processes, and so both the configuration and its successor are encoded by words of length  $k$ . For this reason, on top of the complexity for general transducers we also investigate the complexity for the length-preserving case.

A summary of our results is shown in Tables 1 and 2 (recall that  $\Pi_1^0$  are the co-recursively enumerable languages and  $\Pi_1^1$  the corresponding level of the analytic hierarchy [42]). In all cases, the problem consists of deciding, given an RTS, a nondeterministic transducer recognizing the reachability relation, and an NFA recognizing the set  $L$  of configurations, whether the corresponding property holds. Our main results are:

- There is a big gap in the complexities of  $\mathbf{EF} L$  and  $\mathbf{AF} L$  (the first can be solved in polynomial time, while the second is undecidable) even in the length-preserving case.
- A.s. recurrent reachability is easier to check than sure reachability in the length-preserving case. This is surprising, because, as mentioned above, sure reachability is often used as an easier-to-check approximation to a.s. recurrent reachability.
- For sure properties, the general case is harder than the length-preserving case. Indeed, since they are  $\Pi_1^1$ - and  $\Pi_1^0$ -complete, respectively, there is no recursive reduction from the first to the second. This is contrary to the case of reachability and recurrent reachability, where the general case can be reduced to the length-preserving case.

**Regular overapproximations of the reachability relation.** The reachability relation is regular for some classes of RTSs, but not for many others. In particular, this is not the case for most models of concurrent and distributed computing, like Vector Addition Systems

■ **Table 1** Complexity of several decision problems for RTSs given a transducer for the reachability relation.  $T$  denotes the set of configurations without a successor. Note that sure termination is the special case of recurrent reachability where  $L = \mathcal{C}$ .

property	CTL*	length-preserving	general
reachability	$\mathbf{EF} L$	NL-complete	NL-complete
recurrent reachability	$\mathbf{EGF} L$	NL-complete (see [12])	NL-complete (see [12])
sure reachability	$\mathbf{AF} L$	$\Pi_1^0$ -complete	$\Pi_1^1$ -complete
sure recurrent reachability	$\mathbf{AGF} L$	$\Pi_1^0$ -complete	$\Pi_1^1$ -complete
sure termination	$\mathbf{AF} T$	NL-complete (see [12])	NL-complete (see [12])
deadlock-freedom	$\mathbf{AG} \bar{T}$	PSPACE-complete	PSPACE-complete

■ **Table 2** Complexity of several decision problems for probabilistic RTSs given a transducer for the reachability relation.  $T$  denotes the set of configurations without a successor. Note that a.s. deadlock-freedom is equivalent to deadlock-freedom.

property	LTL	length-preserving	general
a.s. reachability	$\mathcal{P}(\mathbf{F} L) = 1$	$\Pi_1^0$ -complete	undecidable
a.s. recurrent reachability	$\mathcal{P}(\mathbf{GF} L) = 1$	PSPACE-complete	undecidable
a.s. termination	$\mathcal{P}(\mathbf{F} T) = 1$	EXSPACE-complete	undecidable
a.s. deadlock-freedom	$\mathcal{P}(\mathbf{G} \bar{T}) = 1$	PSPACE-complete	PSPACE-complete

(VAS, aka Petri nets) and many of their extensions [41, 13], lossy channel systems [7, 37], broadcast protocols [22, 27] or population protocols [11]. However, several techniques exist for computing a regular *overapproximation* of the reachability relation [6, 35, 19]. In this paper, we are interested in the regular abstraction frameworks of [19]. In this approach, transducers are used not only to model the transition relation of the system, but also, in the terminology of abstract interpretation, to model abstract domains [18]. After the user fixes an abstract domain by choosing a transducer, the system automatically computes another transducer recognizing the *potential reachability* relation, the best overapproximation in the abstract domain of the reachability relation – in a certain technical sense<sup>1</sup>. It is shown in [19] that, while the safety problem (whether, given an RTS and a regular set of unsafe configurations, some reachable configuration is unsafe) is undecidable, the abstract safety problem (whether, given additionally a transducer for an abstract domain, some *potentially* reachable configuration is unsafe) is EXSPACE-complete.

We study whether the decision algorithms for sure and almost-sure properties become semi-decision algorithms when one uses a regular abstraction, i.e., whether they become algorithms that always terminate and answer “yes” or “don’t know” (or “no” and “don’t know”). We first show that this is the case for recurrent reachability and prove that deciding whether some *potential run* of the RTS visits  $L$  infinitely often, is also EXSPACE-complete. We then study almost-sure recurrent reachability. It is easy to see that in general one does not obtain a semi-decision algorithm. However, our last result shows that one does for systems in which the set of predecessors of the set  $L$  of goal configurations is effectively

<sup>1</sup> The precise notion of best approximation is not the same as in standard abstract interpretation.

computable, a condition satisfied by all well-structured transition systems [4, 31] under weak conditions satisfied all the models of [41, 13, 7, 37, 22, 27, 11]. In particular, we prove that the abstract version of a.s. recurrent reachability is EXPSPACE-complete for population protocols, whereas the problem itself is equivalent to the reachability problem for Petri nets under elementary reductions [28], and therefore Ackermann-complete [20, 21, 39].

## 2 Preliminaries

**Relations.** Let  $R \subseteq X \times Y$  be a relation. The *complement* of  $R$  is the relation  $\bar{R} := \{(x, y) \in X \times Y \mid (x, y) \notin R\}$ . The *inverse* of  $R$  is the relation  $R^{-1} := \{(y, x) \in Y \times X \mid (x, y) \in R\}$ . The *projections* of  $R$  onto its first and second components are the sets  $R|_1 := \{x \in X \mid \exists y \in Y : (x, y) \in R\}$  and  $R|_2 := \{y \in Y \mid \exists x \in X : (x, y) \in R\}$ . The *join* of two relations  $R \subseteq X \times Y$  and  $S \subseteq Y \times Z$  is the relation  $R \circ S := \{(x, z) \in X \times Z \mid \exists y \in Y : (x, y) \in R, (y, z) \in S\}$ . The *post-image* of a set  $X' \subseteq X$  under a relation  $R \subseteq X \times Y$ , denoted  $X' \circ R$  or  $R(X')$ , is the set  $\{y \in Y \mid \exists x \in X' : (x, y) \in R\}$ ; the *pre-image*, denoted  $R \circ Y$  or  $R^{-1}(Y)$ , is defined analogously. Throughout this paper, we only consider relations where  $X = \Sigma^*$  and  $Y = \Gamma^*$  for some alphabets  $\Sigma, \Gamma$ . We just call them relations. A relation  $R \subseteq \Sigma^* \times \Gamma^*$  is *length-preserving* if  $(u, w) \in R$  implies  $|u| = |w|$ .

**Automata.** Let  $\Sigma$  be an alphabet. A *nondeterministic finite automaton (NFA)* over  $\Sigma$  is a tuple  $A = (Q, \Sigma, \delta, Q_0, F)$  where  $Q$  is a finite set of *states*,  $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$  is the *transition function*,  $Q_0 \subseteq Q$  is the set of *initial states*, and  $F \subseteq Q$  is the set of *final states*. A *run* of  $A$  on a word  $w = w_1 \cdots w_l \in \Sigma^l$  is a sequence  $q_0 q_1 \cdots q_l$  of states where  $q_0 \in Q_0$  and  $\forall i \in [l] : q_i \in \delta(q_{i-1}, w_i)$ . A run on  $w$  is *accepting* if  $q_l \in F$ , and  $A$  *accepts*  $w$  if there exists an accepting run of  $A$  on  $w$ . The language *recognised* by  $A$ , denoted  $L(A)$ , is the set of words accepted by  $A$ . If  $|Q_0| = 1$  and  $|\delta(q, a)| = 1$  for every  $q \in Q, a \in \Sigma$ , then  $A$  is a *deterministic finite automaton (DFA)*.

**Convolutions and transducers.** Let  $\Sigma, \Gamma$  be alphabets, let  $\# \notin \Sigma \cup \Gamma$  be a padding symbol, and let  $\Sigma_{\#} := \Sigma \cup \{\#\}$  and  $\Gamma_{\#} := \Gamma \cup \{\#\}$ . The *convolution* of two words  $u = a_1 \dots a_k \in \Sigma^*$  and  $w = b_1 \dots b_l \in \Gamma^*$ , denoted  $\begin{bmatrix} u \\ w \end{bmatrix}$ , is the word over the alphabet  $\Sigma_{\#} \times \Gamma_{\#}$  defined as follows. Intuitively,  $\begin{bmatrix} u \\ w \end{bmatrix}$  is the result of putting  $u$  on top of  $w$ , aligned left, and padding the shorter of  $u$  and  $w$  with  $\#$ . Formally, if  $k \leq l$ , then  $\begin{bmatrix} u \\ w \end{bmatrix} = \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \cdots \begin{bmatrix} a_k \\ b_k \end{bmatrix} \begin{bmatrix} \# \\ b_{k+1} \end{bmatrix} \cdots \begin{bmatrix} \# \\ b_l \end{bmatrix}$ , and otherwise  $\begin{bmatrix} u \\ w \end{bmatrix} = \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \cdots \begin{bmatrix} a_l \\ b_l \end{bmatrix} \begin{bmatrix} a_{l+1} \\ \# \end{bmatrix} \cdots \begin{bmatrix} a_k \\ \# \end{bmatrix}$ .

