Bachelor Thesis

Recognizing Languages over Infinite Alphabets using Automata on Grids

submitted by

Christopher Spinrath

submitted to the

Faculty of Mathematics, Computer Science and Natural Sciences
at RWTH Aachen University

September 4, 2014

First supervisor
Prof. Dr. Dr.h.c. Wolfgang Thomas
Logic and Theory of Discrete Systems, Informatik 7
RWTH Aachen University

Second supervisor
Dr. Christof Löding
Logic and Theory of Discrete Systems, Informatik 7
RWTH Aachen University

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen sind, sind als solche kenntlich gemacht.

Aachen, 4.9.2014

Christopher Spinrath

Abstract

A new automata model is established that uses specific input words – namely grids – to recognize languages over infinite alphabets. The closure properties of those languages are studied and decidabillity as well as undecidabillity results concerning the new model are presented. In particular, the decidabillity of the emptiness problem is proved. Moreover, it is shown that the expressiveness of this model is different from already existing models – the “Strong Automata” and “Register Automata”.

Contents

1 Introduction

While finite state automata are widely established as canonical automata model for languages over finite alphabets (cf. [RS59]) a different situation emerges for infinite alphabets like $\mathbb{N}$. Since languages over infinite alphabets are useful in computer science, e.g. for the purpose of model checking, several automata models have been proposed. In 1994 Kaminski and Francez introduced Register Automata (cf. [KF94]) which can be considered as generalization of the finite state automata. Additionally, the use of pebble automata to recognize languages over infinite alphabets have been studied and compared to Register Automata in [NSV04]. In 2008 Bès proposed another model that uses logic formulae as labels for transitions to capture infinitely many letters (cf. [Bès08]). Lastly, Spelten, Thomas and Winter introduced Strong Automata which expands the model proposed by Bès by the ability to compare successive letters. However, in comparison with the finite state automata over finite alphabets all these models either lack closure properties, cannot distinguish between infinite subsets of the alphabet or cannot recognize common languages like the language that contains all words where the last letter equals the first one (cf. the given resources and chapter 5 of this thesis).

In this thesis we will introduce another finite automata model to recognize languages over the infinite alphabet $\mathbb{N}$. Note that the letters in any countable alphabet can easily be identified with the natural numbers. Thus, this is a moderate restriction. To avoid confusion we remark that $0 \in \mathbb{N}$ throughout the whole paper. While we will focus on languages of finite words we will extend our results to languages of infinite words over $\mathbb{N}$ wherever possible. Therefore, we are going to adapt the Büchi-acceptance condition (cf. [Tho90]) to our automata model.

There are two major challenges in the design of finite automata models over infinite alphabets. Firstly, an automaton should be able to remember a letter read previously. Secondly, it is desirable to maintain a finite representation of the transitions. In the case of finite alphabets the automaton can remember patterns in its finite state space and the set of transitions is naturally finite. We will avoid both problems in the definition of the model itself and handle them in the semantics. Therefore, we will demand that our automata operate on a special kind of input words which are quite naturally isomorphic to the words over the designated alphabet $\mathbb{N}$ and have a discrete grid structure. Using this structure the automaton can compare and distinguish between different letters. Furthermore, a letter is not read atomically but the automaton processes the grid structure induced by this letter (the complete grid structure is induced by all letters of the word and their order). Since these grids are discrete this "processing" requires only a finite set of transitions.

In the second chapter we will introduce the special input words as well as a deterministic and a non-deterministic variant of our automata model and their semantics. Moreover, we will establish the isomorphism between those words and words over $\mathbb{N}$. The third chapter deals with the closure properties of the languages recognized by the automata model. We will see that the closure properties are the weakness of our model and that they differ

between the deterministic and non-deterministic automata variant. Further on, we will study some common decision problems with respect to our model in the fourth chapter. In particular, we will see that the emptiness problem is decidable but not for a "two-way" variant. Finally, in the fifth chapter we will compare our model with Strong Automata as well as Register Automata. Thereby, we will show that our automata model recognizes a set of languages that is incomparable to the set of languages recognized by those models.

2 Grid Words and Grid Word Automata

The objective of this chapter is the introduction of the automata model which will be studied in this paper. Since those automata operate on grids – or more precisely Grid Words – we will formalize them as well as some useful properties first.

2.1 Grid Words

Definition 1 (Grid Word)**.** Let $n \in \mathbb{N}$. A function $w : \mathbb{N} \times \{0, \dots, n+1\} \to \{\#, 0, 1\}$ such that for all $j \in \{1, \dots, n\}$:

1) $w(i, j) = \# \Leftrightarrow i = 0$,

2) $w(i, j) = 1 \Rightarrow w(k, j) = 1$ for all $1 \leq k \leq i$,

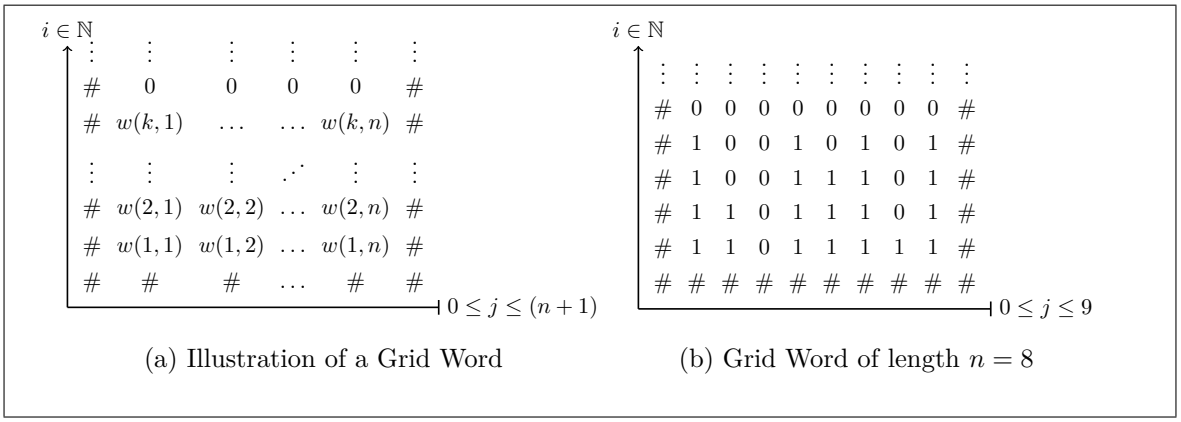
3) $w(i, j) = 0$ for some $i > 0$

and $w(i, 0) = w(i, n + 1) = \#$ for all $i \in \mathbb{N}$ is called a *Grid Word* (of length n).

Definition 2 (ω-Grid Word)**.** A function $v : \mathbb{N} \times \mathbb{N} \to \{\#, 0, 1\}$ such that for all $j \geq 1$ the conditions 1), 2) and 3) from the definition of a Grid Word hold and $v(i, 0) = \#$ for all $i \in \mathbb{N}$ is called an *ω-Grid Word*.

With $\mathcal{G} := \{w : \mathbb{N} \times \{0, \dots, n+1\} \to \{\#, 0, 1\} \mid n \in \mathbb{N}, w \text{ is a Grid Word}\}$ we denote the set of all Grid Words. Furthermore, $\mathcal{G}_\omega := \{v : \mathbb{N} \times \mathbb{N} \to \{\#, 0, 1\} \mid v \text{ is a } \omega\text{-Grid Word}\}$ denotes the set of all ω-Grid Words. Given a subset $L \subseteq \mathcal{G}$ ($L \subseteq \mathcal{G}_\omega$) the *complement* of L is defined as $\overline{L} = \mathcal{G} \setminus L$ ($\overline{L} = \mathcal{G}_\omega \setminus L$).

Example 1. Each Grid Word can be illustrated as a discrete grid in the two-dimensional space as shown in figure 2.1a. An explicit example of a depicted Grid Word of length eight is given in figure 2.1b.



(a) Illustration of a Grid Word

(b) Grid Word of length $n = 8$

Figure 2.1: Grid Word – Examples

In the following we will introduce some more terms which are helpful when arguing about Grid Words. Therefore, we fix two Grid Word $w, v \in \mathcal{G}$. Then $|w|$ denotes the length of w and $w(i,j)$ is the *label* of w at *position* (i,j) for all $(i,j) \in \mathbb{N} \times \{0,\dots,|w|,|w|+1\}$. Analogous to words over finite alphabets the *concatenation* of Grid Words is defined as follows:

$$w \cdot v : \mathbb{N} \times \{0,\dots,|w|+|v|+1\} \to \{\#,0,1\}, (i,j) \mapsto \begin{cases} w(i,j), & j \leq |w| \\ v(i,j-|w|), & j > |w| \end{cases}$$

Note that $w \cdot v$ is a well-defined Grid Word. As usual we omit the dot and write wv instead of $w \cdot v$ if the used operation is clear. If we fix the second parameter of w to some $0 \leq j \leq |w|+1$ the resulting function $w_j : \mathbb{N} \to \{\#,0,1\}, i \mapsto w(i,j)$ is the j-th *letter* or *column* of w. For convenience we extend the concatenation such that we can write w as the unique sequence of its columns $w = w_1 \dots w_n$. The unique Grid Word of length zero, also referred to as the empty Grid Word, is denoted by ϵ.

At last, we define $h(w) := \min\{h \in \mathbb{N} \mid w(h+1,j) = 0 \ \forall 1 \leq j \leq |w|\} \in \mathbb{N}$. $h(w)$ is called the *height* of w. The well-definedness of $h(w)$ follows directly from conditions 2) and 3) of the definition of a Grid Word. This implies that a Grid Word carries only a finite amount of information. As a result, a Grid Word may be part of an input for an algorithm (e.g. a decision procedure).

Example 2. The Grid Word in figure 2.1b has the height $h = 4$, the length $|w| = 8$ and $w_3(i) = 0$ for all $i \in \mathbb{N} \setminus \{0\}$. Besides, it holds that $w_5(1) = w_5(2) = w_5(3) = 1$ as well as $w_5(i) = 0$ for $i > 3$.

For ω-Grid Words the notions of *position*, *label* and *letter* are defined completely analogous. Further on, the *height* of an ω-Grid Word v is $h(v) := \min\{h \in \mathbb{N} \mid w(h+1,j) = 0 \ \forall j \geq 1\}$ or $h(v) := \infty$ if the minimum is not defined. The "concatenation" of ω-Grid Words is not meaningful but each ω-Grid Word v can be written as $v = w \cdot u$ where u is an ω-Grid Word and w is a Grid Word of arbitrary length. Then w is a prefix of v and we demand that the decomposition wu is compatible with the concatenation of Grid Words. Note that in contrast to a Grid Word an ω-Grid Word may contain an infinite amount of information.

The following result establishes the announced connection between words over $\mathbb{N}$ and Grid Words. In particular, it will allow us to describe subsets of $\mathcal{G}$ as subsets of $\mathbb{N}^*$ which is more convenient than describing sets of Grid Words. Moreover, it simplifies the comparison of the automata model introduced in the next section with other models (cf. chapter 5).

Lemma 1. *Let $(\mathbb{N}^*, \odot)$ be the structure of all finite words over the infinite alphabet $\mathbb{N}$ with the usual word concatenation. Then it holds that $(\mathbb{N}^*, \odot) \cong (\mathcal{G}, \cdot)$. In particular, the function Λ that maps a natural number n to the Grid Word of length one where the only column corresponds to the unary representation of n induces an isomorphism between $(\mathbb{N}^*, \odot)$ and $(\mathcal{G}, \cdot)$. Formally Λ is defined as follows:*

$$\Lambda : \mathbb{N} \to \{f : \mathbb{N} \times \{0,1,2\} \to \{\#,0,1\}\}, n \mapsto w_n \text{ where } w_n(i,j) := \begin{cases} \#, & j \neq 1 \vee i = 0 \\ 1, & j = 1 \wedge i \leq n \\ 0, & j = 1 \wedge i > n \end{cases}$$

Proof. First of all, we have that $\Lambda(n)$ is a Grid Word of length one for all $n \in \mathbb{N}$ by definition. Let $n, m \in \mathbb{N}$, $n \neq m$ and w.l.o.g. $n < m$. Then $\Lambda(n) \neq \Lambda(m)$ because $\Lambda(n)(m,1) = 0 \neq 1 = \Lambda(m)(m,1)$. Hence, Λ is injective. Moreover, given a Grid Word $w \in \mathcal{G}$ of length one it holds that $\Lambda(h(w)) = w$. Thus, Λ is surjective and therefore bijective.

The canonical extension of Λ on $(\mathbb{N}^*, \odot)$ is given by $\lambda(\epsilon) := \epsilon$, and $\lambda(n_0 \odot \ldots \odot n_\ell) := \Lambda(n_0) \cdot \ldots \cdot \Lambda(n_\ell)$ for $\ell > 0$. The bijectivity is inherited from Λ and λ is compatible with concatenation by definition. Since each Grid Word is the concatenation of its columns which can be identified with Grid Words of length one the image of λ is $\mathcal{G}$. Hence, λ is an isomorphism between $(\mathbb{N}^*, \odot)$ and $(\mathcal{G}, \cdot)$. □

Definition 3. $\lambda : \mathbb{N}^* \to \mathcal{G}$ is defined as the isomorphism induced by Λ as given in Lemma 1.

Remark 1. Let $\mathbb{N}^\omega$ be the set of all infinite words over $\mathbb{N}$. Then λ can be extended to a bijection between $\mathbb{N}^\omega$ and $\mathcal{G}_\omega$ that preserves the prefix relation by defining $\lambda(n_1 n_2 n_3 \ldots) := \lambda(n_1)\lambda(n_2)\lambda(n_3)\ldots$ for all $n_1 n_2 n_3 \ldots \in \mathbb{N}^\omega$. For convenience we use the same notation for this extension.

Example 3. Let w be the Grid Word depicted in figure 2.1b. Then it holds that $\lambda^{-1}(w) = 42043414 \in \mathbb{N}^*$ and $\lambda(42043414) = w$.

2.2 Progressive Grid Word Automata

Now, we are ready to define the automata model that will be studied throughout this paper – the Progressive Grid Word Automata. We will concentrate on the model accepting finite words at first and adapt it later to accept infinite words. The basic idea is that such an automaton "reads" a Grid Word starting in the lower left corner (cf. the graphical representation 2.1). In each step it reads the current label and decides if it moves up, down or to the right and starts over. The term "progressive" derives from the fact that the automaton cannot move to the left and therefore progresses towards the end of the Grid Word every time it moves horizontally. In the following we define a deterministic as well as a non-deterministic version.

Definition 4 (Deterministic Progressive Grid Word Automaton)**.** A *Deterministic Progressive Grid Word Automaton* is a tuple $\mathcal{A} = (Q, \delta, q_0, F)$ where

- Q is a finite set of states,
- $\delta : Q \times \{\#, 0, 1\} \to Q \times \{\uparrow, \downarrow, \rightarrow\}$ is the transition function with $\delta(Q \times \{\#\}) \subseteq Q \times \{\uparrow, \rightarrow\}$,
- $q_0 \in Q$ is the initial state and
- $F \subseteq Q$ is the set of accepting states.

Definition 5 (Non-deterministic Progressive Grid Word Automaton)**.** A *Non-deterministic Progressive Grid Word Automaton* is a tuple $\mathcal{A} = (Q, \Delta, q_0, F)$ where

- Q is a finite set of states,
- $\Delta \subseteq Q \times \{\#, 0, 1\} \times Q \times \{\uparrow, \downarrow, \rightarrow\}$ is the transition relation with $\Delta \cap Q \times \{\#\} \times Q \times \{\downarrow\} = \emptyset$,

- $q_0 \in Q$ is the initial state and
- $F \subseteq Q$ is the set of accepting states.

Note that the restriction of the transition function in the deterministic case is consistent with the restriction of the transition relation in the non-deterministic case. In both cases it prevents an automaton from "falling below" the lower bound of a Grid Word. Although this could be handled in the semantics it would be necessary to deal with this special case in most proofs which argue about the behavior of an automaton. Besides, each Deterministic Progressive Grid Word Automaton is a Non-deterministic Progressive Grid Word Automaton. Therefore, we argue about a transition relation in the general case.

Semantics of Progressive Grid Word Automata

The aim of this section is the definition of the behavior of Progressive Grid Word Automata on Grid Words. For this purpose we will consider a fixed (Non-)Deterministic Grid Word Automaton $\mathcal{A} = (Q, \Delta, q_0, F)$.

