Finite Automata Over Infinite Alphabets: Two Models with Transitions for Local Change

Christopher Czyba, Christopher Spinrath(✉), and Wolfgang Thomas

RWTH Aachen University, Aachen, Germany
{christopher.czyba,christopher.spinrath}@rwth-aachen.de,
thomas@informatik.rwth-aachen.de

Abstract. Two models of automata over infinite alphabets are presented, mainly with a focus on the alphabet $\mathbb{N}$. In the first model, transitions can refer to logic formulas that connect properties of successive letters. In the second, the letters are considered as columns of a labeled grid which an automaton traverses column by column. Thus, both models focus on the comparison of successive letters, i.e. "local changes". We prove closure (and non-closure) properties, show the decidability of the respective non-emptiness problems, prove limits on decidability results for extended models, and discuss open issues in the development of a generalized theory.

1 Introduction

Automata over infinite alphabets have been studied since the 1990's, starting with the work of Kaminski and Francez [8] on "register automata". Further motivation was added by application areas, e.g. in verification or data base theory. The latter led to the theory of "data words" and "data automata" [4]. Another approach was pursued by Bès [1]. In view of algorithmic applications, a common interest in these theories is to devise models (in automata theory or in logic) where satisfiability problems are decidable, in particular the non-emptiness problem for automata over infinite alphabets.

A fundamental question in this context is the mechanism by which "memory of past inputs" is realized (over an infinite alphabet of possible inputs). The models mentioned above differ in the way this memory is implemented. The technical details in the definitions are not always simple; for example, the access to the memory of register automata is subject to some constraints which are subtle and to some extent artificial. The subject as a whole seems far from finished.

The purpose of this paper is to present two models of automata where the non-emptiness problem is decidable. Both models follow the intuition that the relation between *successive* letters is relevant and has to be controlled by automaton transitions. The results given in this paper are initial; we comment at the end of the paper on further research questions that are motivated by the present work.

I. Potapov (Ed.): DLT 2015, LNCS 9168, pp. 203–214, 2015.
DOI: 10.1007/978-3-319-21500-6_16

The first approach is an extension of the model introduced by A. Bès in [1]. In that paper, the alphabet letters are elements of a relational structure $\mathcal{M}$ over an infinite domain M, and a logic $\mathcal{L}$ allows to express conditions on alphabet letters by formulas $\varphi(y)$. The transition relation of the automaton is specified by such formulas $\varphi_{pq}(y)$ where p, q are from the finite set Q of states. The automaton can proceed from state p via letter $a \in M$ to state q if $\mathcal{M} \models \varphi_{pq}[a]$, i.e. a satisfies $\varphi_{pq}(y)$ in $\mathcal{M}$. A central weakness of this model is the inability to connect successive letters, for example to check the property that two identical successive letters exist (which gives the language $\bigcup_{a \in M} M^* a a M^*$).

Continuing work of Spelten [12], we study an extension of this model where this defect is repaired to some extent, due to a description of the relation between successive letters: We use now formulas $\varphi_{pq}(x, y)$ which allow to move from state p to state q via letter y if state p was reached via x (of course, for the initial transition of a run one uses a transition formula $\varphi_{q_0 q}(y)$). We call them "two-letter transitions". Since successive letters are subject to conditions expressed in the considered logic $\mathcal{L}$, we speak of "local change". In the paradigmatic case of the alphabet $\mathbb{N}$ of the natural numbers, one may express, for example, that in the step from one letter to the next the value can increase or decrease by 1 (or by at most by some constant k). But a transition could also specify that odd and even numbers have to alternate.

The main contribution of the first part of the paper is a proof showing that the resulting automata have a decidable non-emptiness problem (thereby repairing an unclear point in [12]), provided that the MSO-theory of the alphabet structure $\mathcal{M}$ is decidable. It is also shown that the result fails for two natural extensions of the model, namely for the case that alphabet letters are elements of M^2, i.e. pairs of M-elements, and for the case that three successive letters are related in automaton transitions (by formulas $\varphi(x, y, z)$).