A *transducer* over  $\Sigma \times \Gamma$  is an NFA over  $\Sigma_{\#} \times \Gamma_{\#}$ . The binary relation recognised by a transducer  $T$  over  $\Sigma \times \Gamma$ , denoted  $R(T)$ , is the set of pairs  $(u, w) \in \Sigma^* \times \Gamma^*$  such that  $T$  accepts  $\begin{bmatrix} u \\ w \end{bmatrix}$ . A transducer is *deterministic* if it is a DFA. A relation is *regular* if it is recognised by some transducer. A transducer is *length-preserving* if it recognises a length-preserving relation.

**Complexity of operations on automata and transducers.** Given NFAs  $A_1, A_2$  over  $\Sigma$  with  $n_1$  and  $n_2$  states, DFAs  $B_1, B_2$  over  $\Sigma$  with  $m_1$  and  $m_2$  states, and transducers  $T_1$  over  $\Sigma \times \Gamma$  and  $T_2$  over  $\Gamma \times \Sigma$  with  $l_1$  and  $l_2$  states, the following facts are well known (see e.g. chapters 3 and 5 of [25]):

- there exist NFAs for  $L(A_1) \cup L(A_2)$ ,  $L(A_1) \cap L(A_2)$ , and  $\overline{L(A_1)}$  with at most  $n_1 + n_2$ ,  $n_1 n_2$ , and  $2^{n_1}$  states, respectively;
- there exist DFAs for  $L(B_1) \cup L(B_2)$ ,  $L(B_1) \cap L(B_2)$ , and  $\overline{L(B_1)}$  with at most  $m_1 m_2$ ,  $m_1 m_2$ , and  $m_1$  states, respectively;

- there exist NFAs for  $R(T_1)|_1$  and  $R(T_1)|_2$  and a transducer for  $R(T_1)^{-1}$  with at most  $l_1$  states;
- there exists a transducer for  $R(T_1) \circ R(T_2)$  with at most  $l_1 l_2$  states; and
- there exist NFAs for  $L(A_1) \circ R(T_1)$  and  $R(T_1) \circ L(A_2)$  with at most  $n_1 l_1$  and  $l_1 n_2$  states, respectively.

**Turing machines.** We fix the definition of Turing machine used in the paper.

A *Turing machine (TM)* is a tuple  $M = (Q, \Sigma, \Gamma, \square, \delta, q_0, q_f)$  where  $Q$  is the set of states,  $\Sigma$  is the input alphabet,  $\Gamma \supseteq \Sigma$  is the tape alphabet,  $\square \in \Gamma \setminus \Sigma$  is the blank symbol,  $q_0 \in Q$  is the initial state,  $q_f \in Q$  is the (only) final state, and  $\delta: (Q \setminus \{q_f\}) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$  is the transition function. A *nondeterministic Turing machine* is defined the same way as a Turing machine, with the difference that  $\delta$  is now a function from  $(Q \setminus \{q_f\}) \times \Gamma$  to  $2^{Q \times \Gamma \times \{L, R\}}$ . A *configuration* of a (nondeterministic) TM  $M$  is a triple  $(w_l, q, w_r)$  where  $w_l$  is the tape to the left of the head of  $M$ ,  $q$  is the current state of  $M$ , and  $w_r$  is the tape to the right of the head of  $M$ ; the first symbol of  $w_r$  is the symbol currently being read. The successor(s) of a configuration are defined as usual. A configuration  $(w_l, q, w_r)$  is *accepting* if  $q = q_f$ . A *run* of a (nondeterministic) TM  $M$  on an input  $w$  is either a sequence  $(c_i)_{i \in \mathbb{N}_0}$  of configurations of  $M$  where  $c_0 = (\varepsilon, q_0, w)$  and  $c_i$  is a successor of  $c_{i-1}$ , or a tuple  $(c_0, \dots, c_n)$  where  $c_0 = (\varepsilon, q_0, w)$ ,  $c_i$  is a successor of  $c_{i-1}$ , and  $c_n$  is accepting. In the latter case, the run is called *accepting*.  $M$  *accepts* the input  $w$  if there exists an accepting run of  $M$  on  $w$ .

► **Remark 2.1.** Note that Turing machines are (unless specified otherwise) deterministic, have no rejecting state, and the transition function is total, i.e. every non-accepting configuration has a successor. In particular, a Turing machine halts iff it accepts.

**The arithmetical and analytical hierarchies.** We briefly recall the definition of the first levels of the arithmetical and analytical hierarchies (see e.g. [42], part IV).  $\Sigma_1^0$  is the set of all semi-decidable problems.  $\Sigma_1^1$  is the set of sets of the form  $\{n \in \mathbb{N} \mid \exists \varphi_1, \dots, \varphi_k : f(n, \varphi_1, \dots, \varphi_k)\}$  where the  $\varphi_i$  range over functions from  $\mathbb{N}$  to  $\mathbb{N}$  and  $f$  is an arithmetic formula with arbitrary quantification over natural numbers.  $\Pi_1^0$  and  $\Pi_1^1$  are the sets of problems whose complement is in  $\Sigma_1^0$  and  $\Sigma_1^1$ , respectively.  $\Sigma_1^1$ -hard and  $\Pi_1^1$ -hard problems are sometimes called *highly undecidable*.

## 2.1 Regular transition systems

We recall standard notions about regular transition systems and fix some notations. Then we recall how to use regular transition systems to simulate Turing machines.

A *transition system* is a pair  $\mathcal{S} = (\mathcal{C}, \Delta)$  where  $\mathcal{C}$  is the set of all possible *configurations* of the system, and  $\Delta \subseteq \mathcal{C} \times \mathcal{C}$  is a *transition relation*. The *reachability relation*  $Reach$  is the reflexive and transitive closure of  $\Delta$ . Observe that, by our definition of post-set,  $\Delta(\mathcal{C})$  and  $Reach(\mathcal{C})$  are the sets of configurations reachable in one step and in arbitrarily many steps from  $\mathcal{C}$ , respectively.

Regular transition systems are transition systems where configurations are represented by words and the transition relation is regular. We also define regular transition systems to contain a set of initial configurations which all runs start from.

► **Definition 2.2.** A regular transition system (RTS) over an alphabet  $\Sigma$  is a pair  $\mathcal{S} = (\mathcal{I}, \Delta)$  where  $\Sigma$  is an alphabet, the set of configurations is  $\mathcal{C} = \Sigma^*$ ,  $\mathcal{I} \subseteq \mathcal{C}$  is a regular set of initial configurations and  $\Delta \subseteq \mathcal{C} \times \mathcal{C}$  is a regular transition relation.  $\mathcal{S}$  is called *length-preserving* if  $\Delta$  is length-preserving. A configuration  $c \in \mathcal{C}$  is called *terminating* if it has no successor, i.e.

if  $\Delta(c) = \emptyset$ . We denote the set of terminating configurations by  $T$ . A run of  $\mathcal{S}$  is either a sequence  $(c_i)_{i \in \mathbb{N}_0}$  where  $c_0 \in \mathcal{I}$  and  $\forall i \in \mathbb{N} : (c_{i-1}, c_i) \in \Delta$  or a tuple  $(c_0, \dots, c_n)$  where  $c_0 \in \mathcal{I}$ ,  $\forall i \in [n] : (c_{i-1}, c_i) \in \Delta$  and  $c_n \in T$ . In the latter case, the run is called terminating.

► **Example 2.3.** We give two small examples of RTSs loosely inspired by Herman’s randomized protocol for self-stabilization. We model an array of cells, each of which either holds a token ( $\bullet$ ) or not ( $\circ$ ). The alphabet is  $\{\langle, \bullet, \circ, \rangle\}$  and the initial configurations are  $\langle (\bullet + \circ)^* \bullet (\bullet + \circ)^* \rangle$ , where  $\langle$  and  $\rangle$  mark the two ends of the array. A transition moves a token to a neighbour cell, “swallowing” its token if the cell is not empty. The language of the transducer is

$$\left[ \langle \right] [x]_* \left( \left[ \begin{smallmatrix} \bullet \\ \circ \end{smallmatrix} \right] \left[ \begin{smallmatrix} \circ \\ \bullet \end{smallmatrix} \right] + \left[ \begin{smallmatrix} \circ \\ \bullet \end{smallmatrix} \right] \left[ \begin{smallmatrix} \bullet \\ \circ \end{smallmatrix} \right] + \left[ \begin{smallmatrix} \bullet \\ \bullet \end{smallmatrix} \right] \left[ \begin{smallmatrix} \bullet \\ \circ \end{smallmatrix} \right] + \left[ \begin{smallmatrix} \bullet \\ \circ \end{smallmatrix} \right] \left[ \begin{smallmatrix} \bullet \\ \bullet \end{smallmatrix} \right] \right) [x]_* \left[ \rangle \right]$$

where  $[x]_*$  is an abbreviation for  $\left[ \begin{smallmatrix} \bullet \\ \bullet \end{smallmatrix} \right] + \left[ \begin{smallmatrix} \circ \\ \circ \end{smallmatrix} \right]$ . The RTS is length-preserving. Intuitively, it satisfies that the set  $\langle \circ^* \bullet \circ^* \rangle$  of configurations with exactly one token is almost surely reachable and also almost surely recurrently reachable, for any probability distribution assigning non-zero probability to each transition.

