

Model Checking One-Dimensional Cellular Automata

KLAUS SUTNER

Carnegie Mellon University

Pittsburgh, PA 15213

November 1, 2007

Abstract

We show that the first order theory of a one-dimensional cellular automaton, construed as a structure with the global map and equality, is decidable. The argument employs bi-infinite versions of Büchi automata that can also be used to demonstrate that the spectra of cellular automata on finite grids are regular. For existential properties our method can be used to produce witnesses.

1 Introduction

A 1972 paper by Amoroso and Patt [1] arguably contains the first explicit description of a decision algorithm for properties of one-dimensional cellular automata. In the reference, the authors show that both injectivity and surjectivity of the global map of a one-dimensional cellular automaton are decidable. Injectivity is a Π_1 statement by its very definition. By a theorem of Hedlund, surjectivity can also be expressed as a Π_1 statement provided we admit an additional predicate $x \stackrel{*}{=} y$ meaning that x and y differ in only finitely many places. Thus, the two properties are quite simple from the perspective of definability theory. Alas, it was shown by Kari that no generalization to dimensions higher than one is possible: both properties are undecidable, see [8]. The proof involves tilings and thus has no analogue in the one-dimensional situation. On the other hand, automata theoretic methods can be used to obtain a streamlined and easily implementable quadratic time algorithm that tests injectivity, surjectivity as well as openness of the global map, see [15].

The purpose of this paper is show that all first order properties of the phase-space of a one-dimensional cellular automaton are decidable. The underlying language is $\mathcal{L}(\rightarrow)$ where \rightarrow is a binary relation symbol denoting the “next-configuration” relation. We assume equality to be available. Thus, one can describe assertions such as “the cellular automaton is reversible”, “there exists a 5-cycle” or “there are exactly 2 fixed points” but it is not possible in general to make any statements about orbits: the reachability relation is not expressible in this setting. For example, it is impossible to state that all

configurations evolve to a fixed point, a property well-known to be undecidable even for periodic configurations, see [13, 14].

Decidability results based on automata on one-way infinite words, so-called ω -automata, date back to Büchi's work in the early 1960's, in particular his proof of the decidability of the monadic second order theory of the successor, see [2, 3]. More recently, numerous applications in computer science have appeared under the name of model checking, see [4] and [7] for extensive bibliographies. A discussion of recognizable languages of bi-infinite words can be found in [10] and, more recently, in [11]. Two-way infinite words in the context of cellular automata appear in [6] and [5].

2 Automata on Bi-infinite Words

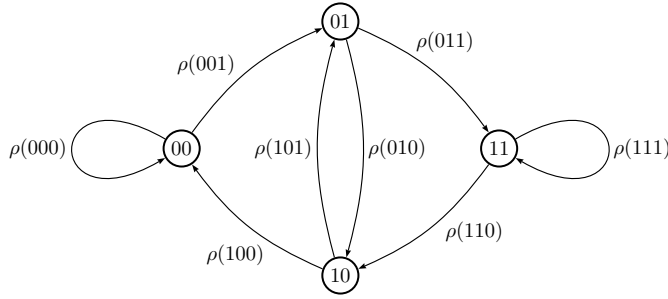
We consider a one-dimensional cellular automaton given by a local map ρ as a first order structure

$$\mathfrak{C}_\rho = \langle \Sigma^{\mathbb{Z}}, \rightarrow \rangle$$

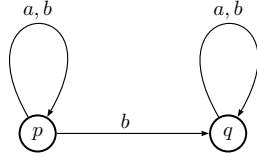
where $\Sigma^{\mathbb{Z}}$ is the space of configurations of the system and \rightarrow is a binary predicate that expresses the application of the global map G_ρ . In the deterministic case \rightarrow corresponds simply to the application of a unary function, but the relational representation is slightly more convenient for our purposes. Without loss of generality we consider only standard local rules of the form $\rho : \Sigma^{2r+1} \rightarrow \Sigma$.

