Periodic Generalized Automata over the Reals

Klaus Meer[1(✉)] and Ameen Naif[1]

Computer Science Institute, Brandenburg University of Technology
Cottbus-Senftenberg, Platz der Deutschen Einheit 1, 03046 Cottbus, Germany
{meer,naif}@b-tu.de

Abstract. In [4] Gandhi, Khoussainov, and Liu introduced and studied a generalized model of finite automata able to work over arbitrary structures. The model mimics the finite automata over finite structures but has an additional ability to perform in a restricted way operations attached to the structure under consideration. As one relevant area of investigations for this model the authors of [4] identified studying the new automata over uncountable structures such as the real numbers. This research was started in [7]. However, there it turned out that many elementary properties known from classical finite automata are lost. This refers both to structural properties of accepted languages and to decidability and computability questions. The intrinsic reason for this is that the computational abilities of the new model turn out to be too strong.

We therefore propose a restricted version of the model which we call periodic GKL automata. The new model still has certain computational abilities which, however, are restricted in that computed information is deleted again after a fixed period in time. We show that this limitation regains a lot of classical properties including the pumping lemma and many decidability results. Thus the new model seems to reflect more adequately what might be considered as a finite automata over the reals and similar structures. Though our results resemble classical properties, for proving them other techniques are necessary. One fundamental proof ingredient will be quantifier elimination over real closed fields.

Keywords: Unconventional models of computation · Computational complexity

1 Introduction

In recent work Gandhi, Khoussainov, and Liu [4] introduced a generalized model of finite automata called $(\mathcal{S}, k)$-automata. It is able to work over an arbitrary structure $\mathcal{S}$, and here in particular over infinite alphabets like the real numbers. A structure is characterized by an alphabet (also called universe) together with a finite number of binary functions and relations over that alphabet. Intuitively the model processes words over the underlying alphabet componentwise. Each

A. Naif was supported by a 'Promotionsstipendium nach Graduiertenförderungsverordnung des Landes Brandenburg GradV'. The support is gratefully acknowledged.

A.-H. Dediu et al. (Eds.): LATA 2016, LNCS 9618, pp. 168–180, 2016.
DOI: 10.1007/978-3-319-30000-9_13

single step is made of finitely many test operations relying on the fixed relations as well as finitely many computational operations relying on the fixed functions. For performing the latter an $(\mathcal{S}, k)$-automaton can use a finite number k of registers. It moves between finitely many states and finally accepts or rejects an input.

The motivation to study such generalizations is manifold. In [4] the authors discuss different previous approaches to design finite automata over infinite alphabets and their role in program verification and database theory. One goal is to look for a generalized framework that is able to homogenize at least some of these approaches. As the authors remark, many classical automata models like pushdown automata, Petri nets, visible pushdown automata can be simulated by the new model. Another major motivation results from work on algebraic models of computation over structures like the real and complex numbers. Here, the authors suggested their model as a finite automata variant of the Blum-Shub-Smale BSS model [1,2]. They then ask to analyze such automata over structures like real or algebraically closed fields.

This line of research has been started recently by the present authors in [7]. Given the tremendous impact finite automata have in classical computability theory it looks promising to introduce and study a similar concept as restriction of the real computational model introduced by Blum, Shub, and Smale. However, the main lesson from [7] is that the general automata model by Gandhi et al. turns out to be too strong when applied to computations over the real or complex numbers. Almost all basic automata problems turn out to be undecidable in the BSS framework for these two uncountable structures. As another consequence, only weak structural properties can be derived for real languages accepted by such an automaton. The intrinsic reason for this is the automata's ability to store intermediate results during an entire computation, something obviously not possible in the finite automata world. We therefore suggest a restricted version of the Ghandi-Khoussainov-Liu (for short: GKL) model. The basic idea is to force the automaton to periodically forget after a constant number of steps its intermediate results. This still enables the automaton to perform operations present in the given structure, but in a limited way.

For the resulting restricted periodic version of real GKL automata we shall prove both structural as well as several decidability results. Since the automata can perform basic arithmetic operations over $\mathbb{R}$ semi-algebraic sets naturally show up; computability and decidability results then naturally rely on quantifier elimination algorithms for the first order theory of the reals.

Our results hopefully indicate that the restricted model is a meaningful alternative giving back many of the features finite automata have, but non-trivially related to the uncountable structure under consideration. Some of the further questions arising are discussed at the end.

In the next section we recall the automata model by Gandhi et al., equipped with the additional restriction of periodicity. The main results are proved in Sects. 3 and 4, which collect both decidability results for several questions about our periodic automata and structural properties of the languages accepted by them.

The paper intends to be a further step towards the development of a generalized model of finite automata. It might be promising to analyse our variant as well for the many further scenarios treated in [4].