Consider now another RTS in which the array can additionally grow or shrink on the right end. We can model this by adding to the language of the transducer the transitions

$$\left[ \langle \right] [x]_* \left[ \begin{smallmatrix} \circ \\ \circ \end{smallmatrix} \right] \left[ \begin{smallmatrix} \bullet \\ \# \end{smallmatrix} \right] + \left[ \langle \right] [x]_* \left[ \begin{smallmatrix} \circ \\ \circ \end{smallmatrix} \right] \left[ \begin{smallmatrix} \rangle \\ \# \end{smallmatrix} \right].$$

The new RTS is no longer length-preserving, and the property above no longer holds for every probability distribution.

**Simulation of Turing machines by regular transition systems.** In our undecidability proofs we make use of the fact that RTSs can simulate Turing machines. More precisely, a configuration  $(w_l, q, w_r)$  of a TM  $M$  can be encoded as a word  $w_l q w_r$ . The transition function  $\delta$  of  $M$  can then be simulated by  $\Delta$  as only the letters at positions next to the head can change. If the RTS is length-preserving, one can set  $\mathcal{I} := \{q_0 w\} \{\square\}^*$  where  $w$  is the input to  $M$  and terminate if the head of  $M$  gets to the last position of the configuration of the RTS; in this case,  $M$  accepts  $w$  iff there exists a run of the RTS (starting in a configuration of a high enough length) which reaches a configuration containing  $q_f$ .

### 3 Decision problems

It was proved in [43] that the problem whether an RTS with a regular reachability relation has a run visiting a set of configurations given by an NFA infinitely often is in P for both length-preserving and general transition relations. Extending this result, we analyse the complexities of infinite reachability problems in an RTS with a regular reachability relation.

We start with a lemma showing that a basic problem about RTSs is  $\Sigma_1^1$ -complete in the general case and  $\Sigma_1^0$ -complete in the length-preserving case. The proof of the length-preserving case is very simple, while the general case requires to use a clever result by Harel, Pnueli, and Stavi [33].

► **Lemma 3.1.** *Deciding whether an RTS  $\mathcal{S}$  has an infinite run is  $\Sigma_1^1$ -complete. If  $\mathcal{S}$  is length-preserving, the problem is  $\Sigma_1^0$ -complete.*

**Proof.** In the length-preserving case, only finitely many configurations are reachable from any configuration; hence, if there exists an infinite run, there exists a path from some  $c_0 \in \mathcal{I}$  to some  $c \in \mathcal{C}$  which visits  $c$  twice. This path is a checkable certificate, proving membership in  $\Sigma_1^0$ . For hardness, we reduce from the problem whether a Turing machine  $M$  accepts the

empty input. First, we construct a TM  $M'$  which behaves like  $M$ , but swaps acceptance and looping:  $M'$  simulates  $M$  while writing down all visited configurations; if  $M'$  detects that  $M$  visits a configuration for the second time,  $M'$  accepts; if  $M'$  detects that  $M$  accepts,  $M'$  goes in an infinite loop. We then simulate  $M'$  by  $\mathcal{S}$ ;  $\mathcal{S}$  has an infinite run iff  $M'$  goes in an infinite loop iff  $M$  accepts the empty input.

We now show that the problem is  $\Sigma_1^1$ -complete in the general case by reducing from and to the following problem: Given a nondeterministic Turing machine  $M$ , does  $M$  have an infinite run on the empty input which visits its initial state  $q_0$  infinitely often? This problem is known to be  $\Sigma_1^1$ -complete [33, Proposition 5.1].

Given  $\mathcal{S}$ ,  $M$  simulates a run of  $\mathcal{S}$  by nondeterministically writing down an initial configuration, repeatedly guessing transitions in the transducer for  $\Delta$  before nondeterministically stopping in a final state of the transducer, writing down the successor, and repeating the process.  $M$  never visits  $q_0$  during the guessing process (otherwise  $M$  might never stop guessing an infinitely long successor) and every time a successor is guessed,  $M$  visits  $q_0$ .

For the other direction,  $\mathcal{S}$  can simulate  $M$  with a counter which indicates when the next time  $M$  visits  $q_0$  will be. Every time  $M$  visits  $q_0$ ,  $\mathcal{S}$  nondeterministically sets the counter to a number  $k$ , and every time  $M$  does a transition without visiting  $q_0$ ,  $k$  is decreased by 1. If  $k$  reaches 0,  $\mathcal{S}$  terminates. (Note that  $k$  can be set arbitrarily high with only one transition.)  $\blacktriangleleft$

In the following problems, we assume that the input consists of an NFA for  $\mathcal{I}$ , a transducer for  $\Delta$ , and, where applicable, a transducer for  $Reach$  and an NFA for  $L$ .

► **Theorem 3.2.** *Sure reachability and sure recurrent reachability are  $\Pi_1^1$ -complete. If  $\mathcal{S}$  is length-preserving, then they are  $\Pi_1^0$ -complete.*

**Proof.** We prove that the complements of both problems are equivalent to the problem whether an RTS has an infinite run, both in the length-preserving and in the general case. We do that by reducing these four problems to each other in a cycle:

- (a) Given  $\mathcal{S}$ ,  $L$ , and  $Reach$ , does there exist a run which only visits  $L$  finitely often?
- (b) Given  $\mathcal{S}$ ,  $L$ , and  $Reach$ , does there exist a run which never visits  $L$ ?
- (c) Given  $\mathcal{S}$ , does there exist an infinite run?
- (d) Given  $\mathcal{S}$  where all configurations are initial, does there exist an infinite run?

Here, (a) is the complement of the sure recurrent reachability problem, (b) is the complement of the sure reachability problem, and (c) is the problem from Lemma 3.1.

(a)  $\leq$  (b): A run visits  $L$  finitely often if and only if it visits a configuration from which it never visits  $L$  again. Hence the reduction just changes the initial configurations to  $Reach(\mathcal{I})$  while leaving everything else the same.

(b)  $\leq$  (c): Given  $\mathcal{S} = (\mathcal{I}, \Delta)$  and  $L$ , define  $\mathcal{S}' := (\mathcal{I} \setminus L, \Delta \cap (L \times \bar{L}))$  and add self-loops to all terminating configurations.

(c)  $\leq$  (d): Given  $\mathcal{S} = (\mathcal{I}, \Delta)$ , we construct an RTS  $\mathcal{S}'$  which has an infinite run starting from any configuration iff  $\mathcal{S}$  has an infinite run. The idea is that  $\mathcal{S}'$  checks an infinite run of  $\mathcal{S}$  for correctness. The configurations of  $\mathcal{S}'$  are of the form  $c_0 \# c_1 \# \dots \# c_n$  where  $c_0, \dots, c_n$  are configurations of  $\mathcal{S}$ .

In the length-preserving case,  $\mathcal{S}'$  checks (using multiple transitions) that  $c_0 \in \mathcal{I}$ , that  $(c_{i-1}, c_i) \in \Delta$  for all  $i \in [n]$ , and that there exists an  $i < n$  such that  $c_i = c_n$ .  $\mathcal{S}'$  can do that by e.g. working like a Turing machine and marking the already checked letters. If any checks

fail,  $\mathcal{S}'$  terminates; if all checks succeed,  $\mathcal{S}'$  unmarks all letters and repeats the process.  $\mathcal{S}$  has an infinite run iff  $\mathcal{S}$  has a run which loops, i.e. visits the same configuration twice, and  $\mathcal{S}'$  can check that run forever, which also results in an infinite run. For the converse, note that  $\mathcal{S}'$  can only have an infinite run if all checks succeed.

The non-length-preserving case is similar. Instead of checking for a cycle,  $\mathcal{S}'$  checks that the current prefix  $c_0\#c_1\#\dots\#c_n$  of a run of  $\mathcal{S}$  is valid, and after all checks succeed, nondeterministically adds a configuration  $c_{n+1}$  and restarts the process. It is clear that  $\mathcal{S}'$  has an infinite run iff  $\mathcal{S}$  does.

(d)  $\leq$  (a): Given  $\mathcal{S} = (\mathcal{I}, \Delta)$  with  $\mathcal{I} = \mathcal{C}$ , we construct an RTS  $\mathcal{S}' = (\mathcal{I}', \Delta')$  and an NFA for  $L$  such that  $\mathcal{S}'$  has a run which only visits  $L$  finitely often iff  $\mathcal{S}$  has an infinite run. We let  $\mathcal{S}'$  behave the same way as  $\mathcal{S}$ , but add a special configuration  $s$  (in the length-preserving case, we add infinitely many such configurations, one for each length), set  $L = \mathcal{I}' = \{s\}$  and add the transitions  $(s, c)$  and  $(c, s)$  for all  $c \in \mathcal{C}$  to  $\Delta'$ . Then the reachability relation of  $\mathcal{S}'$  is regular as all configurations (or all configurations of the same length) can reach each other. If  $\mathcal{S}$  has an infinite run, then  $\mathcal{S}'$  has the same infinite run which does not visit  $L$ ; if  $\mathcal{S}$  does not have an infinite run, then every run of  $\mathcal{S}'$  must visit  $L$  again and again.  $\blacktriangleleft$

► **Remark 3.3.** The undecidability of the sure reachability problem for arbitrary transducers can be immediately deduced from well-known results. Theorem 3.3 of [23] proves that the reachability relation of Basic Parallel Processes (BPPs) [17, 16] is expressible in Presburger arithmetic, and so regular under standard encodings of tuples of natural numbers as words [32, 26]. So BPPs are a special case of RTSs in which *Reach* can be encoded as a transducer. Further, it was shown in [29, 24] that sure reachability for BPPs is undecidable. The novelty of Theorem 3.2 is to show that the problem remains undecidable in the length-preserving case (and to give a simpler proof for arbitrary transducers). Observe that the length-preserving case requires a novel argument: indeed, since sure reachability is  $\Pi_1^1$ -complete in the general case, it cannot be recursively reduced to sure reachability for length-preserving transducers, which is only  $\Pi_1^0$ -complete.