Since we are dealing with two-way infinite words we need to generalize the notion of a Büchi automaton. A ζ -Büchi automaton \mathcal{A} consists of a finite, nondeterministic transition system $\langle Q, \Sigma, \tau \rangle$ and an acceptance condition (I, F) where $I, F \subseteq Q$. Given a bi-infinite word $x \in \Sigma^\infty$, a run of \mathcal{A} on x is a bi-infinite sequence $\pi = (p_i)_{i \in \mathbb{Z}}$ of states in Q such that $\tau(p_i, x_i, p_{i+1})$ for all $i \in \mathbb{Z}$. Let $\text{Inf}^-(\pi) = \{p \in Q \mid \exists^\infty i \in \mathbb{N} (p_{-i} = p)\}$ and $\text{Inf}^+(\pi) = \{p \in Q \mid \exists^\infty i \in \mathbb{N} (p_i = p)\}$ denote the set of negatively and positively recurrent states in π . Then \mathcal{A} accepts x if there exists some run π on x such that $\text{Inf}^-(\pi) \cap I \neq \emptyset$ and $\text{Inf}^+(\pi) \cap F \neq \emptyset$. A set of bi-infinite words is *recognizable* if there is some ζ -Büchi automaton that accepts it. This definition is justified by the existence of other machine models such as Muller or Rabin automata that recognize the same class of languages. Furthermore, recognizable languages are precisely the rational ones: those that can be written as finite unions of languages of the form ${}^\omega U W V {}^\omega$ where U , W and V are all recognizable languages of finite words. Lastly, one can obtain a characterization of recognizable ζ -languages in terms of morphisms of appropriate semigroups, see [11].

The canonical de Bruijn automaton that recognizes the collection of all words produced by one application of the global map of a cellular automaton can be construed as a ζ -Büchi automaton. For example, for any elementary cellular automaton the corresponding de Bruijn automaton looks like so.



The understanding is that $Q = I = F$, so that acceptance simply means the existence a bi-infinite path with the proper labels. Thus this automaton decides the unary predicate $\exists x (x \rightarrow y)$. We refer to Büchi automata with $Q = I = F$ as full. A simple example that requires non-full automata is given by the language $L = {}^\omega\{a, b\}b\{a, b\}^\omega$. The natural nondeterministic ζ -Büchi automaton for L has $I = \{p\}$, $F = \{q\}$. Structurally similar automata can handle inequality $x \neq y$ and almost-everywhere equality, $x \stackrel{*}{=} y$.



In general we use ζ -Büchi automata on bi-infinite words over alphabets of the form $\Gamma = \Sigma^k$ and will construe a bi-infinite word over Γ to be comprised of k tracks, each holding a bi-infinite word over Σ . For example, the canonical automaton $\mathcal{A}_\rho(x, y)$ that tests whether two configurations $x, y \in \Sigma^\infty$ are related by $x \rightarrow y$ will scan words of the form

...	x_{-3}	x_{-2}	x_{-1}	x_0	x_1	x_2	x_3	...
...	y_{-3}	y_{-2}	y_{-1}	y_0	y_1	y_2	y_3	...

Thus, each variable in the formula corresponds to a separate track in the bi-infinite word. Since y_i depends uniformly on the application of the local map ρ on a finite block of the form $x_{i-r}, \dots, x_i, \dots, x_{i+r}$ the appropriate automaton has state set $\Gamma^{2r} \times \Gamma^r$ and transitions of the form

$$\begin{array}{ccc} x_1, \dots, x_{2r} & \xrightarrow{\begin{array}{c} a \\ b \end{array}} & x_2, \dots, x_{2r}, a \\ y_1, \dots, y_r & & y_2, \dots, y_r, b \end{array}$$

provided that $\rho(x_1, \dots, x_{2r}, a) = y_1$. Thus, the diagram is a substructure of the complete de Bruijn automaton over Σ^2 of order $2r$. Note that these automata are also full, acceptance simply means that there has to be a bi-infinite run of the automaton with

label sequence $\begin{smallmatrix} x \\ y \end{smallmatrix}$. Equality can be considered to be the special case associated with a trivial local rule of radius $r = 0$.

One minor complication to bear in mind is that that bi-infinite words, unlike their finite or one-way infinite counterparts, fail to carry a natural coordinate system. In fact, there are distinct bi-infinite words that cannot be distinguished by any finite state machine. This is easy to see when the words in question are shifts of each other. Indistinguishability also arises if the the two words are recurrent (every finite subword that appears anywhere already appears bi-infinitely often) and have the same cover (i.e., the same set of finite factors). These are the only circumstances under which indistinguishability can arise, see [11]. Still, we can obtain a representation of a recognizable ζ -language L in terms of pairs of recognizable ω -languages:

$$L = \bigcup U_i^{\text{op}} V_i$$

where U^{op} stands for the co- ω -words obtained from U by reversing all the words. More precisely, for any ω -word $x : \mathbb{N} \rightarrow \Sigma$ define the co- ω -word $x^{\text{op}} : \mathbb{Z}^- \rightarrow \Sigma$, where \mathbb{Z}^- denotes the negative integers, by $x^{\text{op}}(-(i+1)) = x(i)$, $i \geq 0$. Clearly one can join a co- ω -word x^{op} obtained by reversal with an ω -word y to produce a ζ -word $x^{\text{op}}y$. In the decomposition of L the constituent parts U_i and V_i can be handled by suitable Büchi automata. As a consequence, one can exploit classical algorithms for ω -languages to deal with recognizable ζ -languages. In particular the determinization algorithm for ω -languages due to Safra, see [12] can be adapted to determinize ζ -automata.

