A New Approach for Showing Termination of Parameterized Transition Systems

Roland Herrmann[1][0009−0003−5748−7921] and Philipp Rümmer[1,2][0000−0002−2733−7098]

[1] University of Regensburg, Germany
[2] Uppsala University, Sweden

Abstract. We investigate the termination problem of regular transition systems, i.e., of transition systems whose transition relations can be represented by finite-state automata. Leveraging this automata-theoretical perspective, we propose a new, efficient approach to termination analysis. Our method encodes termination arguments, called statements, as words in an auxiliary language, enabling the verification of whether configurations satisfy those termination arguments via an interpretation automaton. The choice of a fixed interpretation automaton enables the synthesis of new automata dedicated to searching for statements and verifying properties within any given regular transition system. The central contribution of this work is the construction of a specialized finite-state automaton that recognizes statements ensuring the system's termination according to predefined conditions. Depending on the fragment of regular transitions considered, we define two sets of sufficient conditions for termination and provide an automata-based method to verify them within a given regular transition system.

1 Introduction

We consider the termination problem of transition systems. The termination problem is an instance of a liveness property and is, in general, undecidable, as illustrated by the halting problem [15]. Consequently, we must either identify suitable fragments of transition systems that retain decidability, or rely on sound but incomplete procedures that deliver good results in practical applications. In this paper, we examine fragments of regular transition systems (RTS), which can be viewed as an instance of parameterized transition systems. The transition relation in an RTS is modeled by a length-preserving transducer, with the length of the initial configuration acting as an implicit parameter of the system.

The standard approach for proving termination involves over-approximating the transition relation with a well-founded relation. However, finding such a relation efficiently is often challenging. To tackle this problem, we leverage an automata-theoretical framework. Recently, a new perspective on analyzing RTS was introduced by Welzel-Mohr [17], which we adopt. It reasons about properties of the configurations of the system. Those properties are represented by words over an auxiliary alphabet and called statements. To verify whether a given

configuration satisfies a statement, a fixed interpretation automaton is used. In this way, the entire verification process is described by automata. The synthesis of the relevant automata leads to a framework that transforms the process of invariant synthesis into automata queries, in particular checking non-emptiness.

In previous work, this approach was utilized only for analyzing safety properties. In order to apply the framework for termination analysis, we adapt the alphabet to reason about relations between configurations. In particular, then a statement can be viewed as a relation by considering the set of configurations that satisfy the statement according to the interpretation automaton. The search for well-founded relations is incorporated into the synthesized automaton. The challenge is no longer the automation of the process, but rather to find suitable preconditions that the synthesized automata should search for. We give two example automata that verify (1) that a regular transition system is lexicographically ordered, or (2) that it is letter-wise ordered, which both imply termination. The approach is sound for any given RTS and complete for the respective fragment. Our work can be seen as a recipe for transferring the approach to other areas.

Related Work Our work lies within the broader field of **regular model checking,** which was first developed by Bouajjani et al. [7]. In general, questions in regular model checking can be formulated as second-order formulae over some automatic structure [12]. The most common focus is on **safety** properties, that is, whether error states can be reached from initial states. A typical approach involves searching for inductive invariants, e.g., via abstract regular model checking [6] or active automata learning using the L^* algorithm [5,8]. An alternative approach, utilized and extended in this paper, was recently introduced by Welzel-Mohr [17], determining inductive invariants with the help of interpretation automata.

In contrast, **liveness** properties, such as **termination**, have received less attention in regular model checking so far. Podelski et al. [13] introduce the notion of a transition invariant as a tool to prove termination of programs, which directly applies to transition systems as well. A combination with predicate abstraction for automation is described in [14]. A key result shown is that a program terminates if and only if a disjunctively well-founded transition invariant exists, which is a generalization of well-founded relations. However, in our work, we are only concerned with ordinary well-founded relations.

To compute transition invariants of **parameterized transition systems,** it is necessary to over-approximate the transitive closure of the transition relation, which already by itself is a hard problem. An approach that avoids this construction is described by Abdullah et al. [3], where instead backward calculation is used to determine which states necessarily lead to terminating states, leveraging existing methods for safety analysis. Closely related to well-founded relations is the concept of ranking functions. Fang et al. [10] use a heuristic called projection & generalize to construct a ranking function, which is intended to verify that the transition relation eventually makes progress towards termination by reducing

the rank of the current state at each step. Lin et al. [11] use a CEGAR approach to show termination of parameterized transition systems modeling two-player reachability games, applying both Angluin's L^*-algorithm and SAT-solving.

Structure of the Paper In section 2, we give preliminaries on finite-state automata, regular transition systems, relations and fix some notation. We recall the approach from [17] in section 3, which is initially used to prove safety properties. Our main contribution is to carry over the approach from [17] to liveness properties, more precisely to termination analysis. In section 4, we consider two fragments of regular transition systems that we prove terminating: (1) lexicographically ordered systems and (2) letter-wise ordered systems. Both sections follow a common pattern of identifying preconditions that are sufficient to prove termination and then constructing an automaton that searches for witnesses for these conditions. We conclude in section 5 with future work.