2 Periodic GKL Automata

We suppose the reader to be familiar with the basics of the Blum-Shub-Smale model of computation and complexity over $\mathbb{R}$. Very roughly, algorithms in this model work over finite strings of real numbers. The operations either can be computational, in which case addition, subtraction, and multiplication are allowed; without much loss of generality we do not consider divisions in this paper to avoid technical inconveniences. Or an algorithm can branch depending on the result of a binary test operation. The latter will be inequality tests of form 'is $x \geq 0$?' The size of a string is the number of components it has, the cost of an algorithm is the number of operations it performs until it halts. For more details see [1].

The generalized finite automata introduced in [4] work over structures. Here, a structure $\mathcal{S}$ consists of a universe D together with finite sets of (binary) functions and relations over the universe. An automaton informally works as follows. It reads a word from the universe, i.e., a finite string of components from D and processes each component once. Reading a component the automaton can set up some tests using the relations in the structure. The tests might involve a fixed set of constants from the universe D that the automaton can use. It can as well perform in a limited way computations on the current component. Towards this aim, there is a fixed number k of registers that can store elements from D. Those registers can be changed using their current value, the current input and the functions related to $\mathcal{S}$. The new aspect we include here is that such computations cannot be performed unlimited; after a fixed number of steps performed all register values are reset to the intial assignment 0, thus forgetting intermediate results. After having read the entire input word the automaton accepts or rejects it depending on the state in which the computation stops. These automata can both be deterministic and non-deterministic.

The approach in particular easily can be adapted to define generalized finite automata over structures like $\mathbb{R}$ and $\mathbb{C}$. We shall in the rest of the paper focus on the real numbers, though all our results hold as well in their corresponding variant for the complex numbers. Statements about computability and decidability refer to the real BSS model of computation.

We consider exclusively the structure $\mathcal{S}_{\mathbb{R}} := (\mathbb{R}, +, -, \bullet, pr_1, pr_2, \geq, =)$ of reals as ring with order. As in the original work [4] we include the projection operators pr_1, pr_2 which give back the first and the second component of a tuple, respectively. In order to avoid technicalities for the subtraction operation we allow both orders of the involved arguments, i.e., applying $-$ to two values x, v can mean $x - v$ or $v - x$. Similarly, the order test can be performed both as $x \leq v$? and $v \leq x$? We do not include division as an operation. This will not significantly change our results.

The following definition makes these ideas precise. We alternatively call the resulting automata real periodic $(S_{\mathbb{R}}, k, T)$-automata or (a bit less clumsy) real periodic GKL automata, where GKL refers to the initials of the authors of [4]. The definition below adapts the general automata definition in [4] and its real number version from [7].

Definition 1. *(Periodic GKL automata over $\mathbb{R}$) Let $k, T \in \mathbb{N}$ be fixed.*

(a) A deterministic periodic $(\mathcal{S}_{\mathbb{R}}, k, T)$-automaton $\mathcal{A}$, also called real periodic GKL automaton, consists of the following objects:
- *a finite state space Q and an initial state $q_0 \in Q$,*
- *a set $F \subseteq Q$ of final (accepting) states,*
- *a set of ℓ registers which contain fixed given constants $c_1, \ldots, c_\ell \in \mathbb{R}$,*
- *a set of k registers which can store real numbers denoted by $v_1, \ldots, v_k$,*
- *a counter containing a number $t \in \{0, 1, \ldots, T-1\}$; we call T the periodicity of the automaton,*
- *a transition function $\delta : Q \times \mathbb{R} \times \mathbb{R}^k \times \{0,1\}^{k+\ell} \times \{0, 1, \ldots, T-1\} \mapsto Q \times \mathbb{R}^k \times \{0, 1, \ldots, T-1\}$.*

The automaton processes elements of $\mathbb{R}^* := \bigsqcup_{n \geq 1} \mathbb{R}^n$, i.e., words of finite length with real components. For such an $(x_1, \ldots, x_n) \in \mathbb{R}^n$ it works as follows. The computation starts in q_0 with initial assignment $0 \in \mathbb{R}$ for the values $v_1, \ldots, v_k \in \mathbb{R}$. The automaton has a counter which stores an integer $t \in \{0, 1, \ldots, T-1\}$. At the beginning of a computation its value is 0. $\mathcal{A}$ reads the input components step by step. Suppose a value x is read in state $q \in Q$ with counter value t. The next state together with an update of the values v_i and t is computed as follows:
- *$\mathcal{A}$ performs the $k + \ell$ comparisons $x\sigma_1 v_1?, x\sigma_2 v_2?, \ldots, x\sigma_k v_k?, x\sigma_{k+1} c_1?, \ldots, x\sigma_{k+\ell} c_\ell?$, where $\sigma_i \in \{\geq, \leq, =\}$. This gives a vector $b \in \{0,1\}^{k+\ell}$, where a component 0 indicates that the comparison that was tested is violated whereas 1 codes that it is valid;*
- *depending on state q and b the automaton moves to a state $q' \in Q$ (which could again be q);*
- *if the value of the counter is $t = T - 1$, then the counter as well as all register entries v_i are reset to 0. Otherwise, the counter value is increased by 1 and the values of all v_i are updated applying one of the operations in the structure: $v_i \leftarrow x \circ_i v_i$. Here, $\circ_i \in \{+, -, \bullet, pr_1, pr_2\}, 1 \leq i \leq k$ depends on q and b only.*

