Regular Model Checking with Regular Relations

Vrunda Dave[1], Taylor Dohmen[2(✉)], Shankara Narayanan Krishna[1], and Ashutosh Trivedi[2]

[1] IIT Bombay, Mumbai, India
{vrunda,krishnas}@cse.iitb.ac.in
[2] Univeristy of Colorado, Boulder, USA
{taylor.dohmen,ashutosh.trivedi}@colorado.edu

Abstract. Regular model checking is an exploration technique for infinite state systems where state spaces are represented as regular languages and transition relations are expressed using rational relations over infinite (or finite) strings. We extend the regular model checking paradigm to permit the use of more powerful transition relations: the class of regular relations, of which the rational relations are a strict subset. We use the language of monadic second-order logic (MSO) on infinite strings to specify such relations and adopt streaming string transducers (SSTs) as a suitable computational model. We introduce nondeterministic SSTs over infinite strings (ω-NSSTs) and show that they precisely capture the relations definable in MSO. We further explore theoretical properties of ω-NSSTs required to effectively carry out regular model checking. In particular, we establish that the regular type checking problem for ω-NSSTs is decidable in PSPACE. Since the post-image of a regular language under a regular relation may not be regular (or even context-free), approaches that iteratively compute the image can not be effectively carried out in this setting. Instead, we utilize the fact that regular relations are closed under composition, which, together with our decidability result, provides a foundation for regular model checking with regular relations.

1 Introduction

Regular model checking [2,3,13,24,31] is a symbolic exploration and verification technique where sets of configurations are expressed as regular languages and transition relations are encoded as rational relations [27–29] in the form of generalized sequential machines. A generalized sequential machine (GSM) is essentially a finite state machine with output capability; on every transition an input symbol is read, the state changes, and a finite string is appended to an output string (see Fig. 1, for instance, where the label α/s indicates that the machine reads the symbol α and writes the string s on any such transition). While regular model checking is undecidable in general, a number of approximation schemes and heuristics [1,8,12,13,18,22,23,30] have made it a practical verification approach. It has, for example, been applied to verify programs with

E. Bampis and A. Pagourtzis (Eds.): FCT 2021, LNCS 12867, pp. 190–203, 2021.
https://doi.org/10.1007/978-3-030-86593-1_13

unbounded data structures such as lists and stacks [3,13]. Moreover, since infinite strings over a finite alphabet can be naturally interpreted as real numbers in the unit interval, regular model checking over infinite strings provides a framework [7,9,10,14,25,26] to analyze properties of dynamical systems.

This paper generalizes the regular model checking approach so that transition relations can be expressed using *regular relations* over infinite strings. We propose the computational model of nondeterministic streaming string transducers on infinite strings (ω-NSST), and explore theoretical properties of ω-NSSTs required to effectively carry out regular model checking.

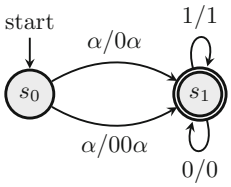

Fig. 1. A GSM that shifts a string to the right by 1 or 2, or equivalently realizing division of the binary encoding of real numbers in $[0, 1]$ by 2 or 4.

Regular Relations. While rational relations are capable of modelling a rich set of transition systems, their limitations can be observed by noting their inability to express common transformations such as copy $\stackrel{\text{def}}{=}$ $w \mapsto ww$ and reverse $\stackrel{\text{def}}{=}$ $w \mapsto \overleftarrow{w}$, where the string $\overleftarrow{w}$ is the reverse of the string w. Courcelle [16,17] initiated the use of monadic second-order logic (MSO) in defining deterministic and nondeterministic graph-to-graph transformations which are known to include some non-rational transformations like copy and reverse. Engelfriet and Hoogeboom [20] showed that deterministic MSO-definable transformations (DMSOT) over finite strings coincide exactly with the transformations that can be realized by generalizations of GSMs that can read inputs in two directions (2GSM). Furthermore, they showed that this correspondence does not extend to the set of nondeterministic MSO-definable transformations (NMSOT) and nondeterministic 2GSMs (N2GSM).

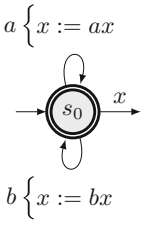

Fig. 2. SST implementing reverse. Here, x is a string variable and input strings ending in the final state s_0 output variable x (as shown by the label on the outgoing arrow from s_0.)

Alur and Černý [4] proposed a one-way machine capable of realizing the same transformations as DMSOTs. These machines, known as *streaming string transducers* (SST), work by storing and combining partial outputs in a finite set of variables, and enjoy a number of appealing properties including decidability of functional equivalence and type-checking (see Fig. 2 for an SST realization of reverse). Alur and Deshmukh followed up this work by introducing nondeterministic streaming string transducers (NSST) as a natural generalization [5] and proved this model captures precisely the same set of relations as NMSOTs. Since the connection between automata and logic is often used as a yardstick for regularity, MSO-definable functions and relations over finite strings are often called regular functions and regular relations.

Regular Relations over Infinite Strings. The expressiveness of SSTs and MSO-definable transformations also coincide when representing functions over infinite strings [6]. Deterministic SSTs operating on infinite strings are known as ω-DSSTs, however, for regular relations of infinite strings, no existing computational model exists. We combine and generalize results in the literature on NSSTs and ω-DSSTs to propose the computational model of nondeterministic streaming ω-string transducers (ω-NSST) capturing regular relations of ω-strings.