► **Theorem 3.4.** *Deadlock-freedom is PSPACE-complete.*

**Proof.** The set of terminating configurations is  $\overline{\Delta^{-1}(\mathcal{C})}$ , and thus there exists a terminating run iff  $\text{Reach}(\mathcal{I}) \cap \overline{\Delta^{-1}(\mathcal{C})} \neq \emptyset$ . This is decidable in nondeterministic polynomial space by e.g. guessing a configuration  $c$  letter by letter and checking on the fly that  $c \in \text{Reach}(\mathcal{I})$  and  $c \notin \Delta^{-1}(\mathcal{C})$ ; indeed, this only requires enough memory to store a state of the NFA for  $\text{Reach}(\mathcal{I})$  and a set of states of the NFA for  $\Delta^{-1}(\mathcal{C})$ . For hardness, we can reduce from the universality problem: Given an NFA  $A$ , setting  $\mathcal{I} := \mathcal{C}$  and  $\Delta := L(A) \times \mathcal{C}$  makes the problem equivalent to deciding whether  $A$  is universal.  $\blacktriangleleft$

## 4 Almost sure properties

In this section we present our results on the decidability and complexity of almost sure properties. We see the RTS  $\mathcal{S}$  as a Markov chain, i.e. we have a probability measure  $\mathbb{P}: \Delta \rightarrow (0, 1]$  which assigns a positive probability to every transition  $(c, c') \in \Delta$  and satisfies  $\forall c \in \mathcal{C} : \sum_{c' \in \Delta(c)} \mathbb{P}(c, c') = 1$ . This induces as usual a probability space  $(\text{run}(c), \mathbb{F}, \mathcal{P})$  for every configuration  $c$ , where  $\text{run}(c)$  is the set of runs starting at  $c$ ,  $\mathbb{F}$  is the  $\sigma$ -field generated by all basic cylinders  $\text{run}(w)$  where  $w$  is a finite path starting at  $c$ , and  $\mathcal{P}: \mathbb{F} \rightarrow [0, 1]$  is the unique probability function such that  $\mathcal{P}(\text{run}(w)) = \prod_{i=1}^m \mathbb{P}(c_{i-1}, c_i)$  where  $w = (c_0, \dots, c_m)$ .

In the length-preserving case, a.s. reachability, recurrent reachability and termination only depend on the topology of the Markov chain  $\mathcal{S}$ , and not on the numerical values of the probabilities of its transitions. This fact is well-known for finite Markov chains. To show that it also holds for  $\mathcal{S}$ , which can be infinite, observe that, since the number of configurations of a given length is finite, every configuration can only reach finitely many configurations, and so  $\mathcal{S}$  is the disjoint union of a finite or countably infinite family of finite Markov chains. It follows that our almost-sure properties hold iff they hold for every Markov chain in the family, and therefore do not depend on the numerical values either.

In the general case, the property depends on the numerical values. For example, consider a random walk on  $\{a\}^*$  with transitions  $\Delta = \{(a^n, a^{n+1}), (a^{n+1}, a^n) \mid n \in \mathbb{N}_0\}$  and assume that  $p := \mathbb{P}(a^n, a^{n+1})$  is the same for all  $n$ . The probability that a run starting at  $a$  eventually reaches  $\varepsilon$  is 1 if and only if  $p \leq 0.5$ .

Again, we assume that the input consists of an NFA for  $\mathcal{I}$ , a transducer for  $\Delta$ , and, where applicable, a transducer for  $Reach$  and an NFA for  $L$ .

## 4.1 The length-preserving case

We make use of the fact that a run of  $\mathcal{S}$  almost surely eventually visits a bottom strongly connected component (bSCC) of  $\mathcal{S}$ . Further, once in a bSCC, the run either terminates (if the bSCC is trivial, i.e. consists of a single terminating configuration) or stays in that bSCC forever, visiting each configuration of the bSCC infinitely often almost surely.

► **Proposition 4.1.** *Almost sure reachability is  $\Pi_1^0$ -complete for length-preserving RTSs.*

**Proof.** If there exists an initial configuration  $c_0 \in \mathcal{I}$  from which reaching  $L$  has probability less than 1, then there exists a bSCC reachable from  $c_0$  such that neither the bSCC nor the path from  $c_0$  to the bSCC intersects  $L$ . In other words, there exists a path  $(c_0, \dots, c_n)$  such that  $c_0 \in \mathcal{I}$ ,  $c_i \notin L$  for all  $i$ , and  $c_n$  is in a bSCC which does not intersect  $L$ , i.e.  $Reach(c_n) \subseteq Reach^{-1}(c_n)$  and  $Reach(c_n) \cap L = \emptyset$ . Given such a path, these conditions can be checked, proving semi-decidability of the complement and thus proving that almost sure reachability is in  $\Pi_1^0$ .

For hardness, we reduce from the non-emptiness problem for Turing machines. Given a TM  $M$ , we define  $\mathcal{I}$  to be the set of input configurations of  $M$  with any number of blank symbols, that is, the set of all words  $wv$  where  $w$  is the encoding of an input configuration of  $M$ , and  $v \in \{\square\}^*$  is a word of blanks. Further, we let  $\Delta$  simulate the transitions of  $M$ . Moreover, for every length, we add two special configurations  $s, t$  to the RTS and add the following transitions to  $\Delta$ :  $(c, s)$  and  $(s, c)$  for every configuration  $c$  of  $M$ ,  $(a, t)$  for every accepting configuration  $a$  of  $M$ , and  $(s, t)$ . Then  $Reach = ((C \cup \{s\}) \times (C \cup \{s, t\})) \cup \{(t, t)\}$  where  $C$  is the set of configurations of  $M$ , so  $Reach$  is regular. Let  $L = \{s\}$  (of all lengths). If  $M$  has an input  $w$  which it accepts, then there exists a run of  $\mathcal{S}$  starting from  $q_0w$  (with enough blank symbols) which reaches an accepting configuration of  $M$  and then goes to  $t$ , never visiting  $s$ . This run has positive probability as the number of steps until  $t$  is reached is finite, and thus the probability of visiting  $L$  is not 1. Conversely, if  $M$  does not have an input which it accepts, then no run can reach  $t$  without visiting  $s$  first, and a transition to  $s$  will occur eventually almost surely, i.e. the probability of visiting  $L$  is 1. ◀

For almost sure recurrent reachability, we introduce the following characterisation, which will also be useful in later sections.

► **Lemma 4.2.**  *$\mathcal{S}$  reaches  $L$  infinitely often almost surely iff  $\text{Reach}(\mathcal{I}) \subseteq \overline{T} \cap \text{Reach}^{-1}(L)$ .*

**Proof.** Every run of  $\mathcal{S}$  eventually visits a bSCC almost surely, and thus every infinite run visits  $L$  infinitely often iff every reachable bSCC is non-trivial and contains a configuration in  $L$ . This is the case iff no terminating configuration is reachable and  $L$  can be reached from every reachable configuration, or formally,  $\text{Reach}(\mathcal{I}) \subseteq \overline{T} \cap \text{Reach}^{-1}(L)$ . ◀

► **Proposition 4.3.** *Almost sure recurrent reachability is PSPACE-complete for length-preserving RTSs.*

**Proof.** By Lemma 4.2, it suffices to show that deciding whether  $\text{Reach}(\mathcal{I}) \subseteq \overline{T} \cap \text{Reach}^{-1}(L)$  is PSPACE-complete. One can decide this in nondeterministic polynomial space by e.g. guessing a configuration  $c$  letter by letter and checking on the fly that  $c \in \text{Reach}(\mathcal{I})$ ,  $c \in \overline{T}$ , and  $c \notin \text{Reach}^{-1}(L)$ . For that, one can run  $c$  on the corresponding NFAs by memorizing the current set of states after each letter of  $c$ , and checking whether that set contains a final state of the NFA at the end. Since all involved NFAs have polynomial size (note that  $\overline{T} = \Delta^{-1}(\mathcal{C})$ ), this algorithm takes polynomial space, and membership in PSPACE follows from NPSPACE = PSPACE.

For hardness, we can reduce from the NFA subset problem, i.e., whether  $L(A) \subseteq L(B)$  for given NFAs  $A, B$ : By setting  $\mathcal{I} := L(A)$ ,  $\Delta := \{(c, c) \mid c \in \mathcal{C}\}$ , and  $L := L(B)$ , almost sure recurrent reachability holds iff  $L(A) \subseteq L(B)$ . ◀

We now consider almost sure termination, the property for which the proof is most involved. Again, we first prove a characterisation. We say that  $\mathcal{S}$  *terminates almost surely* if for every initial configuration  $c$ , the set of runs starting at  $c$  that end in a terminating configuration has probability 1.