A *configuration* of $\mathcal{A}$ is an element in $Q \times \mathbb{N} \times \mathbb{N}$. Informally a configuration consists of the "current" state of $\mathcal{A}$ and a position of a Grid Word. Let $w \in \mathcal{G}$. 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:

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 $0 \leq i < m : c_i \in Q \times \mathbb{N} \times \{0, \dots, |w|\}$,
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*. In case π does not satisfy condition 1) or 4) it is called a *partial run* of $\mathcal{A}$ on w. Moreover, if $\mathcal{A}$ is deterministic then for all partial runs $\rho_1 = (q_0, 1, 1) \dots c_{m_1}$,$\rho_2 = (q_0, 1, 1) \dots c_{m_2}$ it holds that either $\rho_1 = \rho_2$ or $m_1 \neq m_2$. In particular, if there is an accepting run of $\mathcal{A}$ on w then this run is unique. Aside from that we just omit the specification of $\mathcal{A}$ and w if they are implied by the context. $\mathcal{A}$ *halts* on w if the following conditions are satisfied:

- There is no infinite sequence $(q_0, 1, 1) \dots$ of configurations such that every finite prefix is a partial run of $\mathcal{A}$ on w,
- and every partial run $(q_0, 1, 1) \dots (q, h, \ell)$ can be extended to a run of $\mathcal{A}$ on w.

$\mathcal{A}$ *accepts* $w \in \mathcal{G}$ if there is an accepting run of $\mathcal{A}$ on w. Note that $\mathcal{A}$ accepts ϵ if and only if $q_0 \in F$. Furthermore, if $\mathcal{A}$ is deterministic and accepts w then $\mathcal{A}$ halts on w because the run is unique and has no proper extension. If $\mathcal{A}$ does not accept w then $\mathcal{A}$ *rejects* w. Besides, the set

$$L(\mathcal{A}) := \{w \in \mathcal{G} \mid \mathcal{A} \text{ accepts } w\} \subseteq \mathcal{G}$$

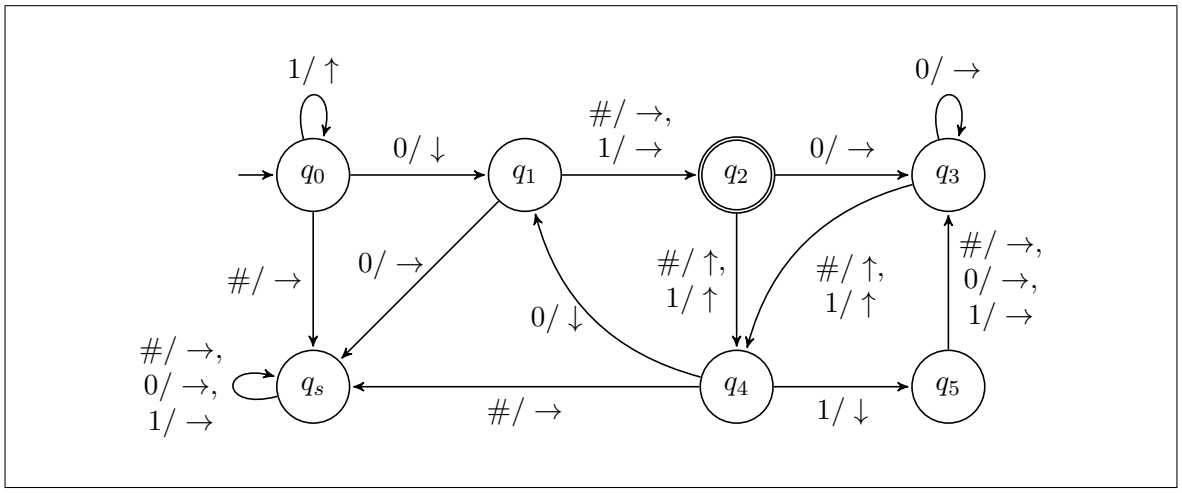


Figure 2.2: Example of a Deterministic Progressive Grid Word Automaton

is the language *recognized* by $\mathcal{A}$. Conversely, a language $L \subseteq \mathcal{G}$ is *(non)-deterministically grid recognizable* if and only if there is a (Non-)Deterministic Progressive Grid Word Automaton $\mathcal{A}$ with $L(\mathcal{A}) = L$. As we will see in chapter 3 the distinction between deterministically and non-deterministically is important here.

Further on, we define analogous acceptance conditions for words over $\mathbb{N}^*$: $\mathcal{A}$ *accepts* $w \in \mathbb{N}^*$ if there is an accepting run of $\mathcal{A}$ on $\lambda(w)$ and $L \subseteq \mathbb{N}^*$ is *(non)-deterministically grid recognizable* if and only if there is a (Non-)Deterministic Progressive Grid Word Automaton $\mathcal{A}$ with $L(\mathcal{A}) = \lambda(L)$.

Example 4. As usual in automata theory a (Non-)Deterministic Progressive Grid Word Automaton can be illustrated as a transition graph. An edge from the state q to the state p in the transition graph is labeled with x/d if (q, x, p, d) is a transition of the automaton.

The Deterministic Progressive Grid Word Automaton $\mathcal{A}$ pictured in figure 2.2 operates as follows on a Grid Word. In the beginning the automaton moves to the "highest" 1 in the first column. If there is no 1 then $\mathcal{A}$ moves to the bottom symbol #. Afterwards the internal state is q_1 and $\mathcal{A}$ moves to the right (note that the transition labeled with 0 cannot be taken) and enters the state q_2. Now, $\mathcal{A}$ tries to verify that the current column equals the first one. Indeed, if the current column equals the first column then $\mathcal{A}$ reads a 1 and the position above is labeled with a 0. This corresponds to the state sequence q_2, q_4, q_1. Since q_4 can only be entered by reading a 1 and moving upwards and the transition to q_1 implies a move downwards, $\mathcal{A}$ will read a 1 (or the bottom symbol) in q_1. In particular, $\mathcal{A}$'s internal state is q_1 if and only if the current column equals the first one and $\mathcal{A}$ reads the "highest" 1. Note that $\mathcal{A}$ is in the same situation as before. That is, it moves to the right and tries to verify that the next column equals the first column. If the verification fails then $\mathcal{A}$ eventually enters the state q_3 and moves to the right but keeps the vertical position in the Grid Word. As in the positive case $\mathcal{A}$ tries to verify that the next column equals the first column. Finally, $\mathcal{A}$ moves to the right and enters the only accepting state q_2 if and only if the current column equals the first one because the only incoming transitions are those from q_1. In conclusion, $\mathcal{A}$ recognizes the language

$$L := \{n_0 \dots n_p \in \mathbb{N}^* \mid p \in \mathbb{N} \wedge n_0 = n_p\}.$$

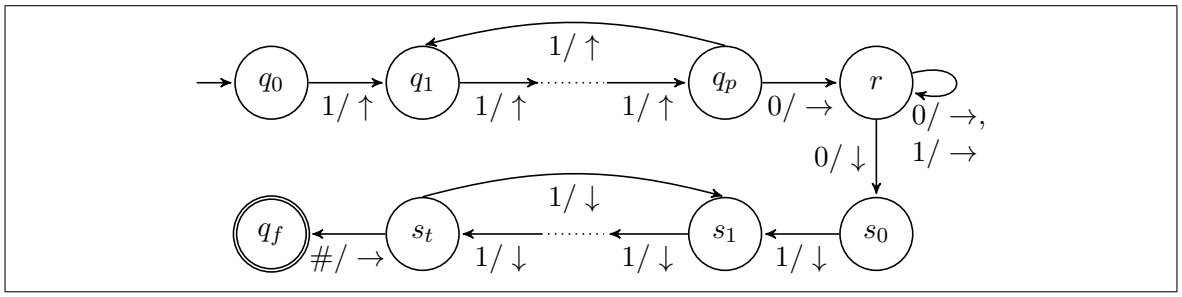


Figure 2.3: Non-deterministic Progressive Grid Word Automaton discussed in example 5

Therefore L is deterministically grid recognizable and $\mathcal{A}$ accepts $w = \lambda(121)$. The accepting run witnessing $w \in L(\mathcal{A})$ is as follows:

$$(q_0, 1, 1)(q_0, 2, 1)(q_1, 1, 1)(q_2, 1, 2)(q_4, 2, 2)(q_5, 1, 2)(q_3, 1, 3)(q_4, 2, 3)(q_1, 1, 3)(q_2, 1, 4)$$

This example also demonstrates that the length of a run may be greater than the length of an input word. In addition, the run above is an accepting run of $\mathcal{A}$ on $\lambda(141)$, too. Thus, an accepting run may witness the membership of several Grid Words to $L(\mathcal{A})$.

Example 5. Consider the Non-deterministic Progressive Grid Word Automaton with parameters $p, t \in \mathbb{N}$ illustrated in figure 2.3. On a Grid Word $w = \lambda(n) \dots \lambda(m)$ it behaves as follows: Firstly, it checks whether n is a multiple of p. Afterwards, it moves to the right until it reaches the last column of w. Note that the automaton guesses in r if the current column is the last one and that the automaton requires that $n, m > 0$. Then the automaton verifies that $m = n$ and m is a multiple of t. We can observe three important properties of Progressive Grid Word Automata:

1) If p and t are different prim numbers then the automaton has to visit the states $q_1, \dots, q_p$ as well as $s_1, \dots, s_t$ multiple times. More precisely t and p times, respectively.

2) Furthermore, if n is a multiple of p but either m is not a multiple of t or $n \neq m$ then the following situation arises. The automaton "leaves" the first column and until it reads the last column of w there is no indication that the run will "fail".

3) If all transitions $(q, 1, q', d)$ of the automaton are replaced with $(q, 0, q', d)$ the automaton accepts 0^k for all $k > 1$. However, an accepting run visits positions (h, ℓ) where $h = \text{lcm}(p, t) + 1 > h(w) = 0$.

These observations will have an influence on the hardness of some decision problems as we will see later.

Remark 2. Consider the automaton pictured in figure 2.3 and explained in example 5. It can be transformed in the following way: The state r is removed and a transition $(q_p, 0, s_0, \downarrow)$ is introduced. Furthermore, a sink state has to be introduced and all transitions missing in the figure should lead to this state. This transformation yields a Deterministic Progressive Grid Word Automaton $\mathcal{A}_{p,t}$ that recognizes the language $L_{p,t} = \{n \in \mathbb{N} \mid p \text{ and } t \text{ divide } n\}$. Note that $L_{p,t}$ contains only words of length one. If p and t are prime numbers then $\mathcal{A}_{p,t}$ is a

minimal (w.r.t. the number of states) Deterministic Progressive Automaton that recognizes $L_{p,t}$. Besides, it holds that $L(\mathcal{A}_{p,t}) = L(\mathcal{A}_{t,p})$ but $\mathcal{A}_{p,t} \neq \mathcal{A}_{t,p}$. We conclude that given a deterministically grid recognizable language L there is no unique minimal Deterministic Grid Word Automaton that recognizes L.

2.3 Progressive Grid Word Büchi Automata

If we turn towards accepting ω-Grid Words the definition of the automata model stays essentially the same. Nevertheless, we will give a brief definition for the sake of completeness.

Definition 6 (Progressive Grid Word Büchi Automata)**.** A *Deterministic Progressive Grid Word Büchi Automaton* is a tuple $\mathcal{A} = (Q, \delta, q_0, F)$. The single components are precisely the same as for a Deterministic Progressive Grid Word Automaton in definition 4.

Furthermore, a *Non-deterministic Progressive Grid Word Büchi Automaton* is a tuple $\mathcal{A} = (Q, \Delta, q_0, F)$. Again, the single components are precisely the same as for a Non-deterministic Progressive Grid Word Automaton in definition 5.

However, ω-Grid Words do not have a "right border". Thus, the semantics need an adaption. Let $\mathcal{A} = (Q, \Delta, q_0, F)$ be a Progressive Grid Word Büchi Automaton and $w \in \mathcal{G}_\omega$. An infinite sequence $\pi = c_0 c_1 \dots$ of configurations is a *run* of $\mathcal{A}$ on w if it satisfies the following conditions:

1) $c_0 = (q_0, 1, 1)$,

2) for all $i \in \mathbb{N}$ 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) and for all $\ell > 0$ there is $m \in \mathbb{N}$ such that $c_m = (q, h, \ell)$ for some $q \in Q$ and $h \in \mathbb{N}$.

Note that the last condition forces the automaton to move to the right infinitely times while processing w. Analogous to the finite case π is called a *partial run* if π does not satisfy condition 1) or 3). Further on, π is an *accepting run* if for all $\ell > 0$ there are $\ell', h, m \in \mathbb{N}$, $q \in Q$ and $q_f \in F$ such that $c_{m-1} = (q, h, \ell + \ell')$ and $c_m = (q_f, h, \ell + \ell' + 1)$. That is, $\mathcal{A}$ enters infinitely often an accepting state by moving to the right. This acceptance condition corresponds to the acceptance condition introduced by Büchi for accepting infinite words over an infinite alphabet (cf. [Büc62; Tho90]). The remaining notations are again completely analogous to the case of finite words. $\mathcal{A}$ *accepts* $w \in \mathcal{G}_\omega$ ($w \in \mathbb{N}^\omega$) if there is an accepting run of $\mathcal{A}$ on w (on $\lambda(w)$). Otherwise, $\mathcal{A}$ rejects w. Moreover, the language recognized by $\mathcal{A}$ is defined as

$$L(\mathcal{A}) = \{w \in \mathcal{G}_\omega \mid \mathcal{A} \text{ accepts } w\} \subseteq \mathcal{G}_\omega.$$

Finally, a language $L \subseteq \mathcal{G}_\omega$ ($L \subseteq \mathbb{N}^\omega$) is *(non)-deterministically ω-grid recognizable* if and only if there is a (Non-)Deterministic Progressive Grid Word Büchi Automaton $\mathcal{A}$ with $L(\mathcal{A}) = L$ (with $L(\mathcal{A}) = \lambda(L)$).

Example 6. If we understand the automaton illustrated in figure 2.2 as a Deterministic Progressive Grid Word Büchi Automaton it recognizes precisely the language of all ω-Grid Words where the first letter appears infinitely often.

2.4 General Grid Word Automata

We finalize this chapter with the definition of a more general extension of Progressive Grid Word (Büchi) Automata. Basically we allow that an automaton "moves left on a Grid Word". Since we will see later in chapter 4 that this ability renders most decision problems undecidable we restrict ourselves to a brief introduction of the non-deterministic version.

Definition 7 (General Grid Word Automaton)**.** A *General Grid Word Automaton* is a tuple $\mathcal{A} = (Q, \Delta, q_0, F)$ where

- Q is a finite set of states,
- $\Delta \subseteq Q \times \{\#, 0, 1\} \times Q \times \{\uparrow, \downarrow, \rightarrow, \leftarrow\}$ is the transition relation that satisfies $\Delta \cap Q \times \{\#\} \times Q \times \{\downarrow\} = \emptyset$,
- $q_0 \in Q$ is the initial state and
- $F \subseteq Q$ is the set of accepting states.

The definition of the acceptance conditions of a General Grid Word Automaton are very similar to those of a Progressive Grid Word Automaton. So, instead of redefining everything we just name the differences. In fact only conditions 2) and 3) of the definition of a run need an adaption:

2') Alongside the cases in 2) $c_{i+1} = (q_{i+1}, h_i, \ell_i - 1)$ and $(q_i, w(h_i, \ell_i), q_{i+1}, \leftarrow) \in \Delta$ is allowed,

3') and for all $0 \leq i < m : c_i \in Q \times \mathbb{N} \times \{0, \ldots, |w| + 1\}$.

Essentially, the last condition allows the automaton to move to the left after it has read the right "border" of a Grid Word. Of course, we can also generalize the Progressive Grid Word Büchi Automata.

Definition 8 (General Grid Word Büchi Automaton)**.** A *General Grid Word Büchi Automaton* is a tuple $\mathcal{A} = (Q, \Delta, q_0, F)$ where the single components are exactly the same as for General Grid Word Automata.

This time we have to adapt only condition 2) of the definition of a run of Progressive Grid Word Büchi Automata:

2') Alongside the cases in 2) $c_{i+1} = (q_{i+1}, h_i, \ell_i - 1)$ and $(q_i, w(h_i, \ell_i), q_{i+1}, \leftarrow) \in \Delta$ is allowed.

3 Closure Properties

In this chapter the closure properties of deterministically as well as non-deterministically grid recognizable languages are worked out. It turns out that they are quite different. Most results also apply for (non-)deterministically ω-grid recognizable languages with only minor modifications. Therefore, we will show our findings for grid recognizable languages and state the differences afterwards in most cases. An overview over all results which are proved in this chapter is given in table 3.1. Further on, the proofs of the positive results are all constructive. That is, if it is claimed that a language L is (non-)deterministically (ω)-grid recognizable then the proof contains a construction of a proper automaton recognizing L. We start with two theorems which are intuitively accessible but the formal proofs need some technical details.