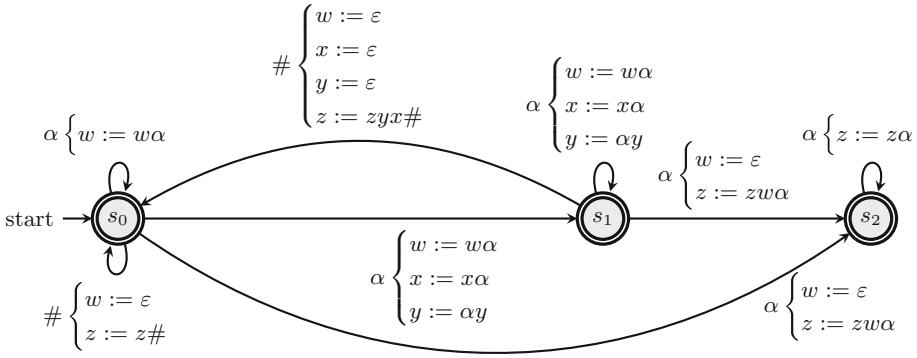

Fig. 3. An ω-NSST implementing the relation $R_{\overleftarrow{u}\,u}$ from Example 1. Let α denote all symbols in A, excluding #. Variable w remembers the string since the last #, while x and y store the chosen suffix and its reverse. The output variable is z.

Example 1. Let A be a finite alphabet and # be a special separator not in A. For $u, v \in A^*$, we say that $v \preceq u$ if v is a suffix of u. Consider a relation $R_{\overleftarrow{u}\,u}$ that transforms strings in $(A \cup \{\#\})^\omega$ such that each maximal #-free finite substring u occurring in the input string is transformed into $\overleftarrow{v}\,v$ for some suffix v of u. Formally, $R_{\overleftarrow{u}\,u}$ is defined as

$$\{(u_1\# \cdots \#u_n\#w, \overleftarrow{v_1}v_1\# \cdots \#\overleftarrow{v_n}v_n\#w) : u_i, v_i \in A^*, w \in A^\omega, \text{ and } v_i \preceq u_i\}$$
$$\cup \{(u_1\#u_2\# \ldots, \overleftarrow{v_1}v_1\#\overleftarrow{v_2}v_2\# \ldots) : u_i, v_i \in A^* \text{ and } v_i \preceq u_i\},$$

and can be implemented as an ω-NSST with Büchi acceptance condition (accepting states are visited infinitely often for accepting strings) as shown in Fig. 3.

Contributions and Outline. In Sect. 2 we introduce ω-NSSTs and their semantics as a computational model for regular relations. In Sect. 3 we prove that the ω-NSST-definable relations coincide exactly with MSO-definable relations of infinite strings. In Sect. 4 we consider regular model checking with regular relations. To enable regular model checking with regular relations, we study the following key verification problem. The *type checking problem* for ω-NSSTs asks to decide, given two ω-regular languages L_1, L_2 and an ω-NSST, whether $[\![T]\!](L_1) \subseteq L_2$, where $[\![T]\!]$ is the regular relation implemented by T. We show that type checking for ω-NSSTs is decidable in PSPACE.

2 Regular Relations for Infinite Strings

An alphabet A is a finite set of letters. A *string* w over an alphabet A is a finite sequence of symbols in A. We denote the empty string by ε. We write A^* for the set of all finite strings over A, and for $w \in A^*$ we write $|w|$ for its length. A language L over A is a subset of A^*. An *ω-string* x over A is a function $x : \mathbb{N} \to A$, and written as $x = x(0)x(1)\cdots$. We write A^ω for the set of all ω-strings over A, and A^∞ for $A^* \cup A^\omega$. An ω-language L over A is a subset of A^ω.

2.1 MSO Definable Relations

Strings may be viewed as ordered structures encoded over the signature $\mathcal{S}_A = \{(a)_{a\in A}, <\}$ and interpreted with respect to A^* or A^ω. The domain of a string in this context refers to the set of valid positions in the string, and the relation $<$ in $\mathcal{S}_A$ ranges over this domain. The expression $a(x)$ holds true if the symbol at position x is a, and $x < y$ holds if x is a lesser index than y.

Formulae in MSO over $\mathcal{S}_A$ are defined relative to a countable set of first-order variables $x, y, z, \ldots$ that range over individual elements of the domain and a countable set of second-order variables $X, Y, Z, \ldots$ that range over subsets of the domain. The syntax for well-formed formulae is given as:

$$\phi ::= \exists X.\ \phi \mid \exists x.\ \phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid a(x) \mid x < y \mid x \in X$$

MSO transducers are particular specifications in this logic that define transformations between strings. Intuitively, each such transducer copies each input string some fixed number of times and treats the positions in each copy as nodes in a graph, which are then relabeled and and rearranged in accordance with the formulae of the transducer to produce an output.

Definition 1. *A deterministic* MSO *ω-string transducer (ω-*DMSOT*) is a tuple*

$$\left(A, B, \mathsf{dom}, N, (\phi_b^n(x))_{b\in B}^{n\in N}, (\psi^{n,m}(x,y))^{n,m\in N}\right),$$

where A and B are input and output alphabets, $N = \{1, \ldots, n\}$ is a set of copy indices, dom *is an* MSO *sentence that defines an input language, the* node formulae $(\phi_b^n(x))_{b\in B}^{n\in N}$ *specify the labels of positions in the output, and the* edge formulae $(\psi^{n,m}(x,y))^{n,m\in N}$ *specify which positions in the output will be adjacent.*

A ω-DMSOT operates over N disjoint copies of the string graph of an input. Each formula ϕ_b^n has a single free variable and should be interpreted such that if a position satisfies ϕ_b^n, then that position will be labeled by the symbol b in the n^{th} disjoint string graph comprising the output. Each formula $\psi^{(n,m)}$ has two free variables and a satisfying pair of indices indicates that there is a link between the former index in copy n and the latter index in copy m.