When the final component of an input is read $\mathcal{A}$ performs the tests for this component and moves to its last state without any further computation. It accepts the input if this final state belongs to F, otherwise $\mathcal{A}$ rejects.

(b) Non-deterministic $(\mathcal{S}_{\mathbb{R}}, k, T)$-automata are defined similarly with the only difference that δ becomes a relation in the following sense: If in state q the tests result in $b \in \{0,1\}^{k+\ell}$ the automaton can non-deterministically choose for the next state and the update operations one among finitely many tuples $(q', \circ_1, \ldots, \circ_k) \in Q \times \{+, -, \bullet, pr_1, pr_2\}^k$. The counter, however, is changed as in the deterministic case and if $t = T - 1$ the register values have to be reset to 0 as well in the non-deterministic case.

As usual, a non-deterministic automaton accepts an input if there is at least one accepting computation.

(c) The language of finite strings accepted by $\mathcal{A}$ is denoted by $L(\mathcal{A}) \subseteq \mathbb{R}^*$.

(d) A configuration of $\mathcal{A}$ is a tuple from $Q \times \mathbb{R}^k \times \{0, \ldots, T-1\}$ specifying the current data during a computation.

Example 2. (a) Every classically regular language $L \subseteq \{0,1\}^*$ when considered as language in $\mathbb{R}^*$ can be accepted by a real periodic GKL automaton. This can be seen easily by interpreting a corresponding finite automaton for L as GKL automaton which does not use its registers.

(b) For every semi-algebraic set $S \in \subseteq \mathbb{R}^n, n \in \mathbb{N}$ there is an $m \in \mathbb{N}$ and a real periodic GKL automaton $\mathcal{A}$ such that S is the projection of the set $L(\mathcal{A}) \cap \mathbb{R}^m$ onto its first n components. This was shown in [7] for the real GKL automata model, but the proof applies as well in the periodic case; this is true because the main point is to give a bound on the number of steps needed to evaluate the polynomial conditions defining S. The peridocity of the new automaton then should be larger than this bound.

Below in Sect. 4 we outline a more systematic description of acceptable languages which resembles the structure of regular sets over finite alphabets.

3 Decidability

Let us start with the study of some typical decision problems for periodic GKL automata. To each periodic $(S_{\mathbb{R}}, k, T)$-automaton $\mathcal{A}$ we attach a directed graph $G_{\mathcal{A}}$. Additionally, the edges are labeled by certain semi-algebraic sets. Both the graph and those sets turn out to be crucial for solving many of the fundamental decision problems about periodic GKL automata. Before being more precise let us describe the intuition behind those definitions. Since $\mathcal{A}$ has periodicity T we are naturally led to consider the following two questions: Starting in a state q with register values 0, which other states are reachable within precisely a number of t steps from q, where $1 \leq t \leq T$? And this happens when processing which inputs from $\mathbb{R}^t$? We therefore split any computation on inputs $x \in \mathbb{R}^n$ in s blocks of length T and t remaining steps, where $n = s \cdot T + t, s \in \mathbb{N}_0, 0 \leq t < T$.

Definition 3. *Let $\mathcal{A}$ be a deterministic $(S_{\mathbb{R}}, k, T)$-automaton with state set Q, initial state q_0 and final states $F \subseteq Q$.*

(a) The directed graph $G_{\mathcal{A}} = (V, E)$ attached to $\mathcal{A}$ is defined as follows: For each $q \in Q$ the set V contains a vertex q and vertices $q(t)$ for $1 \leq t < T$. $G_{\mathcal{A}}$ has an edge (p, q) iff there is a computation of $\mathcal{A}$ that when starting in p with register values 0 reaches q after T steps. $G_{\mathcal{A}}$ has an edge $(p, q(t))$ for an $1 \leq t < T$ iff there is a computation of $\mathcal{A}$ that when starting in p with register values 0 reaches q after t steps. No other edges are present in $G_{\mathcal{A}}$.

(b) For edges (p,q) and $(p,q(t))$ of $G_{\mathcal{A}}$, respectively, define $S(p,q) \subseteq \mathbb{R}^T$ and $S(p,q(t)) \subseteq \mathbb{R}^t$ as set of those $x \in \mathbb{R}^T$ or $x \in \mathbb{R}^t$, respectively, for which $\mathcal{A}$ moves from p to q when reading x according to the conditions under (a).