The second approach is presented here for the case of the alphabet $\mathbb{N}$ rather than an arbitrary structure, in order to allow for a concise presentation (see the final section for a more general framework). A word $n_1 n_2 \dots n_\ell$ is represented by a $\{1, \perp\}$-labeled grid: we imagine a sequence of grid columns where the i-th column represents number n_i; this column starts from below with n_i nodes

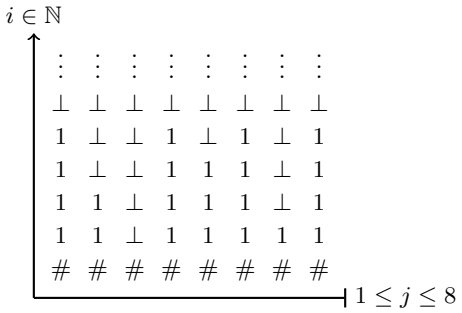

Fig. 1. Grid representing the word 4 2 0 4 3 4 1 4 $\in \mathbb{N}^*$

colored 1, followed by nodes colored $\perp$. For technical reasons we add a #-labeled boundary at the bottom. For instance, Figure 1 shows the grid for the word 4 2 0 4 3 4 1 4. This grid is traversed by a finite automaton in a three-way mode (hence a special type of the grid-walking automata of Blum, Hewitt and Rosenfeld [2],[10]). Each column can be traversed in a two-way mode up and down (which means that an input letter is analyzed), and after a move to the right via a horizontal edge no return to a previous column is allowed. The move to the right enables the automaton to check, in the example of the alphabet $\mathbb{N}$ as in the Figure, whether the move to a next letter gives the same value as before, respectively a larger or smaller value. Thus, again a natural test on "local change" is possible. But also non-local properties of words can be checked, for example (in the non-deterministic version) the existence of two equal letters from $\mathbb{N}$ in the input word.

In the second part of the paper we study this model regarding closure properties, show that the non-emptiness problem is decidable but that the universality problem is undecidable. The decidability of the emptiness problem resolves a question of [7], where a weaker model is studied, processing finite grids labeled with a single letter alphabet.

The paper is structured as follows. The subsequent section is devoted to automata with two-letter transitions. In order to have a shorter name for them, we call them "strong automata", following [12]. After this, we present the mentioned three-way automata, called "progressive grid automata". In the final section we relate these automata, discuss connections to known models of automata on infinite alphabets, and address a number of open issues.

Throughout we assume that the reader is acquainted with the basics of automata and logic, in particular first-order logic FO and monadic second-order logic MSO; see e.g. [14]. Due to lack of space, proofs are outlined; more details can be found in [5].

2 Automata with Two-Letter Transitions

2.1 Definitions

We allow elements of an arbitrary structure $\mathcal{M}$ as alphabet letters. Central examples are $(\mathbb{N}, +1)$ and $(\mathbb{N}, +, 0)$. If $\mathcal{M}$ is a structure with domain M and $\mathcal{L}$ a logic[1] (used with the appropriate signature), we call $(\mathcal{M}, \mathcal{L})$ an "alphabet frame". We only use logics in which the Boolean connectives are available and where first-order variables $x, y, \ldots$ can be employed to express properties of (tuples of) M-elements by formulas $\varphi(x), \varphi(x, y)$, etc. We denote the set of formulas with k free first-order variables by Ψ_k (note that nevertheless such a formula may contain quantified second-order variables).

Let $(\mathcal{M}, \mathcal{L})$ be an alphabet frame where M is the domain of $\mathcal{M}$. A *strong automaton* over $(\mathcal{M}, \mathcal{L})$ has the form $\mathfrak{A} = (Q, M, q_0, \Delta, F)$ where Q is a finite set of states, the input alphabet is M (the domain of $\mathcal{M}$), $q_0 \in Q$ is the initial

[1] The term "logic" is used here as in abstract model theory, see [6] Chapter 13.1.

state, $\Delta \subseteq Q \times (\Psi_1 \cup \Psi_2) \times Q$ is the finite transition relation, and lastly $F \subseteq Q$ is the set of accepting states.