Nondeterminism is introduced through additional set variables $X_1, \ldots, X_k$ called *parameters*. Fixing a valuation—sets of positions of the input graph satisfying the domain formula—of these parameters determines an output graph,

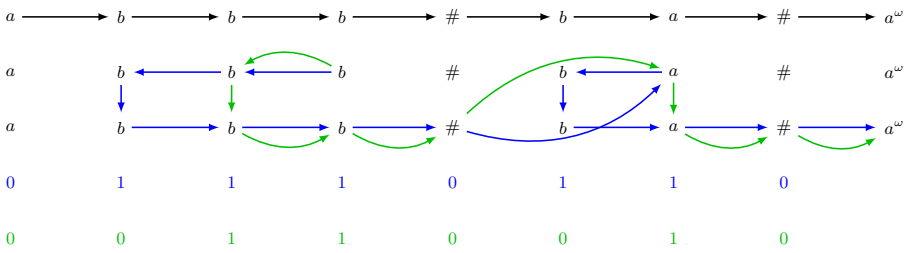

Fig. 4. Two possible outputs of the relation given in Example 1 constructed according ot the ω-NMSOT from Example 2.

just as in the deterministic case. Each possible valuation may result in a different output graph for the same input graph, and thus nondeterminism arises from the choice of valuation.

Definition 2. *A nondeterministic* MSO *ω-string transducer (ω-*NMSOT*) with k free set variables $\mathbf{X}_k = (X_1, \ldots, X_k)$ is given as a tuple*

$$\left(A, B, \mathsf{dom}(\mathbf{X}_k), N, (\phi_b^n(x, \mathbf{X}_k))_{b\in B}^{n\in N}, (\psi^{n,m}(x, y, \mathbf{X}_k))^{n,m\in N}\right),$$

where all formulae are parameterized by the free second-order variables in addition to the required first-order parameters.

A relation between strings is a *regular relation* if it is definable by a ω-NMSOT. Since ω-DMSOTs can map each input to at most one output, the relations definable by ω-DMSOTs are called the *regular functions*.

Example 2. We now describe a ω-NMSOT capturing the relation given in Example 1. Set $A = \{a, b, \#\} = B$, $N = \{1, 2\}$, and consider a single parameter $\mathbf{X}_1 = \{X_1\}$. The domain of the relation is simply A^ω, so we omit the formula. For all symbols $\beta \in B$ and copy indices $n \in N$, the node formulae labels each position with the same symbol as the corresponding position in the input string: $\phi_\beta^n(x, X_1) \stackrel{\text{def}}{=} \beta(x)$. We omit formal specifications of the edge formulae (which can be found in the extended version of this work [19]) and describe them informally. The formula for edges from copy 1 to copy 1 connects adjacent non-# positions that belong to X_1 in the reverse order. The formula for edges from copy 1 to copy 2 connects non-# positions to themselves when the predecessor position is not in X_1. The formula for edges from copy 2 to copy 2 links the right-most sequence of positions in X_1 that preceed a # symbol and also connect all those positions coming after the final # if required. Finally, the formula for edges from copy 2 to copy 1 links # symbols to the last position in X_1 left of the next #.

Two possible outputs from the relation of Example 1 are displayed in Fig. 4 which shows how the above ω-NMSOT constructs an output string for two different valuations of X_1. A 1 in the blue (resp. green) row signifies that the position at that column is in X_1, while a 0 indicates that it is not in X_1.

2.2 Nondeterministic Streaming String Transducers

Definition 3. *A nondeterministic streaming string transducer T over ω-strings (ω-NSST) is a tuple $(A, B, S, I, \mathsf{Acc}, \Delta, f, X, U)$, where*

- *A and B are finite input and output alphabets,*
- *S is a finite set of states,*
- *$I \subseteq Q$ is a set of initial states,*
- *Acc is an acceptance condition,*
- *X is a finite set of string variables,*
- *U is a finite set of variable update functions of type $X \to (X \cup B)^*$,*
- *Δ is a transition function of type $(S \times A) \to 2^{U \times S}$, and*
- *$f \in X$ is an append-only output variable.*

Such a machine is deterministic (a ω-DSST) if $|\Delta(s, a)| = 1$, for all states $s \in S$ and symbols $a \in A$, and $|I| = 1$; it is nondeterministic otherwise.

On each transition $s_k \xrightarrow[u_k]{a_k} s_{k+1}$, the transducer changes state and applies the update u_k to each variable of X in parallel. An ω-NSST is *copyless* if every variable in X occurs at most once in the image $\mathsf{im}(u)$ of every update $u \in U$. Alternately stated, an update $u \in U$ is copyless if the string $u(x_0)u(x_1)\dots u(x_{n-1})$ has at most one occurrence of each $x \in X$, and an ω-NSST is copyless if all of its updates are copyless.

A run of an ω-NSST on an infinite string $a_1 a_2 \cdots \in A^\omega$ is an infinite sequence of states and transitions $s_0 \xrightarrow[u_0]{a_0} s_1 \xrightarrow[u_1]{a_1} \dots$ where $s_0 \in I$ and $(s_{k+1}, u_k) \in \Delta(s_k, a_k)$ for all $k \in \mathbb{N}$. Let $\mathsf{Runs}_T(w)$ be the set of all runs in T, given input w. An update function $u : X \to (X \cup B)^*$ can easily be extended to $\widehat{u} : (X \cup B)^* \to (X \cup B)^*$ such that $\widehat{u}(w) \stackrel{\text{def}}{=} \varepsilon$ if $w = \varepsilon$, $\widehat{u}(w) \stackrel{\text{def}}{=} b\widehat{u}(w')$ if $w = bw'$, and $u(x)\widehat{u}(w')$ if $w = xw'$. The effect of two updates $u_1, u_2 \in U$ in sequence can be *summarized* by the function composition $\widehat{u}_1 \circ \widehat{u}_2$; likewise a sequence of updates of arbitrary length would be summarized by $\widehat{u}_0 \circ \widehat{u}_1 \circ \ldots \circ \widehat{u}_{n-1}$. For notational convenience, we often omit the hats when the extension is clear from context. Notice that if all updates in a sequence of compositions are copyless, then so is the entire summary.