Intuitively, vertices named by $p \in Q$ are used for dealing with sequences of T computational steps of $\mathcal{A}$, whereas the copies $q(t), 1 \leq t < T$ are used to reflect the final t steps of a computation. Therefore, there are no directed edges of form $(q(t), p)$.

Theorem 4. *Let $\mathcal{A}$ be an $(S_{\mathbb{R}}, k, T)$-automaton with attached directed graph $G_{\mathcal{A}} = (V, E), S(u,v)$ be defined as above for vertices $u, v \in V$. Then the following holds:*

(a) All $S(p,q)$ are semi-algebraic in $\mathbb{R}^T$, all $S(p,q(t))$ are semi-algebraic in $\mathbb{R}^t$.[1]
(b) The edge relation of $G_{\mathcal{A}}$ is BSS-computable, i.e., for given $u, v \in V$ one can decide by a BSS algorithm whether $(u,v) \in E$.
(c) For each $q \in Q, 0 \leq t \leq T-1$ the set $V(q,t)$ of register values $v \in \mathbb{R}^k$ that occur as valid entries during a computation of $\mathcal{A}$ which reaches q such that the counter contains t is semi-algebraic (and thus BSS decidable).

Proof. We are in this paper not interested in efficiency results and thus only argue how the questions under consideration lead to certain quantifier elimination tasks in the first order theory of the reals. Since this theory is BSS decidable, see [8] for more on the history of respective algorithms, the claimed results then follow.

Ad (a) Let $p, q \in Q$ be fixed. The arguments below will be the same for a pair $(p, q(t))$ of vertices. Suppose $\mathcal{A}$ uses ℓ constants and is in state p with all register values equal to 0. We enumerate all sequences $\mathcal{P} := (p_0, p_1, p_2, \ldots, p_T)$ of states $p_i \in Q$, where $p_0 = p$ and $p_T = q$, together with sequences $\beta := (b_1, \ldots, b_T), b_i \in \{0,1\}^{k+\ell}$ of decision vectors such that in $\mathcal{A}$ it is possible to move from state p_i to p_{i+1} if the actual input component x_{i+1} yields the test results coded by the components of b_i.

We now construct for each such pair $(\mathcal{P}, \beta)$ a first order formula $\Phi_{(\mathcal{P},\beta)}(x)$ expressing for which inputs $x \in \mathbb{R}^T$ the path $\mathcal{P}$ is representing $\mathcal{A}$'s computation on x. Towards this aim we must record the changes of the register values as well as check the related test results. For each $1 \leq i \leq T$ let $h_1^{(i)}, \ldots, h_k^{(i)} : \mathbb{R}^i \mapsto \mathbb{R}$ be polynomials such that $h_j^{(i)}(x_1, \ldots, x_i)$ is the entry in register v_j if the computation follows $(\mathcal{P}, \beta)$ on input $x_1, \ldots, x_i$ starting from register values 0. The $h_j^{(i)}$ clearly are polynomials of degree at most i given the way $\mathcal{A}$ can compute.

Next, first order formulas $\varphi_i(x_1, \ldots, x_i)$ express that if $\mathcal{A}$ is in state p_{i-1} with register values $v_1 = h_1^{(i-1)}(x_1, \ldots, x_{i-1}), \ldots, v_k = h_k^{(i-1)}(x_1, \ldots, x_{i-1})$ and reads x_i, then all performed tests give a result according to b_i, $\mathcal{A}$ moves to state p_i, and the new register values are $v_1 = h_1^{(i)}(x_1, \ldots, x_i), \ldots, v_k = h_k^{(i)}(x_1, \ldots, x_i)$. Once again, all above conditions can be expressed in first order logic over $\mathbb{R}$

[1] A semi-algebraic set in $\mathbb{R}^n$ is a set that can be defined as a finite union of solution sets of polynomial equalities and inequalities.

due to the form of the tests that can be performed. $\Phi_{(\mathcal{P},\beta)}(x)$ now is the conjunction of all these formulas for all $1 \leq i \leq T$. It follows that $S(p,q) = \{x \in \mathbb{R}^T \mid \bigcup_{(\mathcal{P},\beta)\in\Gamma} \Phi_{(\mathcal{P},\beta)}(x)\}$, where Γ contains all suitable sequences leading in T steps from p to q respecting the constraints described above. Since there are only finitely many suitable pairs and for each of it the set of x satisfying $\Phi_{(\mathcal{P},\beta)}(x)$ is semi-algebraic, the claim follows. The argument for vertices $(p, q(t))$ is the same.