2 Preliminaries

Finite-State Automata We assume basic familiarity with finite automata. A non-deterministic finite-state automaton (NFA) is a tuple $\mathcal{A} = (Q_\mathcal{A}, \Sigma_\mathcal{A}, s_\mathcal{A}, \delta_\mathcal{A}, F_\mathcal{A})$, where $Q_\mathcal{A}$ is the set of states, $\Sigma_\mathcal{A}$ the finite alphabet, $s_\mathcal{A} \in Q_\mathcal{A}$ the initial state, $\delta_\mathcal{A} \subseteq Q_\mathcal{A} \times \Sigma_\mathcal{A} \times Q_\mathcal{A}$ the transition relation, and $F_\mathcal{A} \subseteq Q_\mathcal{A}$ the set of accepting states. We will stick to this subscript notation for the corresponding components of an automaton $\mathcal{A}$. The language accepted by $\mathcal{A}$ is denoted by $\mathcal{L}(\mathcal{A})$. We assume that for all $p \in Q_\mathcal{A}, s \in \Sigma_\mathcal{A}$, there exists a $q \in Q_\mathcal{A}$ such that $(p, s, q) \in \delta_\mathcal{A}$, since otherwise we may simply introduce a fresh non-accepting "sink state". For the sake of presentation we will not explicitly depict the sink state in figures.

A Σ_1-Σ_2-transducer for two alphabets Σ_1, Σ_2 is an NFA with alphabet $\Sigma_1 \times \Sigma_2$. Let $\mathcal{T}$ be a Σ_1-Σ_2-transducer. Let $u = u_1 \cdots u_n \in \Sigma_1^*, v = v_1 \cdots v_n \in \Sigma_2^*$ be two words, then we write $u \otimes v$ for the word $(u_1, v_1) \dots (u_n, v_n)$ in $(\Sigma_1 \times \Sigma_2)^*$ and $\binom{u_i}{v_i}$ for its letters instead of (u_i, v_i). In case $u \otimes v \in \mathcal{L}(\mathcal{T})$, we interpret u as the input and v as the output of $\mathcal{T}$. If $\Sigma_1 = \Sigma_2 = \Sigma$, then $\mathcal{T}$ defines a relation $R_\mathcal{T}$ on Σ^* by $(u, v) \in R_\mathcal{T} :\Leftrightarrow u \otimes v \in \mathcal{L}(\mathcal{T})$.[3] When writing $u \otimes v$, we we implicitly assume that the lengths of the words u, v is equal. Furthermore, our notion of transducer does not allow ε-transitions, neither in the input nor in the output, as opposed to other definitions from the literature.

Regular Transition Systems

Definition 1 (regular transition system[1][2]). *A regular transition system (RTS) is a tuple $(\Sigma, \mathcal{T})$, where Σ is a finite alphabet and $\mathcal{T}$ is a Σ-Σ transducer. We call $\mathcal{T}$ the transition relation, the letters $s \in \Sigma$ states, words $w \in \Sigma^*$ configurations and the i-th position of a configuration an agent. In other words, a configuration is a finite sequence of the current states of agents.*

[3] We write "$A :\Leftrightarrow B$" to denote that A is defined to be equivalent to B.

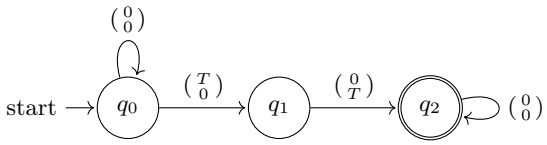


Fig. 1: Transition relation $\mathcal{T}$ for token passing

Remark 1 (initial configurations). We consider termination from any configuration. Therefore, we do not specify initial configurations. In this regard, our definitions deviate from other standard terminology also found in the literature, which usually emphasizes the origin from transition systems and then additionally demands the initial configurations and transition relation to be a regular expression.

Let $(\Sigma, \mathcal{T})$ be a RTS, $a = a_1 \dots a_n \in \Sigma^*$, $b = b_1 \dots b_n \in \Sigma^*$ two words. Then we write $a \to_{\mathcal{T}} b$, or simply $a \to b$ if $\mathcal{T}$ is clear from the context, if we have $\binom{a_1}{b_1} \dots \binom{a_n}{b_n} \in \mathcal{L}(\mathcal{T})$ and call it a transition from a to b. The number of agents in a configuration is invariant under the transition relation, since transducers are defined to be length-preserving in our setting. One may view the number of agents as a parameter, making RTSs an instance of parametrized transition systems. Furthermore, since our alphabet is finite, there are at most $|\Sigma|^n < \infty$ reachable configurations from any initial configuration $w \in \Sigma^n$ for n agents. We call transition systems in which for any configuration the set of reachable configurations is finite, weakly-finite [9].

Example 1 (Token passing). Let $\Sigma = \{0, T\}$. Consider the transition relation described by the Σ-Σ-transducer depicted in Figure 1. To illustrate the transition system, we also give all transitions by hand, starting from a specific configuration $T000$ and the parameter value $n = 4$:

$$T000 \to 0T00 \to 00T0 \to 000T \tag{1}$$

The transition relation can equivalently be defined using the corresponding regular expression:

$$\tbinom{0}{0}^* \tbinom{T}{0} \tbinom{0}{T} \tbinom{0}{0}^* . \tag{2}$$

The RTS $(\Sigma, \mathcal{T})$ models a token passing protocol. The state T represents that at this position there is a token and 0 represents that there is none. The transition relation $\mathcal{T}$ models that a token is handed over from the i-th to the $(i+1)$-th agent, starting at the first and ending in the last. The transition relation assumes that there exists only one token (otherwise there are no transitions). The token passing protocol serves as an easy running example of a terminating RTS.

Relations We define some terms and notation on relations for later use.

Definition 2. *Let $R \subseteq S \times S$ be a relation. We call R*

- *reflexive if for all $x \in S$, we have $(x, x) \in R$.*
- *transitive if for all $x, y, z \in S$ with $(x, y), (y, z) \in R$, we have $(x, z) \in R$.*
- *irreflexive if for all $x \in S$, we have $(x, x) \notin R$.*
- *total if for all $x, y \in S$ with $x \neq y$, we have $(x, y) \in R$ or $(y, x) \in R$.*
- *partial if it is not total.*
- *preorder if it is reflexive and transitive.*

Remark 2. Let $\preccurlyeq$ be a preorder on a set S. Then

$$a \prec b \;:\Leftrightarrow\; a \preccurlyeq b \wedge \neg(b \preccurlyeq a). \tag{3}$$

defines an irreflexive, transitive relation and every irreflexive, transitive relation arises in this way. This motivates the following notation.