A valuation is a function $X \to B^*$ mapping each variable to a string value. The initial valuation val_ε of all variables is the empty string ε. A valuation is well-defined after any finite prefix r_n of a run r and is computed as a composition of updates occurring on this prefix: $\mathsf{val}_{r_n} = \mathsf{val}_\epsilon \circ u_0 \circ u_1 \circ \cdots \circ u_{n-1}$. The output $T(r) \stackrel{\text{def}}{=} \lim_{n\to\infty} \mathsf{val}_{r_n}(f)$ of T on r is well-defined only if r is accepted by T. Since the output variable f is only ever appended to and never prepended, this limit exists and is an ω-string whenever r is accepted, otherwise we set $T(r) = \bot$. The relation $[\![T]\!]$ realized by an ω-NSST T is given by $[\![T]\!] \stackrel{\text{def}}{=} \{(w, T(r)) : r \in \mathsf{Runs}_T(w)\}$. An ω-NSST T is *functional* if for every w the set $\{w' : (w, w') \in [\![T]\!]\}$ has cardinality at most 1.

We consider both Büchi and Muller acceptance conditions for ω-NSSTs and reference these classes of machines by the initialisms NBT and NMT (DBT and

DMT for their deterministic versions), respectively. For a run $r \in \mathsf{Runs}_T(w)$, let $\mathsf{Inf}(r) \subseteq S$ denote the set of states visited infinitely often.

1. A *Büchi acceptance condition* is given by a set of states $F \subseteq S$ and is interpreted such that a NBT is defined on an input $w \in A^\omega$ if there exists a run $r \in \mathsf{Runs}_T(w)$ for which $\mathsf{Inf}(r) \cap F \neq \emptyset$.
2. A *Muller acceptance condition* is given as a set of sets $\mathbb{F} = \{F_0, \ldots, F_n\} \subseteq 2^S$, interpreted such that a NMT is defined on input $w \in A^\omega$ if there exists a run $r \in \mathsf{Runs}_T(w)$ for which $\mathsf{Inf}(r) \in \mathbb{F}$.

Proposition 1. *A relation is* NBT *definable if, and only if, it is* NMT *definable.*

The equivalence of NBT and NMT -definable relations follows from a straightforward application of the equivalence of nondeterministic Büchi automata and nondeterministic Muller automata. Equivalence of these acceptance conditions in transducers allows us to switch between them whenever convenient.

Remark 1. Observe that DMTs and functional NMTs, both of which were introduced in [6], have a slightly different output mechanism, which is defined as a function $\Omega : 2^S \rightharpoonup X^*$ such that the output string $\Omega(S')$ is copyless and of the form $x_1 \ldots x_n$, for all $S' \subseteq S$ for which $\Omega(S') \neq \perp$. Furthermore, there is the condition that if $s, s' \in S'$ and $a \in A$ s.t. $(u, s') \in \Delta(s, a)$, then (1) $u(x_k) = x_k$ for all $k < n$ and (2) $u(x_n) = x_n w$ for some $w \in (X \cup B)^*$.

In contrast, our definition has a unique append-only output variable $f \in X$. However, our model with the Muller acceptance is as expressive as that studied in [6]. One can use nondeterminism to guess a position in the input after which states in a Muller accepting set S' will be visited infinitely often. The output function can be defined by guessing a Muller set, and keeping an extra variable for the output. Upon making the guess, it will move the contents of $x_1 \ldots x_n$ to the variable f and make a transition to a copy $T_{S'}$ of the transducer where $\mathsf{Acc} = \{S'\}$. If any state outside the set S' is visited, or the variables $x_1 \ldots, x_{n-1}$ are updated, or the variable f is assigned in non-appending fashion, then $T_{S'}$ makes a transition to a rejecting sink state. Alur, Filiot, and Trivedi [6] showed the equivalence of functional NMT with DMT. This implies that the transductions definable using *functional* NMTs or *functional* NBTs (in our definition) are precisely those definable by ω-DMSOT.

3 Equivalence of ω-NMSOT and ω-NSST

Alur and Deshmukh [5] showed that relations over finite strings definable by nondeterministic MSO transducers coincide with those definable by nondeterministic streaming string transducers. We generalize this result by proving that a relation is definable by an ω-NMSOT if, and only if, it is definable by an ω-NSST. We provide symmetric arguments to connect ω-NSST, ω-DSST and ω-NMSOT, ω-DMSOT, resulting in a simple proof.

Our arguments use the concept of a relabeling relation, following Engelfriet and Hoogeboom [20]. A relation $\rho \subseteq A^\omega \times B^\omega$ is a *relabeling*, if there exists another relation $\rho' \subseteq A \times B$ such that $(aw, bv) \in \rho$ iff $(a, b) \in \rho'$ and $(w, v) \in \rho$. In other words, ρ is obtained by lifting the letter-to-letter relation ρ', in a straightforward manner, to ω-strings. Let $\mathsf{Let}(\rho)$ denote the letter to letter relation $\rho' \subseteq A \times B$ corresponding to ρ and let RL be the set of all such relabelings.

Theorem 1. ω-NMSOT $= \omega$-NSST.

The proof of Theorem 1 proceeds in two stages. In the first part (Lemma 1), we show that every ω-NSST is equivalent to the composition of a nondeterministic relabeling and a ω-DSST. In the second part (Lemma 2), we show that every ω-NMSOT is equivalent to the composition of a nondeterministic relabeling and a ω-DMSOT. These two lemmas, in conjunction with the equivalence of DMTs and functional NMTs [6], allow us to equate these two models of transformation via a simple assignment.

Lemma 1. ω-NSST $= \omega$-DSST $\circ$ RL

Proof. We first show ω-DSST $\circ$ RL $\subseteq \omega$-NSST by proving that for every DMT $T \stackrel{\text{def}}{=} (B, C, S, I, \mathbb{F}, \Delta, f, X, U)$ and nondeterministic relabeling $\rho \subseteq A^\omega \times B^\omega$, there is a NMT $T' \stackrel{\text{def}}{=} (A, C, S, I, \mathbb{F}, \Delta_\rho, f, X, U)$ such that $[\![T']\!] = [\![T]\!] \circ \rho$. As indicated by the tuple given to specify T', the only distinct components between the two machines are their input alphabets and their transition functions Δ and Δ_ρ. The latter is given as $\Delta_\rho \stackrel{\text{def}}{=} (s, a) \mapsto \bigcup_{(a,b) \in \mathsf{Let}(\rho)} \Delta(s, b)$. The nondeterminism of ρ is therefore captured in Δ_ρ. This results in a unique run through T', for every possible relabeling of inputs for T. Since the remaining pieces of T are untouched in the process of constructing T', it is clear that $[\![T']\!] = [\![T]\!] \circ \rho$.