A transition for the initial state has the form $(q_0, \varphi(y), q)$, other transitions have the form $(p, \varphi(x, y), q)$. A run ρ of $\mathfrak{A}$ on the word $w = a_1 ... a_m \in M^*$ is a finite sequence $\rho = \rho(0)...\rho(m)$ where

1. $\rho(0) = q_0$,
2. $\rho(1) = q$ with $(q_0, \varphi(x), q) \in \Delta$ such that $\mathcal{M} \models \varphi[a_1]$,
3. $\rho(i) = q$ where for $p = \rho(i-1)$, $(p, \varphi(x, y), q) \in \Delta$ such that $\mathcal{M} \models \varphi[a_{i-1}, a_i]$ for $1 < i \leq m$.

Note that several transitions from p to q may be condensed into one, by taking the disjunction of the respective transition formulas. Then one can indicate a transition just by writing $\varphi_{pq}(x, y)$ as done in the Introduction.

A run ρ of $\mathfrak{A}$ on the word w is successful if $\rho(m) \in F$. We say $\mathfrak{A}$ accepts w if there is a successful run of $\mathfrak{A}$ on w. Furthermore, $L(\mathfrak{A}) := \{w \in M^* \mid \mathfrak{A} \text{ accepts } w\}$.

Call a strong automaton deterministic if Δ is functional, i.e. for all $p \in Q$ and all $a, b \in M$ there is exactly one state q reachable by a corresponding transition.

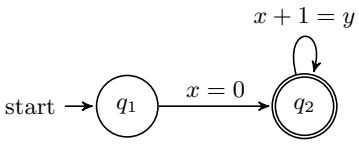

Fig. 2. Strong automaton recognizing $\{01 \cdots n \mid n \in \mathbb{N}\}$

A very simple example language recognized by a strong automaton is $L = \{01 \cdots n \mid n \in \mathbb{N}\}$ (for the alphabet frame $((\mathbb{N}, +1), FO)$), as depicted by the automaton in Figure 2.

The next two subsections lift results of Bès [1] from automata with single-letter transitions to strong automata (with two-letter transitions).

2.2 Boolean Closure Properties

The languages (over a given alphabet frame) that are recognized by strong automata form a Boolean algebra. The key for this claim is the following lemma on determinization, proved in analogy to [1].

Lemma 1. *Given a strong automaton $\mathfrak{A}$, one can construct a deterministic strong automaton $\mathfrak{A}'$ such that $L(\mathfrak{A}) = L(\mathfrak{A}')$.*

Proof. The idea is to take all the (finitely many) formulas $\varphi_i(x, y)$ $(i = 1, \ldots, m)$ that occur in a given automaton $\mathfrak{A} = (Q, M, q_1, \Delta, F)$ over the alphabet frame $(\mathcal{M}, \mathcal{L})$, and to form the Boolean min-terms of the relations $R_1, \ldots, R_m$ defined

by the φ_i. Each of these min-terms is defined by a conjunction of formulas φ_i and their negations, and each φ_i is equivalent to a disjunction of these min-terms. The transitions of the given automaton can now be partitioned into sets of transitions labeled by disjoint properties of letter pairs, each of them given by a min-term formula (for the initial transitions with formulas $\varphi(x)$ one proceeds analogously). Now the standard subset construction is applied to the automaton with these new transitions; whence a deterministic strong automaton as desired is obtained. Moreover, this construction shows that $\mathfrak{A}'$ can be obtained such that its size is at most exponential in the size of $\mathfrak{A}$.

For deterministic strong automata, complementation is shown by exchanging final with non-final states. Closure of strong automata under union is trivial by taking the union of given automata and a new initial state. Thus, we obtain the claim:

Proposition 1. *The languages recognized by strong automata (over a given alphabet frame) form an effective Boolean algebra (where the Boolean operations are realized by effective constructions of strong automata).*

2.3 The Non-Emptiness Problem

Theorem 1. *If the MSO-theory of $\mathcal{M}$ is decidable, then the non-emptiness problem for strong automata over the alphabet frame ($\mathcal{M}$, MSO) is decidable.*

Proof. The main point in this result is to exploit MSO-logic for expressing a reachability condition. Given a strong automaton, say $\mathfrak{A} = (\{0, \ldots, n\}, M, 0, \Delta, F)$ over the alphabet frame ($\mathcal{M}$, MSO), we have to express that a successful run of $\mathfrak{A}$ exists. Then the existence of a successful run is captured by a path through the domain $M \times \{0, \ldots, n\}$,

1. starting with an arbitrary element $(m, 0)$ succeeded by an element (m', j') such that the triple $(0, m', j')$ is an admissible initial transition,
2. continuing with steps from a pair (m, j) to (m', j') if (m, j, m', j') is an admissible transition,
3. ending with a pair (m, j) with $j \in F$.