Notation 1. Let $R \subseteq S \times S$ be a relation. For $x, y \in S$, we write

- $x \geq_R y$ if $(x, y) \in R$.
- $x >_R y$ if $(x, y) \in R$ and $(y, x) \notin R$.
- $x =_R y$ if $(x, y), (y, x) \in R$.

We abbreviate $\geq_i, >_i, =_i$ for $\geq_{R_i}, >_{R_i}, =_{R_i}$ for a relation R_i with index i.

As seen in [13], well-founded relations play a crucial role for termination.

Definition 3 (well-founded [16]). *Let S be a set. A relation $R \subseteq S \times S$ is called well-founded if there exists no sequence $(s_n)_{n\in\mathbb{N}}$ such that $(s_n, s_{n+1}) \in R$ for all $n \in \mathbb{N}$.*

Lemma 1. *Let R be an irreflexive, transitive relation on a finite set S. Then R is well-founded.*

Proof. Assume for a contradiction that R is not well-founded, i.e., there exists a sequence $(w_i)_{i\in\mathbb{N}}$ such that $(w_i, w_{i+1}) \in R$ for all $i \in \mathbb{N}$. Since S is finite there exists $i, j \in \mathbb{N}$ such that $i \neq j$ and $w_i = w_j$. Without loss of generality we may assume $i < j$. Then transitivity of R along $w_i \geq_R w_{i+1} \geq_R \cdots \geq_R w_j$ implies $w_i \geq_R w_j = w_i$ which contradicts irreflexivity of R.

3 Properties of RTS as Statements

We revise the approach presented in [17] for reasoning about configurations in an RTS. Statements are formulated as words over an auxiliary alphabet and an interpretation automaton justifies whether a configuration satisfies a statement.

Fix a new (finite) alphabet Γ for statements. A statement is then a word $I \in \Gamma^*$. An interpretation automaton is a Σ-Γ-transducer $\mathcal{V}$. A statement $I \in \Gamma^n$ holds in a configuration $w \in \Sigma^n$ if and only if $w \otimes I \in \mathcal{L}(\mathcal{V})$. For $\Gamma = 2^\Sigma$, an example of an interpretation automaton is depicted in Figure 2 by $\mathcal{V}_{Trap}$. The labels "$w \in I$" and "$w \notin I$" summarize the exponentially many transition labels $\{(w, I) \in \Sigma \times 2^\Sigma \mid w \in I\}$, respectively $\{(w, I) \in \Sigma \times 2^\Sigma \mid w \notin I\}$ and the

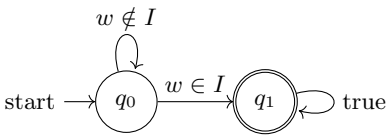


Fig. 2: The interpretation automaton $\mathcal{V}_{Trap}$

transition "true" is always available. In natural language $\mathcal{V}_{Trap}$ checks whether at least one of the states of the configuration is included in the corresponding set (at same position) in the statement. We can define the set of configurations that satisfy I as $\{w \in \Sigma^* \mid w \otimes I \in \mathcal{L}(\mathcal{V})\}$. That way, I naturally describes an invariant candidate.

Example 2. We come back to our token passing example. Consider the word $I = \{T\}\{T\}\{T\}\{T\} \in (2^\Sigma)^*$. It can be easily verified that $\Sigma^4 \setminus \{0000\}$ is the set of all configurations that satisfy I in the interpretation $\mathcal{V}_{Trap}$. In natural language, I can be reformulated as "has at least one token".

The main result of this construction is now that we can automatically verify whether a statement holds (for certain words) and therefore also automatically search for statements that have certain properties. In the work of [17], this is done for inductive statements. In other words, the statement alphabet and the interpretation automaton yield a set of invariant candidates (one for each statement) and we need to search within this set for actual invariants with the desired properties, like being inductive. The challenge in this approach is no longer the automation of the search but rather finding appropriate alphabets for statements and interpretations. Ideally, one tries to avoid a choice of alphabet and interpretation that is specific for one single RTS.

4 Termination of RTS

In this section we show how the approach from [17] can be used to show termination of regular transition systems. The proofs of this section are straightforward and rather technical, hence they are deffered to the appendix.

Definition 4 (Termination of a RTS). *Let $(\Sigma, \mathcal{T})$ be an RTS. We say that $(\Sigma, \mathcal{T})$ terminates on $w_0 \in \Sigma^*$ if there exists no sequence $(w_i)_{i\in\mathbb{N}}$ such that $w_i \otimes w_{i+1} \in \mathcal{L}(\mathcal{T})$ for all $i \in \mathbb{N}_0$. We say that $(\Sigma, \mathcal{T})$ terminates if it terminates for all $w_0 \in \Sigma^*$.*

Theorem 1. *The termination problem of RTS is undecidable in general.*

Proof. Consider the halting problem whether a turing machine M halts on input x. This can be reduced to termination of a turing machine M' using at most N tape cells on inputs of at most N cells of memory for each $N \in \mathbb{N}$ uniformly. M' can be modelled by a length preserving transducer $\mathcal{T}$. Hence, M' halts for all $N \in \mathbb{N}$ and every input if and only if the RTS $(\Sigma, \mathcal{T})$ terminates where Σ represents both the tape content and the state of the turing machine M'.