Theorem 1. *Given two Non-deterministic Progressive Grid Word (Büchi) Automata $\mathcal{A}_1$ and $\mathcal{A}_2$ one can effectively construct a Non-deterministic Progressive Grid Word (Büchi) Automaton $\mathcal{B}$ such that $L(\mathcal{B}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$. In particular, non-deterministically grid recognizable languages and non-deterministically ω-grid recognizable languages are closed under union.*

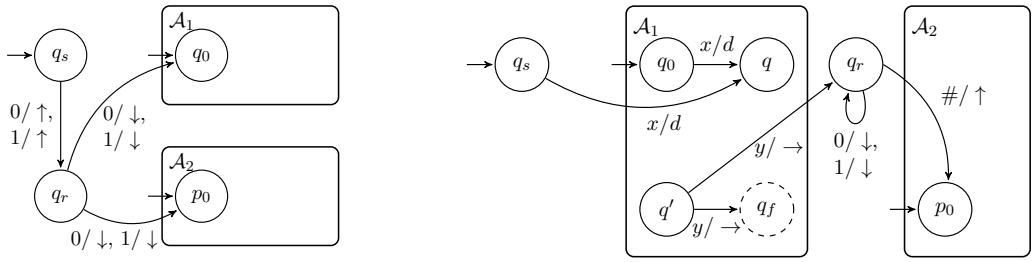
Proof. Let L_1 and L_2 be non-deterministically grid recognizable and $\mathcal{A}_1 = (Q, \Delta_1, q_0, F_1)$ as well as $\mathcal{A}_2 = (P, \Delta_2, p_0, F_2)$ be the witnessing Non-deterministic Progressive Grid Word Automata, respectively.

We have to show the closure under union. Therefore, we define $\mathcal{B} := (Q_\mathcal{B}, \Delta_\mathcal{B}, q_s, F_\mathcal{B})$ where

- $Q_\mathcal{B} := Q \dot\cup P \dot\cup \{q_s, q_r\}$,
- $\Delta_\mathcal{B} := \Delta_1 \dot\cup \Delta_2 \dot\cup \{(q_s, x, q_r, \uparrow), (q_r, x, q_0, \downarrow), (q_r, x, p_0, \downarrow) \mid x \in \{0, 1\}\}$
- and $F_\mathcal{B} := F_1 \dot\cup F_2 \dot\cup \{q_s\}$ if $q_0 \in F_1$ or $p_0 \in F_2$, otherwise $F_\mathcal{B} := F_1 \dot\cup F_2$.

	grid recognizable		ω-grid recognizable	
closed under	deterministically	non-deterministically		deterministically
Complement	Yes ✓	No	No	No
Intersection	No	No	No	No
Union	No	Yes ✓	Yes ✓	No
Concatenation	No	Yes ✓	-	-

Table 3.1: Overview of the closure properties of non-deterministically as well as deterministically (ω-)grid recognizable languages



(a) Graphical representation of $\mathcal{B}$ with $L(\mathcal{B}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

(b) Simplified illustration of $\mathcal{C}$ with $L(\mathcal{C}) = L(\mathcal{A}_1)L(\mathcal{A}_2)$ where the initial state has only one outgoing and the acceptance state of $\mathcal{A}_1$ has only one incoming transition. The dashed state q_f is an accepting state w.r.t. $\mathcal{A}_1$ but not w.r.t. $\mathcal{C}$.

Figure 3.1: Illustrations of the constructions of the proofs of theorems 1 and 2

Intuitively, $\mathcal{B}$ decides non-deterministically if it simulates either $\mathcal{A}_1$ or $\mathcal{A}_2$. The intermediate state q_r is required to return to the correct position on the Grid Word. The construction is illustrated in figure 3.1a. We show that $L_1 \cup L_2 = L(\mathcal{A}_1) \cup L(\mathcal{A}_2) = L(\mathcal{B})$. Then $L_1 \cup L_2$ is non-deterministically grid recognizable by definition. Let $w \in L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$. W.l.o.g. we can consider the case $w \in L(\mathcal{A}_1)$. Then there is a run $\pi = (q_0, 1, 1) \dots (q_f, h, \ell)$ of $\mathcal{A}_1$ on w where $q_f \in F_1$. If $w = \epsilon$ then $\pi = (q_0, 1, 1)$ and therefore $q_0 \in F_1$. Hence, $q_s \in F_\mathcal{B}$ and $(q_s, 1, 1)$ is an accepting run of $\mathcal{B}$ on w. Otherwise, $|w| > 0$. Thus, $(q_s, 1, 1)(q_r, 2, 1)\pi$ is an accepting run of $\mathcal{B}$ on w by the definition of $\Delta_\mathcal{B}$. In both cases it follows that $w \in L(\mathcal{B})$. Conversely, if $w \in L(\mathcal{B})$ then there is an accepting run ρ of $\mathcal{B}$ on w. If $w = \epsilon$ then $q_s \in F_\mathcal{B}$. It follows that $q_0 \in F_1$ or $p_0 \in F_2$. Therefore, $w \in L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$. If $w \neq \epsilon$ then $\rho = (q_s, 1, 1)(q_r, 2, 1)(s, 1, 1)\hat{\rho}$ for $s \in \{q_0, p_0\}$ by the definition of $\Delta_\mathcal{B}$. Thus, $(s, 1, 1)\hat{\rho}$ is an accepting run of $\mathcal{A}_1$ or $\mathcal{A}_2$ on w. Hence, $w \in L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

For non-deterministically ω-grid recognizable languages the construction is precisely the same. In the correctness the special case for the empty word would be removed without substitution and π, ρ and $\hat{\rho}$ are replaced by infinite sequences (note that we only argued about prefixes of a run). □

Theorem 2. *Given two Non-deterministic Progressive Grid Word Automata $\mathcal{A}_1$ and $\mathcal{A}_2$ one can effectively construct a Non-deterministic Progressive Grid Word Automaton $\mathcal{C}$ such that $L(\mathcal{C}) = L(\mathcal{A}_1)L(\mathcal{A}_2)$. In particular, non-deterministically grid recognizable languages are closed under concatenation.*

Proof. The proof idea is analogous to the one to show the closure under union. Let L_1 and L_2 be non-deterministically grid recognizable and $\mathcal{A}_1 = (Q, \Delta_1, q_0, F_1)$ as well as $\mathcal{A}_2 = (P, \Delta_2, p_0, F_2)$ be the witnessing Non-deterministic Progressive Grid Word Automata, respectively. Let $\mathcal{C} := (Q_\mathcal{C}, \Delta_\mathcal{C}, q_s, F_\mathcal{C})$ where

- $Q_\mathcal{C} := Q \dot\cup P \dot\cup \{q_s, q_r\}$,
- $\Delta_\mathcal{C} := \Delta_1 \dot\cup \Delta_2 \dot\cup \{(q_s, x, q, d) \mid (q_0, x, q, d) \in \Delta_1\} \dot\cup \{(q_s, x, q_r, \uparrow) \mid x \in \{0, 1\}, q_0 \in F_1\}$
  $\dot\cup \{(q, x, q_r, \rightarrow) \mid (q, x, q_f, \rightarrow) \in \Delta_1, q_f \in F_1\}$
  $\dot\cup \{(q_r, 1, q_r, \downarrow), (q_r, 0, q_r, \downarrow), (q_r, \#, p_0, \uparrow)\}$

- and $F_{\mathcal{C}}$ is the smallest set that satisfies:

$$F_2 \subseteq F_{\mathcal{C}},\ p_0 \in F_2 \Rightarrow q_r \in F_{\mathcal{C}}, \text{ and } q_0 \in F_1 \wedge p_0 \in F_2 \Rightarrow q_s \in F_{\mathcal{C}}$$

$\mathcal{C}$ simulates $\mathcal{A}_1$ and every time it may move to the right and enter an accepting state it may also start to simulate $\mathcal{A}_2$ on the next column. Again we need an intermediate state q_r to return to the correct position. The constructed automaton $\mathcal{C}$ is depicted in figure 3.1b. Note that it is not possible to simply insert transitions from all accepting state to q_r. Since an accepting state may be entered by an up or down movement the resulting automaton would possibly accept a strict super set of L_1L_2. Let $w \in \mathcal{G}$. It suffices to show $w \in L_1L_2 \Leftrightarrow w \in L(\mathcal{C})$ to prove the correctness of the construction.

Case 1: $w = \epsilon$. Then $\epsilon \in L_1L_2 \Leftrightarrow q_0 \in F_1 \wedge p_0 \in F_2 \Leftrightarrow q_s \in F_{\mathcal{C}} \Leftrightarrow \epsilon \in L(\mathcal{C})$✓

Case 2: $w = uv, \epsilon \neq u \in L_1, v \in L_2$. Then there is an accepting run of $\mathcal{A}_1$ on u of the form $(q_0, 1, 1) \dots (q', h, |u|)(q_f, h', |u| + 1)$ and $(q', w(h, |u|), q_f, \rightarrow) \in \Delta_1$. If $v = \epsilon$ then $q_r \in F_{\mathcal{C}}$ and therefore $(q_s, 1, 1) \dots (q', h, |u|)(q_r, h, |u| + 1)$ is an accepting run of $\mathcal{C}$ on w. Otherwise, there is an accepting run $(p_0, 1, 1) \dots (p_f, \hat{h}, |v| + 1)$ of $\mathcal{A}_2$ on v. It follows that $(q_s, 1, 1) \dots (q', h, |u|)(q_r, h, |u| + 1) \dots (q_r, 0, |u| + 1)(p_0, 1, |u| + 1) \dots (p_f, \hat{h}, |u| + |v| + 1)$ is an accepting run of $\mathcal{C}$ on w. In conclusion $w \in L(\mathcal{C})$.✓

Case 3: $w = uv, u = \epsilon \in L_1, \epsilon \neq v \in L_2$. This case follows by a similar reasoning as above in the second sub case. Here the prefix of an accepting run of $\mathcal{C}$ on w is $(q_s, 1, 1)(q_r, 2, 1)$. ✓

Case 4: $\epsilon \neq w \notin \lambda(L_1L_2)$. Then for each composition $w = uv$ it holds that $u \notin L_1$ or $v \notin L_2$. If $u \notin L_1$ then there is no accepting run of $\mathcal{A}_1$ on u and therefore there is no partial run that starts with $(q_s, 1, 1)$ and ends with (q_r, h, ℓ). Then there is no accepting run of $\mathcal{C}$ on w. The other case for v is symmetric. Note that if $v = \epsilon$ (and the first case does not apply) then $q_r \notin F_{\mathcal{C}}$. Thus, $w \notin L(\mathcal{C})$.✓ □

Theorem 3. *Given a Deterministic Progressive Grid Word Automaton $\mathcal{A}$ one can effectively construct a Deterministic Progressive Grid Word Automaton $\mathcal{B}$ such that $L(\mathcal{B}) = \overline{L(\mathcal{A})}$. In particular, the set of deterministically grid recognizable languages is closed under complement.*

First of all, note that this result is not trivial because it is not sufficient to swap the accepting states like in the case of finite state automata since a Deterministic Progressive Grid Word Automaton might not halt on a given input $w \in \mathcal{G}$. Hence, the constructed automaton (with swapped accepting states) does not halt on w, too. But w should be accepted.

Example 7. Consider the Deterministic Progressive Grid Word Automaton $\mathcal{A} = (\{q_0\}, \delta, q_0, \{q_0\})$ where $\delta(q_0, x) := (q_0, \uparrow)$ for all $x \in \{\#, 0, 1\}$. It holds that $L(\mathcal{A}) = \{\epsilon\}$. The Deterministic Grid Word Automaton $\mathcal{B} = (\{q_0\}, \delta, q_0, \emptyset)$ recognizes $\emptyset \neq \overline{L(\mathcal{A})} = \mathcal{G} \setminus \{\epsilon\}$.

Moreover, note that the theorem does not mention ω-grid recognizable languages since a different situation emerges for them (cf. theorem 4). Before we prove the theorem we first show the following lemma which helps us to handle the situation described above.

Lemma 2. *Let $\mathcal{A}$ be a Deterministic Progressive Grid Word Automaton. Then one can effectively construct a Deterministic Progressive Grid Word Automaton $\mathcal{A}'$ that halts on every input with $L(\mathcal{A}) = L(\mathcal{A}')$.*

To prove this lemma we adapt the proof approach from Inoue and Takanami for three-way automata over unary alphabets (cf. [IT80]).

Proof. The proof consists of two parts. Firstly, we will prepare some conditions that can be checked by a Deterministic Progressive Grid Word Automaton and are equivalent to the situation that the automaton does not halt if it progresses further. Secondly, we will construct $\mathcal{A}'$ such that it checks those conditions while imitating the behavior of $\mathcal{A}$.

W.l.o.g. $\mathcal{A}$ is given by $\mathcal{A} = (Q, \delta, 1, F)$ with $Q = \{1, \dots, n\}$ for some $n \in \mathbb{N}$. Let $w \in \mathcal{A}$ and $\pi = (1,1,1)c_1c_2\dots$ a possibly infinite sequence of configurations such that every partial run of $\mathcal{A}$ on w is a prefix of π. Since $\mathcal{A}$ is deterministic π is unique. Furthermore, π is finite if and only if $\pi = (1,1,1)\dots(q,h,|w|+1)$ for some $h, \ell \in \mathbb{N}, q \in Q$. That is if and only if $\mathcal{A}$ halts on w. On the other hand, $\mathcal{A}$ does not halt on w if and only if π is infinite. This is the case if and only if

$$\exists k \in \mathbb{N} \text{ and } \ell \leq |w| \text{ such that } c_i = (q_i, h_i, \ell) \text{ with } q_i \in Q, h_i \in \mathbb{N} \text{ for all } i \geq k \qquad (\star)$$

Hence, the following conditions imply that $\mathcal{A}$ dos not halt on w:

1) A configuration of the form $(q, 0, \ell)$ with $q \in Q$ occurs more than $|Q|$ times in π for a fixed $1 \leq \ell \leq |w|$. In other words $\mathcal{A}$ reads more than $|Q|$ times the symbol # in the same column. Then for some $p \in Q$ the configuration $(p, 0, \ell)$ occurs twice in π. But since the behavior $\mathcal{A}$ is deterministic it occurs infinitely often.

2) A subsequence $(q, h, \ell)(q', h', \ell)$ with $q, q' \in Q$ and $w(h, \ell) = 1 \neq 0 = w(h', \ell)$ (or vice versa) occurs more than $|Q|$ times in π. Then it follows that π is infinite by the same reasoning as in 1).

3) $\pi = \pi_1\pi_2\pi_3$ with $\pi_2 = (q, h, \ell)\dots(p, h', \ell)$ where $h' > h + |Q| > h(w_\ell) + |Q|$ (w_ℓ is the ℓ-th letter of w). Then there is a state repetition and $(\star)$ holds since $\mathcal{A}$ is deterministic and $w(\hat{h}, \ell) = 0\ \forall \hat{h} \geq h$.

4) $\pi = \pi_1\pi_2\pi_3$ with $\pi_2 = (q, h, \ell)\dots(q, h, \ell)$ and $|\pi_2| \leq |Q| + 1$. It follows immediately that $(\star)$ holds since $\mathcal{A}$ is deterministic.

Next we will show that this conditions are already necessary. Assume $(\star)$ holds but 1) and 2) do not. Then there is a subsequence $\pi' = (q_0, h_0, \ell)\dots(q_{|Q|+1}, h_{|Q|+1}, \ell)$ of π such that either $w(h_i, \ell) = 0$ or $w(h_i, \ell) = 1$ for all $0 \leq i \leq |Q| + 1$. Furthermore, π' contains two configurations (q, h_k, ℓ) and (q, h_j, ℓ) for some $q \in Q$. If $d := |h_k - h_j| \neq 0$ then 3) holds because $\mathcal{A}$ is deterministic. If $d = 0$ then either 4) holds or $\pi' = (q, h_k, \ell)\dots(q, h_k, \ell)$ because $|\pi'| = |Q| + 2$. In the latter case (4) does not hold) either π' contains another configuration of the form $(q, \hat{h}, \ell)$ or there are configurations (p, h'', ℓ) and (p, h''', ℓ),$p \neq q$ in π'. In both cases we can apply the same arguments again with a shorter subsequence until this case does not occur again. Thus, the conditions above are necessary ones.

Note that these conditions are "local" and independent of w. In fact, we construct $\mathcal{A}'$ such that it checks the conditions above. If one of them is true $\mathcal{A}'$ halts and rejects. The detection is achieved with finitely many finite counters which will be encoded in the state space. We need $|Q| + 3$ counters in total. The first $n = |Q|$ counters $c_1 \dots c_n$ are for the last condition (the counter c_q observes the state $q \in Q = \{1, \dots, n\}$). The remaining

three counters $c_\#, c_{01}, c_0$ are for the conditions 1) up to 3), respectively. In addition to the counters, we save the last label of the input word in the state space to be able to identify condition 2).