Ad (b) Given the result in (a) for each pair (u, v) of vertices of the graph $G_{\mathcal{A}}$ it is decidable whether $S(u, v) \neq \emptyset$ by (one of) the well known algorithms for quantifier elimination over real closed fields [8]. Therefore, the edge relation of $G_{\mathcal{A}}$ is BSS computable.

Ad (c) The argument is similar to those in (a). First, any computation of n steps that reaches state q such that the counter has value t can be decomposed into s blocks of T steps followed by t final steps, where $n = sT + t$. In order to determine the realizable register assignments we have to figure out for which states $p \in Q$ automaton $\mathcal{A}$ can reach q in t steps when starting with register values 0. Among those states we are only interested in the ones reachable in sT steps, i.e., those p for which there is a path in $G_{\mathcal{A}}$ from q_0 to p. The latter can be checked by a usual search algorithm on directed graphs once $G_{\mathcal{A}}$ has been computed according to (b). Let $H(q, t)$ be the set of states p reachable in the above sense and such that $S(p, q(t)) \neq \emptyset$. Then a $v \in \mathbb{R}^k$ is in $V(q, t)$ iff there is a $p \in H(q, t)$ and an $x \in S(p, q(t))$ with $v = (h_1^{(t)}(x), \ldots, h_k^{(t)}(x))$, where $h_j^{(t)}$ are as defined in the proof of part (a). Clearly, this set is again semi-algebraic. □

The above theorem implies several decidability results of fundamental questions about real periodic GKL automata. Decidability here refers to the real number BSS model. Whereas for most of the problems treated below decidability follows relatively straightforwardly from the proof of Theorem 4, the equivalence problem is a bit harder to handle. Note that we do not currently know about a state minimization algorithm, thus the idea behind the classical algorithm for deciding equivalence of deterministic finite automata using minimal ones is not applicable. Note also that given the significantly extended computability features of the original definition of real GKL automata in [4] none of the problems listed below is decidable in this more general model, as was shown in [7].

Theorem 5. *The problems below are decidable in the real BSS model:*

(a) Emptiness: Given an $(S_{\mathbb{R}}, k, T)$-automaton $\mathcal{A}$, is $L(\mathcal{A}) = \emptyset$?
(b) Reachability I: Given $\mathcal{A}$ as in (a) with state set Q together with a state $q \in Q$, is there a computation starting in $\mathcal{A}$'s initial state with register entries 0 that reaches q?
(c) Reachability II: Given an automaton $\mathcal{A}$, a state q, a counter value $t \in \{0, \ldots, T-1\}$, and a $v \in \mathbb{R}^k$, is there a computation starting in $\mathcal{A}$'s initial state with register entries 0 that reaches p such that the counter's value is t and the register values equal v?

(d) Reachability III: Similar to Reachability II, but without t being specified?
(e) Equivalence: Given two real periodic GKL automata $\mathcal{A}_1, \mathcal{A}_2$*, is* $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?

Proof. Decidability of the questions under consideration is a relative immediate consequence of (the proof of) Theorem 4.

Ad (a) For emptiness we compute a set H of states reachable by $\mathcal{A}$ from its starting configuration in some $s \cdot T$ steps, where $s \in \mathbb{N}_0$. The computation of H can be done using the arguments from the proof of parts (a) and (b) in Theorem 4. If $H \cap F \neq \emptyset$ it follows $L(\mathcal{A}) \neq \emptyset$. Otherwise, for each $p \in H, 1 \leq t < T$, and $q \in Q$ we compute a description of the semi-algebraic set $S(p, q(t))$ and check whether it is empty or not using quantifier elimination. It follows that now $L(\mathcal{A}) = \emptyset$ iff all those $S(p, q(t))$ are empty.

Ad (b) Using $G_{\mathcal{A}}$ and the corresponding sets $S(u, v)$ it is easy to check whether a state q or one of its copies $q(t), 1 \leq t < T$ are reachable in $G_{\mathcal{A}}$ from the starting state. This can be done, for example, by a breadth-first search.

Ad (c) Check whether q (if $t = 0$) or $q(t)$ (if $t > 0$) are reachable in $G_{\mathcal{A}}$. In case it is we analyze the set of attainable register values $V(q, t)$ as in the proof of Theorem 4.

Ad (d) As in (c), but instead of checking one fixed t the algorithm has to be performed for all $0 \leq t \leq T - 1$.

Ad (e) Let $\mathcal{A}_1, \mathcal{A}_2$ be given with state sets Q_1, Q_2 and parameters $(k_i, T_i), i \in \{1, 2\}$, respectively. The key idea is to give a bound N for the dimension of an $x \in \mathbb{R}^N$ which is accepted by exactly one of the two automata in case they are not equivalent. Knowing such a bound we can search for x by using once more the previous arguments together with quantifier elimination algorithms.