As a first step, we exploit the fact that regular transition systems are weakly-finite. Therefore, it is sufficient for termination that there exists an irreflexive, transitive relation on the set of reachable configurations for each value of the parameter, over-approximating the transition relation. This relation is then immediately well-founded by Lemma 1. Formally, we have the following Theorem.

Theorem 2. *Let $(\Sigma, \mathcal{T})$ be an RTS, $n \in \mathbb{N}$. Let R be a well-founded relation that over-approximates the transition relation, i.e. $(\Sigma^n \times \Sigma^n) \cap R_{\mathcal{T}} \subseteq (\Sigma^n \times \Sigma^n) \cap R$. Then the RTS terminates on any input $w \in \Sigma^n$.*

In order to reason about relations, we consider $\Sigma \times \Sigma$ as underlying alphabet and $\Gamma = 2^{\Sigma \times \Sigma}$ as alphabet for statements. For a fixed interpretation automaton $\mathcal{V}$, a statement $I \in \Gamma^*$ then defines a relation on the configurations by

$$R_I = \{(w_1, w_2) \in \Sigma^* \times \Sigma^* \mid ((w_1 \otimes w_2) \otimes I) \in \mathcal{L}(\mathcal{V})\} \subseteq \Sigma^{|I|} \times \Sigma^{|I|}. \quad (4)$$

We give two frameworks to prove termination of fragments of regular transition systems in the following.

4.1 Lexicographically Ordered Systems

We first treat lexicographically ordered systems. Intuitively, this is how words in a dictionary are arranged.

Lemma 2 (lexicographical order). *Let $R_\Sigma \subseteq \Sigma \times \Sigma$ be a relation on Σ. Then R_Σ induces a relation R_{lex} on Σ^* by*

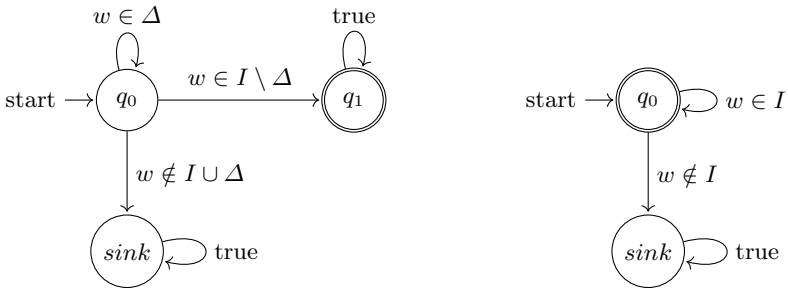
$$\begin{aligned} u_1 \dots u_n R_{lex} v_1 \dots v_m :\Leftrightarrow \exists i \in \{1, \dots, n\}.\ (u_i R_\Sigma v_i \wedge \forall j < i.\ u_j = v_j) \\ \vee (n > m \wedge u_1 \dots u_m = v_1 \dots v_m). \end{aligned} \quad (5)$$

If R_Σ has any of the properties in definition 2, then so does R_{lex}. If R_Σ is a total, irreflexive, transitive relation, we call R_{lex} lexicographical order. We call an RTS $(\Sigma, \mathcal{T})$ lexicographically ordered if there exists a lexicographical order R_{lex} such that $R_{\mathcal{T}} \subseteq R_{lex}$.

Let $\Delta := \{\binom{x}{x} \mid x \in \Sigma\}$. We define the interpretation automaton that captures the essential steps to recognize lexicographical orders as the automaton $\mathcal{V}_{lex}$ depicted in Figure 3 over the alphabet $(\Sigma \times \Sigma) \times 2^{\Sigma \times \Sigma}$. In particular $I \cup \Delta, I \setminus \Delta \in 2^{\Sigma \times \Sigma}$. The automaton $\mathcal{V}_{lex}$ demands for some $i \in \mathbb{N}$ the first $i-1$ letters to be equal and then make a transition according to our input statement on the i-th position in order to reach the accepting state.

For a fixed statement $I \in (2^{\Sigma \times \Sigma})^*$, we consider the induced relation R_I from (4). By Lemma 1 and Theorem 2 it suffices to find for all $n \in \mathbb{N}$ an irreflexive, transitive relation that over-approximates the transition relation on Σ^n, i.e.,

(i) Irreflexive: $R_I \subseteq \{(x, y) \mid x \neq y\} \cap (\Sigma \times \Sigma)^n$.
(ii) Transitive: $\{(x, z) \mid \exists y.\{(x, y), (y, z)\} \subseteq R_I\} \subseteq R_I$.
(iii) over-approximates transition relation $R_{\mathcal{T}} \cap (\Sigma \times \Sigma)^n \subseteq R_I$



(a) $\mathcal{V}_{lex}$ (b) $\mathcal{V}_{All}$

Fig. 3: Interpretation automata for lexicographical, respectively letterwise order

Example 3. The token passing protocol is lexicographically ordered. We simply define $R_\Sigma = \{(T, 0)\}$. The corresponding statement that recognises this lexicographical order is then $I_n = \{\binom{T}{0}\}^n$ for all $n \in \mathbb{N}$ and we have

$$R_{I_n} = \mathcal{L}\left(\left(\binom{0}{0} + \binom{T}{T}\right)^* \binom{T}{0} (\Sigma \times \Sigma)^*\right) \cap (\Sigma \times \Sigma)^n. \tag{6}$$

We now construct for a given RTS $(\Sigma, \mathcal{T})$ an automaton $\mathcal{A}_{lex}$ that searches within the set of relations R_I for one that satisfies the conditions (i)-(iii). $\mathcal{A}_{lex}$ should have $2^{\Sigma\times\Sigma}$ as alphabet and accept a statement $I \in (2^{\Sigma\times\Sigma})^*$ if and only if R_I satisfies (i)-(iii). However, the conditions (i)-(iii) are universally quantified. In order to eliminate the universal quantifier on our three conditions, we first construct the complement of the desired automaton, that is, an automaton that accepts I if and only if there exist $x, y, z \in \Sigma^*$ such that one of (i)-(iii) does not hold, i.e.