Let $R := \{-(|Q|+1), \ldots, 0, \ldots, |Q|+1\} \cup \{\bot\}$ be the range of the counters. $\bot$ signals that the counter is inactive. For convenience we define $\bot + z := \bot$ for all $z \in \mathbb{Z}$. Furthermore, we identify each $k \in \mathbb{Z} \setminus R$ with $\bot$. Then $\mathcal{A}' := (Q', \delta', q_0', F')$ where $Q' := Q \times R^m \times \{\#, 0, 1\} \cup \{q'_{\text{sink}}\}$, $m := |Q| + 3$, $q_0' := (0, \bot, \ldots, \bot, 0, 0, 0, 0)$ and $F' := \{(q, c_0, \ldots, c_m, y) \mid q \in F\}$. q'_{sink} is a sink state used by $\mathcal{A}'$ to halt and reject.

Lastly, δ' needs to be defined. Let $q' = (q, c_1, \ldots, c_n, c_\#, c_{01}, c_0, y) \in Q'$,$x \in \{\#, 0, 1\}$ and $p \in Q, d \in \{\uparrow, \downarrow, \rightarrow\}$ such that $\delta(q, x) = (p, d)$. Then we distinguish the following cases:

Case 1: $d = \rightarrow$. If $\mathcal{A}'$ enters a new column of the input word then the automaton resets all counters and starts over. Hence, we define $\delta'(q', x) := ((p, \bot, \ldots, \bot, 0, 0, 0, 0), \rightarrow)$.

Case 2: $d \neq \rightarrow$ and $c_\# = |Q| + 1$ or $c_{01} = |Q| + 1$ or $c_0 = |Q| + 1$. Then 1), 2) or 2) holds, respectively. Therefore, $\mathcal{A}'$ moves to a sink state, halts and rejects. $\delta'(q', x) := (q'_{\text{sink}}, \rightarrow)$.

Case 3: Otherwise, if $c_q = 0$ then 4) holds. Therefore, $\mathcal{A}'$ halts and rejects by entering the sink state. $\delta'(q', x) := (q'_{\text{sink}}, \rightarrow)$.

Case 4: $d = \uparrow$. In this remaining case $\mathcal{A}'$ have to keep track of the distance to the last occurrence of each state within the required range and apart from that imitate $\mathcal{A}$. Thus, $\delta'(q', x) := ((p, c_1', \ldots, c_n', c_\#', c_{01}', c_0', x), \uparrow)$ where $c_i' = c_i + 1 \; \forall 1 \leq i \leq n, i \neq q$ and $c_q' = 1$. Furthermore, $c_\#' = c_\# + 1$ if $x = \#$, otherwise $c_\#' = c_\#$. Analogous $c_0' = c_0 + 1$ if $x = 0$, otherwise $c_0' = c_0$ and $c_{01}' = c_{01} + 1$ if $y = 1 \wedge x = 0$, otherwise $c_{01}' = c_{01}$.

Case 5: $d = \downarrow$. This case is symmetric to $d = \uparrow$. Instead of increasing the counters by one they are decreased by one.

Finally, $\delta'(q'_{\text{sink}}, x) := (q'_{\text{sink}}, \rightarrow)$ for all $x \in \{\#, 0, 1\}$.

$\mathcal{A}'$ halts on every input by construction. Furthermore, if $\mathcal{A}'$ accepts a Grid Word $w \in \mathcal{G}$, then the accepting run is induced only by transitions defined in the first and in the last two cases. Then it holds by construction that the projection on the first component of the states of this run induces an accepting run of $\mathcal{A}$ on w. Hence, $L(\mathcal{A}') \subseteq L(\mathcal{A})$. On the other hand, if $\mathcal{A}$ accepts $w \in \mathcal{A}$ then $\mathcal{A}$ halts on w. Therefore, $\mathcal{A}'$ halts and accepts since it imitates the behavior of $\mathcal{A}$. In conclusion, we have that $L(\mathcal{A}') = L(\mathcal{A})$. □

Proof of the theorem. Let L be deterministically grid recognizable. Then there is a Deterministic Progressive Grid Word Automata $\mathcal{A} = (Q, \delta, q_0, F)$ such that $L(\mathcal{A}) = L$. W.l.o.g. we can assume that $\mathcal{A}$ halts on every input due to lemma 2.

We define $\mathcal{B} := (Q, \delta, q_0, Q \setminus F)$. Then $\mathcal{B}$ is a well-defined Deterministic Progressive Grid Word Automaton and it holds that

$$
\begin{aligned}
& w \notin L(\mathcal{A}) \\
\Leftrightarrow\ & \exists \text{ a unique run } (q_0, 1, 1) \ldots (p, h, |w| + 1) \text{ of } \mathcal{A} \text{ on } w \text{ and } p \notin F \\
\Leftrightarrow\ & \exists \text{ a unique run } (q_0, 1, 1) \ldots (p, h, |w| + 1) \text{ of } \mathcal{B} \text{ on } w \text{ and } p \in Q \setminus F \\
\Leftrightarrow\ & w \in L(\mathcal{B})
\end{aligned}
$$

Not that the first equivalence is valid because $\mathcal{A}$ halts on every input. We conclude that $L(\mathcal{B}) = \overline{L(\mathcal{A})} = \overline{L}$. Hence, $\overline{L}$ is deterministically grid recognizable. □

As already stated earlier in contrast to deterministically grid recognizable languages we have that deterministically ω-grid recognizable languages are not closed under complement. Note that this corresponds to the situation that applies for finite alphabets (cf. [Tho90]).

Theorem 4. *Deterministically ω-grid recognizable languages are not closed under complement.*

Proof. Consider the language of all ω-Grid Words where the first letter appears infinitely often. Due to example 6 it is deterministically ω-grid recognizable. Assume $\overline{L} = \{xuv \mid x \in \mathbb{N}, u \in \mathbb{N}^*, v \in (\mathbb{N} \setminus \{x\})^\omega\}$ is deterministically ω-grid recognizable. In other words, the complement of L consists of all ω-Grid Words where from some point on the first letter does not occur any more. Then there is a Deterministic Progressive Grid Word Büchi Automaton $\mathcal{A}$ that recognizes $\overline{L}$. Furthermore, we fix a $v \in (\mathbb{N} \setminus \{1\})^\omega$. Clearly, $\mathcal{A}$ accepts $w_0 := 1v$. Hence, $\mathcal{A}$ enters infinitely often an accepting state and moves to the right infinitely often while processing w_0. Then there is a decomposition $w_0 = 1u_1v_1$ such that $\mathcal{A}$ enters an accepting state while reading the prefix $1u_1$. Moreover, $\mathcal{A}$ accepts $1u_11v$ and we know that $\mathcal{A}$ enters an accepting state while reading the prefix $1u_11$ since $\mathcal{A}$ is deterministic. If we apply the same argumentation to prefixes $1u_11u_21u_3 \dots 1u_i$ it follows that $\mathcal{A}$ enters an accepting state while processing $1u_11u_21u_3 \dots \in \mathbb{N}^\omega$ infinitely often. That is, $\mathcal{A}$ accepts $1u_11u_21u_3 \dots \notin \overline{L}$. This is a contradiction to the assumption $L(\mathcal{A}) = \lambda(\overline{L})$. Thus, $\overline{L}$ is not deterministically ω-grid recognizable and it follows that deterministically ω-grid recognizable languages are not closed under complement. □

Theorem 5. *Neither non-deterministically (ω-)grid recognizable nor deterministically (ω-) grid recognizable languages are closed under intersection.*

On the first look it seems quite surprising that both, non-deterministically and deterministically (ω-)grid recognizable languages, are not closed under intersection. But intuitively, theorem 5 can be interpreted as follows: A Progressive Grid Word (Büchi) Automaton (deterministic or not) can only remember one letter at a time it has seen before. This letter "is saved" in the run or in other words the current vertical position of the automaton on the (ω-)Grid Word corresponds to the letter that should be remembered. Moreover, while remembering some letter the automaton can only act limited. Actually, this limitation is used in formal proof.

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. We show that $L := L_1 \cap L_2$ is not non-deterministically grid recognizable. Then the result follows immediately. For the sake of contradiction assume that L is non-deterministically grid recognizable. Thus, there is a Non-deterministic Progressive Grid Word Automaton $\mathcal{A} = (Q, \Delta, q_0, F)$ with $L(\mathcal{A}) = \lambda(L)$. Let $L' := \{(0m)^n \mid n, m \in \mathbb{N}, n > 0\} \subseteq L$. Then $\mathcal{A}$ accepts all $w \in \lambda(L')$.

Furthermore, it holds that if π is an accepting run of $\mathcal{A}$ on some $w = (0m)^n \in L'$ then $\pi = \pi_1(q, 1, \ell)\pi_2$ for appropriate π_1, π_2, $1 \leq \ell \leq 2n$ and $q \in Q$. Otherwise, there is an

accepting run π on some $(0m)^n \in L'$ and $0 \leq p \leq n$ such that $\mathcal{A}$ accepts $(0m)^{n-p}(1m)(0m)^p$ since π is an accepting run of $\mathcal{A}$ on $(0m)^{n-p}(1m)(0m)^p$. But $(0m)^{n-p}(1m)(0m)^p \notin L$.

Since $|L'|$ is infinite and $|Q|$ is finite there are $u = (0m_u)^{n_u}, v = (0m_v)^{n_v} \in L'$ with $m_u \neq m_v$ as well as accepting runs $\pi = \pi_1(q,1,\ell)\pi_2, \rho = \rho_1(q,1,\ell)\rho_2$ of $\mathcal{A}$ on u and v, respectively, for some $q \in Q$ due to the pigeonhole principle. It follows that $\pi_1(q,1,\ell)\rho_2$ is an accepting run of $\mathcal{A}$ on $(0m_u)^{n_u}(0m_v)^{n_v} \notin L$. This contradicts the assumption that $\lambda(L) = L(\mathcal{A})$.

Regarding ω-grid recognizable languages the same argumentation applies for the deterministically ω-grid recognizable languages $L_i' := \{n_0 n_1 \ldots \in \mathcal{G}_\omega \mid \forall k \in \mathbb{N} : \ n_0 \ldots n_k \in L_i\}$ for $i \in \{1,2\}$. Furthermore, w' has the shape $(0m)^\omega$ and π', ρ', π_2' and ρ_2' are infinite sequences satisfying the acceptance condition for Progressive Grid Word Büchi Automata. Then the result follows because we can consider the same finite prefixes as for grid recognizable languages. □

With the theorems proved so far and De Morgan's laws we can conclude the following result.

Corollary 1. *Non-deterministically (ω-)grid recognizable languages are not closed under complement.*

Proof. For the sake of contradiction assume that non-deterministically (ω-)grid recognizable languages are closed under complement. Additionally, non-deterministically (ω-)grid recognizable languages are closed under union due to theorem 1. Let L_1, L_2 be non-deterministically (ω-)grid recognizable languages. Then it follows with De Morgan's laws that $\overline{\overline{L_1} \cup \overline{L_2}} = L_1 \cap L_2$ is non-deterministically (ω-)grid recognizable. Hence, non-deterministically (ω-)grid recognizable languages are closed under intersection. This contradicts theorem 5. □

Theorem 6. *Deterministically (ω)-grid recognizable languages are not closed under union.*

Proof. Using theorem 3 instead of theorem 1 the result for grid recognizable languages follows with the same reasoning as corollary 1. However, we cannot conclude this for ω-grid recognizable languages analogously since they are not closed under complement. Note that the following proof does also works for grid recognizable languages. Let L_1 be the language of all words in $\mathbb{N}^\omega$ where the first and the third letter are the same. That is, $L_1 = \{xyxv \mid x, y \in \mathbb{N}, \ v \in \mathbb{N}^\omega\}$. Furthermore, let $L_2 := \{z0v \mid z \in \mathbb{N}, \ v \in \mathbb{N}^\omega\}$. Both are clearly deterministically ω-grid recognizable.

Now, suppose there is a Deterministic Progressive Grid Word Büchi Automaton $\mathcal{A} = (Q, \delta, q_0, F)$ recognizing $L_1 \cup L_2$. Let $v \in \mathbb{N}^\omega$, $y \in \mathbb{N}$ and for all $x \in \mathbb{N}$,$x \neq y$ let $\pi_x = (q_0,1,1)\ldots$ be the accepting run of $\mathcal{A}$ on $x0yv \in L_1 \cup L_2$. Then for all x it holds that π_x has the shape $\pi_x = (q_0,1,1)\ldots(q_x,1,2)\pi_x'$ because otherwise $\mathcal{A}$ would accept $x1yv \notin L_1 \cup L_2$ (cf. proof of theorem 5). On the other hand $\mathcal{A}$ accepts $x1xv \in L_1$ for all $v \in \mathbb{N}^\omega$, $x \in \mathbb{N}$ and since $\mathcal{A}$ is deterministic the run of $\mathcal{A}$ on $x1xv$ has the prefix $(q_0,1,1)\ldots(q_x,1,2)$ for all $x \in \mathbb{N}$. Moreover, there are $a, b \in \mathbb{N}$ such that $q_a = q_b$ since Q is finite. Hence, there is an accepting run $(q_0,1,1)\ldots(q_a,1,2)\ldots$ of $\mathcal{A}$ on $a1bv \notin L_1 \cup L_2$. But this contradicts $L(\mathcal{A}) = \lambda(L_1 \cup L_2)$. Thus, deterministically ω-grid recognizable languages are not closed under union. □

It remains to cover the question if deterministically grid recognizable languages are closed under concatenation. Note that we used explicit non-determinism to prove that non-deterministically grid recognizable languages are closed under concatenation, indeed (cf. proof of theorem 2).

Theorem 7. *Deterministically grid recognizable languages are not closed under concatenation.*

Proof. We prove the theorem by contradiction. Therefore, we define the following two languages:

$$L_1 := \{n_0 \dots n_p \in \mathbb{N}^* \mid p \in \mathbb{N} \wedge n_0 = n_p\}$$

$$L_2 := \{0^n \mid n \in \mathbb{N}\}$$

We have seen in example 4 that L_1 is deterministically grid recognizable. Moreover, L_2 is trivially deterministically grid recognizable. Assume $L := L_1 \cdot L_2$ is deterministically grid recognizable. Then there is a Deterministic Progressive Grid Word Automaton $\mathcal{A} = (Q, \delta, q_0, F)$ with $L(\mathcal{A}) = \lambda(L)$.

It follows that $\mathcal{A}$ accepts $w = n_0 n_1^s n_0 0^{2k+1} \in L$ where $s = k = 2|Q| + 1$ and $n_0 > 3|Q|$. Then all configurations $(q_i, 1, s + l_i)$ where $3 \leq l_i \leq 2k + 3$ occur in the accepting run π of $\mathcal{A}$ on w. Otherwise, $\mathcal{A}$ accepts $n_0 n_1^s n_0 0^{2k-p} 10^p$ for some $0 \leq p \leq 2k$ since π would be the witnessing run.

On the other hand $\mathcal{A}$ accepts $w' := n_0 n_1^s n_0 0^k n_0 0^k \in L$. Let ρ be the accepting run of $\mathcal{A}$ on w'. Since $\mathcal{A}$ is deterministic and w, w' have the same prefix $n_0 n_1^s n_0 0^k$ the accepting runs of both words have the same prefix $(q_0, 1, 1) \dots (p, 1, s + 2 + k)$. We distinguish two cases:

Case 1: $(q, n_0, s + 2 + k + 1)$ does not occur in ρ for any $q \in Q$. Then ρ is an accepting run of $\mathcal{A}$ on $n_0 n_1^s n_0 0^k (n_0 - 1) 0^k \notin L$↯.

Case 2: $(q, n_0, s+2+k+1)$ does occur in ρ for some $q \in Q$. Then there is a state repetition in $\rho = (q_0, 1, 1) \dots (p, 1, s+2+k) \dots (q', h, \ell) \dots (q', h', \ell) \dots (q, n_0, s+2+k+1)$ with $n_0 \geq h' > h, q' \in Q$ because $n_0 > 3|Q|$. Note that the third component of the relevant configurations is the same, that is the state repetition occurs in one column of $\lambda(w')$. Then the partial run $(q_0, 1, 1) \dots (p, 1, s + 2 + k) \dots (q', h, \ell) \dots (q', h', \ell)$ can be extended to an accepting run of $\mathcal{A}$ on some $\hat{w} = n_0 n_1^s n_0 0^k n_2 0^k \notin L$ with $n_2 > n_0$ by repeating the sequence of states. This follows by a symmetric argumentation that respects if some $(\hat{q}, 0, \hat{\ell})$ occurs in ρ later on $(\hat{\ell} \geq \ell)$↯.