This existence claim can be expressed in MSO-logic over $M \times \{0, \ldots, n\}$ by saying

> each set X containing the pairs satisfying condition 1, and closed under the steps according to condition 2, contains an element according to condition 3.

It is easy to express all three parts of this statement in MSO, using the fact that the transition formulas of $\mathfrak{A}$ are MSO-formulae themselves.

Finally, one employs the assumption that the MSO-theory of $\mathcal{M}$ is decidable. It is well known that this implies that also the MSO-theory of $\mathcal{M} \times \{0, \ldots, n\}$ is decidable (see e.g. [3], Prop. 3.12). Alternatively, one can show the claim directly by working over the domain M and using a tuple of universal set quantifiers over variables $X_0, \ldots, X_n$, rather than using "$\forall X$" over $M \times \{0, \ldots, n\}$.

Due to the effective closure of the class of languages recognized by strong automata over an alphabet frame $(\mathcal{M}, MSO)$ under Boolean operations, we conclude the following:

Corollary 1. *Over an alphabet frame $(\mathcal{M}, MSO)$ where the MSO-theory of $\mathcal{M}$ is decidable, the inclusion problem, the equivalence problem, and the universality problem for strong automata are decidable.*

2.4 Extensions and Undecidability Results

There are two natural extensions of the model of strong automaton: First we can proceed from input letters of M to input letters of M^2 when M is the domain of an alphabet structure, and secondly we can introduce transitions that connect more than two successive input letters. In both cases, the transition formulae have more free variables. We show that both extensions cause the non-emptiness problem to be undecidable, even for the basic alphabet frame $((\mathbb{N}, +1), FO)$.

In these results we use – without going into all details – the obvious extension of the framework of strong automata to "two-dimensional" input letters. The transition formulas can now be written as $\varphi_{pq}(x_1, x_2, y_1, y_2)$, connecting two successive input letters (x_1, x_2), (y_1, y_2) from $M \times M$, where M is the alphabet.

Theorem 2. *The non-emptiness problem for strong automata over $\mathbb{N}^2$ is undecidable when the alphabet frame is $((\mathbb{N}, +1), FO)$.*

We use the well-known result that for 2-register machines the reachability problem is undecidable (cf. [9]). The two registers are x_1, x_2, and the operations are $\texttt{INC}_j$ (increase the value of x_j by one), $\texttt{DEC}_j$, $\texttt{IF } x_j = 0 \texttt{ GOTO } m$, and $\texttt{GOTO } m$ with their usual semantics. The last instruction is always the `HALT` instruction. A configuration of such a machine R (with k instructions) is a tuple $(i, r_1, r_2) \in \{1, \ldots, k\} \times \mathbb{N}^2$ where i is the number of an instruction and r_j is the value stored in x_j. The reachability problem for 2-register machines with k instructions asks if a configuration (k, r_1, r_2) is reachable from $(1, 0, 0)$.

Proof. Let k be the number of instructions of the given 2-register machine R. One constructs a strong automaton $\mathfrak{A}_R = (Q, \mathbb{N}^2, q_0, \Delta, \{q_k\})$ whose language is non-empty iff R terminates from configuration $(1, 0, 0)$. Indeed, it is easy to build $\mathfrak{A}_R$ such that for a sequence of register contents $(0, 0)(m_2, n_2) \cdots (m_r, n_r)$ of a *halting* computation this word of $(\mathbb{N}^2)^*$ is the only word accepted by $\mathfrak{A}_R$, while the lack of such a halting computation causes $L(\mathfrak{A}_R)$ to be empty.

Let us turn to strong automata with higher "history extension", taking the view that the strong automata defined above have a "1-history" (they are allowed to look back one letter). Now we use transitions of the form $(p, \varphi(x_0, x_1, x_2), q)$, allowing to move from p to q via m_2 if the previous *two* letters were m_0, m_1, in this order, and $\mathcal{M} \models \varphi[m_0, m_1, m_2]$. We call these automata "strong automata with 2-history".