► **Lemma 4.4.**  *$\mathcal{S}$  terminates almost surely iff  $\text{Reach}(\mathcal{I}) \setminus \text{Reach}^{-1}(T) = \emptyset$ .*

**Proof.**  $\text{Reach}(\mathcal{I}) \setminus \text{Reach}^{-1}(T)$  is the set of all reachable configurations  $c$  such that no terminating configuration can be reached from  $c$ . If such a  $c$  exists, then reaching it has a positive probability, and since the run cannot terminate from  $c$ ,  $\mathcal{S}$  terminates with probability less than 1. For the converse, if terminating configurations can be reached from any reachable configuration, then no reachable non-trivial bSCCs exist, i.e. every reachable bSCC is a terminating configuration. Since a run of  $\mathcal{S}$  eventually reaches a bSCC almost surely, it follows that  $\mathcal{S}$  terminates almost surely. ◀

► **Proposition 4.5.** *Almost sure termination is EXPSPACE-complete for length-preserving RTSs. If an NFA for the set of terminating configurations is provided in the input, almost sure termination becomes PSPACE-complete.*

**Proof.** From the characterisation of Lemma 4.4, it is easy to show that the problem is PSPACE-complete in the case where an NFA for  $T$  is given in the input, one can e.g. reduce from and to the NFA subset problem. For membership in EXPSPACE, observe that  $T = \overline{\Delta^{-1}(\mathcal{C})}$  has an NFA of exponential size in the input. One can thus build a polynomial NFA for  $\text{Reach}(\mathcal{I})$  and an exponential NFA for  $\text{Reach}^{-1}(T)$ , guess a configuration  $c$  letter by letter and check that  $c \in \text{Reach}(\mathcal{I})$  and  $c \notin \text{Reach}^{-1}(T)$ . This is a nondeterministic algorithm using exponential space, and membership follows from NEXPSPACE = EXPSPACE.

For EXPSPACE-hardness, we reduce from the problem whether an exponentially space-bounded Turing machine  $M = (Q, \Sigma', \Gamma, \delta, \square, q_0, q_f)$  accepts the empty input. The configurations of our RTS  $\mathcal{S}$  are going to be encodings of runs of  $M$ , and a run of  $\mathcal{S}$  cannot terminate if and only if it starts with the encoding of the accepting run of  $M$  on the empty input. We encode a run of  $M$  as a word consisting of encodings of configurations of  $M$  separated by #.

Let  $n := |M|$  and let  $p_1, \dots, p_n$  be the first  $n$  prime numbers. Then  $\sum_{i=1}^n p_i$  is polynomial in  $n$  by the prime number theorem while  $m := \prod_{i=1}^n p_i$  is exponential in  $n$ ; we can thus assume that the run of  $M$  on the empty input uses only the first  $m$  tape cells. An NFA recognizing the language of words not divisible by  $p_i$  only needs  $p_i$  states. Thus, by putting  $n$  NFAs side by side, each of which only accepts words of length not divisible by  $p_i$ , one can construct an NFA of polynomial size in  $n$  which only accepts words of length not divisible by  $m$ . This fact will be used in our reduction.

Let  $\Sigma_1 := Q \cup \Gamma \cup \{\#\}$  and  $\Sigma_2 := \{x' \mid x \in \Sigma_1\}$ . We call symbols in  $\Sigma_1$  *unmarked* and symbols in  $\Sigma_2$  *marked*. We denote the symbol in position  $i$  of a configuration of  $\mathcal{S}$  by  $x_i$ . Let  $\Sigma := \Sigma_1 \cup \Sigma_2$ ,  $\mathcal{C} := \Sigma^*$  and  $\mathcal{I} := \{\#q_0\}\{\square\}^*\{\#\}\Sigma_1^*\{q_f\}$ , i.e. the first configuration of  $M$  in an initial configuration of  $\mathcal{S}$  is the empty input to  $M$ , and the last symbol is the accepting state of  $M$ .  $\Delta$  has three types of transitions:  $\Delta_{mark}$ , which marks some symbols, and  $\Delta_{unmark}$ , which unmarks all symbols, and  $\Delta_{end}$ , which is used to terminate certain runs. Specifically,  $\Delta_{mark}$  nondeterministically chooses a position  $i$  and a distance  $k > 2$  and marks the symbols at positions  $i - 1, i, i + 1, i + 2$ , and  $i + k$ . For this transition to be executed, three conditions are required:

- No symbol is marked.
- There is exactly one  $\#$  in the subword  $x_{i+1} \cdots x_{i+k}$ .
- If  $x_{i-1}x_i x_{i+1}x_{i+2}$  is part of a configuration of  $M$ , then  $x_k$  is *not* the symbol appearing in place of  $x_i$  in the next configuration of  $M$ . In particular, if  $x_i = \#$ , then  $x_k \neq \#$ .

The last point can be intuitively described as the encoding of the run having a “mistake” at position  $x_k$ . This description only fits if  $\Delta_{mark}$  chooses the correct distance; for other distances, even the correct run has a mistake. It will be relevant, however, that the encoding of the correct run does not have a mistake if distance  $m$  is chosen.

$\Delta_{unmark}$  takes as input a configuration with exactly five marked symbols, checks that the distance between the second and the fifth is not a multiple of  $m$ , and unmarks all symbols. If the distance is a multiple of  $m$ , the transition cannot occur. As explained above, this can be done with a transducer of polynomial size in  $n$ .

$\Delta_{end}$  takes as input a configuration where the distance between any two  $\#$ s is not a multiple of  $m$ , and replaces all symbols by  $\#'$ , after which the run terminates.

Observe that transitions only mark and unmark symbols and never change them. Thus every reachable configuration can be reached in at most two steps, and  $Reach = \{(c, c) \mid c \in \mathcal{C}\} \cup \Delta \cup \Delta^2$  is regular and has a transducer of polynomial size in  $n$ . Furthermore, a run of  $\mathcal{S}$  can only terminate if either the initial configuration has  $\#$ s at a distance not divisible by  $m$ , or the run reaches a configuration where the distance between the marked symbols  $x_i$  and  $x_{i+k}$  chosen by  $\Delta_{mark}$  is a multiple of  $m$ , i.e.  $\Delta_{mark}$  finds a mistake at a distance divisible by  $m$ .

If  $M$  accepts the empty input, a run of  $\mathcal{S}$  can start with the encoding of the accepting run of  $M$ . Since the run only uses the first  $m$  tape cells, there exists such an encoding where the distance between two consecutive  $\#$  is always  $m$ ; therefore, the run of  $\mathcal{S}$  can never terminate with  $\Delta_{end}$ . Moreover,  $\Delta_{mark}$  can only choose distances  $k \leq 2m - 1$  as otherwise there will be two  $\#$  between position  $i + 1$  and  $i + k$ . If  $k \neq m$ , then  $\Delta_{unmark}$  will unmark the positions in the next transition; and  $k = m$  cannot be chosen because the correct run has no “mistakes” for  $\Delta_{mark}$  to find. Hence such a run cannot terminate.

If  $M$  does not accept the empty input, then there exists no run of  $M$  starting with the empty input and ending with an accepting configuration. That is, every encoding of such a run in  $\mathcal{S}$  must have a mistake, i.e. there must exist symbols  $x_{i-1}, x_i, x_{i+1}, x_{i+2}, x_{i+lm}$  where  $x_{i+lm}$  does not result from  $x_{i-1}x_i x_{i+1}x_{i+2}$ , where  $l \in \mathbb{N}$  s.t.  $lm$  is the distance between

two consecutive  $\#$ . Such a mistake will almost surely be eventually “found” by  $\Delta_{mark}$ , and the run will terminate. If the distance between two consecutive  $\#$  is not a multiple of  $m$ , then the run can terminate with  $\Delta_{end}$ . In both cases, the run will almost surely eventually terminate.  $\blacktriangleleft$

## 4.2 The general case

In the general case, not every run ends up in a bSCC almost surely; bSCCs do not even necessarily exist. Furthermore, the probabilities of the transitions can affect the answer, which introduces the question of how  $\mathbb{P}$  is provided in the input, and how to deal with configurations with infinitely many successors. We show that the problems are undecidable even in the special case where every configuration has finitely many successors and  $\mathbb{P}$  is the uniform probability measure, i.e.  $\forall(c, c') \in \Delta : \mathbb{P}(c, c') = \frac{1}{|\Delta(c)|}$ .

We prove undecidability of the almost sure termination problem; the other problems can be easily reduced to it. For that, we reduce from the problem whether a deterministic Turing machine  $M$  loops on the empty input, i.e. reaches the same configuration twice. A run of our RTS  $\mathcal{S}$  will simulate the run of  $M$  on the empty input with high probability and have a low probability (the longer the configuration of  $\mathcal{S}$ , the lower) to jump to a special configuration  $s$ . From  $s$ , the run can terminate, restart the simulation of the run of  $M$  on the empty input, or, with low probability, jump to any configuration of  $M$  (this is done to ensure that *Reach* is regular). If  $M$  halts or the head goes too far to the right,  $\mathcal{S}$  restarts the simulation of  $M$  on the empty input while increasing the length by 1. This way, if  $M$  loops on the empty input, the run will eventually jump to  $s$  and have a chance to terminate. If  $M$  does not loop on the empty input, but instead halts or the head goes arbitrarily far, then, with high probability, the configuration of  $\mathcal{S}$  will become longer and longer, and the probability of reaching  $s$  will decrease, leading to termination with probability  $< 1$ . For the full proof, see the full version [30].