In both cases $\mathcal{A}$ accepts some $w \in \overline{L}$. We conclude $L(\mathcal{A}) \neq \lambda(L_1 \cdot L_2)$. Therefore, $L_1 \cdot L_2$ is not deterministically grid recognizable. □

Corollary 2. *The set of deterministically (ω-)grid recognizable languages is a strict subset of the set of non-deterministically (ω-)grid recognizable languages.*

Proof. It is clear that every deterministically (ω-)grid recognizable language is non-deterministically (ω-)grid recognizable because every Deterministic Progressive Grid Word (Büchi) Automaton can be considered as a Non-deterministic Progressive Grid Word (Büchi) Automaton. The strict containment is a direct result from theorem 1 and theorem 6. Due to the latter one there are deterministically (ω-)grid recognizable language L_1, L_2 such that

$L_1 \cup L_2$ is not deterministically (ω-)grid recognizable. Assume that every non-deterministically (ω-)grid recognizable language is deterministically (ω-)grid recognizable. Since non-deterministically (ω-)grid recognizable languages are closed under union it follows that $L_1 \cup L_2$ is deterministically (ω-)grid recognizable. This is a contradiction. In conclusion, we found a language $L_1 \cup L_2$ that is non-deterministically (ω-)grid recognizable but not deterministically grid recognizable. □

Remark 3. To show that a non-deterministically grid recognizable language L is not deterministically grid recognizable it suffices to show that the complement $\overline{L}$ is not non-deterministically grid recognizable. This follows directly from the result that non-deterministically grid recognizable languages are not closed under complement in contrast to deterministically grid recognizable languages (cf. corollary 1 and theorem 3, respectively). In other words, each non-deterministically grid recognizable language whose complement is not non-deterministically grid recognizable is not deterministically grid recognizable.

Example 8. An explicit example of a language that is non-deterministically grid recognizable but not deterministically grid recognizable is $L := \{n_0 n_1 n_2 u \mid n_0, n_1, n_2 \in \mathbb{N}, u \in \mathbb{N}^*, n_1 = 0 \vee n_0 = n_1\}$. The complement is given by $\overline{L} = \{v = v_0 \dots v_n \in \mathbb{N}^* \mid |v| < 3 \vee (n_1 \neq 0 \wedge n_0 \neq n_1)\}$. It is not non-deterministically grid recognizable. This can be proven with a combination of the arguments used in the proofs of theorems 5 and 7. Therefore, L is not deterministically grid recognizable.

4 Decision Problems

In this chapter we study some common decision problems for (Non-)deterministic Progressive Grid Word Automata as well as General Grid Word Automata. Additionally, we will extend our results to (Non-)deterministic Progressive Grid Word Büchi Automata and General Grid Word Büchi Automata wherever it is suitable. More precisely, we will consider the following problems.

- The word problem for General Grid Word Automata

  `Input:` A General Grid Word Automaton $\mathcal{A}$ and a Grid Word $w \in \mathcal{G}$.
  `Output:` Is $w \in L(\mathcal{A})$?

- The emptiness problem for Progressive (General) Grid Word (Büchi) Automata

  `Input:` A Progressive (General) Grid Word (Büchi) Automaton $\mathcal{A}$.
  `Output:` Is $L(\mathcal{A}) = \emptyset$?

- The universality problem for (Non-)deterministic Progressive Grid Word (Büchi) Automata

  `Input:` A (Non-)deterministic Progressive Grid Word (Büchi) Automaton $\mathcal{A}$.
  `Output:` Is $L(\mathcal{A}) = \mathcal{G}$?

- The equivalence problem for Non-deterministic Progressive Grid Word (Büchi) Automata

  `Input:` Non-deterministic Progressive Grid Word (Büchi) Automata $\mathcal{A}$ and $\mathcal{B}$.
  `Output:` Is $L(\mathcal{A}) = L(\mathcal{B})$?

Note that the word problem is not defined for Grid Word Büchi Automata because an ω-Grid Word may contain an infinite amount of information. Hence, an ω-Grid Word is not a suitable input for an algorithm.

4.1 The Word Problem

Theorem 8. *The word problem for General Grid Word Automata is decidable.*

One may argue that this result is trivial since the given General Grid Word Automaton can be simulated on the given Grid Word while keeping track of all configurations occurred

in every possible run. If there is a configuration repetition the automaton will not halt and therefore reject. Unfortunately, there are infinitely many possible configurations that may occur since a Grid Word is not bounded with respect to the first component. The third observation in example 5 demonstrates that the height of a Grid Word is not a proper boundary. Moreover, the boundary is at least $\text{lcm}(p,t)+1$ where p,t are prim numbers and $p+t$ is the number of states of the concerned automaton.

Instead of giving a combinatorial proof that tries to provide an upper limit for configurations we will reduce the word problem to the emptiness intersection problem for two-way finite state automata. As a side effect this approach implies a more elegant decision procedure. We start with a brief definition of two-way finite state automata and the corresponding emptiness intersection problem.

Definition 9 (two-way finite state automaton)**.** A two-way finite state automaton is a tuple $\mathcal{A} := (Q, \Sigma, q_0, \Delta, F)$ where Q is the finite set of states, $q_0 \in Q$ is the initial state and $F \subseteq Q$ is the set of acceptance states. Furthermore, Σ is a finite alphabet and $\Delta : Q \times \Sigma \to Q \times \{\leftarrow, \rightarrow, \circ\}$ is the transition relation. Intuitively, in each step of a run $\mathcal{A}$ can move left, right or stay in addition to changing its internal state. $\mathcal{A}$ accepts $v \in \Sigma^*$ if it "leaves" v on the right side while being in an acceptance state.

Note that in most definitions a two-way finite state automaton has markers on both sides of the input word to detect the end or beginning of the word. However, we do not need them here although it would not change our reduction.

Rabin and Scott have shown that it is decidable whether $L(\mathcal{B}) \cap L(\mathcal{C}) \neq \emptyset$ where $\mathcal{B}$ and $\mathcal{C}$ are two-way finite state automata (cf. [RS59]). Given a General Grid Word Automaton $\mathcal{A} = (Q, \Delta, \overline{q}, F)$ and a Grid Word $w \in \mathcal{G}$ we construct two-way finite state automata $\mathcal{B}$ and $\mathcal{C}$ such that

$$L(\mathcal{B}) \cap L(\mathcal{C}) \neq \emptyset \Leftrightarrow w \in L(\mathcal{A})$$

Then it follows immediately that the word problem for General Grid Word Automata is decidable. Furthermore, a decision procedure for the emptiness intersection problem together with the following construction yields a decision procedure for the word problem.
Construction: Let $w \in \mathcal{G}$, $\ell := |w|$ and $\mathcal{A} = (Q, \Delta, \overline{q}, F)$ be a General Grid Word Automaton. The idea of the construction is to "turn w by ninety degree". Then each row of w can be seen as a letter over the alphabet $\Sigma := \{\#, 0, 1\}^{\ell+2}$. Moreover, the "rotated" Grid Word w can be translated to a word over Σ. Therefore, let $u_i := (w(i,0), \ldots, w(i,(\ell+1)))^T$. Then the word $u := u_0 \ldots u_{h(w)} \in \Sigma^*$ contains all essential information of w. Remember that $h(w)$ is the height of w. Eventually, $\mathcal{B}$ is the two-way finite state automaton that recognizes

$$L(\mathcal{B}) = \{u z_{\ell+2}^k \mid z_{\ell+2} = (\#, 0, \ldots, 0, \#)^T \in \Sigma, k \in \mathbb{N}\} \subseteq \Sigma^*$$

It is obvious that $\mathcal{B}$ can be constructed. Intuitively, $\mathcal{B}$ takes care of the word structure of w. It remains to construct $\mathcal{C}$ which takes care of the behavior of $\mathcal{A}$ (note that $\mathcal{B}$ is independent of $\mathcal{A}$ while $\mathcal{C}$ will be independent of w). $\mathcal{C}$ will be constructed such that if $\mathcal{A}$ operates on the i-th column of w then $\mathcal{C}$ ignores all components of the input word but the i-th one. Moreover, if $\mathcal{A}$ moves left or right then $\mathcal{C}$ changes the internal state such that the $(i-1)$-th or the $(i+1)$-th component of the input word is the relevant one. Up or down movements of $\mathcal{A}$ are translated to right or left movements of $\mathcal{C}$, respectively. Then $\mathcal{C}$ imitates the behavior of $\mathcal{A}$. Note that $\mathcal{A}$ may operate on arbitrary words in Σ^*. The Grid

Word structure is induced by the intersection with $\mathcal{B}$. Formally, let $\mathcal{C} := (Q', \Sigma, \overline{q}_1, \Delta', \{q_f\})$ where $Q' := \{q_i \mid 0 \leq i \leq \ell + 1, q \in Q\} \cup \{q_f\}$. In connection to the explanation above $\mathcal{C}$ is in state q_i if and only if $\mathcal{A}$ is in state q in the i-th column of w. Finally, Δ' is the union of the following sets:

- As mentioned above up or down movements of $\mathcal{A}$ are translated to right or left movements of $\mathcal{C}$, respectively. Only the component of the letters that corresponds to the column of w w.r.t. $\mathcal{A}$ is relevant for these transitions.

$$\bigcup_{0 \leq i \leq \ell+1} \{(q_i, x, q'_i, d) \mid \exists (q, y, q', d') \in \Delta : d' \in \{\uparrow, \downarrow\}, d = \begin{cases} \leftarrow, & d' = \downarrow \\ \rightarrow, & d' = \uparrow \end{cases} \text{ and } x_i = y\}$$

- If $\mathcal{A}$ changes the column $\mathcal{C}$ adapts the relevant component.

$$\bigcup_{0 \leq i \leq \ell} \{(q_i, x, q'_{i+1}, \circ) \mid \exists (q, y, q', \rightarrow) \in \Delta : x_i = y\} \cup \bigcup_{1 \leq i \leq \ell+1} \{(q_i, x, q'_{i-1}, \circ) \mid \exists (q, y, q', \leftarrow) \in \Delta : x_i = y\}$$

- And finally, if $\mathcal{A}$ enters the $(\ell + 1)$-th column in an accepting state it accepts w, so $\mathcal{C}$ should accept.

$$\{(q_{\ell+1}, x, q_f, \rightarrow) \mid q \in F, x \in \Sigma\} \cup \{(q_f, x, q_f, \rightarrow) \mid x \in \Sigma\}$$

Correctness: Let $c_0 \dots c_m$ be a partial run of $\mathcal{A}$ on w with $c_i = (q(i), h(i), \ell(i))$ for all $1 \leq i \leq m$. Let $k := max\{h(i) \mid 0 \leq i \leq m\}$ and $v := uz^k_{\ell+2} \in L(\mathcal{B})$. Then $(q(1)_{\ell(1)}, h(1)) \dots (q(m)_{\ell(m)}, h(m))$ is a partial run of $\mathcal{C}$ on v by construction of $\mathcal{C}$. On the contrary, if $(q(1)_{\ell(1)}, h(1)) \dots (q(m)_{\ell(m)}, h(m))$ is a partial run of $\mathcal{C}$ on some $v' = uz^{k'}_{\ell+2} \in L(\mathcal{B})$ then there exists a partial run of $\mathcal{A}$ on w with the same shape. It follows that $w \in L(\mathcal{A})$ if and only if $\mathcal{C}$ accepts some word in $L(\mathcal{B})$. Note that $\mathcal{C}$ can non-deterministically accept a word if and only if it is in some state $q_{|w|+1}$ and $q \in F$ by moving to the state q_f. That is, if $\mathcal{A}$ accepts w. □

4.2 The Emptiness Problem

This section consists of two parts. Firstly, we will study the emptiness problem for Progressive Grid Word Automata and prove that it is decidable. In the second part we show the undecidability of the emptiness problem for General Grid Word Automata.

4.2.1 The Emptiness Problem for Progressive Grid Word Automata

First of all, note that this problem cannot be solved with a simple depth-first search (or something similar) since some transitions may have to be taken several times to accept some Grid Word. We have already observed this behavior in example 5. However, the problem is decidable.

Theorem 9. *The emptiness problem for Progressive Grid Word Automata is decidable.*

We show this theorem by providing a Turing-Reduction from the emptiness problem to an already decidable problem. We choose the decision problem of the MSO-theory of $(\mathbb{N}, S)$ which is defined as follows:

Definition 10. (The decision problem of the MSO-theory of $(\mathbb{N}, S)$).

`Input:` A Monadic-Second-Order (MSO) sentence φ over the signature $\{S\}$.
`Output:` `Yes`, if $(\mathbb{N}, S) \models \varphi$, otherwise `No`. Here S is the successor relation over $\mathbb{N}$.

The decidability of this problem was shown by Büchi in 1962 (cf. [Büc62]). Thus, the decidability of the emptiness problem follows as soon as we can provide a Turing-Reduction.

The basic idea of the reduction is the encoding of a run of a given Progressive Grid Word Automaton $\mathcal{A}$ in a formula φ. However, we cannot encode a run directly because we neither know the maximal length and height of a minimal word accepted by $\mathcal{A}$ nor it is possible to quantify a relation $R \subseteq \mathbb{N} \times \mathbb{N}$ in a MSO-formula over $\{S\}$. But either of them is a requirement to encode a sequence of configurations directly and hence, a run. That is why some preparation is necessary.

In the following an alternative notion of a run of $\mathcal{A}$ is worked out which does not change the semantics but is easier to encode in a formula. Especially, we want to get rid of the third component of a configuration which represents the horizontal position on a Grid Word. At the same time it should still be possible to identify the positions in a run where the automaton moves to the right. Note that this is important to verify the acceptance conditions because the last action of $\mathcal{A}$ has to be a movement to the right. Since the alternative notion can be understood as a projection of a run to the first two components of its configurations, we will call the new notion "projected run". We start with an alternative view of partial runs within a single column of a Grid Word.

Definition 11. Let $\mathcal{A} = (Q, \Delta, q_0, F)$ be a Progressive Grid Word Automaton, $c \in Q \times \mathbb{N}$, $k \in \mathbb{N}$ and $w = \lambda(k) \in \mathcal{G}$. Then T_c^k is the smallest set satisfying the following conditions:

- $c \in T_c^k$,
- if $(p, h) \in T_c^k$ and $(p, w(h, 1), q, \uparrow) \in \Delta$ then $(q, h + 1) \in T_c^k$, and
- if $(p, h) \in T_c^k$ and $(p, w(h, 1), q, \downarrow) \in \Delta$ then $(q, h - 1) \in T_c^k$.

Furthermore, for all $p, q \in Q$ the relation $R_{p,q}$ is defined as

$$R_{p,q} := \{(x, y, k) \in \mathbb{N}^3 \mid (q, y) \in T_{(p,x)}^k\}$$

T_c^k can be seen as the set of all projections of possible configurations that can occur in a partial run of $\mathcal{A}$ on a Grid Word of length one starting in c to the first and second component. This is formalized in the next lemma.

Lemma 3. *Let $\mathcal{A} = (Q, \Delta, q_0, F)$ be a Progressive Grid Word Automaton, $c = (p, h), d = (q, h') \in Q \times \mathbb{N}$ and $w \in \mathcal{G}$. Then for all $1 \leq \ell \leq |w|$ it holds that there is a partial run $\pi = (p, h, \ell) \dots (q, h', \ell)$ of $\mathcal{A}$ on w if and only if $d \in T_c^k$ where $k = \lambda^{-1}(w_\ell)$.*

Proof. Let $\ell \in \{1, \dots, |w|\}$.
"$\Rightarrow$": Assume there is a partial run $\pi = (p_1, h_1, \ell)(p_2, h_2, \ell) \dots (p_m, h_m, \ell)$ of $\mathcal{A}$ on w with $(p_1, h_1) = c$ and $(p_m, h_m) = d$. Note that the third component stays the same since $\mathcal{A}$ is a Progressive Grid Word Automaton. Let $1 \leq i < m$ and $x = w(h_i, \ell)$. Assume $(p_i, h_i) \in T_c^k$. Furthermore, it holds that $(h_{i+1} - h_i) \in \{1, -1\}$ by the definition of a run. W.l.o.g. $(h_{i+1} - h_i) = 1$. Then $(p_i, x, p_{i+1}, \uparrow) \in \Delta$ (otherwise, if $(h_{i+1} - h_i) = -1$ then replace $\uparrow$ with $\downarrow$). Since $x = w_\ell(h_i, 1)$ we have that $(p_{i+1}, h_i + 1) = (p_{i+1}, h_{i+1}) \in T_c^k$. Hence, it follows that $d = (p_m, h_m) \in T_c^k$ by an inductive argument because $c = (p_1, h_1) \in T_c^k$.
"$\Leftarrow$": On the contrary, assume $d \in T_c^k$. Then there is a sequence

$$c = (p_1, h_1)(p_2, h_2) \dots (p_m, h_m) = d$$

such that each (p_i, h_i) forces the membership of (p_{i+1}, h_{i+1}) to T_c^k. The non-existence of such a sequence would yield a contradiction to the minimalism of T_c^k. It follows that

$$(p_1, h_1, \ell)(p_2, h_2, \ell) \dots (p_m, h_m, \ell)$$

is a partial run of $\mathcal{A}$ on w since $(p_i, w_\ell(h_i, 1), p_{i+1}, d_i) \in \Delta$ for appropriate $d_i \in \{\uparrow, \downarrow\}$ and $w_\ell(h_i, 1) = w(h_i, \ell)$ for all $1 \leq i < m$. □

Now, we extend the previous result to Grid Words of arbitrary length which leads us to the "projected runs". After the definition we will show that these are equivalent to the "classical" runs of an automaton with respect to the semantics. In particular, this will show that it is sufficient to encode such a "projected run" in the reduction.