Theorem 3. *The non-emptiness problem for a strong automaton with 2-history is undecidable over the alphabet frame* $((\mathbb{N}, +1), FO)$.

Proof. From a 2-register machine R we construct a strong automaton $\mathfrak{A}_R$ with 2-history such that R has a terminating computation from $(1, 0, 0)$ iff $L(\mathfrak{A}_R) \neq \emptyset$. In order to simulate a computation of R, the strong automaton will in the even steps take care of simulation for the first register while in the odd steps will do the simulation for the second register. Since it is possible to look back 2 letters, an update of the two register contents can be done in two successive steps.

3 Three-Way-Grid Traversal Automata

3.1 Definitions

In this section, we consider automata that accept words whose letters are again words (over some finite alphabet Σ). We may write a sequence $w_1 w_2 \dots w_n$ of words w_i as a sequence of entries in a grid structure, where each w_i is noted in a column, starting from the bottom line of the grid upwards and using the symbol $\perp$ after the last letter of w_i is written. In special cases, we use words w_i as unary representations of natural numbers; here each w_i is a word on $\Sigma = \{1\}$ (cf. Figure 1).

Let Σ be a finite alphabet and $n \in \mathbb{N}$. Formally, a *grid word* of length n is given by a function that maps grid positions to letters, i.e. $w : \mathbb{N} \times \{1, \dots, n\} \to \Sigma \dot\cup \{\#, \perp\}$ such that for all $j \in \{1, \dots, n\}$:

1. $w(i, j) = \# \Leftrightarrow i = 0$,
2. $w(i, j) \in \Sigma \Rightarrow w(k, j) \in \Sigma$ for all $1 \leq k \leq i$,
3. $w(i, j) = \perp$ for some $i > 0$

With $\mathcal{G}(\Sigma) := \{w : \mathbb{N} \times \{1, \dots, n\} \to \Sigma \dot\cup \{\#, \perp\} \mid n \in \mathbb{N}, w \text{ is a grid word}\}$ we denote the set of all grid words on alphabet Σ. Given a subset $L \subseteq \mathcal{G}(\Sigma)$ the complement of L is defined as $\overline{L} = \mathcal{G}(\Sigma) \setminus L$. Finally, as mentioned above, note that $(\mathcal{G}(\{1\}), \cdot)$ is isomorphic to $(\mathbb{N}^*, \cdot)$ where "$\cdot$" is the usual word concatenation in both cases. For instance, the word $2\,1 \in \mathbb{N}^*$ corresponds to the grid word which has two 1's in the first column and a single 1 in the second column.

A *progressive grid automaton* is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ where Q is a finite set of states, Σ is the finite grid label alphabet, q_0 is the initial state, $\Delta \subseteq Q \times \Sigma \dot\cup \{\#, \perp\} \times Q \times \{\uparrow, \downarrow, \rightarrow\}$ is the transition relation (with $\Delta \cap Q \times \{\#\} \times Q \times \{\downarrow\} = \emptyset$), and $F \subseteq Q$ is the set of accepting states.

We call $\mathcal{A}$ a deterministic progressive grid automaton if Δ is functional. A *configuration* of $\mathcal{A}$ is an element in $Q \times \mathbb{N} \times \mathbb{N}$, consisting of the "current" state of $\mathcal{A}$ and its position in a grid word.

Let $w \in \mathcal{G}(\Sigma)$. Then a run of $\mathcal{A}$ on w is a finite sequence $\pi = c_0 \dots c_m$ of configurations that satisfies the following conditions (which define the three-way movement through the grid):

1. $c_0 = (q_0, 1, 1)$,
2. for all $0 \leq i < m$ it holds that if $c_i = (q_i, h_i, \ell_i)$ then
   - $c_{i+1} = (q_{i+1}, h_i, \ell_i + 1)$ and $(q_i, w(h_i, \ell_i), q_{i+1}, \rightarrow) \in \Delta$, or
   - $c_{i+1} = (q_{i+1}, h_i + 1, \ell_i)$ and $(q_i, w(h_i, \ell_i), q_{i+1}, \uparrow) \in \Delta$, or
   - $c_{i+1} = (q_{i+1}, h_i - 1, \ell_i)$ and $(q_i, w(h_i, \ell_i), q_{i+1}, \downarrow) \in \Delta$,
3. for all $0 \leq i < m : c_i \in Q \times \mathbb{N} \times \{0, \ldots, |w|\}$,
4. and $c_m = (q_m, h_m, |w| + 1)$ for some $q_m \in Q$ and $h_m \in \mathbb{N}$.