Theorem 2.1 *Model checking for one-dimensional cellular automata in the first order logic $\mathcal{L}(\rightarrow)$ with equality is decidable.*

Proof.

Consider an arbitrary first order formula $\varphi(x_1, \dots, x_k)$ with k free variables as indicated. We will construct a ζ -Büchi automaton \mathcal{A}_φ that accepts precisely those k -track bi-infinite words that satisfy the formula:

$$\mathcal{L}(\mathcal{A}_\varphi) = \{ (u_1, u_2, \dots, u_k) \in \Gamma^\infty \mid \mathfrak{C}_\rho \models \varphi(u_1, u_2, \dots, u_k) \}$$

where $\Gamma = \Sigma^k$. If φ is atomic, the full automaton \mathcal{A}_ρ or the obvious equality testing automaton, padded so as to operate over the appropriate alphabet, suffices. If φ is of the form $\varphi_1 \vee \varphi_2$, \mathcal{A}_φ can be chosen to be the disjoint union of \mathcal{A}_{φ_1} and \mathcal{A}_{φ_2} . Universal quantifiers are translated into existential quantifiers. The automaton for $\varphi = \exists x \varphi_0$ is obtained from \mathcal{A}_{φ_0} by erasing the track corresponding to variable x . For example, the de Bruijn automaton from above can be obtained by projecting \mathcal{A}_ρ along x .

It remains to deal with negation, $\varphi = \neg \varphi_0$. We need to complement the automaton \mathcal{A}_{φ_0} which in turn requires determinization. To this end, we split a given ζ -Büchi automaton \mathcal{A} on n states into n pairs of ordinary Büchi automata $(\mathcal{B}_p, \mathcal{C}_p)$, one for each state p of \mathcal{A} . More precisely,

$$\mathcal{L}(\mathcal{A}) = \bigcup_p \mathcal{L}^{\text{op}}(\mathcal{B}_p) \mathcal{L}(\mathcal{C}_p)$$

Here \mathcal{L}^{op} refers to the co- ω -language obtained from the ordinary ω -language of an automaton by reversing all words. The individual Büchi automata have the form

$\mathcal{B}_p = \langle Q, \tau, \Sigma; I, p \rangle^{\text{op}}$ and $\mathcal{C}_p = \langle Q, \tau, \Sigma; p, F \rangle$ and are the result of splitting the original automaton \mathcal{A} at state p . As usual, the reversal of a Büchi automaton is obtained by reversing all transitions and interchanging initial and final states. In terms of languages this means that we use the decomposition $\mathcal{L}(\mathcal{A}) = \bigcup U_i^{\text{op}} V_i$ where the U_i and V_i are recognizable ω -languages. Since the latter form an effective boolean algebra we can construct a ζ -Büchi automaton for the complement of $\mathcal{L}(\mathcal{A})$. \square

For sentences φ the number of free variables k in the proof is 0. To handle this case properly it is convenient to adopt the convention that $\Sigma^{k\infty}$ is the 2-element Boolean algebra $\{\top, \perp\}$. Projection yields \perp if the set is empty and \top otherwise. If the leading quantifier in the original formula is universal conversion to an existential quantifier will leave a negation operation in front which can be handled easily by our convention.

Also note that the proof produces a slightly stronger result: the sets of bi-infinite words satisfying a given formula with free variables are effectively regular. Hence we can use the automata to construct examples for existential quantifiers. For example, consider the question whether phase-space contains a 3-cycle, corresponding to the formula

$$\exists x, y, z (x \rightarrow y \wedge y \rightarrow z \wedge z \rightarrow x \wedge x \neq y \wedge x \neq z \wedge y \neq z)$$

As written, the three occurrences of inequality would require a cumbersome product machine construction. However, the matrix of the formula is equivalent to

$$\psi = x \rightarrow y \wedge y \rightarrow z \wedge z \rightarrow x \wedge x \neq y$$

which leads to a smaller automaton $\mathcal{A}_\psi(x, y, z)$. Projecting we solve the decision problem; moreover, the language of the automaton provides all possible witnesses of a 3-cycle and can be used to check additional properties. For example, we can check whether the 3-cycle is unique. Of course, existence and uniqueness are expressible directly in a first order formula; however, the corresponding formula is more complicated and the corresponding machines are significantly larger.