$$(xR_Iy \wedge x = y) \vee (xR_Iy \wedge yR_Iz \wedge \neg xR_Iz) \vee (x\mathcal{T}y \wedge \neg xR_Iy). \tag{7}$$

Any of the occurring predicates can be expressed by an automaton, e.g., xR_Iy evaluates to true if and only if $((x \otimes y) \otimes I) \in \mathcal{L}(\mathcal{V}_{lex})$, $\mathcal{V}_{lex}$ just ignores the z argument. Hence, we need three copies of $\mathcal{V}_{lex}$ for the predicates xR_Iy, yR_Iz, xR_Iz, each one ignoring the argument that does not occur in it. For the equality check, $x = y$, we take a two state automaton that accepts a pair (x, y) if and only if $x = y$. Lastly $xR_\mathcal{T}y$ is expressed by ignoring the z input and have a run in the transducer $\mathcal{T}$. We can check which of these predicates hold simultaneously by going through the product of all these automata states in parallel, that is, we have

$$Q_{\mathcal{A}^C_{lex}} = \underbrace{Q_\mathcal{T}}_{xR_\mathcal{T}y} \times \underbrace{Q_=}_{x=y} \times \underbrace{Q_{\mathcal{V}_{lex}}}_{xR_Iy} \times \underbrace{Q_{\mathcal{V}_{lex}}}_{yR_Iz} \times \underbrace{Q_{\mathcal{V}_{lex}}}_{xR_Iz}. \tag{8}$$

The transition relation is then given according to the transition relations of the respective factors, i.e.,

$$((p_\mathcal{T}, p_=, p_1, p_2, p_3), (x_i, y_i, z_i, I_i), (q_\mathcal{T}, q_=, q_1, q_2, q_3)) \in \delta \tag{9}$$

if and only if all of the following conditions hold:

$$\begin{aligned}
(p_{\mathcal{T}}, (x_i, y_i), q_{\mathcal{T}}) &\in \delta_{\mathcal{T}} && (x \mathcal{T} y)\\
(p_{=}, (x_i, y_i), q_{=}) &\in \delta_{=} && (x = y)\\
(p_1, (x_i, y_i, I_i), q_1) &\in \delta_{\mathcal{V}_{lex}} && (x R_I y)\\
(p_2, (y_i, z_i, I_i), q_2) &\in \delta_{\mathcal{V}_{lex}} && (y R_I z)\\
(p_3, (x_i, z_i, I_i), q_3) &\in \delta_{\mathcal{V}_{lex}} && (x R_I z)
\end{aligned}$$

We can now translate our formula one to one to the accepting states of $\mathcal{A}^C_{lex}$, where we use that the formula 7 is already in disjunctive normal form (each clause stands for the negation of one of the conditions (i)-(iii)):

$$\begin{aligned}
F_{\mathcal{A}^C_{lex}} =& Q_{\mathcal{T}} \times F_{=} \times F_{\mathcal{V}_{lex}} \times Q_{\mathcal{V}_{lex}} \times Q_{\mathcal{V}_{lex}} && \neg\text{(i)}\\
&\cup Q_{\mathcal{T}} \times Q_{=} \times F_{\mathcal{V}_{lex}} \times F_{\mathcal{V}_{lex}} \times (Q_{\mathcal{V}_{lex}} \setminus F_{\mathcal{V}_{lex}}) && \neg\text{(ii)}\\
&\cup F_{\mathcal{T}} \times Q_{=} \times (Q_{\mathcal{V}_{lex}} \setminus F_{\mathcal{V}_{lex}}) \times Q_{\mathcal{V}_{lex}} \times Q_{\mathcal{V}_{lex}} && \neg\text{(iii)}
\end{aligned}$$

That way, our automaton accepts a tuple (x, y, z, I) if and only if at least one of the conditions (i)-(iii) fails. In order to get rid of the first three input tapes, we project onto the statement input tape, ignoring x_i, y_i, z_i in the transition relation δ. We can do this because x, y, z are existentially quantified in the negated statements of (i)-(iii). Let $\delta_{\mathcal{V}^C_{lex}}$ be the transition relation that we obtain from δ by projecting onto the statement tape. Finally, we can define

$$\mathcal{A}^C_{lex} = (Q_{\mathcal{A}^C_{lex}}, \Gamma, (s_{\mathcal{T}}, s_{=}, s_{\mathcal{V}_{lex}}, s_{\mathcal{V}_{lex}}, s_{\mathcal{V}_{lex}}), \delta_{\mathcal{A}^C_{lex}}, F_{\mathcal{A}^C_{lex}}). \tag{10}$$

$\mathcal{A}^C_{lex}$ accepts a statement $I \in \Gamma^*$ if and only if there exist $x, y, z \in \Sigma^*$ such that one of (i)-(iii) does not hold. We obtain the following theorem:

Theorem 3. *Let $(\Sigma, \mathcal{T})$ be a RTS, $\mathcal{A}^C_{lex}$ the corresponding automaton according to our construction above. If $(\mathcal{A}^C_{lex})^C = \mathcal{A}_{lex}$ accepts a word of every length, i.e. $\mathcal{L}(\mathcal{A}_{lex}) \cap \Gamma^n \neq \emptyset$ for all $n \in \mathbb{N}$, then $(\Sigma, \mathcal{T})$ terminates.*

Proof. Suppose $\mathcal{A}_{lex}$ accepts a word of every length. Let $w_0 \in \Sigma^*$, $I \in \mathcal{L}(\mathcal{A}_{lex})$ with $|w_0| = |I|$. Assume for a contradiction that there exists $(w_i)_{i \in \mathbb{N}}$ with $w_i \otimes w_{i+1} \in \mathcal{L}(\mathcal{T})$. By construction R_I over-approximates the transition relation, hence, $(w_i, w_{i+1}) \in R_I$. Furthermore, R_I is irreflexive and transitive since $\mathcal{A}_{lex}$ accepts only statement with these properties. In particular, R_I is well-founded which is a contradiction to $(w_i, w_{i+1}) \in R_I$ for all $i \in \mathbb{N}$.