If $c_m \in F \times \mathbb{N} \times \mathbb{N}$ then π is called an accepting run. A language $L \subseteq \mathcal{G}(\Sigma)$ is called grid recognizable, if L is recognized by a progressive grid automaton $\mathcal{A}$. We will mainly consider words and languages over $\mathbb{N}$, where the automaton operates on the corresponding grid words in $\mathcal{G}(\{1\})$.

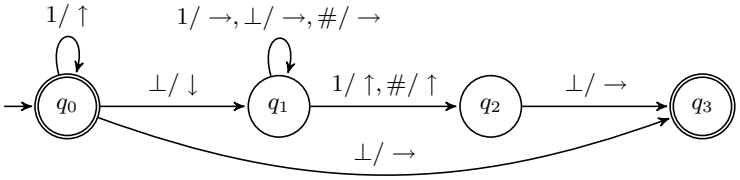

Fig. 3. A progressive grid automaton recognizing $L := \{n_0 \ldots n_p \in \mathbb{N}^* \mid n_0 = n_p\}$

Example 1. The progressive grid automaton pictured in Figure 3 recognizes the language $L := \{n_0 \ldots n_p \in \mathbb{N}^* \mid n_0 = n_p\}$. It operates as follows on a grid word in $\mathcal{G}(\{1\})$. In the beginning, the automaton moves to the "highest" 1 in the first column (or # if there is none). Afterwards the internal state is q_1. Then it moves non-deterministically to the last column and verifies that it equals the first one. Indeed, if this is the case, then the current position is labeled 1 (or #) and the position above is labeled with $\perp$. Otherwise, the last letter would be smaller or greater, respectively. This property is checked in states q_2 and q_3.

3.2 Closure Properties

Proposition 2. *The class of deterministically grid recognizable languages is effectively closed under complement.*

This result can be proven similarly to the complementation proof given in [7]. The basic idea is the introduction of several finite counters to check whether the automaton halts (i.e. moves to the right infinitely often).

Theorem 4. *Neither the class of non-deterministically grid recognizable nor the class of deterministically grid recognizable languages is closed under intersection (even over $\mathcal{G}(\{1\})$).*

Intuitively, theorem 4 can be justified as follows: A progressive grid automaton (deterministic or not) can only remember one letter (i.e. column) at a time it has seen before. This can be done by using the vertical position of the automaton upon a move to the right. Moreover, while remembering some letter the movement of the automaton is restricted.

Proof. We consider the following two languages:

$$L_1 := \{n_0 \dots n_k \mid k \in \mathbb{N}, \forall i \in \{2j \mid 0 \leq j \leq \lfloor \frac{k}{2} \rfloor\} : n_i = 0\}$$

$$L_2 := \{n_0 \dots n_k \mid k \in \mathbb{N}, \forall i_0, i_1 \in \{2j+1 \mid 0 \leq j \leq \lfloor \frac{k}{2} \rfloor\} : n_{i_0} = n_{i_1}\}$$

Both are clearly deterministically grid recognizable. It suffices to show that $L := L_1 \cap L_2$ is not grid recognizable. Let $L' := \{(0m)^n \mid n, m \in \mathbb{N}, n > 0\} \subseteq L$. Using a pumping argument it can be shown that every progressive grid automaton recognizing a superset of L' accepts a word not in L. Informally, for sufficient large m, n the automaton has to forget the value of m to check that every second column is labeled only with $\perp$.

Corollary 2. *1. The class of grid recognizable languages is not closed under complement.*
2. The class of deterministically grid recognizable languages is not closed under union.
3. The class of deterministically grid recognizable languages is strictly included in the class of (non-deterministically) grid recognizable languages.

Proof. For claim 1, observe that the class of grid recognizable languages is naturally closed under union (using non-determinism). Suppose it is closed under complement. Then closure under intersection follows immediately. However, this contradicts Theorem 4. Claim 2 follows immediately from Theorem 4. Thus, we also obtain claim 3.