2.1 Algorithmic Issues

The construction of the basic model checking automata \mathcal{A}_ρ is $O(n\sqrt{n})$ where $n = |\Sigma|^{2r+1}$ is the size of the rule table for ρ . The k -variable version, however, requires time and space $O(n^k)$. Taking the disjoint union of two automata as well as the projection needed to deal with existential quantifiers are linear in the size of the given automata. Note that logical *and* can be dealt with directly by a product machine construction of quadratic complexity. Alas, complementation fails to be polynomial time. First off, the construction of the split automata is quadratic in the number of states of the original ζ -Büchi automaton. Moreover, the disjointness condition required for complementation can cause an exponential number of Büchi automata to be constructed. Determinization of each of these machines is $2^{O(n \log n)}$. Note that for full automata Safra's algorithm degenerates into standard Rabin-Scott determinization and thus avoids super-exponential blowup, though exponential blowup may still occur. Once the appropriate ζ -automaton has been constructed the emptiness test is linear time, as we have seen. Projecting to $\{\top, \perp\}$ comes down to solving an instance of the following Emptiness problem:

Problem: ζ -Büchi Emptiness
Instance: A ζ -Büchi automaton \mathcal{A} .
Question: Is $\mathcal{L}(\mathcal{A}) = \emptyset$?

Proposition 2.1 *The Emptiness problem for ζ -Büchi automata is solvable in time linear in the size of the automaton.*

Proof. Consider a ζ -Büchi automaton $\mathcal{A} = \langle Q, \Sigma, \tau; I, F \rangle$. Using Tarjan’s algorithm we construct the strongly connected components of the diagram of \mathcal{A} in linear time. We can then identify all non-trivial components that contain an initial state and those that contain a final state. Lastly, we check if there is a path from one of the former components to one of the latter. Clearly, all of this can be handled with standard graph algorithms in linear time. \square

As the examples of injectivity and surjectivity show, it may be preferable on occasion to avoid the elimination of universal quantifiers and instead rely on compactness to reduce a decision problem on ζ -words to a problem about finite words.

At any rate, overall the complexity of the decision algorithm in general fails to be elementary due to the iterated application of the negation procedure. As a consequence, one can only hope to deal with formulae of very limited complexity in practice. Indeed, if one applies ad hoc local optimizations one can derive the quadratic time algorithm for injectivity in [15]. The decidability of surjectivity can be established in this setting without the use of Hedlund’s No-Diamond theorem that surjectivity is equivalent to injectivity on configurations of finite support. The resulting algorithm, though, is exponential, much as the direct universality test for the de Bruijn automaton, [16].

2.2 Extensions

Since the decision algorithm is based on ζ -Büchi automata one can generalize the language $\mathcal{L}(\rightarrow)$ to include finitely many unary predicates to be interpreted as recognizable languages over Σ^∞ . For example, we can adjoin a predicate F to be interpreted as the collection of configurations of finite support, ${}^\omega 0 \Sigma^* 0^\omega$. In the light of the Cook-Wolfram argument for the computational universality of elementary cellular automaton 110 another interesting constraint is to consider backgrounds, configurations of the form ${}^\omega u \Sigma^* v^\omega$ where u and v are fixed finite words. The proof of universality establishes the fact that ECA 110 can simulate cyclic tag systems given appropriate backgrounds. As a consequence, the reachability problem for ECA 110 is trivially decidable for finite and one-way infinite configurations but is r.e.-complete for backgrounds. As we have seen, our machinery is sufficient to discover short cycles in phase-space, which cycles can be used in the construction of suitable background patterns. Of course, verification of the whole Cook-Wolfram argument lies outside of the scope of our methods.

Lastly, since we are dealing with nondeterministic machines, the decision algorithm easily generalizes to nondeterministic cellular automata: the local “map” of the cellular automaton may well be a relation $\rho \subseteq \Sigma^w \times \Sigma$. In fact, any finite state transducer can be used in place of the global map of the cellular automaton.