Definition 12 (projected run)**.** Let $\mathcal{A} = (Q, \Delta, q_0, F)$ be a Progressive Grid Word Automaton, and $w \in \mathcal{G}$ with $n := |w|$. Then a sequence $d_1 \dots d_{n+1}$ with $d_i = (q_i, h_i) \in Q \times \mathbb{N}$ is a *projected run* of $\mathcal{A}$ on w if for all $1 \leq \ell \leq n$ there is a $q'_\ell \in Q$ such that

1) $(q'_\ell, w(h_{\ell+1}, \ell), q_{\ell+1}, \rightarrow) \in \Delta$

2) $(h_\ell, h_{\ell+1}, \lambda^{-1}(w_\ell)) \in R_{q_\ell, q'_\ell}$.

Lemma 4. *Let $\mathcal{A} = (Q, \Delta, q_0, F)$ be a Progressive Grid Word Automaton, and w be a Grid Word of length n. Then there is a run $c_0 \dots c_m$ of $\mathcal{A}$ on w with $c_m = (q, h, n+1)$ if and only if there is a projected run $d_1 \dots d_{n+1}$ of $\mathcal{A}$ on w with $d_1 = (q_0, 1)$ and $d_{n+1} = (q, h)$.*

Proof. We proof this result by induction over $n := |w| \in \mathbb{N}$.
Induction Base: $|w| = 0 \Rightarrow w = \epsilon \Rightarrow (q_0, 1, 1)$ is a run and $d_1 = (q_0, 1)$ is a projected run of $\mathcal{A}$ on w.

Induction Step: $w = vw_n$, $|v| = n$ and $|w_n| = 1$. Assume there is a run π of $\mathcal{A}$ on w. Then π can be written as

$$\pi = (q_0, 1, 1) \dots (q', h, n)(q, h, n+1) \dots (p', g, n+1)(p, g, n+2),\ q, q', p, p' \in Q, h, g \in \mathbb{N}$$

Furthermore, $(p', w(g, n+1), p, \rightarrow) \in \Delta$ and by the induction hypothesis there is a projected run $d_1 \dots d_{n+1}$ with $d_{n+1} = (q, h)$ of $\mathcal{A}$ on v. Let $d_{n+2} := (p, g)$. Since there is partial run $(q, h, n+1) \dots (p', g, n+1)$ we have that $(p', g) \in T^k_{(q,h)}$ where $k = \lambda^{-1}(w_n)$ due to lemma 3. It follows that $(h, g, \lambda^{-1}(w_n)) \in R_{q,p'}$. Thus, $d_1 \dots d_{n+1} d_{n+2}$ is a projected run of $\mathcal{A}$ on w.

On the contrary, assume there is a projected run $d_1 \dots d_{n+2}$ of $\mathcal{A}$ on w. Then $d_1 \dots d_{n+1}$ is a projected run of $\mathcal{A}$ on v. Hence, there is a run $\rho = (q_0, 1, 1) \dots (q_{n+1}, h_{n+1}, n+1)$ of $\mathcal{A}$ on v by the induction hypothesis. Apart from that, there exists a $d' = (q', h_{n+2})$ such that $(h_{n+1}, h_{n+2}, \lambda^{-1}(w_n)) \in R_{q_{n+1},q'}$ and $(q', w(h_{n+2}, n+1), q_{n+2}) \in \Delta$. With lemma 3 it follows that there is a partial run $\rho' = (q_{n+1}, h_{n+1}, n+1) \dots (q', h_{n+2}, n+1)$. In conclusion, $\rho\rho'(q_{n+2}, h_{n+2}, n+2)$ is a run of $\mathcal{A}$ on w. □

To encode a projected run in a formula it is essential that the relations $R_{p,q}$ can be encoded. And in fact, they are definable in $(\mathbb{N}, S)$ with MSO-logic. Before we show this we recall some other useful relations which are definable in $(\mathbb{N}, S)$. For better readability we denote first order variables with lower case letters and second order variables with capital letters from now on.

Remark 4. The relations $<$, 0 and 1 with their usual interpretation over $\mathbb{N}$ are definable in $(\mathbb{N}, S)$. The corresponding formulae are

- $\varphi_<(x, y) := x \neq y \wedge \forall X(X(x) \wedge \forall x' \forall y'((X(x') \wedge S(x', y')) \to X(y'))) \to X(y)$,
- $\varphi_0(x) := \forall y(x = y \vee x < y)$,
- and $\varphi_1(x) := \exists x_0(\varphi_0(x_0) \wedge S(x_0, x))$, respectively.

Lemma 5. *Let $\mathcal{A} = (Q, \Delta, q_0, F)$ be a Progressive Grid Word Automaton and $p, q \in Q$. Then $R_{p,q}$ is definable in $(\mathbb{N}, S)$.*

Proof. We have to construct a formula $\psi_{p,q}(x, y, z)$ over $\{S\}$ such that for all $h, g, k \in \mathbb{N}$:

$$(\mathbb{N}, S) \models \psi_{p,q}[h, g, k] \Leftrightarrow (h, g, k) \in R_{p,q} \overset{\text{Def.}}{\Leftrightarrow} (q, g) \in T^k_{(p,h)}$$

W.l.o.g. $Q = \{1, \dots, n\}$ for some $n \in \mathbb{N}$. Firstly, we define an auxiliary formula $\vartheta_{p,q}$:

$$\vartheta_{p,q}(x, y, z, X_1, \dots X_n) := X_p(x) \wedge \forall v \forall w(\tag{4.1}$$

$$\bigwedge_{(p',0,q',\uparrow) \in \Delta} ((S(v, w) \wedge X_{p'}(v) \wedge v > z) \to X_{q'}(w)) \wedge \tag{4.2}$$

$$\bigwedge_{(p',1,q',\uparrow) \in \Delta} ((S(v, w) \wedge X_{p'}(v) \wedge v \leq z \wedge \neg v = 0) \to X_{q'}(w)) \wedge \tag{4.3}$$

$$\bigwedge_{(p',\#,q',\uparrow) \in \Delta} ((S(v, w) \wedge X_{p'}(v) \wedge v = 0) \to X_{q'}(w)) \wedge \tag{4.4}$$

$$\bigwedge_{(p',0,q',\downarrow) \in \Delta} ((S(w, v) \wedge X_{p'}(v) \wedge v > z) \to X_{q'}(w)) \wedge \tag{4.5}$$

$$\bigwedge_{(p',1,q',\downarrow) \in \Delta} ((S(w, v) \wedge X_{p'}(v) \wedge v \leq z \wedge \neg v = 0) \to X_{q'}(w))) \tag{4.6}$$

Then the final formula is $\psi_{p,q}(x, y, z) := \forall X_1 \dots \forall X_n \vartheta_{p,q}(x, y, z, X_1, \dots, X_n) \to X_q(y)$. The overall idea of this formula is that subsets of the X_i form a "partition" of the set $T^z_{(p,h)}$, e.g. if $(q, g) \in T^k_{(p,h)}$ then $g \in X_q$ and vice versa.
Correctness: Let $h, g, k \in \mathbb{N}$. Further on, let $T_1, \dots, T_n \subseteq \mathbb{N}$. First of all, we show that if $(\mathbb{N}, S) \models \vartheta_{p,q}[h, g, k, T_1, \dots, T_n]$ then $T^k_{(p,h)} \subseteq T_{\text{part}} := \{(s, e) \mid 1 \leq s \leq n, e \in T_s\}$. Now,

assume $(\mathbb{N}, S) \models \vartheta_{p,q}[h, g, k, T_1, \ldots, T_n]$. It is clear that $h \in T_p$ and thus $(p, h) \in T_{\text{part}}$ (cf. line (4.1)). Therefore, it suffices to show that T_{part} satisfies the conditions from the definition of $T^k_{p,h}$ (Def. 11). Suppose $(s, e) \in T_{\text{part}}$ and $(s, w(e, 1), s', \uparrow) \in \Delta$ where $w = \lambda(k)$. Then $e \in T_s$. Moreover, assume $w(e, 1) = 0$. Then $e > k$. It follows that $(e + 1) \in T_{s'}$ if we identify the variable w with $(e+1)$ and v with e in line (4.2). Hence, $(s', e+1) \in T_{\text{part}}$. The other cases are analogous and follow by lines (4.3) till (4.6). Note that the formula may not handle the case that $\mathcal{A}$ moves downwards and the current label is # since this is forbidden by the definition of Progressive Grid Word Automata. In conclusion we can deduce that

$$\begin{aligned}(q, g) \in T^k_{(p,h)} &\Rightarrow \text{for all } T_1, \ldots T_n \text{ with } (\mathbb{N}, S) \models \vartheta_{p,q}[h, g, k, T_1, \ldots, T_n] : (q, g) \in T_{\text{part}} \\ &\Rightarrow \text{for all } T_1, \ldots T_n \text{ with } (\mathbb{N}, S) \models \vartheta_{p,q}[h, g, k, T_1, \ldots, T_n] : g \in T_q \\ &\Rightarrow (\mathbb{N}, S) \models \psi_{p,q}[h, g, k]\end{aligned}$$

Further on, if $(\mathbb{N}, S) \models \psi_{p,q}[h, g, k]$ then there are $T'_1, \ldots, T'_n \subseteq \mathbb{N}$ such that $(\mathbb{N}, S) \models \vartheta_{p,q}[h, g, k, T'_1, \ldots, T'_n]$ and $T'_{\text{part}} = T^k_{(p,h)}$ since lines (4.1) till (4.6) cover exactly the conditions of the definition of $T^k_{(p,h)}$ (T'_{part} is defined analogous to T_{part}). Note that $\vartheta_{p,q}$ is trivially satisfiable if all second order variables are identified with $\mathbb{N}$. Finally, it follows that

$$(\mathbb{N}, S) \models \psi_{p,q}[h, g, k] \Rightarrow (\mathbb{N}, S) \models \vartheta_{p,q}[h, g, k, T'_1, \ldots, T'_n] \text{ and } g \in T'_q \Rightarrow (q, g) \in T'_{\text{part}} = T^k_{(p,h)}$$

This concludes the proof. □

Note that the construction in the proof of lemma 5 depends on the structure of Grid Words. It is used to verify that the automaton can move according to a transition or in other words to identify the label of the Grid Word $\lambda(k)$ at a specific position. Eventually, we can now prove the emptiness problem for Progressive Grid Word Automata.

Proof of the Decidability of the Emptiness Problem for Progressive Grid Word Automata

As stated at the beginning of this section the emptiness problem is reduced to the decision problem of the MSO-theory of $(\mathbb{N}, S)$. For this purpose it suffices to map a given Progressive Grid Word Automaton $\mathcal{A}$ to a MSO-sentence φ such that

$$(\mathbb{N}, S) \models \varphi \Leftrightarrow L(\mathcal{A}) \neq \emptyset$$

W.l.o.g. we can assume $\mathcal{A} = (\underline{n}, \Delta, 1, F)$ for some $n \in \mathbb{N}$ where $\underline{n} := \{1, \ldots, n\}$. As already discussed the goal is to encode a projected run of $\mathcal{A}$ on some Grid Word w in φ. Note that there is at least the trivial projected run on the empty Grid Word. Let $p, q \in Q$. Consider the following formula over $\{S\}$:

$$\theta_{p,q}(x, y, z) := \qquad y = 0 \rightarrow \bigvee_{(q', \#, q, \rightarrow) \in \Delta} R_{p,q'}(x, y, z) \tag{4.7}$$

$$\wedge(\neg y = 0 \wedge y \leq z) \rightarrow \bigvee_{(q', 1, q, \rightarrow) \in \Delta} R_{p,q'}(x, y, z) \tag{4.8}$$

$$\wedge(y > z) \rightarrow \bigvee_{(q', 0, q, \rightarrow) \in \Delta} R_{p,q'}(x, y, z) \tag{4.9}$$

First of all θ is well-defined since Δ is finite and $R_{p,q'}$ is definable in $(\mathbb{N}, S)$ due to lemma 5 (for convenience we use the same notations for the syntactic symbol and the relation). Furthermore, for each assignment of the free variables of θ exactly one of the premises is true. Let $d = (p, h), d' = (q, g) \in Q \times \mathbb{N}$.
Claim: $(\mathbb{N}, S) \models \theta_{p,q}[h, g, k]$ if and only if dd' is a projected run of $\mathcal{A}$ on $\lambda(k)$.

Proof. If dd' is a projected run of $\mathcal{A}$ on $w := \lambda(k)$ then there is a $q' \in Q$ such that $(q', x, q, \rightarrow) \in \Delta$ where $x := w(g, 1)$. Moreover, $(h, g, k) \in R_{p,q'}$. If $x = 1$ then $0 < g \leq k$ and therefore it holds that $(\mathbb{N}, S) \models \theta_{p,q}[h, g, k]$ (cf. line (4.8)). The claim follows for $x = \#$ and $x = 0$ analogously with lines (4.7) and (4.9), respectively.

Suppose $(\mathbb{N}, S) \models \theta_{p,q}[h, g, k]$. Consider the case $0 < g \leq k$. Then there is a transition $(q', 1, q, \rightarrow) \in \Delta$ for some $q' \in Q$ (cf. line (4.8)). Furthermore, $(h, g, k) \in R_{p,q'}$ and $\lambda(k)(g, 1) = 1$. Hence, dd' is a projected run of $\mathcal{A}$ on w. Again the other cases are similar. The claim follows since for all assignments exactly one of the premises in θ is true. □

It remains to generalize the above result to words of arbitrary lengths. But since we do not know the length of a word in $L(\mathcal{A})$ (and if there is any) we cannot simply quantify all letters and verify the existence of a projected run. However, since $\mathcal{A}$ is a Progressive Grid Word Automaton we can quantify a single letter and build the transitive closure of all projected runs of $\mathcal{A}$ on all Grid Words. This works because the automaton can "forget" all letters it has read because it cannot move backwards. Finally, the formula φ is defined as follows:

$$\varphi := \exists h_0[h_0 = 1 \wedge \forall X_1 \ldots \forall X_n \, (X_1(h_0) \wedge \varphi_{\text{closure}}(X_1, \ldots, X_n)) \rightarrow \exists h(\bigvee_{q \in F} X_q(h))]$$

$$\text{where } \varphi_{\text{closure}}(X_1, \ldots, X_n) := \forall h \forall g \left(\bigwedge_{p,q \in \underline{n}} (X_p(h) \wedge \exists k(\theta_{p,q}(h, g, k))) \rightarrow X_q(g) \right)$$

Correctness: Suppose $L(\mathcal{A}) \neq \emptyset$. Then there is a Grid Word $w \in L(\mathcal{A})$ and a projected run $(q_1, h_1) \ldots (q_{m+1}, h_{m+1})$ of $\mathcal{A}$ on w where $m = |w|$ due to lemma 4. Furthermore, $h_1 = 1$ and $q_{m+1} \in F$ since we can assume that the projected run is induced by an accepting run of $\mathcal{A}$ on w. With the preliminary considerations above we have that $(\mathbb{N}, S) \models \theta_{q_i, q_{i+1}}[h_i, h_{i+1}, \lambda^{-1}(w_i)]$ for all $1 \leq i \leq m$. Hence, for all $S_1, \ldots, S_n \subseteq \mathbb{N}$ with $(\mathbb{N}, S) \models X_1(1) \wedge \varphi_{\text{closure}}[S_1, \ldots, S_n]$ we have that $h_i \in S_{q_i}$ for all $1 \leq i \leq m + 1$ since $h_1 = 1 \in S_1$ and $q_1 = 1$ is the initial state of $\mathcal{A}$. Thus, $h_{m+1} \in S_f$ where $f = q_{m+1} \in F$. Therefore, we have that $(\mathbb{N}, S) \models (\exists h(\bigvee_{q \in F} X_q(h)))[S_1, \ldots, S_n]$. In conclusion, $(\mathbb{N}, S) \models \varphi$.