For both automata we consider computational blocks of length $T := T_1 \cdot T_2$. It then follows that for each integer $r \in \mathbb{N}$ after $r \cdot T$ steps both automata have 0 as its register assignments. Suppose $L(\mathcal{A}_1) \neq L(\mathcal{A}_2)$, then there exist an $N \in \mathbb{N}$ and an $x \in \mathbb{R}^N$ such that without loss of generality $x \in L(\mathcal{A}_1) \setminus L(\mathcal{A}_2)$. Decompose $N = s \cdot T + t, s \in \mathbb{N}_0, 0 \leq t < T$. We want to find an absolute upper bound s_0 for s such that a witness x can be proved to exist up to dimension $s_0 \cdot T + t < (s_0 + 1)T$ if one exists at all.

In order to determine s_0 define once again directed graphs $G_{\mathcal{A}_1}, G_{\mathcal{A}_2}$ attached to the given automata as in Definition 3. However, this time both graphs are defined with respect to computational blocks of length T instead of taking the respective periodicities of the two automata. In the first reasoning below we want to decide existence of a witness $x \in \mathbb{R}^N$ for some $N = s \cdot T$, i.e., if $t = 0$.

For $0 \leq i \leq s$ let $(p_i^{(1)}, p_i^{(2)}) \in Q_1 \times Q_2$ denote the pairs of states that $\mathcal{A}_1, \mathcal{A}_2$ attain after $i \cdot T$ steps of processing input x. Define $s_0 := |Q_1| \cdot |Q_2| - 1$. Then no matter how x looks like after at most $s_0 \cdot T$ steps a pair of states occurs for the second time. Since the configurations of both automata at these steps are the same, we can neglect the corresponding part of the computation. By removing other loops along the computation path in a similar way it follows that if a witness x exists at all (for choice $t = 0$) there is one of dimension at most $N := s_0 \cdot T$. This reduces the question to a finite set of quantifier

elimination problems in the spaces $\mathbb{R}^T, \mathbb{R}^{2T}, \ldots, \mathbb{R}^{s_0 T}$. For each $\mathbb{R}^{iT}, 1 \leq i \leq s_0$ express the question whether there is an $x \in \mathbb{R}^{iT}$ such that $x \in L(\mathcal{A}_1) \setminus L(\mathcal{A}_2)$ as closed existential first order formula. This can be done using the arguments in Theorem 4. We then decide truth of the formula by quantifier elimination.

If this decision procedure shows that there is no x of a dimension $N \in \{T, 2T, 3T, \ldots\}$ witnessing the difference of the two automata we next decide the respective question for dimensions of the form $N = sT + t$ with $1 \leq t < T$. As before, we first compute the set H of all pairs $(p, q) \in Q_1 \times Q_2$ that can be reached by $\mathcal{A}_1, \mathcal{A}_2$ in a sequence of some sT steps. For each fixed $1 \leq t < T$ and all pairs $(p', q') \in F_1 \times F_2$ of final states we decide, whether there are edges in $G_{\mathcal{A}_1}$ and $G_{\mathcal{A}_2}$, respectively, from p to $p'(t)$ and from q to $q'(t)$. In order to guarantee that both corresponding computations by $\mathcal{A}_1, \mathcal{A}_2$ are followed for the same input $x \in \mathbb{R}^t$ we consider the conjunction of the first order formulas expressing reachability of p' from p and of q' from q and only then apply quantifier elimination. If pairs $(p, q) \in H, (p', q') \in F_1 \times F_2, t \in \{1, \ldots, T-1\}$ are found it follows $L(\mathcal{A}_1) \neq L(\mathcal{A}_2)$, otherwise the two automata are equivalent. □

4 Structural Results

In this section the focus is on elaborating elementary structural results for languages acceptable by real periodic GKL automata. They extend the corresponding ones for discrete finite automata and include a pumping lemma, the equivalence of non-deterministic and deterministic periodic GKL automata, and some initial ideas about a generalization of regular expressions to our setting. Note that in its usual form the pumping lemma does not hold for general GKL automata over $\mathbb{R}$ and no variant is known to be true; similarly, non-deterministic real GKL automata are strictly more powerful than deterministic ones, see [7] for both issues and [4] for similar results over other structures. Thus, periodicity significantly reduces the computational power of GKL automata and brings them probably closer to what one would expect from a real version of finite automata.