3 Finite Cellular Automata and Spectra

The discussion so far is limited to infinite cellular automata where the underlying grid has the structure of \mathbb{Z} . Naturally one can also consider finite grids, in particular a path of length n with fixed or periodic boundary conditions. The latter case also corresponds to circular grids or periodic configurations in the infinite case. Note that n is a free parameter and the questions about the behavior of all cellular automata with a fixed local rule on finite grids may well be undecidable. For example, it is Π_1 -complete to determine whether all configurations evolve to a fixed point for all n .

Given any first order sentence φ we define its spectrum to be

$$\text{spec}(\varphi) = \{ 0^n \mid \mathfrak{C}_\rho^n \models \varphi \}$$

Note that the numbers are written in unary in order to make the size of the grid accessible to a finite state machine. Also, we are not concerned with the cardinalities of all finite models of φ , just the finite grids for which the sentence holds. When construed as a set of natural numbers the spectrum of any first order sentence will turn out to be ultimately periodic.

Theorem 3.1 *Any sentence in the logic $\mathcal{L}(\rightarrow)$ has regular spectrum.*

Proof. We will construct an automaton \mathcal{A}_φ such that

$$\mathcal{L}(\mathcal{A}_\varphi) = \{ (0^n, u_1, u_2, \dots, u_k) \in 0^* \times \Gamma^\infty \mid \mathfrak{C}_\rho^n \models \varphi(u_1, u_2, \dots, u_k) \}$$

We focus on the case of cyclic boundary conditions and comment on modifications necessary to deal with fixed boundary conditions in the end.

The construction of the automaton is analogous to the bi-infinite case except that now we need to constrain the strings u_i to be of length n . This is accomplished by scanning 0's in an additional top track. In particular the transitions are the same as in the infinitary case. To deal with the boundary conditions the automaton checking for $x \rightarrow y$ is comprised of several disjoint copies of the machine \mathcal{A}_ρ from above, augmented by initial and final states of the form $I = F = \{p\}$ where p ranges over the state set $\Gamma^{2r} \times \Gamma^r$. The inductive construction of compound automata then proceeds as in the infinitary case. As a consequence, for any sentence φ we have an automaton \mathcal{A}_φ that accepts 0^n if, and only if, $\mathfrak{C}_\rho^n \models \varphi$.

To deal with fixed boundary conditions we use a single machine that has multiple initial states of the form $00 \dots 0a_1 \dots a_r$ and final states $b_1 \dots b_r 00 \dots 0$ (upper tracks only). \square

Note that the machines for initial boundary conditions are larger since we have to ensure that the beginning and end of the scanned strings match up.

Corollary 3.1 *Any sentence in the logic $\mathcal{L}(\rightarrow)$ has ultimately periodic spectrum.*

As we have seen, local optimizations can help to reduce the size of the model checking automata. First off, the additional track for 0^n is not needed: we can simply

rewrite all the transitions to be labeled 0. As an example, recall the simplified formula used in the test for a 3-cycle in phase-space:

$$\exists x, y, z (x \rightarrow y \wedge y \rightarrow z \wedge z \rightarrow x \wedge x \neq y)$$

In the computation of the spectrum for cyclic boundary conditions we may safely assume that the words x and y differ in their first position. Thus, one can simply select proper initial and final states in the machine checking $x \rightarrow z \wedge y \rightarrow z \wedge z \rightarrow x$. The latter can be implemented on a three-track alphabet and requires only 160 states on average for elementary cellular automata. The largest machine encountered has 640 states and is associated with elementary cellular automaton number 105. As it turns out, this is the only elementary cellular automaton whose 3-cycles appear exactly in grids of size $20n$. A complete listing of the spectra of 3-cycles for elementary cellular automata together with some sample machines can be found in the following table. The table is organized by increasing size of the minimal automaton recognizing the spectrum. The first column shows the spectrum as an ultimately periodic set of natural numbers, the second gives the number of elementary cellular automaton that have this spectrum for 3-cycles, and the last column shows some examples (or a complete listing if space allows).

spectrum	# ECA	examples
\emptyset	106	0, 1, 4, 5, 6, \dots , 251, 252, 253, 254, 255
$3\mathbb{N}$	108	2, 3, 10, 11, 16, \dots , 245, 246, 247, 248, 249
$5\mathbb{N}$	3	90, 150, 165
$6\mathbb{N}$	18	15, 26, 27, 38, 39, \dots , 167, 180, 181, 210, 211
$3, 5 + \mathbb{N}$	4	62, 118, 131, 145
$7 + \mathbb{N}$	2	73, 109
$9\mathbb{N}$	8	9, 65, 110, 111, 124, 125, 137, 193
$7, 9 + \mathbb{N}$	2	94, 133
$12\mathbb{N}$	4	30, 86, 135, 149
$20\mathbb{N}$	1	105