3.3 Decidability of the Non-emptiness Problem

Theorem 5. *The non-emptiness problem for progressive grid automata is decidable.*

To prove this result, we will use the MSO-theory of $(\mathbb{N}, +1)$ (known to be decidable). We describe a run of a progressive grid automaton $\mathcal{A}$ on some grid word w in MSO. W.l.o.g. we can assume $\mathcal{A} = (\{1, \dots, n\}, \Sigma, \Delta, 1, F)$ for some $n \in \mathbb{N}$. At first, a single column of a grid word can be described in MSO-logic with the help of second order variables. The existence of an accepting run is certified by a sequence over $\{1, \dots, n\} \times \mathbb{N}$ that

1. starts with $(1, 1)$,
2. contains $(p, k)(q, \ell)$ if and only if there is a well-labeled column (i.e. the labeling is in $\#1^*\bot^\omega$) and a tuple (p', ℓ), such that there is an applicable transition $(p', a, q, \rightarrow)$ and (p', ℓ) is reachable from (p, k) in the considered column,
3. and ends with a pair in $F \times \mathbb{N}$.

Note that the third component of a configuration, which is the index of the column, is dropped here. The reachability within a single fixed column used in the second condition, i.e. if there is a path from a configuration (p, k, i) to (p', ℓ, i), can be described by the following least fixed point:

- (p, k) is in the fixed point, and
- if (q, m) is in the fixed point, the m-th row is labeled a, and $(q, a, q', \uparrow)$ or $(q, a, q', \downarrow)$ is a transition, then $(q', m + 1)$ resp. $(q', m - 1)$ is in the fixed point.

Then (p', ℓ, i) is reachable iff (p', ℓ) is in the fixed point. This fixed point can be expressed in MSO using universal set quantification as in Theorem 1. Finally, the conditions 1 to 3 can be formalized in MSO analogously.

3.4 The Non-universality Problem

Since the non-emptiness problem is decidable and deterministic progressive grid automata are closed under complement, we can immediately conclude that the non-universality problem is decidable. We show that the situation changes for progressive grid automata in general.

Theorem 6. *The non-universality problem for non-deterministic progressive grid automata is undecidable.*

Proof. We fix $\Sigma = \{1\}$ and consider a 2-register machine R. A computation of R can be encoded in a grid word u. For the i-th computation step the unary encodings of the first and second register correspond to the $(2i-1)$-th and $2i$-th column of u, respectively. The claim follows by constructing a progressive grid automaton $\mathcal{B}$ with

$$L(\mathcal{B}) := \mathcal{G}(\{1\}) \setminus \{uv \mid u \text{ encodes a terminating computation of } R\}.$$

That is, $L(\mathcal{B}) = \mathcal{G}(\{1\})$ iff there is no terminating computation of $\mathcal{M}$. To achieve this property, $\mathcal{B}$ needs to accept all grid words that do not have a prefix encoding a valid computation of R starting in $(1, 0, 0)$. The idea is that $\mathcal{B}$ remembers the current instruction in the state space and guesses if the encoding is wrong at some point.

4 Discussion, Related Work, Perspectives

4.1 Comparison Between the Two Models

The language classes given by strong automata and progressive grid automata cannot be compared directly since the underlying alphabet structures differ (alphabet frames involving a logic vs. input letters as sequences ("columns")). On the other hand, we can provide a comparison for the fixed alphabet frame $((\mathbb{N}, +1), MSO)$ on one side and the alphabet $\mathbb{N}$ on the other: *Each language* $L \subseteq \mathbb{N}^*$ *recognized by a strong automaton over* $((\mathbb{N}, +1), MSO)$ *is non-deterministically grid recognizable.*[2] *The converse fails in general (e.g., using the language* L *presented in Example 1).*

4.2 Comparison with Other Models

Let us compare our models with the register automata introduced by Kaminski and Francez [8] which recognize the "quasi-regular" languages over $\mathbb{N}$. The language of words $0123 \ldots \ell$ is not quasi-regular, but recognizable both by strong automata and progressive grid automata. On the other hand, the language $L = \{(0m)^n \mid m, n \in \mathbb{N}, n > 0\}$ is quasi-regular, but neither recognizable by a strong automaton nor by a progressive grid automaton. Also, the "data automata" of [4] only refer to the equality (and non-equality) of input letters from an abstract data domain and do not incorporate a possibility of comparison (as with $<$ over $\mathbb{N}$).