Lemma 6 (Pumping Lemma for Real Periodic GKL Automata). *Let $L \subseteq \mathbb{R}^*$ be accepted by a real periodic GKL automaton $\mathcal{A}$ with periodicity T, k registers and s states. Then for all $w \in L$ of algebraic size $|w| \geq sT$ there exist $x, y, z \in \mathbb{R}^*$ such that $w = xyz, |y| \geq T, |xy| \leq sT$, and $\forall i \in \mathbb{N}_0\ xy^i z \in L$.*

Proof. Periodicity of $\mathcal{A}$ implies that for all $r \in \mathbb{N}$ after rT steps of any computation the register values are all 0. The actual configuration thus after rT steps only depends on the current state. For inputs w of length at least sT there occurs at least one state twice among the current states after one of the time steps $\{0, T, 2T, \ldots, sT\}$. The rest of the proof is as usual: If p is a state occuring twice, x the prefix of w processed by $\mathcal{A}$ until p is reached for first time, y the part processed until p occurs for the second time, and z the rest, then $w = xyz$ by definition and $xy^i z \in L$ for all $i \in \mathbb{N}_0$ because $\mathcal{A}$'s configuration is $(p, 0, 0)$ both when it starts to process y and when it has finished. The remaining conditions obviously are satisfied. □

The lemma shows that the language $\{0^n 1^n | n \in \mathbb{N}\}$, as in the finite automata world, is neither acceptable by a periodic real GKL automaton. It might be interesting to sharpen the statement by not only considering loops having a length being an integer multiple of the periodicity; this, however, would require to control the evolvement of the register entries, something that seemed to hinder a sharper structural result in [7].

Without requiring periodicity non-deterministic real GKL automata are more powerful than deterministic ones [7]. This difference vanishes if we include the periodicity requirement into the model.

Theorem 7. *The classes of languages over $\mathbb{R}^*$ acceptable by non-deterministic and by deterministic periodic real GKL automata are the same.*

Proof. Let $\mathcal{A}$ be a periodic non-deterministic $(S_{\mathbb{R}}, k, T)$-automaton with state set Q and final states $F \subseteq Q$. The construction of an equivalent deterministic automaton $\mathcal{A}'$ in principle uses the classical powerset idea; however, it is more complicated because of the automata's ability to compute. The new automaton has not only to record states that can be reached non-deterministically, but also register entries. In particular, it might be possible that $\mathcal{A}$ in a state has several choices how to continue to the same successor state but with different computations performed on the registers. Therefore, instead of (unordered) subsets of Q we are lead to consider (ordered) tuples of elements in Q as states of the new automaton $\mathcal{A}'$.

Let $M \in \mathbb{N}$ upper bound the maximal number of non-deterministic transitions $\mathcal{A}$ can choose from for any of its states. $\mathcal{A}$ begins its computations in a start state q_0 with register assignment 0. For $t < T$ steps $\mathcal{A}$ can achieve at most M^{t-1} different configurations. After T steps, i.e., one sweep of length the periodicity, there are at most $|Q|$ different configurations since the register values are reset. For the following sweeps of length T the same reasoning shows that at most $m := |Q| \cdot M^{T-1}$ different configurations can occur at any stage of a computation of $\mathcal{A}$.

This motivates the definition of the state set Q' of $\mathcal{A}'$: Q' will be used to code the multi-set of states of Q that can occur with potentially different register values at a certain point of a computation. Since not necessarily m different configurations are realizable let $q^* \notin Q$ denote a new dummy symbol and define Q' as a subset of $(Q \cup \{q^*\})^m$; a state in Q' lists with the respective cardinalities the set of states in Q reachable non-deterministically within a given number of steps. If less than m different configurations are reachable the lacking components are filled with q^*.[2] The starting state of $\mathcal{A}'$ is $(q_0, q^*, q^*, \ldots, q^*) \in (Q \cup \{q^*\})^m$, final states in Q' are those which contain at least one state from F as a component. $\mathcal{A}'$ uses km registers which are divided into m blocks of length k; each block is used to code the evolvement of $\mathcal{A}$'s k registers during one possible computation.

[2] In order to make the coding unique we could order the components of any tuple in Q' according to an order of $Q \cup \{q^*\}$, but we refrain from elaborating on this because it will likely not increase understandability.

The transition function can now be defined straightforwardly; in order not to overload the presentation notationally we just describe its functioning informally. If $\mathcal{A}$ after t steps on input x can reach a configuration $(p, v_1, \ldots, v_k, t)$, then $\mathcal{A}'$ after t steps is in a state p' which has p as one of its components; attached to each component is one block of k registers - and the one attached to the component containing the above p has $(v_1, \ldots, v_k)$ as its register values. Now each of the at most M many non-deterministic transitions $\mathcal{A}$ can perform next is coded via changing the corresponding components of p' and their attached register blocks accordingly. If $\mathcal{A}'$ has less than M many choices, in p the lacking components are set to q^* and the corresponding register values to 0. Of course, it has to be specified which components of a state p' and which register blocks of $\mathcal{A}'$ code which potential computation path of $\mathcal{A}$. However, it should be obvious that this easily can be done. Finally, $\mathcal{A}$ has an accepting computation on x ending in a state $q_f \in F$ if and only if $\mathcal{A}'$ reaches at the end a state that has q_f as one of its components, i.e., $\mathcal{A}'$ accepts x. □

The results especially of Theorem 4 resemble a strong similarity between the structures of real languages acceptable by periodic GKL automata and of regular languages. Due to space limitations we just outline this similarity and postpone a more complete treatment to a full version.