Remark 3. Let $\mathcal{A}$ be an NFA. In order to show that $\mathcal{A}$ accepts a word of every length we can apply the following construction. Construct the automaton $\mathcal{A}_*$ which is the same as $\mathcal{A}$ except that we replace the alphabet by the singleton $\Sigma_{\mathcal{A}_*} = \{*\}$ and accordingly $(p, *, q) \in \delta_{\mathcal{A}_*}$ if and only if there exists an $s \in \Sigma_{\mathcal{A}}$ such that $(p, s, q) \in \delta_{\mathcal{A}}$. Then $\mathcal{A}$ accepts a word of every length if and only if $\mathcal{A}_*$ is universal which is easy to check.

Corollary 1. *Let $(\Sigma, \mathcal{T})$ be a lexicographically ordered RTS. Then $\mathcal{L}(\mathcal{A}_{lex}) \cap \Gamma^n \neq \emptyset$ for all $n \in \mathbb{N}$.*

Proof. Let $>_\Sigma$ be the irreflexive, transitive relation on Σ that induces the lexicographical order on Σ^*. Then $(>_\Sigma)^n \in \mathcal{L}(\mathcal{A}_{lex})$ for all $n \in \mathbb{N}$.

Remark 4 (positionwise different orders). The interpretation automaton does even recognise more than only lexicographically ordered systems. We are allowed to consider different orders on every position. For example, we can take the system where even agents count down from some fixed $n \in \mathbb{N}$, and odd agents count up from zero to n, i.e. the transition relation is given by

$$\left(\left(\binom{0}{1} + \cdots + \binom{n-1}{n}\right)\left(\binom{n}{n-1} + \cdots + \binom{1}{0}\right)\right)^* . \tag{11}$$

Then $\mathcal{A}_{lex}$ would recognise all words with the following letters on odd, respectively even positions

$$I_{odd} = \left\{\binom{x-1}{x} \mid x \in \{1, \ldots, n\}\right\} \qquad I_{even} = \left\{\binom{x}{x-1} \mid x \in \{1, \ldots, n\}\right\} \tag{12}$$

In particular, the system is proved terminating by our automaton $\mathcal{A}_{lex}$.

4.2 Letterwise ordered systems

In lexicographically ordered systems, one only needs to wait for the next letter to strictly progress towards some terminating configuration. In this section we will add letterwise loops, i.e., the i-th agent of the system may execute a loop but the RTS still terminates, since the loop can only be executed finitely often. The following example explains the class of systems we want to tackle.

Example 4 (polite Mexican standoff). Let n agents be alive (state A) as an initial configuration of our RTS. Every agent is armed and wants to kill all remaining agents, such that the agent is the last one alive (such situations are called Mexican standoff). In order to bring the whole thing to a neat and tidy end, they decide to randomly pick two agents to have a transition into a shooting state (S) and have a duel, while the others are waiting until the duel has finished. In the duel, randomly one of the two duelists dies (changes state to D) and the other stays alive (changes state back to A). Formally, let $\Sigma = \{A, S, D\}$ be the alphabet. The transition relation $\mathcal{T}$ is described in Figure 4, where $X_i \in (A + D)^*$ for $i \in \{1, \ldots, 3\}$. We labelled the transitions for later reference. It is now possible for the first agent who transitions into S to go back to A afterwards, as long as a later agent changes his state to D, contributing towards the terminating configuration D^*AD^*.

The polite Mexican standoff is not lexicographically ordered, since we do not know whether the first or the second agent of the duel dies. Formally, both choices $A > S$ and $S > A$ contradict either *second* or *init* in the induced lexicographical order. However, we can say that at some point we may progress from A to D and can never go back. Hence, to model just $first$, $second$, $skipFirst$ and $skipSecond$

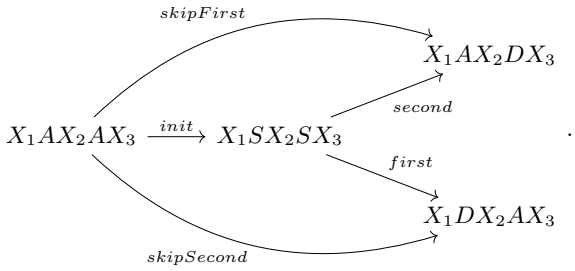


Fig. 4: transition graph of the polite Mexican standoff

of Figure 4 we may set $A =_1 S >_1 D$ for a relation R_1 and observe that all agents of our configuration follow this order. This relation can be modelled by the set

$$\{\binom{A}{A}, \binom{S}{S}, \binom{D}{D}, \binom{A}{S}, \binom{S}{A}, \binom{A}{D}, \binom{S}{D}\} \tag{13}$$

In order to verify that every agent follows this relation we define the interpretation $\mathcal{V}_{All}$, depicted in Figure 3. A configuration $w_1 \cdots w_n \in \Sigma^n$ satisfies $I_1 \cdots I_n \in (2^\Sigma)^n$ in $\mathcal{V}_{All}$ if and only if $w_i \in I_i$ for all $i \in \{1, \ldots, n\}$. Consequently, the statements that describe the desired order relation on configuration is given by applying the Kleene-star to (13).