Even in a high level proof-of-concept type implementation in a computer algebra system the whole classification takes less than 20 seconds to compute on a standard dual-core desktop machine. Checking for 3-cycles in the infinite case is even easier; the corresponding Büchi automaton is obtained from a product machine construction between a full machine that tests for the cycle and another that checks for inequality. The resulting automaton has 128 states, and one can solve the corresponding Emptiness problems for all 256 elementary cellular automata in less than 3 seconds. Needless to say, 106 of the phase-spaces turn out not to have a 3-cycle.

4 Conclusion

We have shown that assertions expressible in the first order theory of phase-space of an infinite one-dimensional cellular automaton are decidable. Similarly, the first order spectra of the finite instances of a one-dimensional cellular automaton are effectively

regular. The algorithm converts first order sentences into ζ -Büchi automata and reduces validity to the Emptiness problem for an appropriate automaton. The conversion process requires determinization to handle negation, and, indirectly, universal quantification. As a consequence, the running time fails to be elementary, making it unlikely that the method can be applied to complicate assertions. In particular formulae with several alternations of quantifiers pose significant challenges for any concrete implementation. For sufficiently simple queries even a high level implementation in a computer algebra system produces very satisfying results.

It remains to be seen how far this approach can be pushed given a custom implementation in a low level language. It is also conceivable that methods used in model checking to avoid determinization could be brought to bear on our problem, see [9] for references. Another area under consideration is the analysis of phase-space for particularly simple cellular automata. For example, in the case of linear rules one may expect to be able to handle more complicated assertions. At present, we do not know how to classify the first order theories arising from one-dimensional cellular automaton; indeed, even for elementary cellular automata this is an open problem. Another interesting question is whether there are non-trivial classes of cellular automaton for which the decision algorithm can be generalized to a more expressive logic. For example, one could add a binary predicate denoting reachability in phase space. Alternatively, we could consider monadic second order logic or transitive closure logic. Since evolution of all configurations to a fixed point is undecidable in general [14] the decision problem for these extended logics must also be undecidable; we do not know where the boundary between decidability and undecidability lies.

References

- [1] S. Amoroso and Y. N. Patt. Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures. *Journal of Computer and Systems Sciences*, 6:448–464, 1972.
- [2] J. R. Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik and Grundl. Math.*, 6:66–92, 1960.
- [3] J. R. Büchi. On a decision method in restricted second-order arithmetic. In *Logic, Methodology and Philosophy of Science*, Proc. 1960 Internat. Congr., pages 1–11, 1962.
- [4] Edmund Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. MIT Press, 2000.
- [5] K. Culik. Global cellular automata. *Complex Systems*, 9:251–266, 1995.
- [6] K. Culik and Sheng Yu. Cellular automata, $\omega\omega$ -regular sets, and sofic systems. *Discrete Applied Mathematics*, 32:85–101, 1991.
- [7] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge UP, 2000.
- [8] J. Kari. Reversibility of 2D cellular automata is undecidable. *Physica D*, 45:397–385, 1990.

- [9] O. Kupferman. Avoiding determinization. In *Proc. 21st IEEE Symp. on Logic in Computer Science*, 2006.
- [10] M. Nivat and D. Perrin. Ensembles reconnaissable de mots biinfinis. *Canad. J. Math.*, 38:513–537, 1986.
- [11] D. Perrin and J.-E. Pin. *Infinite Words*, volume 141 of *Pure and Applied Math.* Elsevier, 2004.
- [12] S. Safra. On the complexity of ω -automata. In *Proc. 29th FOCS*, pages 319–327. IEEE Computer Soc. Press, 1988.
- [13] K. Sutner. A note on Culik-Yu classes. *Complex Systems*, 3(1):107–115, 1989.
- [14] K. Sutner. Classifying circular cellular automata. *Physica D*, 45(1–3):386–395, 1990.
- [15] K. Sutner. De Bruijn graphs and linear cellular automata. *Complex Systems*, 5(1):19–30, 1991.
- [16] K. Sutner. The size of power automata. In J. Sgall, Ales Pultr, and Petr Kolman, editors, *Mathematical Foundations of Computer Science*, volume 2136 of *SLNCS*, pages 666–677, 2001.