It has been shown in Sect. 3 that computation cycles of length the periodicity T play a crucial role in the analysis of periodic automata. If such an automaton can move in T steps from state p to q assuming all registers have been initialized to 0, then the (non-empty) semi-algebraic sets $S(p, q)$ introduced in Definition 3 constitute building blocks of words being processed by the automaton; similarly at the end of a computation with sets of form $S(p, q(t))$. Thus, for the development of a theory of 'regular expressions' in our context such sets could serve as elementary objects. Of course, this requires as well taking more care about what kind of semi-algbraic sets can be allowed here.

Next, there is as well a natural way to define a Kleene-$*$ operation on those sets. Starting from the automata side, each cycle in the directed graph $G_{\mathcal{A}}$ indicates that a corresponding sequence of operations can be performed by $\mathcal{A}$ arbitrarily many times, each time processing a word from a corresponding semi-algebraic set; the latter has the concatenated structure $S(p_1, p_2)S(p_2, p_3) \ldots S(p_r, p_1)$, where $p_1 \to p_2 \to \ldots \to p_r \to p_1$ denotes the underlying cycle. That way, we obtain a recursive construction of expressions being built on base of certain semi-algebraic sets. It is then not hard to work out a decomposition result for all real languages L accepted by a real periodic GKL automaton. Each such L can be decomposed using the operations of concatenation of fundamental semi-algebraic sets together with the $*$-operation related to the cycle structure in $G_{\mathcal{A}}$. It is probably more demanding to work out the other direction: Which kind of semi-algebraic ground-pieces can be allowed to obtain in the above way exactly those languages that are accepted by real periodic GKL automata? Nevertheless, the structural similarity to the concept of regular languages seems striking.

5 Further Questions

There are several ways to go. First, one could continue the line of research in this paper and analyze further properties of real languages acceptable by periodic GKL automata. For example, is there a way to minimize the number of states and of registers, and what impact has the choice of the periodicity? Next, above we completely disregarded complexity issues and only applied elimination results in their general form. In many of our problems requiring quantifier elimination the formulas obtained have a very specific structure because the automata process an input component only once and it is only influencing T steps of a computation. Therefore, one might ask which of the problems treated above are efficiently solvable, which are $\mathrm{NP}_{\mathbb{R}}$-complete in the BSS framework? Especially finding new $\mathrm{NP}_{\mathbb{R}}$-complete problems is interesting since the list of known such problems still is relatively limited.

A third area of further research is considering other underlying structures than the reals. One major motivation of [4] was to have a generalized automata model for many different structures which also homogenizes existing ones. So the question is in how far our restricted version of GKL automata also is meaningful for further structures like those considered in [4]? Finally, it seems promising to analyze the new automata model as well for infinite computations over countably infinite sequences of reals. The classical theory initiated by Büchi, see [9] for a longer introduction into the topic, established a close relation between such infinite automata and logic. For working with infinite structures meta-finite model theory was developed in [5] and applied to BSS computability theory in [3,6]. We believe it to be interesting to investigate potential links between the latter and a kind of periodic real Büchi automata.

References

1. Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and Real Computation. Springer, Berlin (1998)
2. Blum, L., Shub, M., Smale, S.: On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. Bull. Am. Math. Soc. New Ser. **21**(1), 1–46 (1989)
3. Cucker, F., Meer, K.: Logics which capture complexity classes over the reals. J. Symb. Log. **64**(1), 363–390 (1999)
4. Gandhi, A., Khoussainov, B., Liu, J.: Finite automata over structures. In: Agrawal, M., Cooper, S.B., Li, A. (eds.) TAMC 2012. LNCS, vol. 7287, pp. 373–384. Springer, Heidelberg (2012)
5. Grädel, E., Gurevich, Y.: Metafinite model theory. Inf. Comput. **140**(1), 26–81 (1998)
6. Grädel, E., Meer, K.: Descriptive complexity theory over the real numbers. In: Leighton, F.T., Borodin, A. (eds.) Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, Las Vegas, Nevada, USA, pp. 315–324. ACM, 29 May–1 June 1995
7. Meer, K., Naif, A.: Generalized finite automata over real and complex numbers. Theor. Comput. Sci. **591**, 85–98 (2015)

8. Renegar, J.: On the computational complexity and geometry of the first-order theory of the reals, parts 1–3. J. Symb. Comput. **13**(3), 255–352 (1992)
9. Thomas, W.: Automata on infinite objects. In: van Leeuven, J. (ed.) Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B), pp. 133–192. Elsevier, Amsterdam (1990)