On the contrary, assume $(\mathbb{N}, S) \models \varphi$. Let $S'_1, \ldots, S'_n \subseteq \mathbb{N}$ minimal such that $(\mathbb{N}, S) \models X_1(1) \wedge \varphi_{\text{closure}}[S'_1, \ldots, S'_n]$. That is, if an element is removed from any of the S'_i the satisfaction relation does not hold anymore. Note that those S'_i exist since $1 \in S_1$ can be realized trivially and the implications are pairwise independent. By assumption there exists a $f \in F \subseteq \{1, \ldots, n\}$ and $h_f \in \mathbb{N}$ such that $h_f \in S_f$. Furthermore, there are sequences

$$q_1, \ldots, q_{m+1} \in \{1, \ldots, n\}, \; m \in \mathbb{N} \text{ where } q_1 = 1 \text{ and } q_m = f$$
$$\text{and } h_1, \ldots, h_{m+1}, k_1, \ldots, k_m \in \mathbb{N} \text{ where } h_1 = 1, h_{m+1} = h_f$$

such that $(\mathbb{N}, S) \models \theta_{q_i, q_{i+1}}[h_i, h_{i+1}, k_i]$ for all $1 \leq i \leq m$. Otherwise, some S_i would not be minimal. With the preliminary considerations it follows that $(q_1, h_1) \ldots (q_{m+1}, h_{m+1})$ is a

projected run of $\mathcal{A}$ on $w := \lambda(k_1 \dots k_m)$. Since $q_{m+1} = f \in F$ there is an accepting run of $\mathcal{A}$ on w due to lemma 4. Hence, $w \in L(\mathcal{A}) \neq \emptyset$. This concludes the proof of the decidability of the emptiness problem for Progressive Grid Word Automata. □

The decidability of the emptiness problem can be extended to Progressive Grid Word Büchi Automata with some minor changes in the proof.

Proposition 1. *The emptiness problem for Progressive Grid Word Büchi Automata is decidable.*

Proof approach. The proof of the decidability of the emptiness problem for Progressive Grid Word Automata needs two adaptations. Firstly, a run of a Progressive Grid Word Büchi Automaton is an infinite sequence. Therefore, a projected run of a Progressive Grid Word Büchi Automaton needs to be an infinite sequence, too. The relationship between an item of this sequence and its successor stays the same as in the definition of a projected run. Note that our formulae in the proof and lemma 5 do not need to be adapted because they consider only an infix of a projected run or the transitive closure of all possible projected runs of arbitrary length. Furthermore, no constraint demands the end of a (projected) run.

Secondly, Progressive Grid Word Büchi Automata have a different acceptance condition. They have to enter an accepting state infinitely often by moving to the right. This condition is inherited by the adapted projected run. That is, given a projected run $\pi = (q_1, h_1)(q_2, h_2) \dots$ on a ω-Grid Word v of some Progressive Grid Word Büchi Automaton $\mathcal{A}$, v is accepted if and only if there are infinitely many $i \in \mathbb{N}$ such that q_i is an accepting state. Since the state space of $\mathcal{A}$ is finite this is the case if and only if π has the shape $\pi = (1,1) \dots (f,h) \dots (f,h') \dots$ for some $f \in F$ and $h' \geq h$. In other words, at some point $\mathcal{A}$ moves to the right and enters an accepting state f. Afterwards, it can enter the state f again by moving to the right after a finite number of steps. Then there is a ω-Grid Word such that $\mathcal{A}$ can enter f infinitely often by moving to the right because the infix between the two occurrences of f can be repeated. Note that if $h' < h$ then $\mathcal{A}$ hits the "bottom" of the ω-Grid Word after a finite number of repetitions.

All in all, only the acceptance check in the formula φ needs to be exchanged. That is, the sub-formula $\exists h(\bigvee_{q \in F} X_q(h))$ has to be replaced by

$$\exists h \exists h' \left[h' \geq h \wedge \bigvee_{f \in F} \left(X_f(h) \wedge \forall Y_1 \dots \forall Y_n \left(Y_f(h) \wedge \varphi_{\text{closure}}(Y_1, \dots Y_n) \right) \rightarrow Y_f(h') \right) \right]$$

which encodes precisely the discussed acceptance condition. □

Note that the proof of the decidability of the emptiness problem makes use of the special structure of a Grid Word (cf. proof of lemma 5) as well as of the fact that a column cannot be visited twice by a Progressive Grid Word Automaton. Allowing that the positions of an input word are labeled with arbitrary symbols from a finite alphabet Σ yields *three-way automata*. A formal definition may be found in [Ros79].

Proposition 2. *The emptiness problem for three-way automata is decidable.*

Proof approach. The structure of a Grid Word is used in lemma 5 and the formula $\theta_{p,q}$ to identify the labeling of a position in a single column of a Grid Word. For this purpose, we

quantified a variable z. Informally, we intended that $\lambda(k)$ is the column concerned. Then a label is inferred by comparing the vertical position with z, e.g. it holds that $x \geq z \Leftrightarrow x = 0$. In the case of three-way automata we have different labels from a finite alphabet Σ on all positions of the column. Let $\Sigma' := \Sigma \dot{\cup} \{\#\}$. Instead of quantifying z we quantify second order variables $Z_1, \dots Z_n$ where $n := \lceil log|\Sigma'| \rceil$. Furthermore, each symbol of Σ' can be identified by a unique number in the range 0 till n. Finally, we can look up the label at a position in a column by interpreting the $Z_1, \dots, Z_n$ as bit vector. That is, the vertical position x (in a column) is labeled with $a \in \Sigma'$ if and only if the bit vector $Z_1(x) \dots Z_n(x)$ represents the number identifying a. The decidability of the emptiness problem follows if all occurrences of z in the formulae in lemma 5 and the decidability proof of the emptiness problem are replaced with $Z_1, \dots, Z_n$ as described. □

Note that the definition of three-way automata does include an upper "border" of the input word while Grid Words have none. However, this has no influence on the proof approach above since the positions can easily be limited globally. To sum up, the special structure of Grid Words is not necessary for the decidability of the emptiness problem. On the other hand, it is necessary for the decidability that no column can be visited twice. We will show this by proving that the emptiness problem for General Grid Word Automata is undecidable in the next section. Prior to this we infer the decidability of the universality problem for Deterministic Progressive Grid Word Automata.

Corollary 3. *The universality problem for Deterministic Progressive Grid Word Automata is decidable.*

Proof. Let $\mathcal{A}$ be a Deterministic Progressive Grid Word Automaton. Then we can effectively construct a Deterministic Progressive Grid Word Automaton $\mathcal{A}'$ such that $L(\mathcal{A}') = \overline{L(\mathcal{A})}$ due to theorem 3. Hence, it holds that $L(\mathcal{A}) = \mathcal{G} \Leftrightarrow L(\mathcal{A}') = \emptyset$. The latter statement is decidable due to theorem 9. Thus, the universality problem is decidable. □

4.2.2 The Emptiness Problem for General Grid Word Automata

Theorem 10. *The emptiness problem for General Grid Word (Büchi) Automata is undecidable.*

Again we prove this result by providing a Turing-Reduction. This time an undecidable problem is reduced to the emptiness problem for General Grid Word (Büchi) Automata. Since this problem refers to register machines we will briefly introduce them.

Definition 13 (k-register machine)**.** A *k-register machine* $\mathcal{M}$ consists of k registers denoted as $R_1, \dots, R_k$ which can store elements of $\mathbb{N}$ and an ordered list of instructions $\ell_1, \dots, \ell_n$ for some $n \in \mathbb{N}$ where each ℓ_i for $1 \leq i \leq n-1$ is one of the following instructions:

- `INC`$_j$ (increase the value of register R_j by one),
- `DEC`$_j$ (decrease the value of register R_j by one),
- `IF` $R_j = 0$ `GOTO` m (if 0 is stored in the j-th register the next instruction is ℓ_m),
- `GOTO` m (the next instruction is ℓ_m)

for some $0 \leq j < k$. If not stated otherwise the next instruction is ℓ_{i+1}. The last instruction ℓ_n is `HALT` and has no successor instruction. Furthermore, a *configuration* of $\mathcal{M}$ is a tuple $(i, r_1, \ldots, r_k) \in \{1, \ldots, n\} \times \mathbb{N}^k$ where i is the index of an instruction and r_j is the value stored in register R_j for all $1 \leq j < k$. A configuration c of $\mathcal{M}$ is *reachable* from another configuration c_0 if the computation of $\mathcal{M}$ starting in c_0 yields the configuration c after a finite number of steps (in each step one instruction is "executed").

It has been proven by Minsky (cf. [Min67]) that the following problem regarding 2-register machines is undecidable.

Definition 14. (The simple reachability problem of 2-register machines).

`Input:` A 2-register machine $\mathcal{M}$ with n instructions.
`Output:` Is a configuration $(n, r_1, r_2),\ r_1, r_2 \in \mathbb{N}$ reachable from $(1, 0, 0)$?

Consequently, it suffices to reduce the simple reachability problem of 2-register machines to the emptiness problem of General Grid Word (Büchi) Automata to show that the latter one is undecidable. Therefore, given a 2-register machine $\mathcal{M}$ with n instructions we have to construct a General Grid Word (Büchi) Automaton $\mathcal{A}$ such that

$$\exists r_1, r_2 \in \mathbb{N} : (n, r_1, r_2) \text{ is reachable from } (1, 0, 0) \Leftrightarrow L(\mathcal{A}) \neq \emptyset$$

As usual the proof is presented for General Grid Word Automata and adapted to General Grid Word Büchi Automata afterwards. The basic idea of the construction is naturally that $\mathcal{A}$ checks if some prefix of a given Grid Word encodes a valid computation of $\mathcal{M}$ starting in $(1, 0, 0)$ and at the end of this computation $\mathcal{M}$ "executes" the `HALT` instruction. Formally every computation of $\mathcal{M}$ is a sequence $(i_1, r_1^1, r_2^1) \ldots (i_p, r_1^p, r_2^p)$ of configurations where p is the number of steps, i_k the instruction and r_j^k the value of register R_j in the k-th step for all $j \in \{1, 2\}$,$1 \leq k \leq p$. We borrow the encoding of such a sequence from [Czy13]. While the current instruction is stored in the state space of $\mathcal{A}$ the register values are encoded in a Grid Word as follows:

$$w = \lambda(r_1^1)\lambda(r_2^1)\lambda(r_1^2)\lambda(r_2^2) \cdot \ldots \cdot \lambda(r_1^p)\lambda(r_2^p) \in \mathcal{G}$$

In other words each column of w encodes a register value in unary format. We define $\text{Enc}(\mathcal{M})$ as the set of all Grid Words $uv \in \mathcal{G}$ where u encodes a valid computation of $\mathcal{M}$ as above starting in $(1, 0, 0)$ and yielding in (n, r_1, r_2) for some $r_1, r_2 \in \mathbb{N}$. Note that there is exactly one prefix u since the computation of $\mathcal{M}$ is deterministic. That is, formally it should hold that $L(\mathcal{A}) = \text{Enc}(\mathcal{M})$. To achieve this $\mathcal{A}$ will operate on a Grid Word $w \in \mathcal{G}$ as follows:

1) Check if the input word starts with $\lambda(00)$. If it does return to the initial position. Otherwise, w cannot encode a computation starting in $(1, 0, 0)$. Hence, reject.

2) If the current instruction is $\ell_n =$ `HALT` accept w.

3) For the current instruction check if the value of the first register is assigned correctly. If not reject.

4) Otherwise move back and check if the values of the second register are correct. If not reject.

5) Update the current instruction and proceed with 2) on the next register values.

Assume that $\mathcal{A}$ can be constructed. Clearly, if $\mathcal{A}$ accepts some $w \in \mathcal{G}$ then $w \in \mathrm{Enc}(\mathcal{M})$. Therefore, there is a configuration (n, r_1, r_2) of $\mathcal{M}$ reachable from $(1, 0, 0)$. On the other hand, if there is such a computation of $\mathcal{M}$ then $\mathcal{A}$ accepts the encoded computation w because it does never reject but moves steadily to the right (cf. 5)) and encounters the `HALT` instruction.

We conclude the proof by providing the construction of $\mathcal{A} = (Q, \Delta, q_0, F)$. Since $\mathcal{A}$ is huge we define it component wise where each component corresponds to one instruction. For the i-th instruction of $\mathcal{M}$ we denote the number of states required to verify it by $s(i)$. Then the state space of $\mathcal{A}$ is $Q := \{q_{i,j} \mid 0 \leq i \leq n,\ 1 \leq j \leq s(i)\}$ and $F := \{q_{n,1}\}$. The states with index zero are used for the initialization (cf. step 1)). Further on, $\Delta := \bigcup_{0 \leq i \leq n} \Delta_i$ and $q_0 := q_{0,1}$. We start with the initialization: $s(0) := 2$ and $\Delta_0 := \{(q_{0,1}, 0, q_{0,2}, \rightarrow), (q_{0,2}, 0, q_{1,1}, \leftarrow)\}$. Indeed, this implements 1) and $\mathcal{A}$ starts verifying the first instruction. Eventually, the verification of the i-th instruction is realized as follows:

Case 1: $i = n$. Then the instruction is `HALT` and according to 2) $\mathcal{A}$ should accept the input. Hence, $s(n) := 1$ and $\Delta_n := \{q_{n,1}, x, q_{n,1}, \rightarrow) \mid x \in \{\#, 0, 1\}\}$. Recall that $F = \{q_{n,1}\}$.

Case 2: $\ell_i = \texttt{DEC}_1$. We need eleven states to verify this instruction. Thus, $s(i) := 11$. We assume that the current position is either $(q_{i,1}, 0, l)$ or $(q_{i,1}, 1, l)$ where w_l encodes the value of the first register in the correct computation step of $\mathcal{M}$. The upper two cases satisfy this. In this and the following cases we will maintain this condition. An example of $\mathcal{A}$ verifying the instruction $\texttt{DEC}_1$ is illustrated in figure 4.1a.

$$\Delta_i := \left.\begin{array}{l} \{(q_{i,1}, \#, q_{i,1}, \uparrow), (q_{i,1}, 1, q_{i,1}, \uparrow), \\ (q_{i,1}, 0, q_{i,2}, \rightarrow)\} \end{array}\right\}$$

Move to the first zero in the column. Then the automaton knows the value of R_1.

$$\cup \quad \{(q_{i,2}, x, q_{i,3}, \rightarrow) \mid x \in \{\#, 0, 1\}\}$$

Move to the next value of $R_1 \ldots$

$$\cup \left.\begin{array}{l} \{(q_{i,3}, 0, q_{i,4}, \downarrow), (q_{i,4}, 0, q_{i,5}, \downarrow) \\ (q_{i,4}, \#, q_{i,6}, \leftarrow), (q_{i,5}, 1, q_{i,6}, \leftarrow) \\ (q_{i,5}, \#, q_{i,6}, \leftarrow)\} \end{array}\right\}$$

$\ldots$ and check if it is decreased by one. That is, there are two zeros below the current label or in the case that the value was zero the automaton reads the bottom symbol #. Afterwards, move left to check the second register.

$$\cup \left.\begin{array}{l} \{(q_{i,6}, 0, q_{i,6}, \downarrow), (q_{i,6}, 1, q_{i,7}, \uparrow) \\ (q_{i,6}, \#, q_{i,7}, \uparrow), (q_{i,7}, 1, q_{i,7}, \uparrow)\} \end{array}\right\}$$

Look up the value of the second register (note that the value may be smaller than the one of the first register).

$$\cup \left.\begin{array}{l} \{(q_{i,7}, 0, q_{i,8}, \rightarrow), (q_{i,8}, 1, q_{i,9}, \rightarrow) \\ (q_{i,8}, 0, q_{i,9}, \rightarrow)\} \end{array}\right\}$$

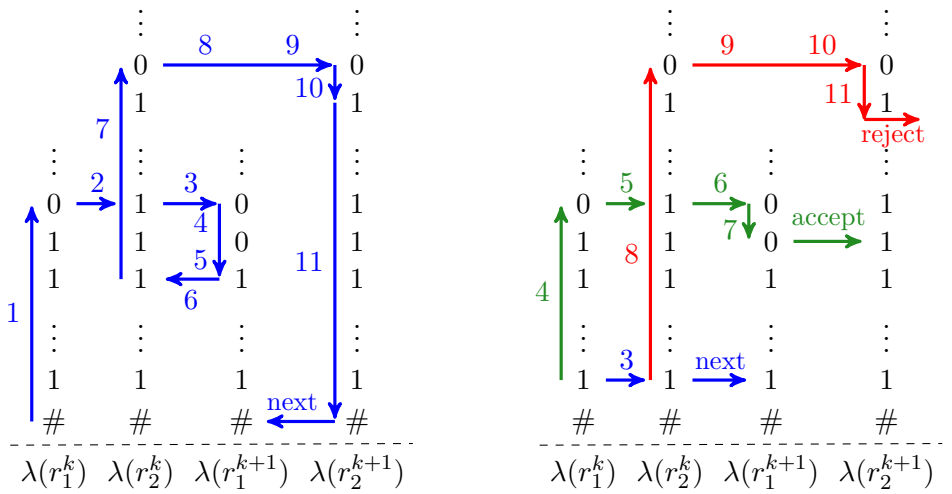
Move to the next value of R_2.

$$\cup \quad \{(q_{i,9}, 0, q_{i,10}, \downarrow), (q_{i,10}, 1, q_{i,11}, \downarrow)\}$$

Verify that the value is the same.

$$\cup \quad \{(q_{i,11}, 1, q_{i,11}, \downarrow), (q_{i,11}, \#, q_{i+1,1}, \leftarrow)\}$$

Finally, move to the bottom to maintain the condition above and adapt the instruction. Note that the register value for R_1 is stored in the previous column.