► **Proposition 4.6.** *Almost sure termination is undecidable.*

► **Corollary 4.7.** *Almost sure reachability and almost sure recurrent reachability are undecidable.*

**Proof.** One can reduce from almost sure termination by adding a new configuration  $a$ , setting  $L := \{a\}$ , and adding transitions  $(a, a)$  and  $(t, a)$  for every  $t \in T$ .  $\blacktriangleleft$

## 5 Sure and almost-sure properties in regular abstraction frameworks

As mentioned in the introduction, while some formalisms that can be modeled as RTSs have an effectively computable regular reachability relation, many others do not, including lossy channel systems, Vector Addition Systems and their many extensions, broadcast protocols, and population protocols. On the other hand, there exist different techniques to effectively compute a regular *overapproximation* of *Reach*, which in this section we generically call *potential reachability* and denote *PReach*. We consider the particular case of *regular abstraction frameworks*, a recent technique for the computation of *PReach* [19] introduced by the authors and colleagues. The technique automatically computes an overapproximation *PReach* for any *regular abstraction* representable – in a certain technical sense – by a transducer, and the computational complexity of the automatic procedure is precisely known (see Section 5.1 below).

The section is structured as follows. Section 5.1 recalls the main notions and results of [19]. Sections 5.2 and 5.3 study sure and almost-sure properties, respectively.

## 5.1 Regular abstraction frameworks

We briefly recall the main idea behind regular abstraction frameworks. Recall that the configurations of an RTS  $\mathcal{S}$  are finite words over an alphabet  $\Sigma$ . A regular abstraction framework introduces a second alphabet  $\Gamma$  and a deterministic transducer  $\mathcal{V}$  over  $\Gamma \times \Sigma$ , called an *interpretation*. Words over  $\Gamma$  are called *constraints*. Intuitively, words over  $\Gamma$  are “identifiers” for sets of configurations, and  $\mathcal{V}$  “interprets” an identifier  $w \in \Gamma^*$  as the set of configurations  $\mathcal{V}(w) := \{w\} \circ \mathcal{R}(\mathcal{V})$ . A configuration  $c$  satisfies a constraint  $w$  if  $c \in \mathcal{V}(w)$ .

A constraint  $w \in \Gamma^*$  is *inductive* if  $\mathcal{V}(w) \circ \Delta \subseteq \mathcal{V}(w)$ . Given an inductive constraint  $w \in \Gamma^*$ , we have  $\mathcal{V}(w) \circ \text{Reach} \subseteq \mathcal{V}(w)$ , i.e.,  $\mathcal{V}(w)$  is closed under the reachability relation. An inductive constraint  $w$  *separates* two configurations  $c, c'$  if  $c \in \mathcal{V}(w)$  and  $c' \notin \mathcal{V}(w)$ . Observe that if  $w$  separates  $(c, c')$ , then  $c'$  is not reachable from  $c$ . This leads to the definition of the *potential reachability relation induced by  $\mathcal{V}$* , denoted  $P\text{Reach}$ , as the set of all pairs  $(c, c')$  that are *not* separated by any inductive constraint. Formally:

$$P\text{Reach} = \{(c, c') \in \mathcal{C} \times \mathcal{C} \mid \forall w \in \Gamma^* : w \text{ is not inductive or does not separate } (c, c')\}$$

The following theorem was shown in [19] and holds for length-preserving as well as general RTSs.

► **Theorem 5.1.** *Let  $\mathcal{S} = (\mathcal{I}, \Delta)$  be an RTS and let  $\mathcal{V}$  be an interpretation.*

1.  *$P\text{Reach}$  is reflexive, transitive, and regular. Further, one can compute in exponential time a transducer recognizing  $P\text{Reach}$  with at most  $n_{\mathcal{V}}^2 \cdot 2^{n_{\Delta} \cdot n_{\mathcal{V}}^2}$  states.*
2. *Given an NFA for  $L$ , deciding whether some configuration in  $L$  is potentially reachable from some initial configuration (formally: whether  $P\text{Reach}(\mathcal{I}) \cap L = \emptyset$  holds) is EXPSPACE-complete.*

The problem of Theorem 5.1.2 is called **ABSTRACT SAFETY** in [19]. We state a corollary.

► **Proposition 5.2.** *Abstract deadlock-freedom is EXPSPACE-complete.*

**Proof.** Abstract deadlock-freedom is the problem whether  $P\text{Reach}(\mathcal{I}) \cap T = \emptyset$ . A non-deterministic algorithm can guess a configuration  $c \in \mathcal{C}$  and check that  $c \notin \overline{P\text{Reach}(\mathcal{I})}$  and  $c \notin \overline{T}$ . Since one can construct an exponential-sized NFA for  $\overline{P\text{Reach}(\mathcal{I})}$  and a polynomial-sized NFA for  $\overline{T}$ , this algorithm needs exponential space.

For hardness, we reduce from **ABSTRACT SAFETY**. Given an RTS  $\mathcal{S}$  and a set  $L$  of unsafe configurations, we construct  $\mathcal{S}'$  by adding a new configuration  $t$  (in the length-preserving case, adding such a configuration for each length), adding self-loops to all configurations except  $t$ , adding transitions  $(c, t)$  for every  $c \in L$ , and adding  $t$  to all constraints (i.e. for every constraint  $\varphi$ , we have  $(\varphi, t) \in \mathcal{V}$ ). This makes  $t$  the only terminating configuration while not affecting the inductivity of any constraints. It is easy to see that abstract safety holds for  $\mathcal{S}$  iff abstract deadlock-freedom holds for  $\mathcal{S}'$ . ◀

## 5.2 Sure properties

We define **ABSTRACT LIVENESS**, obtained by substituting “recurrent reachability” for “reachability” in **ABSTRACT SAFETY**.

► **Definition 5.3.** *Let  $\mathcal{S} = (\mathcal{I}, \Delta)$  be an RTS and let  $\mathcal{V}$  be an interpretation. A potential run of  $\mathcal{S}$  is a run of the RTS  $(\mathcal{I}, \Delta')$  with  $\Delta' := (P\text{Reach} \setminus \text{Id}) \cup \Delta$ . The abstract liveness problem is defined as follows:*

*ABSTRACT LIVENESS*

Given: RTS  $\mathcal{S} = (\mathcal{I}, \Delta)$ , interpretation  $\mathcal{V}$ , NFA  $A$  for  $L$ .

Decide: Does some potential run of  $\mathcal{S}$  visit  $L$  infinitely often?

► **Lemma 5.4.** *The reflexive transitive closure of  $\Delta'$  is  $PReach$ . The transitive closure of  $\Delta'$  is  $\Delta' \cup (PReach \setminus \text{Id})^2$ .*

**Proof.** See the full version [30]. ◀

The first statement of Lemma 5.4 shows that our definition of a potential run coincides with the definition of abstract safety in [19]: Abstract safety holds iff no potential run reaches  $L$ . The second statement will be used to characterise abstract liveness.

In the rest of the section, we show that *ABSTRACT LIVENESS* and abstract sure termination, which can be seen as a special case of *ABSTRACT LIVENESS*, are *EXPSPACE*-complete. We start with a lemma, similar to the starting point of To and Libkin in [43].

► **Lemma 5.5.** *Let  $\mathcal{S} = (\mathcal{I}, \Delta)$ ,  $\mathcal{V}$ ,  $A$  be an instance of *ABSTRACT LIVENESS*. Let  $\Delta'$  be as in Definition 5.3. There exists a potential run of  $\mathcal{S}$  that visits  $L(A)$  infinitely often if and only if one of these conditions hold:*

- (a) *There exist configurations  $c_0, c$  such that  $c_0 \in \mathcal{I}$ ,  $c \in L$ ,  $(c_0, c) \in PReach$ , and  $(c, c) \in \Delta \cup (PReach \setminus \text{Id})^2$ .*
- (b) *There exists a sequence  $(c_i)_{i \in \mathbb{N}_0}$  of pairwise distinct configurations such that  $c_0 \in \mathcal{I}$ ,  $\forall i \in \mathbb{N} : c_i \in L$ , and  $\forall i \in \mathbb{N}_0 : (c_i, c_{i+1}) \in PReach$ .*

**Proof.** We use Lemma 5.4. If (a) holds, then the infinite sequence  $c_0, c, c, \dots$  is a potential run. If (b) holds, then  $(c_i)_{i \in \mathbb{N}_0}$  is a potential run. For the converse, let  $(c_i)_{i \in \mathbb{N}_0}$  be a potential run that visits  $L$  infinitely often. If  $c_i = c_j \in L$  for some  $i < j$ , then  $(c_i, c_i) \in \Delta' \cup (PReach \setminus \text{Id})^2$ , and since  $(c_i, c_i) \in \text{Id}$ , (a) holds. Otherwise, removing all configurations not in  $L$  from the sequence yields a sequence of pairwise distinct configurations in  $L$ , and (b) holds. ◀

The sequence in condition (b) is called an *infinite directed clique*, see [12]. Note that (b) never holds for length-preserving RTSs. We prove that both (a) and (b) can be decided in *EXPSPACE*.