4.3 Extension to Infinite Words

It is not difficult to generalize the main results of this paper to the case of infinite words. There are technical details that require some care (e.g. in the proofs for the decidability of the emptiness problem and the complementation of Büchi automata). We do not present these details here, but refer the reader to [5],[11].

4.4 General Framework of Progressive Grid Automata

In this paper, we considered progressive grid automata over the infinite alphabets Σ^* and $\mathbb{N}$. It should be mentioned that the model is easily extensible to all alphabets whose letters are presentable as labellings of a fixed infinite graph (i.e. with fixed edge relation). In the cases of the alphabets Σ^* and $\mathbb{N}$, this graph is the successor structure of the natural numbers (into which elements of Σ^* or $\mathbb{N}$ were encoded by vertex labellings). Using the same underlying graph, one can also handle the alphabet Σ^ω. Using the binary infinite tree instead, different (finite) labellings code different binary terms. The "columns" of the present paper are then replaced by different labellings of the infinite binary tree, and a progressive grid automaton would work through such a labeled tree as a

[2] A proof may be found in [5].

tree-walking automaton before jumping (on a certain tree node position) to the next tree. In this way a theory of recognizable languages over the alphabet of (binary) terms can be developed. Similarly, one can work with any fixed infinite graph and its possible labellings as the letters of an infinite alphabet.

Pursuing a different direction (which is subject of current work) we can lift the idea of progressive grid *word* automata to progressive grid *tree* automata, e.g. working on $\mathbb{N}$-labeled trees.

Acknowledgments. We thank the reviewers of DLT 2015 for their remarks which led to improvements of the presentation.

References

1. Bès, A.: An application of the feferman-vaught theorem to automata and logics for words over an infinite alphabet. Logical Methods in Computer Science **4**(1) (2008)
2. Blum, M., Hewitt, C.: Automata on a 2-dimensional tape. In: IEEE Conference Record of the Eighth Annual Symposium on Switching and Automata Theory, SWAT 1967, pp. 155–160. IEEE (1967)
3. Blumensath, A., Colcombet, T., Löding, C.: Logical theories and compatible operations. In: Flum, J., et al. (eds.) Logic and Automata: History and Perspectives, pp. 73–106. Amsterdam Univ. Press (2008)
4. Bojańczyk, M., David, C., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on data words. ACM Transactions on Computational Logic (TOCL) **12**(4), 27 (2011)
5. Czyba, C., Spinrath, C., Thomas, W.: Finite Automata Over Infinite Alphabets: Two Models with Transitions for Local Change (Full Version). RWTH Aachen University (2015). https://www.lii.rwth-aachen.de/en/86-finite-automata-over-infinite-alphabets.html
6. Ebbinghaus, H.-D., Flum, J., Thomas, W.: Mathematical Logic, 2nd edn. Springer Undergraduate Texts in Mathematics and Technology. Springer (1996)
7. Inoue, K., Takanami, I.: A note on decision problems for three-way two-dimensional finite automata. Information Processing Letters **10**(4), 245–248 (1980)
8. Kaminski, M., Francez, N.: Finite-memory automata. Theoretical Computer Science **134**(2), 329–363 (1994)
9. Minsky, M.L.: Computation: finite and infinite machines. Prentice-Hall Inc, Upper Saddle River (1967)
10. Rosenfeld, A.: Picture languages. Academic Press (1979)
11. Spelten, A., Thomas, W., Winter, S.: Trees over infinite structures and path logics with synchronization. In: Yu, F., Wang, C. (eds.) Proceedings 13th International Workshop on Verification of Infinite-State Systems, INFINITY 2011, vol. 73, pp. 20–34. EPTCS, Taipei (2011)
12. Spelten, A.: Paths in Infinite Trees: Logics and Automata. PhD thesis, RWTH-Aachen University (2013)
13. Thomas, W.: On the bounded monadic theory of well-ordered structures. The Journal of Symbolic Logic **45**(02), 334–338 (1980)
14. Thomas, W.: Languages, automata, and logic. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, pp. 389–455. Springer, Heidelberg (1997)