Remark 5. $\mathcal{V}_{All}$ does not subsume lexicographical orders. Consider counting down in binary. Then the statement $\{\binom{0}{0}, \binom{1}{1}, \binom{1}{0}\}^n$ does not contain all transitions of counting down in binary in the interpretation $\mathcal{V}_{All}$, since it would demand the agents after the first change to follow this order, too.

In order to include the transition *init* from the polite Mexican standoff, we refine our relation by introducing another relation R_2 that specifies what happens if we are stuck in the case $x =_1 y$. Here, it suffices to demand $A >_2 S$. Since R_2 should only be asked if $=_1$ occurred, it is irrelevant how D is related (it could also be unrelated to A and S). Formally, the overall relation R_3 is defined by

$$xR_3y :\Leftrightarrow x >_1 y \vee (x =_1 y \wedge x >_2 y). \tag{14}$$

Lemma 3. *Let $R_1, R_2 \subset \Sigma \times \Sigma$ be two preorders on a set Σ. Then R_3 defined by (14) is irreflexive and transitive.*

Theorem 2 and Lemma 3 result in the following Theorem.

Theorem 4. *Let $(\Sigma, \mathcal{T})$ be an RTS, $n \in \mathbb{N}$. Let R_1, R_2 be two preorders on Σ^n and let R_3 be defined as in (14), such that R_3 over-approximates the transition relation on Σ^n, i.e., we have $(\Sigma \times \Sigma)^n \cap R_\mathcal{T} \subseteq R_3$. Then the RTS terminates on any input $w \in \Sigma^n$.*

As in the case of lexicographical orders, we can construct an automaton $\mathcal{A}_{All}$ that searches for a relation that proves our system terminating. The construction works completely analogue to the previous section. The only difference is that now we need eight copies of $\mathcal{V}_{All}$ to model the predicates $R_i(x,y)$, $R_i(y,z)$, $R_i(x,z)$, $R_i(y,x)$ for each $i \in \{1,2\}$. Formally we have

$$\mathcal{A}^C_{All} = (Q_{\mathcal{T}} \times Q_{=} \times (Q_{\mathcal{V}_{All}})^8, 2^{\Sigma\times\Sigma}, (s_{\mathcal{T}}, s_{=}, (s_{\mathcal{V}_{All}})^8), \delta_{\mathcal{A}^C_{All}}, F). \tag{15}$$

$\delta_{\mathcal{A}^C_{All}}$ is constructed analogously as for $\mathcal{A}^C_{lex}$, following the transition relations of each factor if there exists a triple $(x,y,z) \in \Sigma^3$ that permits a transition in each factor uniformly. The accepting states F correspond to the negated properties of being reflexive, respectively transitive for R_1 and R_2, and R_3 over-approximating the transition relation. For the sake of presentation, we give the accepting states for the negation of R_3 over-approximating the transition relation. The negation of being an over-approximation of R_3 looks as follows.

$$xR_{\mathcal{T}}y \wedge (\neg xR_1y \vee yR_1x) \wedge (\neg xR_1y \vee \neg yR_1x \vee \neg xR_2y \vee yR_2x). \tag{16}$$

Let W_1 be the set of literals in the second conjunct, W_2 the set of literals in the third conjunct of (16). Then the conjunctive normal form of (16) is given by

$$\bigvee_{w_1 \in W_1, w_2 \in W_2} xR_{\mathcal{T}}y \wedge w_1 \wedge w_2. \tag{17}$$

The corresponding accepting states for $\mathcal{A}^C_{All}$ are then given by the union of the accepting states corresponding to each disjunct of (17).

We may carry on this construction inductively. Let in the following R_i always be a preorder. Let $F_1 :\Leftrightarrow xR_1y \wedge \neg yR_1x$. We define

$$S_i \Leftrightarrow \left(\bigwedge_{k=1}^{i} x =_k y\right) \wedge x >_{i+1} y. \tag{18}$$

Then we can replace the relation from (14) in Theorem 4 by $F_{i+1} :\Leftrightarrow F_i \vee S_i$.

Lemma 4. *Let $n \in \mathbb{N}$. Let R_i be preorders on a set Σ for all $i \in \mathbb{N}$ with $i \leq n$. Let R_{n+1} be the relation defined by F_n. Then R_{n+1} is irreflexive and transitive.*

Theorem 5. *Let $(\Sigma, \mathcal{T})$ be an RTS. Let $\mathcal{A}_{All,i} = (\mathcal{A}^C_{All,i})^C$ be the complement of the automaton constructed above with the modifications according to F_i. If there exists an $i \in \mathbb{N}$ such that $\mathcal{A}_{All,i}$ accepts a word of every length, then $(\Sigma, \mathcal{T})$ terminates.*

Proof. Let $i \in \mathbb{N}$. The automaton $\mathcal{A}_{All,i}$ accepts exactly those words whose induced relation is given by F_i and over-approximates the transition relation by construction. Since F_i is well-founded by Lemma 4, $(\Sigma, \mathcal{T})$ terminates if $\mathcal{A}_{All,i}$ accepts a word of every length.