► **Lemma 5.6.** *Deciding (a) is in *EXPSPACE*.*

**Proof.** One can guess a configuration  $c \in \mathcal{C}$  letter by letter and check that  $c \in L$ ,  $c \notin \overline{PReach(\mathcal{I})}$  and  $(c, c) \in \Delta \cup (PReach \setminus \text{Id})^2$ .  $(c, c) \in (PReach \setminus \text{Id})^2$  is equivalent to there existing a  $c' \in \mathcal{C}$  such that  $c' \neq c$  and  $(c, c'), (c', c) \notin \overline{PReach}$ . Since an NFA for  $\overline{PReach(\mathcal{I})}$  can be built using exponential space (see Theorem 5.1), the algorithm amounts to guessing  $c, c'$  letter by letter (in parallel) and running words on NFAs of at most exponential size. To do this, one can memorize the current set of states after each letter and check whether that set contains a final state of the NFA at the end. Since all involved NFAs are at most exponential, this algorithm takes exponential space, and the result follows from  $\text{NEXPSPACE} = \text{EXPSPACE}$ . ◀

For (b), we first need the following lemma, based on Lemma 4 from [43].

► **Lemma 5.7.** *Condition (b) holds if and only if there exist sequences  $(\alpha_i)_{i \in \mathbb{N}_0}$  and  $(\beta_i)_{i \in \mathbb{N}_0}$  in  $\mathcal{C}$  such that*

1.  $\alpha_0 \in \mathcal{I}$  and  $|\alpha_i| > 0$  for every  $i \in \mathbb{N}$ ;
2.  $|\alpha_i| = |\beta_i|$  for every  $i \in \mathbb{N}_0$ ;

3. there exists an infinite run  $r$  of  $A$  on  $\beta_0\beta_1\cdots$  such that, for every  $i \in \mathbb{N}_0$ ,  $\alpha_{i+1}$  is accepted from the state reached by  $r$  on the prefix  $\beta_0\cdots\beta_i$ ,
4. there exists an infinite run  $r'$  of the transducer for  $PReach$  on  $(\beta_0\beta_1\cdots, \beta_0\beta_1\cdots)$  such that  $\forall i \in \mathbb{N}_0 : (\alpha_i, \beta_i\alpha_{i+1})$  is accepted from the state reached by  $r'$  on the prefix  $(\beta_0\cdots\beta_{i-1}, \beta_0\cdots\beta_{i-1})$ .

**Proof.** One direction is easy: if the four conditions hold, the sequence  $(c_i)_{i \in \mathbb{N}_0}$  with  $c_i := \beta_0\cdots\beta_{i-1}\alpha_i$  satisfies (b). For the other direction, we make use of the transitivity of  $PReach$ , see the proof of Lemma 4 in [43]. ◀

► **Lemma 5.8.** *Deciding (b) is in EXPSPACE.*

**Proof.** See the full version [30]. ◀

► **Theorem 5.9.** *ABSTRACT LIVENESS is EXPSPACE-complete both for length-preserving and for general RTSs.*

**Proof.** Membership in EXPSPACE follows from Lemmas 5.5, 5.6, and 5.8. EXPSPACE-hardness follows from an easy reduction from the abstract safety problem: One can add self-loops to all configurations in  $L$  to make sure that (a) holds iff  $PReach(\mathcal{I}) \cap L \neq \emptyset$ . Since self-loops do not affect whether constraints are inductive,  $PReach$  does not change. Note that for length-preserving RTSs, abstract infinite reachability is equivalent to (a). ◀

► **Theorem 5.10.** *Abstract sure termination is EXPSPACE-complete both for length-preserving and for general RTSs.*

**Proof.** Abstract sure termination is the special case of abstract liveness where  $L = \mathcal{C}$  and therefore also in EXPSPACE. The hardness proof is more involved, see the full version [30]. ◀

### 5.3 Almost-sure properties

Unlike recurrent reachability, almost-sure recurrent reachability cannot be semi-decided using only an overapproximation of the reachability relation, not even in the length-preserving case. To see why, recall that a run of a length-preserving RTS visits a regular set  $L$  of configurations infinitely often with probability 1 iff  $Reach(\mathcal{I}) \subseteq \bar{T} \cap Reach^{-1}(L)$  (Lemma 4.2). Unfortunately, given an overapproximation  $PReach \supseteq Reach$ , this condition neither implies nor is implied by  $PReach(\mathcal{I}) \subseteq \bar{T} \cap PReach^{-1}(L)$ . This situation changes when  $Reach^{-1}(L)$  is an effectively computable regular set of configurations. Indeed, in this case  $PReach(\mathcal{I}) \subseteq \bar{T} \cap Reach^{-1}(L)$  is decidable, and implies  $Reach(\mathcal{I}) \subseteq \bar{T} \cap Reach^{-1}(L)$ . We show that this leads to positive results for

ABSTRACT A.S. LIVENESS

Given: RTS  $\mathcal{S} = (\mathcal{I}, \Delta)$ , interpretation  $\mathcal{V}$ , NFA  $A$  for  $L$ .

Decide: Does a potential run of  $\mathcal{S}$  visit  $L$  infinitely often almost surely?

More precisely, in the rest of the section we recall well-structured RTSs [31] and prove a) that ABSTRACT A.S. LIVENESS is decidable for well-structured RTSs and so-called upward-closed sets of configurations, and b) that several well-known classes of parameterized stochastic systems can be modeled as well-structured RTSs for which  $Reach$  is not effectively regular.

**Well-structured RTSs.** The *scattered subword* order on configurations is defined by: for every  $c, c' \in \Sigma^*$ , we have  $c \preceq c'$  if  $c = q_1 \cdots q_n \in \Sigma^*$  and  $c' = w_0 q_1 w_1 q_2 \cdots q_n w_n$  for some words  $w_0, \dots, w_n \in \Sigma^*$ . A set  $L$  of configurations is *upward-closed* if  $c \in L$  and  $c \preceq c'$  implies  $c' \in L$ . It follows from Higman's lemma that every upward-closed set of configurations is regular [34]. An RTS  $\mathcal{S} = (\mathcal{C}, \Delta)$  is *well-structured* if for every  $(c_1, c_2) \in \Delta$  and every  $c'_1 \succeq c_1$  there exists a configuration  $c'_2 \succeq c_2$  such that  $(c'_1, c'_2) \in \text{Reach}$ .

► **Theorem 5.11** ([31]). *Let  $\mathcal{S}$  be a well-structured RTS and let  $A$  be an NFA recognizing an upward-closed set of configurations  $L$ . Then  $\text{Reach}^{-1}(L)$  is regular and effectively computable.*

The algorithm to compute  $\text{Reach}^{-1}(L)$  is very simple: iterate the operation  $L := L \cup \Delta^{-1}(L)$  until a fixpoint is reached. Theorem 5.11 immediately yields:

► **Theorem 5.12.** *ABSTRACT A.S. LIVENESS is decidable for well-structured RTS and upward-closed sets of configurations.*

**Proof.** One needs to decide whether  $\text{PReach}(\mathcal{T}) \subseteq \overline{\mathcal{T}} \cap \text{Reach}^{-1}(L)$ , i.e. whether  $\overline{\text{PReach}(\mathcal{T})} \cup (\overline{\mathcal{T}} \cap \text{Reach}^{-1}(L))$  is universal. An NFA for  $\overline{\text{PReach}(\mathcal{T})}$  can be computed (see Theorem 5.1), and the statement follows. ◀

**Models with non-regular/non-computable reachability relation.** Broadcast protocols [22, 27] and population protocols [11] are two models in which an arbitrarily large number of identical finite-state processes interact by broadcast or rendez-vous, respectively. In both models, configurations with  $n$  processes can be modeled as words of length  $n$  over the alphabet of process states, and the transition function is captured by a length-preserving RTS.

In lossy channel systems (LCS) a fixed set of finite-state processes communicate through unreliable FIFO channels that can lose messages [7]. LCS are made probabilistic by attaching probabilities to message losses [37, 3]. Channels have unbounded capacity, and so they cannot be modeled by length-preserving RTSs. So we consider parameterized lossy channel systems (PLCS), a variant in which channels have bounded capacity, but the capacity is a parameter. In other words, a PLCS is an infinite collection of LCS that only differ in the capacity of their channels (1, 2, ...).

In any of these models, *Reach* is not always effectively regular. (For PLCS see [5], for population protocols it follows from the fact that *Reach* is not always semilinear [36], and for broadcast protocols from the undecidability of recurrent reachability [27].) These negative results justify the interest of Theorem 5.12. Moreover, for population protocols ABSTRACT A.S. LIVENESS is even in EXPSPACE, and so not worse than ABSTRACTSAFETY:

► **Theorem 5.13** ([19, 15]). *ABSTRACT A.S. LIVENESS is EXPSPACE-complete for RTS modeling population protocols, and upward-closed sets of configurations.*

**Proof.** Bozzelli and Ganty show in [15] that  $\text{Reach}^{-1}(L)$  can be computed in EXPSPACE for population protocols.  $\overline{\text{PReach}(\mathcal{T})}$  can also be computed in EXPSPACE, see Theorem 5.1. Hardness follows from the fact that ABSTRACTSAFETY is EXPSPACE-hard [19]. ◀

## 6 Conclusions

We have extended the work of To and Libkin on recurrent reachability of RTSs [43, 44] to the verification of sure and almost-sure properties, and applied our results to the setting of regular abstraction frameworks [19]. In particular, we have shown that ABSTRACT LIVENESS has the same complexity as ABSTRACT SAFETY, and that ABSTRACT A.S. LIVENESS is decidable for some important models of distributed systems.