What remains to be shown is the inclusion ω-NSST $\subseteq \omega$-DSST $\circ$ RL: for any NMT $T \stackrel{\text{def}}{=} (A, B, S, I, \mathbb{F}, \Delta, f, X, U)$, there exists a DMT T' and a nondeterministic relabeling ρ such that $[\![T]\!] = [\![T']\!] \circ \rho$. From T, we can construct a nondeterministic, letter-to-letter relation $\rho' \subseteq A \times (U \times S)$ as follows: $\rho' \stackrel{\text{def}}{=} \{(a, (u, s')) : (u, s') \in \Delta(s, a)\}$. Now let $\rho \subseteq A^\omega \times (U \times S)^\omega$ be the extension of ρ' as described previously. The relation ρ contains the set of all possible runs through T for any possible input in A^ω.

Next, we construct a DMT $T' \stackrel{\text{def}}{=} (U \times S, B, S, I, \mathbb{F}, \Delta_\rho, f, X, U)$ with transition function $\Delta_\rho \stackrel{\text{def}}{=} (s, (u, s')) \mapsto \{(u, s') : (u, s') \in \Delta(s, a) \text{ for some } a \in A\}$. Consequently, T' retains only the pairs in ρ which correspond to valid runs T and encodes them as ω-strings over the alphabet $S \times U$. The DMT T' then simply follows the instructions encoded in its input and thereby simulates only legitimate runs through T. Thus, we may conclude that $[\![T]\!] = [\![T']\!] \circ \rho$. □

Lemma 2. ω-NMSOT $= \omega$-DMSOT $\circ$ RL.

Proof. We begin by showing the inclusion ω-NMSOT $\subseteq \omega$-DMSOT $\circ$ RL: for any ω-NMSOT T, there exists an ω-DMSOT T' and a relabeling ρ such that $[\![T]\!] = [\![T']\!] \circ \rho$. Nondeterministic choice in T is determined by the choice of assignment to

free variables in $\mathbf{X}_k$. Alternatively, the job of facilitating nondeterminism can be placed upon a relabeling relation, thereby allowing us to remove the parameter variables. Define a letter-to-letter relation $\rho' \subseteq A \times (A \times \{0,1\}^k)$ as follows: $\rho' \stackrel{\text{def}}{=} \left\{(a,(a,b)) : b \in \{0,1\}^k\right\}$, and let the relabeling $\rho \subseteq A^\omega \times (A \times \{0,1\}^k)^\omega$ be its extension. This relabeling essentially gives us a new alphabet such that each symbol from A is tagged with encodings of its membership status for each set parameter from $\mathbf{X}_k$. Now, we can construct an ω-DMSOT T' that is identical to T, apart from two distinctions. Firstly, T' is deterministic (i.e. it has no free set variables), and every occurrence of a subformula $x \in X_i$ in T is replaced by a subformula $\bigvee_{b \in \{0,1\}^k \wedge b[i]=1} (a,b)(x)$ in T'. As a result of this encoding, the equality $[\![T]\!] = [\![T']\!] \circ \rho$ holds.

The converse inclusion, ω-DMSOT$\circ$RL $\subseteq$ ω-NMSOT, is much simpler. Every relabeling ρ in RL is ω-NMSOT definable: consider $\rho' = \mathsf{Let}(\rho) \subseteq A \times B$. The ω-NMSOT specifying ρ is similar to identity/copy, except that here we have that the output label is b iff the input label is a and $(a,b) \in \rho'$. This can be implemented using second-order variables X_b for all $b \in B$. Let $\mathbf{X}_B$ represent this set. Only a single copy is required to produce the output. Node formulae are given by $\phi^1_b(x, \mathbf{X}_B) \stackrel{\text{def}}{=} \bigvee_{a \in A} \bigvee_{(a,b) \in \rho'} (a(x) \wedge x \in X_b)$, and the edge formulae by $\psi^{1,1}(x, y, \mathbf{X}_B) \stackrel{\text{def}}{=} x < y$. It is known that ω-NMSOT are closed under composition [17]. Thus, we conclude that any composition of a nondeterministic relabeling and a ω-DMSOT is definable by a ω-NMSOT and that ω-MSOT$\circ$RL $\subseteq$ ω-NMSOT. □

In conjunction Lemmas 1 and 2 along with the results of [6] allow us to write the following equation, thereby proving Theorem 1.

$$\omega\text{-NMSOT} = \omega\text{-DMSOT} \circ \text{RL} = \text{DMT} \circ \text{RL} = \text{NMT} = \omega\text{-NSST}$$

4 MSO-Definable Regular Model Checking

In this section, we explain how algorithms for deciding properties of regular relations can be used to perform regular model checking. Given two relations T_1 and T_2, their *sequential composition* is $[\![T_2 \circ T_1]\!] \stackrel{\text{def}}{=} \{(x,z) : (x,y) \in [\![T_1]\!], (y,z) \in [\![T_2]\!]\}$. Let T^k denote the k-fold composition of a relation T with itself. Let T^* denote the transitive closure of T.

Suppose that INIT and BAD are regular languages representing sets of states in some system that are initial, and unsafe, respectively. Given a generic transition relation T which captures the dynamics of the system, the *regular model checking problem* asks to decide whether any element of BAD is reachable from any element of INIT via repeated applications of T. In precise terms, the regular model checking problem asks to decide whether the equation $[\![T^*]\!](\text{INIT}) \cap \text{BAD} = \emptyset$ holds. Bounded model checking, in this setting, asks to decide, given $n \in \mathbb{N}$, whether $[\![T^k]\!](\text{INIT}) \cap \text{BAD} = \emptyset$ holds, for all $k \leq n$. Unbounded model checking is undecidable (cf. [19] for a proof), so we focus on bounded model checking.