5 Conclusion and Future Work

We provided the constructions of automata that can prove termination of lexicographically, respectively letterwise ordered systems, carrying over the approach of [17] from safety to liveness properties. The implementation of the construction is ongoing work. A natural continuation would be to tackle other fragments of regular transition systems that can not be proved terminating with the methods from section 4, e.g. a "waving" token passing

$$T000 \to 0TTT \to 0T00 \to T0TT \to 00T0 \to TT0T \to 000T. \tag{19}$$

On the other hand, one can loosen restrictions on the alphabet, allowing infinite alphabets. This can be done by using parameterized automata [4] to model the transition relation. An instance of such a transition system is Dijkstra's self-stabilizing protocol. However, these systems have two difficulties to overcome: (1) they are no longer weakly-finite and therefore need more preparation to formulate terminating conditions to search for and (2) complements of parameterized automata do not exist in general and even if they exist, it is hard to complement them but complementing is a crucial step to eliminate universal quantifiers involved in the conditions that are needed to prove termination.

Furthermore, the flexibility of the approach from [17] may allow to tackle completely different verification problems apart from safety and termination. To this end, the interpretation automaton and the corresponding alphabet for statements have to be varied. This seems to be the challenging part. Many choices may result in constructing empty or universal automata in the end.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Acknowledgement. This work was supported by the Swedish Research Council through grant 2021-06327, and by the Knut and Alice Wallenberg Foundation through project UPDATE. The authors thank the CIAA reviewers for insightful comments!

References

1. Abdulla, P., Jonsson, B., Nilsson, M., Saksena, M.: A Survey of Regular Model Checking. vol. 3170, pp. 35–48 (Aug 2004). https://doi.org/10.1007/978-3-540-28644-8_3
2. Abdulla, P.A.: Regular model checking. International Journal on Software Tools for Technology Transfer **14**(2), 109–118 (Apr 2012). https://doi.org/10.1007/s10009-011-0216-8, https://doi.org/10.1007/s10009-011-0216-8
3. Abdulla, P.A., Jonsson, B., Rezine, A., Saksena, M.: Proving Liveness by Backwards Reachability. In: Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J.M., Mattern, F., Mitchell, J.C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M.Y., Weikum, G., Baier, C., Hermanns, H. (eds.) CONCUR 2006 – Concurrency Theory, vol. 4137, pp. 95–109. Springer Berlin Heidelberg, Berlin, Heidelberg (2006). https://doi.org/10.1007/11817949_7, http://link.springer.com/10.1007/11817949_7, series Title: Lecture Notes in Computer Science

4. Alur, R., Henzinger, T.A., Vardi, M.Y.: Parametric real-time reasoning. In: Proceedings of the twenty-fifth annual ACM symposium on Theory of Computing. pp. 592–601. STOC '93, Association for Computing Machinery, New York, NY, USA (Jun 1993). https://doi.org/10.1145/167088.167242, https://dl.acm.org/doi/10.1145/167088.167242
5. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation **75**(2), 87–106 (Nov 1987). https://doi.org/10.1016/0890-5401(87)90052-6, https://www.sciencedirect.com/science/article/pii/0890540187900526
6. Bouajjani, A., Habermehl, P., Vojnar, T.: Abstract regular model checking. In: Alur, R., Peled, D.A. (eds.) Computer Aided Verification, 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3114, pp. 372–386. Springer (2004). https://doi.org/10.1007/978-3-540-27813-9_29, https://doi.org/10.1007/978-3-540-27813-9_29
7. Bouajjani, A., Jonsson, B., Nilsson, M., Touili, T.: Regular Model Checking. In: Emerson, E.A., Sistla, A.P. (eds.) Computer Aided Verification. pp. 403–418. Springer, Berlin, Heidelberg (2000). https://doi.org/10.1007/10722167_31
8. Chen, Y.F., Hong, C.D., Lin, A.W., Rümmer, P.: Learning to prove safety over parameterised concurrent systems. In: Proceedings of the 17th Conference on Formal Methods in Computer-Aided Design. pp. 76–83. FMCAD '17, FMCAD Inc, Austin, Texas (Oct 2017)
9. Esparza, J., Gaiser, A., Kiefer, S.: Proving Termination of Probabilistic Programs Using Patterns. In: Madhusudan, P., Seshia, S.A. (eds.) Computer Aided Verification. pp. 123–138. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31424-7_14
10. Fang, Y., Piterman, N., Pnueli, A., Zuck, L.: Liveness with Invisible Ranking. In: Steffen, B., Levi, G. (eds.) Verification, Model Checking, and Abstract Interpretation. pp. 223–238. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24622-0_19
11. Lin, A.W., Rümmer, P.: Liveness of Randomised Parameterised Systems under Arbitrary Schedulers. In: Chaudhuri, S., Farzan, A. (eds.) Computer Aided Verification. pp. 112–133. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-41540-6_7
12. Lin, A.W., Rümmer, P.: Regular Model Checking Revisited. In: Olderog, E.R., Steffen, B., Yi, W. (eds.) Model Checking, Synthesis, and Learning: Essays Dedicated to Bengt Jonsson on The Occasion of His 60th Birthday, pp. 97–114. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-91384-7_6, https://doi.org/10.1007/978-3-030-91384-7_6
13. Podelski, A., Rybalchenko, A.: Transition invariants. In: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004. pp. 32–41. IEEE, Turku, Finland (2004). https://doi.org/10.1109/LICS.2004.1319598, http://ieeexplore.ieee.org/document/1319598/
14. Podelski, A., Rybalchenko, A.: Transition Invariants and Transition Predicate Abstraction for Program Termination. In: Abdulla, P.A., Leino, K.R.M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 3–10. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19835-9_2
15. Sipser, M.: Introduction to the Theory of Computation; 2nd ed. Thomson Course Technology, Cambridge (2006)
16. Takeuti, G., Zaring, W.M.: Introduction to axiomatic set theory. New York : Springer-Verlag (1982), http://archive.org/details/introductiontoax00take

17. Welzel-Mohr, C.: Inductive Statements for Regular Transition Systems. Ph.D. thesis, Technische Universität München (2024), https://mediatum.ub.tum.de/1721365