---

**References**

---

- 1 Parosh Aziz Abdulla. Regular model checking. *Int. J. Softw. Tools Technol. Transf.*, 14(2):109–118, 2012. doi:10.1007/S10009-011-0216-8.
- 2 Parosh Aziz Abdulla. Regular model checking: Evolution and perspectives. In *Model Checking, Synthesis, and Learning*, volume 13030 of *Lecture Notes in Computer Science*, pages 78–96. Springer, 2021. doi:10.1007/978-3-030-91384-7\_5.
- 3 Parosh Aziz Abdulla, Christel Baier, S. Purushothaman Iyer, and Bengt Jonsson. Reasoning about probabilistic lossy channel systems. In *CONCUR*, volume 1877 of *Lecture Notes in Computer Science*, pages 320–333. Springer, 2000. doi:10.1007/3-540-44618-4\_24.
- 4 Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *LICS*, pages 313–321. IEEE Computer Society, 1996. doi:10.1109/LICS.1996.561359.
- 5 Parosh Aziz Abdulla, Aurore Collomb-Annichini, Ahmed Bouajjani, and Bengt Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods Syst. Des.*, 25(1):39–65, 2004. doi:10.1023/B:FORM.0000033962.51898.1A.
- 6 Parosh Aziz Abdulla, Frédéric Haziza, and Lukás Holík. Parameterized verification through view abstraction. *Int. J. Softw. Tools Technol. Transf.*, 18(5):495–516, 2016. doi:10.1007/S10009-015-0406-X.
- 7 Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. In *LICS*, pages 160–170. IEEE Computer Society, 1993. doi:10.1109/LICS.1993.287591.
- 8 Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Julien d’Orso. Regular model checking made simple and efficient. In *CONCUR*, volume 2421 of *Lecture Notes in Computer Science*, pages 116–130. Springer, 2002. doi:10.1007/3-540-45694-5\_9.
- 9 Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Mayank Saxena. A survey of regular model checking. In *CONCUR*, volume 3170 of *Lecture Notes in Computer Science*, pages 35–48. Springer, 2004. doi:10.1007/978-3-540-28644-8\_3.
- 10 Parosh Aziz Abdulla, A. Prasad Sistla, and Muralidhar Talupur. Model checking parameterized systems. In *Handbook of Model Checking*, pages 685–725. Springer, 2018. doi:10.1007/978-3-319-10575-8\_21.
- 11 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Comput.*, 18(4):235–253, 2006. doi:10.1007/S00446-005-0138-3.
- 12 Pascal Bergsträßer, Moses Ganardi, Anthony W. Lin, and Georg Zetsche. Ramsey quantifiers over automatic structures: Complexity and applications to verification. In Christel Baier and Dana Fisman, editors, *LICS*, pages 28:1–28:14. ACM, 2022. doi:10.1145/3531130.3533346.
- 13 Michael Blondin and Mikhail A. Raskin. The complexity of reachability in affine vector addition systems with states. *Log. Methods Comput. Sci.*, 17(3), 2021. doi:10.46298/LMCS-17(3:3)2021.
- 14 Ahmed Bouajjani, Bengt Jonsson, Marcus Nilsson, and Tayssir Touili. Regular model checking. In *CAV*, volume 1855 of *Lecture Notes in Computer Science*, pages 403–418. Springer, 2000. doi:10.1007/10722167\_31.
- 15 Laura Bozzelli and Pierre Ganty. Complexity analysis of the backward coverability algorithm for VASS. In *RP*, volume 6945 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2011. doi:10.1007/978-3-642-24288-5\_10.
- 16 Søren Christensen, Yoram Hirshfeld, and Faron Moller. Bisimulation equivalence is decidable for basic parallel processes. In *CONCUR*, volume 715 of *Lecture Notes in Computer Science*, pages 143–157. Springer, 1993. doi:10.1007/3-540-57208-2\_11.
- 17 Søren Christensen, Yoram Hirshfeld, and Faron Moller. Decomposability, decidability and axiomatizability for bisimulation equivalence on basic parallel processes. In *LICS*, pages 386–396. IEEE Computer Society, 1993. doi:10.1109/LICS.1993.287569.
- 18 Patrick Cousot. *Principles of abstract interpretation*. MIT Press, 2021.

- 19 Philipp Czermer, Javier Esparza, Valentin Krasotin, and Christoph Welzel-Mohr. Computing inductive invariants of regular abstraction frameworks. In *CONCUR*, volume 311 of *LIPICs*, pages 19:1–19:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.CONCUR.2024.19.
- 20 Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for petri nets is not elementary. *J. ACM*, 68(1):7:1–7:28, 2021. doi:10.1145/3422822.
- 21 Wojciech Czerwinski and Lukasz Orlikowski. Reachability in vector addition systems is ackermann-complete. In *FOCS*, pages 1229–1240. IEEE, 2021. doi:10.1109/FOCS52979.2021.00120.
- 22 E. Allen Emerson and Kedar S. Namjoshi. Automatic verification of parameterized synchronous systems (extended abstract). In *CAV*, volume 1102 of *Lecture Notes in Computer Science*, pages 87–98. Springer, 1996. doi:10.1007/3-540-61474-5\_60.
- 23 Javier Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. In *FCT*, volume 965 of *Lecture Notes in Computer Science*, pages 221–232. Springer, 1995. doi:10.1007/3-540-60249-6\_54.
- 24 Javier Esparza. Decidability of model checking for infinite-state concurrent systems. *Acta Informatica*, 34(2):85–107, 1997. doi:10.1007/S002360050074.
- 25 Javier Esparza and Michael Blondin. *Automata Theory – An Algorithmic Approach*. MIT Press, 2023.
- 26 Javier Esparza and Michael Blondin. *Automata theory: An algorithmic approach*. MIT Press, 2023.
- 27 Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In *LICS*, pages 352–359. IEEE Computer Society, 1999. doi:10.1109/LICS.1999.782630.
- 28 Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. Model checking population protocols. In *FSTTCS*, volume 65 of *LIPICs*, pages 27:1–27:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.FSTTCS.2016.27.
- 29 Javier Esparza and Astrid Kiehn. On the model checking problem for branching time logics and basic parallel processes. In *CAV*, volume 939 of *Lecture Notes in Computer Science*, pages 353–366. Springer, 1995. doi:10.1007/3-540-60045-0\_62.
- 30 Javier Esparza and Valentin Krasotin. Regular model checking for systems with effectively regular reachability relation, 2025. arXiv:2506.18833.
- 31 Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001. doi:10.1016/S0304-3975(00)00102-X.
- 32 Christoph Haase. A survival guide to presburger arithmetic. *ACM SIGLOG News*, 5(3):67–82, 2018. doi:10.1145/3242953.3242964.
- 33 David Harel, Amir Pnueli, and Jonathan Stavi. Propositional dynamic logic of nonregular programs. *J. Comput. Syst. Sci.*, 26(2):222–243, 1983. doi:10.1016/0022-0000(83)90014-4.
- 34 Graham Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 3(1):326–336, 1952.
- 35 Chih-Duo Hong and Anthony W. Lin. Regular abstractions for array systems. *Proc. ACM Program. Lang.*, 8(POPL):638–666, 2024. doi:10.1145/3632864.
- 36 John E. Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theor. Comput. Sci.*, 8:135–159, 1979. doi:10.1016/0304-3975(79)90041-0.
- 37 S. Purushothaman Iyer and Murali Narasimha. Probabilistic lossy channel systems. In *TAPSOFT*, volume 1214 of *Lecture Notes in Computer Science*, pages 667–681. Springer, 1997. doi:10.1007/BFB0030633.
- 38 Ondrej Lengál, Anthony Widjaja Lin, Rupak Majumdar, and Philipp Rümmer. Fair termination for parameterized probabilistic concurrent systems. In *TACAS (1)*, volume 10205 of *Lecture Notes in Computer Science*, pages 499–517, 2017. doi:10.1007/978-3-662-54577-5\_29.

- 39 Jérôme Leroux. The reachability problem for petri nets is not primitive recursive. In *FOCS*, pages 1241–1252. IEEE, 2021. doi:10.1109/FOCS52979.2021.00121.
- 40 Anthony W. Lin and Philipp Rümmer. Liveness of randomised parameterised systems under arbitrary schedulers. In *CAV (2)*, volume 9780 of *Lecture Notes in Computer Science*, pages 112–133. Springer, 2016. doi:10.1007/978-3-319-41540-6\_7.
- 41 Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989. doi:10.1109/5.24143.
- 42 Piergiorgio Odifreddi. *Classical recursion theory: The theory of functions and sets of natural numbers*, volume 125. Elsevier, 1992.
- 43 Anthony Widjaja To and Leonid Libkin. Recurrent reachability analysis in regular model checking. In *LPAR*, volume 5330 of *Lecture Notes in Computer Science*, pages 198–213. Springer, 2008. doi:10.1007/978-3-540-89439-1\_15.
- 44 Anthony Widjaja To and Leonid Libkin. Algorithmic metatheorems for decidable LTL model checking over infinite systems. In *FoSSaCS*, volume 6014 of *Lecture Notes in Computer Science*, pages 221–236. Springer, 2010. doi:10.1007/978-3-642-12032-9\_16.