When T is a rational relation, its image is always a regular language, and this permits the approach of iteratively applying T from INIT and checking whether this set intersects with BAD by standard automata-theoretic methods. If T is a regular relation, its image may not be a regular language, and we must iteratively compute compositions of T with itself and test whether these compositions enter the BAD language. To allow this, we establish decidability of the *type checking problem* for ω-NSSTs: given two ω-regular languages L_1, L_2 and an ω-NSST T, decide if the inclusions $L_1 \subseteq \mathsf{dom}(T)$ and $[\![T]\!](L_1) \subseteq L_2$ hold.

Theorem 2. *The type checking problem for* ω-NSST*s is decidable in* PSPACE.

Proof. Suppose that $T \stackrel{\text{def}}{=} (A, B, S, I, F, \Delta, f, X, U)$ is an NBT and $L_1 \subseteq A^\omega$ and $L_2 \subseteq B^\omega$ are ω-regular languages, encoded, respectively, as deterministic Muller automata (DMA) M_1 and M_2. We first check whether T is defined for all ω-strings $w \in L_1$, i.e. whether $L_1 \subseteq \mathsf{dom}(T)$. A nondeterministic Büchi automaton (NBA) $\mathcal{C}$ that recognizes the domain of T can be constructed in linear time by ignoring variables and output mechanism. The inclusion $L_1 \subseteq \mathsf{dom}(T)$ can be decided in PSPACE by checking emptiness of $M_1' \cap \mathcal{C}$ where M_1' is the NBA equivalent to M_1 and $\mathcal{C}$ is the NBA representing the complement language of $\mathsf{dom}(T)$. It is known that an NBA can be constructed from a DMA with exponential blowup in the number of states [11]. A complement automaton can be constructed for an NBA with exponential increase in the number of states as well [11]. Hence $\mathcal{C}$ has exponentially many states relative to T and M_1. Intersection of M_1' and $\mathcal{C}$ is a standard product construction with a flag so that both M_1' and $\mathcal{C}$ visit good states infinitely often. Thus the intersection NBA $M_1' \cap \mathcal{C}$ has exponentially many states relative to T and M_1. Thanks to the fact that emptiness of NBA can be checked in NLOGSPACE [11], the emptiness of this product automaton, can be decided in NPSPACE = PSPACE.

We now assume that T is well-defined on L_1 and construct a nondeterministic Muller automaton (NMA) $\mathcal{A}$ such that the language of $\mathcal{A}$ is defined as $\{w \in L_1 : \exists w' \in [\![T]\!](w) \text{ s.t. } w' \notin L_2\}$. Next, we construct a DMA $\overline{M_2}$ for $\overline{L_2}$ by complementing the Acc set. The automaton $\mathcal{A}$ simulates M_1, T and $\overline{M_2}$ in parallel. Next, we construct an NMT T' corresponding to the NBT T in order to homogenize the acceptance condition accross these machines. Let us fix the definition for all three machines: (i) $M_1 \stackrel{\text{def}}{=} (A, S_1, p_0, \mathbb{F}_1, \Delta_1)$, (ii) $T' \stackrel{\text{def}}{=} (A, B, S, I, \mathbb{F}', \Delta, f, X, U)$, (iii) $\overline{M_2} \stackrel{\text{def}}{=} (B, S_2, r_0, \mathbb{F}_2, \Delta_2)$.

The NMA $\mathcal{A}$ is defined as the product of M_1 and T' (without the output mechanism), and it stores a state summary map—i.e. the effect of running current valuation of each variable starting from all states of $\overline{M_2}$—in each of its own states. Formally, the states of $\mathcal{A}$ comprise a finite subset of $S_1 \times S \times (S_2 \times X \to S_2 \cup \{\bot\})$. A state (q, p, g) with $g(r, x) = r'$ represents that, starting from state r, if we read the current value of variable x, then we reach state r'. If $g(r, x) = \bot$, it indicates that there is no run on valuation of x starting from r. This information can be updated along the run of $\mathcal{A}$. For instance, if a transition of T updates x as $aybx$, then the summary map g is updated to g' such that $g'(r, x) = g(\Delta_2(g(\Delta_2(r, a), y), b), x)$, and summarizes the effect of reading $x = aybx$ in $\overline{M_2}$ starting from state r.

The set of states of $\mathcal{A}$ is $S_{\mathcal{A}} = S_1 \times S \times (S_2 \times X \to S_2 \cup \{\bot\})$, in which S_1, S, and S_2 represent the state sets of M_1, T', and $\overline{M_2}$, respectively. The transition relation $\Delta_{\mathcal{A}}$ is defined such that $(q', p', g') \in \Delta_{\mathcal{A}}((q, p, g), a)$ iff (i) $\Delta_1(q, a) = q'$, (ii) $(u, p') \in \Delta_1(p, a)$, and (iii) $g'(r, x) = r'$ and $\Delta_2(r, \mathsf{val}_{u(x)}) = r'$, for all $x \in X$ and $r \in S_2$,. Initial states are the product of initial states i.e. a set $I_{\mathcal{A}} = \{(q_0, p_0, r_0) : q_0 \in I\}$. The Muller accepting set of $\mathcal{A}$ is defined as the collection of all $P \subseteq S_{\mathcal{A}}$ such that (i) $\pi_1(P) \in \mathbb{F}_1$, (ii) $\pi_2(P) \in \mathbb{F}$, and (iii) $(\pi_3(P))(r_0, f) \in \mathbb{F}_2$, where π_i is the i^{th} projection. The size of NMA $\mathcal{A}$ is exponential in the number variables of T, polynomial in the number of states of M_1 and T. Thanks to the fact that emptiness of an NMA can be determined in NLogSpace [11], emptiness of $\mathcal{A}$ having exponential states in the inputs T, M_1 and M_2, can be decided in NPspace and thus, by Savitch's theorem, also in Pspace. □

Since regular relations are definable in MSO, they are closed under sequential composition. In combination with Theorems 1 and 2, this establishes the necessary conditions for bounded regular model checking with regular relations to be possible. Thus, we have the following corollary.

Corollary 1. *Bounded model checking with regular relations is decidable.*

Despite the fact that unbounded regular model checking is undecidable, bounded regular model checking provides a refutation procedure. That is, it allows us to search for a witness for proving the system unsafe. Unfortunately, we cannot use bounded model checking of this kind to decide if the system does satisfy the desired property. On the other hand, we identify several special cases of the problem which permit the safety of the system to be verified in finite time. In general, we assume that $\textsc{init} \subseteq \overline{\textsc{bad}}$, where $\overline{\textsc{bad}}$ is the complement of bad.

Functional Fixed Points. The first instance applies when T is functional, i.e. $[\![T]\!]$ is a function, and relies on the following result of Alur, Filiot, and Trivedi [6].

Theorem 3. *Given an ω-NSST T, it is decidable if $[\![T]\!]$ is a function. Given a pair of functional ω-NSSTs T_1 and T_2, it is decidable if $[\![T_1]\!] = [\![T_2]\!]$.*

At every step of the bounded regular model checking procedure, one can check if T^k is functional, if T^{k+1} is functional, and if $[\![T^k]\!] = [\![T^{k+1}]\!]$. If these three conditions hold, then, for all $m \geq 0$, we have that $[\![T^k]\!] = [\![T^{k+m}]\!]$. When this occurs and $[\![T^k]\!](\textsc{init}) \subseteq \overline{\textsc{bad}}$ holds, it follows that $[\![T^k]\!] = [\![T^*]\!]$ and therefore that $[\![T^*]\!](\textsc{init}) \subseteq \overline{\textsc{bad}}$ which implies $[\![T^*]\!](\textsc{init}) \cap \textsc{bad} = \emptyset$. Note that T^k can be functional even when T is not. To see this, consider a non-functional ω-NSST T such that $[\![T]\!](a^\omega) = \{b^\omega, c^\omega\}$, and $[\![T]\!](b^\omega) = d^\omega = [\![T]\!](c^\omega)$. If $a^\omega \in \textsc{init}$ and $|[\![T]\!](w)| = 1$ for every other input w and $a^\omega \notin \mathsf{im}(T)$, then T^2 is functional.

Inductive Invariants. An alternative approach involves showing that $[\![T]\!]$ satisfies some inductive invariant. Select, as a candidate invariant, a regular or ω-regular language L which is contained in the set of safe states $L \subseteq \overline{\textsc{bad}}$. Now, L provides a witness to the unbounded safety of the system if the following pair of conditions

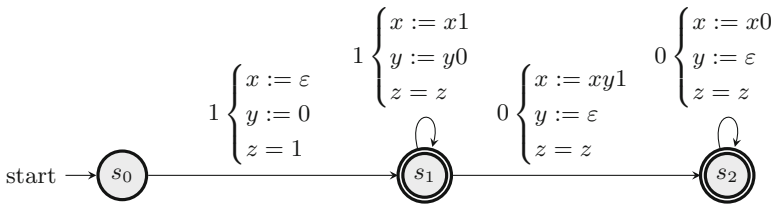

Fig. 5. An ω-SST squaring a number with binary expansion of the form 1^n0^ω. The output at s_1 and s_2 is x. Notice that this function can not be expressed as a GSM.

are met: (i) $\text{INIT} \subseteq L$ and (ii) $[\![T]\!](L) \subseteq L$. Together, (i) and (ii) imply that $[\![T^*]\!](\text{INIT}) \subseteq L$, and in combination with the assumption that $L \subseteq \overline{\text{BAD}}$ this yields that $[\![T^*]\!](\text{INIT}) \cap \text{BAD} = \emptyset$. The necessary inclusions can be formulated as instances of the type checking problem, and so, given an appropriately chosen inductive invariant in the form of an ω-regular language, the global safety of such a system may be verified in polynomial space. This method is easily generalized by searching for k-inductive invariants: ω-regular languages for which there is a $k \in \mathbb{N}$ such that $[\![T^k]\!](L) \subseteq L$. The k-inductive approach complements bounded regular model checking, since, for a given k, bounded regular model checking lets us decide if the system is safe for up to k transitions while k-induction lets us decide if it is safe after at least k transitions.

5 Conclusion

We introduced ω-NSSTs as a computational model for regular relations over infinite strings, and showed that the relations definable by ω-NSST coincide exactly with those definable in MSO. Motivated by potential applications in formal verification, we studied algorithmic properties of these objects and established the minimal theoretical results required for bounded regular model checking to be possible with regular transition relations.

Regular functions and relations provide an intriguing class of models for real valued functions, see Fig. 5 for example. In [15,21] analytic properties such as continuity and differentiability of real functions encoded by ω-automata have been studied. Extending this line of research by going beyond standard ω-automata is both theoretically interesting and could be leveraged towards applications involving verification and control of dynamical systems. The present work indicates the viability of generalizing the automata-theoretic approach to modeling real functions. With this application in mind, it would be worthwhile to study the approximation techniques developed for traditional regular model checking to see if they generalize to handle regular relations.

References

1. Abdulla, P.A., Jonsson, B., Nilsson, M., d'Orso, J.: Regular model checking made simple and effcient. In: Brim, L., Křetínský, M., Kučera, A., Jančar, P. (eds.) CONCUR 2002. LNCS, vol. 2421, pp. 116–131. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45694-5_9
2. Abdulla, P.A., Jonsson, B., Nilsson, M., d'Orso, J., Saksena, M.: Regular model checking for LTL(MSO). Int. J. Softw. Tools Technol. Transf. **14**(2), 223–241 (2012). https://doi.org/10.1007/s10009-011-0212-z
3. Abdulla, P.A., Jonsson, B., Nilsson, M., Saksena, M.: A survey of regular model checking. In: Gardner, P., Yoshida, N. (eds.) CONCUR 2004. LNCS, vol. 3170, pp. 35–48. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28644-8_3
4. Alur, R., Cerný, P.: Expressiveness of streaming string transducers. In: IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS. LIPIcs, vol. 8, pp. 1–12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2010). https://doi.org/10.4230/LIPIcs.FSTTCS.2010.1
5. Alur, R., Deshmukh, J.V.: Nondeterministic streaming string transducers. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6756, pp. 1–20. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22012-8_1
6. Alur, R., Filiot, E., Trivedi, A.: Regular transformations of infinite strings. In: Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS, pp. 65–74. IEEE Computer Society (2012). https://doi.org/10.1109/LICS.2012.18
7. Boigelot, B., Jodogne, S., Wolper, P.: An effective decision procedure for linear arithmetic over the integers and reals. ACM Trans. Comput. Log. **6**(3), 614–633 (2005). https://doi.org/10.1145/1071596.1071601
8. Boigelot, B., Legay, A., Wolper, P.: Iterating Transducers in the Large. In: Hunt, W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 223–235. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45069-6_24
9. Boigelot, B., Legay, A., Wolper, P.: Omega-regular model checking. In: Jensen, K., Podelski, A. (eds.) TACAS 2004. LNCS, vol. 2988, pp. 561–575. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24730-2_41
10. Boigelot, B., Wolper, P.: Representing arithmetic constraints with finite automata: an overview. In: Stuckey, P.J. (ed.) ICLP 2002. LNCS, vol. 2401, pp. 1–20. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45619-8_1
11. Boker, U.: Why these automata types? In: LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning. EPiC Series in Computing, vol. 57, pp. 143–163. EasyChair (2018). https://easychair.org/publications/paper/G5dD
12. Bouajjani, A., Habermehl, P., Vojnar, T.: Abstract regular model checking. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 372–386. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27813-9_29
13. Bouajjani, A., Jonsson, B., Nilsson, M., Touili, T.: Regular model checking. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 403–418. Springer, Heidelberg (2000). https://doi.org/10.1007/10722167_31
14. Bouajjani, A., Legay, A., Wolper, P.: Handling liveness properties in (omega-)regular model checking. In: Proceedings of the 6th International Workshop on Verification of Infinite-State Systems, INFINITY. Electronic Notes in Theoretical Computer Science, vol. 138, pp. 101–115. Elsevier (2004). https://doi.org/10.1016/j.entcs.2005.02.061

15. Chaudhuri, S., Sankaranarayanan, S., Vardi, M.Y.: Regular real analysis. In: 28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS, pp. 509–518. IEEE Computer Society (2013). https://doi.org/10.1109/LICS.2013.57
16. Courcelle, B.: Monadic second-order definable graph transductions: a survey. Theor. Comput. Sci. **126**(1), 53–75 (1994). https://doi.org/10.1016/0304-3975(94)90268-2
17. Courcelle, B., Engelfriet, J.: Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach, Encyclopedia of mathematics and its applications, vol. 138. Cambridge University Press (2012). http://www.cambridge.org/fr/knowledge/isbn/item5758776/?site_locale=fr_FR
18. Dams, D., Lakhnech, Y., Steffen, M.: Iterating transducers. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 286–297. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44585-4_27
19. Dave, V., Dohmen, T., Krishna, S.N., Trivedi, A.: Regular model checking with regular relations. CoRR abs/1910.09072 (2019). http://arxiv.org/abs/1910.09072
20. Engelfriet, J., Hoogeboom, H.J.: MSO definable string transductions and two-way finite-state transducers. ACM Trans. Comput. Log. **2**(2), 216–254 (2001). https://doi.org/10.1145/371316.371512
21. Gorman, A.B., et al.: Continuous regular functions. Log. Methods Comput. Sci. **16**(1) (2020). https://doi.org/10.23638/LMCS-16(1:17)2020
22. Habermehl, P., Vojnar, T.: Regular model checking using inference of regular languages. In: Proceedings of the 6th International Workshop on Verification of Infinite-State Systems, INFINITY. Electronic Notes in Theoretical Computer Science, vol. 138, pp. 21–36. Elsevier (2004). https://doi.org/10.1016/j.entcs.2005.01.044
23. Jonsson, B., Nilsson, M.: Transitive closures of regular relations for verifying infinite-state systems. In: Graf, S., Schwartzbach, M. (eds.) TACAS 2000. LNCS, vol. 1785, pp. 220–235. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-46419-0_16
24. Kesten, Y., Maler, O., Marcus, M., Pnueli, A., Shahar, E.: Symbolic model checking with rich assertional languages. Theor. Comput. Sci. **256**(12), 93–112 (2001). https://doi.org/10.1016/S0304-3975(00)00103-1
25. Legay, A.: Extrapolating (omega-)regular model checking. Int. J. Softw. Tools Technol. Transf. **14**(2), 119–143 (2012). https://doi.org/10.1007/s10009-011-0209-7
26. Legay, A., Wolper, P.: On (omega-)regular model checking. ACM Trans. Comput. Log. **12**(1), 2:1-2:46 (2010). https://doi.org/10.1145/1838552.1838554
27. Löding, C., Spinrath, C.: Decision problems for subclasses of rational relations over finite and infinite words. Discret. Math. Theor. Comput. Sci. 21(3) (2019). http://dmtcs.episciences.org/5141
28. Sakarovitch, J.: Elements of Automata Theory. Cambridge University Press, Cambridge (2009). https://doi.org/10.1017/CBO9781139195218
29. Schützenberger, M.: Sur les relations rationelles entre monoïdes libres. Theor. Comput. Sci. 243–259 (1976)
30. Touili, T.: Regular model checking using widening techniques. Electron. Notes Theor. Comput. Sci. **50**(4), 342–356 (2001). https://doi.org/10.1016/S1571-0661(04)00187-2
31. Wolper, P., Boigelot, B.: Verifying systems with infinite but regular state spaces. In: Hu, A.J., Vardi, M.Y. (eds.) CAV 1998. LNCS, vol. 1427, pp. 88–97. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0028736