(a) Partial run of $\mathcal{A}$ that verifies the instruction $\ell_i = \texttt{DEC}_1$ successfully. The label "next" indicates that $\mathcal{A}$ starts verifying the next instruction.

(b) $\mathcal{B}$ tries to falsify the instruction $\ell_i = \texttt{IF } R_1 = 0 \texttt{ GOTO } m$. Each color corresponds to a possible partial run which is non-deterministically chosen.

Figure 4.1: Illustrations of the General Grid Word Automaton $\mathcal{A}$ and the Non-deterministic Progressive Grid Word Automaton $\mathcal{B}$ constructed in the proofs of theorem 10 and 11, respectively. The automata operate on the column on the right hand side of the line illustrating the run. The numbers correspond to the numbering of the states in the constructions.

This respects 3), 4) and 5). Note that if the word has a different shape as required by the transitions above it does not encode the $\texttt{DEC}_1$ instruction and $\mathcal{A}$ rejects since there is no transition. In particular, $\mathcal{A}$ rejects if the word ends unexpectedly.

Case 3: $\ell_i = \texttt{IF } R_1 = 0 \texttt{ GOTO } m$. We have seen in the case for $\texttt{DEC}_1$ that we can implement the check if the values of both registers stay unchanged. Moreover, before adapting the instruction $\mathcal{A}$ can move back one column and adapt the instruction to ℓ_m by changing the internal state to $q_{m,1}$ instead of $q_{i+1,1}$ if the value is zero. Since the register values have stayed the same this implements the instruction correctly. Formally, the last transition of the $\texttt{DEC}_1$ case is replaced by $(q_{i,11}, \#, q_{i,12}, \leftarrow), (q_{i,12}, \#, q_{i,13}, \uparrow), (q_{i,13}, 1, q_{i+1,1}, \downarrow)$ and $(q_{i,13}, 0, q_{m,1}, \downarrow)$.

The implementations for the remaining instructions are completely analogous. Since there is no further insight in providing them and the construction is large it is reasonable to leave them out.

The construction as well as the reasoning about the correctness is exactly the same for General Grid Word Büchi Automaton. Just $\text{Enc}(\mathcal{M})$ has to be defined as the subset of $\mathcal{G}_\omega$ instead of $\mathcal{G}$. Note that $\mathcal{A}$ does enter the only accepting state if and only if a finite prefix of the ω-Grid Word processed encodes a computation of $\mathcal{M}$. This matches the acceptance condition of General Grid Word Büchi Automata. □

Remark 5. In the construction of the General Grid Word (Büchi) Automaton $\mathcal{A}$ in the proof of theorem 10 the non-determinism is only used to reject a given input by not providing

a transition. Therefore, the construction can easily be made deterministic by introducing all the missing transitions to a sink state that rejects the input. Hence, the emptiness problem would already be undecidable for a deterministic version of General Grid Word (Büchi) Automata.

4.3 The Universality Problem

We have already seen that the universality problem is decidable for Deterministic Progressive Grid Word Automata in corollary 3. The proof of this result is based on the result that deterministically grid recognizable languages are closed under complement. Hence, we cannot use the same approach to handle the non-deterministic case since non-deterministically grid recognizable languages are not closed under complement (cf. corollary 1). In fact, it turns out that the universality problem for Non-deterministic Progressive Grid Word Automata is undecidable.

Theorem 11. *The universality problem for Non-deterministic Progressive Grid Word (Büchi) Automata is undecidable.*

To prove the undecidability of the universality problem we use the same approach as for the undecidability result for the emptiness problem for General Grid Word (Büchi) Automata. That means we reduce the simple reachability problem of 2-register machines to the universality problem for Non-deterministic Progressive Grid Word (Büchi) Automata. Again we will focus on Non-deterministic Progressive Grid Word Automata. Let $\text{Enc}(\mathcal{M})$ be the set of all Grid Words uv where u is the encoded computation of a 2-register machine $\mathcal{M}$ with n instructions starting in $(1,0,0)$ and ending in some (n, r_1, r_2) for $r_1, r_2 \in \mathbb{N}$ as described in section 4.2.2. Given a 2-register machine $\mathcal{M}$ we construct a Non-deterministic Progressive Grid Word Automaton $\mathcal{B}$ such that $L(\mathcal{B}) = \mathcal{G} \setminus \text{Enc}(\mathcal{M})$. Then it holds that

$$\begin{aligned} L(\mathcal{B}) = \mathcal{G} &\Leftrightarrow \text{Enc}(\mathcal{M}) = \emptyset \\ &\Leftrightarrow \forall r_1, r_2 \in \mathbb{N} : (n, r_1, r_2) \text{ is not reachable from } (1,0,0) \text{ w.r.t. } \mathcal{M} \end{aligned}$$

This yields the desired Turing-reduction from the simple reachability problem of 2-register machines to the universality problem for Non-deterministic Progressive Grid Word Automata. To achieve the property above $\mathcal{B}$ needs to accept all Grid Words that do not have a prefix encoding a valid computation of $\mathcal{M}$ starting in $(1,0,0)$. The basic idea is that $\mathcal{B}$ guesses if the encoding is wrong at some point. In more detail, $\mathcal{B}$ operates as follows on a Grid Word w:

1) First of all, $\mathcal{B}$ guesses if the word does not start with $\lambda(00)$. If this is the case then the computation of $\mathcal{M}$ does not start in $(1,0,0)$. Hence, $\mathcal{B}$ validates this and accepts. If the validation fails $\mathcal{B}$ rejects. Otherwise, $\mathcal{B}$ just initializes the current instruction with the first one ℓ_1 (the instruction is saved in the state space).

2) If the current instruction is $\ell_n =$ `HALT` $\mathcal{B}$ enters a sink state and rejects w since the computation ends and no encoding errors were detected before.

3) Suppose $\mathcal{B}$ has checked the first $j-1$ steps of the computation and neither the `HALT` instruction occurred nor $\mathcal{B}$ has found an error yet. Then the current position in the

run of $\mathcal{B}$ on w is given by $(q, 1, 2j-1)$. In other words $\mathcal{B}$ starts to read the infix $\lambda(r_1^j)\lambda(r_2^j)\lambda(r_1^{j+1})\lambda(r_2^{j+1})$ of w (we assume here that the word does not end unexpectedly). If $\mathcal{B}$ decides non-deterministically that the current instruction is not encoded correctly it verifies this and accepts (e.g. the current instruction is $\texttt{INC}_1$ but $r_1^j \neq r_1^{j+1}+1$). Otherwise, $\mathcal{B}$ moves two letters to the right and updates the instruction. The condition in a $\texttt{IF } R_i = 0 \texttt{ GOTO } m$ instruction to determine the next one can easily be checked "on the fly". Afterwards, $\mathcal{B}$ is in the same situation as before but one computation step ahead and continues with 2).

All states of $\mathcal{B}$ but the sink state used in 2) are accepting states. Thus, $\mathcal{B}$ accepts if the word ends unexpectedly. This behavior respects that the word cannot encode a desired computation of $\mathcal{M}$. Before the formal construction of $\mathcal{B}$ is outlined we prove the correctness.
Correctness: Assume that there is a configuration (n, r_1, r_2) of $\mathcal{M}$ reachable from $(1, 0, 0)$. Then $\mathcal{B}$ rejects the Grid Word that encodes this computation since either $\mathcal{B}$ guesses that there is a error in the encoding but fails to validate this or $\mathcal{B}$ encounters the HALT instruction. Note that $\mathcal{B}$ can only accept if either the Grid Word ends unexpectedly or there is an error in the encoding. Furthermore, there exists such a Grid Word by construction (cf. section 4.2.2). Hence, $L(\mathcal{B}) \neq \mathcal{G}$.

On the contrary, if there is no configuration (n, r_1, r_2) of $\mathcal{M}$ reachable from $(1, 0, 0)$ then there is no Grid Word that encodes such a configuration. It follows that for each Grid Word either $\mathcal{B}$ can guess correctly at some point that there is an error or the word ends before the HALT instruction is encountered (e.g. when the computation does not terminate). In both cases there is an accepting run of $\mathcal{B}$ on this Grid Word since all states but the sink state are accepting states. Thus, $L(\mathcal{B}) = \mathcal{G}$.

The formal construction of $\mathcal{B}$ is both technical and huge. Therefore, we will content ourselves with the implementation of an instruction $\ell_i = \texttt{IF } R_1 = 0 \texttt{ GOTO } m$. The other instructions and the initial check are very similar and therefore omitted. We denote the sink state of $\mathcal{B}$ established in 2) as q_r. Remember that all states but q_r are accepting states. Another special state of $\mathcal{B}$ is q_a. Once $\mathcal{B}$ enters this state it accepts the remaining word. Furthermore, states named $q_{i,j}$ for some $j \in \mathbb{N}$ are dedicated to the implementation of the i-th instruction. In particular, the state $q_{i,1}$ is the "initial state" of some instruction. An example of $\mathcal{B}$ falsifying the instruction ℓ_i is depicted in figure 4.1b. In the following we give the transitions that establish the desired behavior of $\mathcal{B}$ w.r.t. ℓ_i as above. We assume that the preconditions from 3) are met. Of course, the construction will maintain this condition.

$$\left.\begin{array}{l}\{(q_{i,1}, 0, q_{i,2}, \rightarrow), (q_{i,2}, x, q_{m,1}, \rightarrow),\\ (q_{i,1}, 1, q_{i,3}, \rightarrow), (q_{i,3}, x, q_{i+1,1}, \rightarrow)\\ \mid x \in \{0,1\}\}\end{array}\right\}$$

If $\mathcal{B}$ decides that there is no error in the encoding it just checks if the value of R_1 equals zero and changes the instruction accordingly.

$$\cup \left\{\begin{array}{l}\{(q_{i,1}, 1, q_{i,4}, \uparrow), (q_{i,1}, 0, q_{i,5}, \rightarrow),\\ (q_{i,4}, 1, q_{i,4}, \uparrow), (q_{i,4}, 0, q_{i,5}, \rightarrow),\\ (q_{i,5}, x, q_{i,6}, \rightarrow), (q_{i,6}, 1, q_a, \rightarrow),\\ (q_{i,6}, 0, q_{i,7}, \downarrow), (q_{i,7}, 0, q_a, \rightarrow),\\ (q_{i,7}, 1, q_r, \rightarrow), (q_{i,7}, \#, q_r, \rightarrow)\\ \mid x \in \{0,1\}\}\end{array}\right.$$

$\mathcal{B}$ may guess that the first register has changed but it should stay the same since the instruction does not change the value. In the state $q_{i,6}$ $\mathcal{B}$ checks if the value has increased and in $q_{i,7}$ if it has decreased. If both tests fail (e.g. in $q_{i,7}$ $\mathcal{B}$ does not read a 0) the value has stayed the same and the guess of $\mathcal{B}$ was wrong. Thus, $\mathcal{B}$ has to reject the word.

$$
\cup \quad \left.\begin{array}{l}
\{(q_{i,2}, 1, q_{i,8}, \uparrow), (q_{i,2}, 0, q_{i,9}, \rightarrow), \\
(q_{i,3}, 1, q_{i,8}, \uparrow), (q_{i,3}, 0, q_{i,9}, \rightarrow), \\
(q_{i,8}, 1, q_{i,8}, \uparrow), (q_{i,8}, 0, q_{i,9}, \rightarrow), \\
(q_{i,9}, x, q_{i,10}, \rightarrow), (q_{i,10}, 1, q_a, \rightarrow), \\
(q_{i,10}, 0, q_{i,11}, \downarrow), (q_{i,11}, 0, q_a, \rightarrow), \\
(q_{i,11}, 1, q_r, \rightarrow), (q_{i,11}, \#, q_r, \rightarrow) \\
\mid x \in \{0, 1\}\}
\end{array}\right\}
$$

Completely analogous $\mathcal{B}$ may guess that the second register has changed and hence, that the encoding is wrong. Note that depending on the value of R_1 $\mathcal{B}$ may be in state $q_{i,2}$ or $q_{i,3}$.

If we understand $\mathcal{B}$ as a Non-deterministic Progressive Grid Word Büchi Automaton and define $\mathrm{Enc}(\mathcal{M})$ as subset of $\mathcal{G}_\omega$ accordingly then the following holds. Let $w \in \mathcal{G}_\omega$. If $w = uv$ such that u encodes a valid (and terminating) computation of $\mathcal{M}$ then either $\mathcal{B}$ fails to verify an error in the encoding or it reads the `HALT` instruction after a finite number of movements to the right. In both cases $\mathcal{B}$ enters the rejecting state q_r. That is, $\mathcal{B}$ moves to the right without entering an accepting state ad infinitum. Thus, it rejects w. Otherwise, if there is no such prefix u then either $\mathcal{B}$ finds an error in the encoding or moves to the right without entering q_r ad infinitum. Since all states but q_r are accepting states it accepts w. Hence, it recognizes $\mathcal{G}_\omega \setminus \mathrm{Enc}(\mathcal{M})$.

In conclusion, the universality problem for Non-deterministic Progressive Grid Word Automata as well as Non-deterministic Progressive Grid Word Büchi Automata is undecidable. □

We finalize this chapter about decision problems with an undecidability result for the equivalence problem for Non-deterministic Progressive Grid Word Automata.

Corollary 4. *The equivalence problem for Non-deterministic Progressive Grid Word (Büchi) Automata is undecidable.*

Proof. Suppose the equivalence problem for Non-deterministic Progressive Grid Word (Büchi) Automata is decidable. Then a decision procedure for the universality problem is as follows: Given a Non-deterministic Progressive Grid Word (Büchi) Automata $\mathcal{A}$, construct a Progressive Grid Word (Büchi) Automaton $\mathcal{A}_\mathcal{G}$ with $L(\mathcal{A}_\mathcal{G}) = \mathcal{G}$ (or $L(\mathcal{A}_\mathcal{G}) = \mathcal{G}_\omega$). Then check whether $L(\mathcal{A}_\mathcal{G}) = L(\mathcal{A})$ holds and return the result. Clearly, $L(\mathcal{A}_\mathcal{G}) = L(\mathcal{A})$ if and only if $L(\mathcal{A}) = \mathcal{G}$ ($L(\mathcal{A}) = \mathcal{G}_\omega$). Hence, the universality problem for Non-deterministic Progressive Grid Word (Büchi) Automata is decidable. This is a contradiction to theorem 11. Thus, the equivalence problem for Non-deterministic Grid Word (Büchi) Automata is undecidable. A proper construction of $\mathcal{A}_\mathcal{G}$ would be $\mathcal{A}_\mathcal{G} := (\{q\}, \{(q, x, q, \rightarrow) \mid x \in \{\#, 0, 1\}\}, q, \{q\})$ which is even deterministic. □

Regarding Deterministic Progressive Grid Word Automata the equivalence problem remains open. Remember that the emptiness as well as the universality problem for Deterministic Progressive Grid Word Automata is decidable. Therefore, a reduction like in the proof of corollary 4 is not possible. On the contrary, if we assume that the equivalence problem is decidable then there are several obstacles that have to be addressed in the proof. Firstly, the recognized languages are not closed under intersection. Hence, there is no simple reduction to the emptiness problem possible. Secondly, an automaton may use the special structure of a Grid Word (e.g. if a position labeled with 1 it is known that all positions

below are labeled with 1, too) while the one that it should be compared with does not. This may lead to a fundamental different behavior of the two given automata. In addition, we have seen in example 5 that the possible behavior is not local. For instance, the behavior on the first column of a given Grid Word may depend on the last column of the Grid Word. Eventually, a minimal Deterministic Progressive Grid Word Automaton is not unique (cf. remark 2). That is, it does not suffice to find a minimization procedure.

A similar situation applies for Deterministic Progressive Grid Word Büchi Automata. In addition to the equivalence problem the universality problem for Deterministic Progressive Grid Word Büchi Automata remains open. Note that the result for Deterministic Progressive Grid Word Automata (cf. corollary 3) cannot be transferred to the Büchi version because they are not closed under complement (cf. theorem 4).

5 Comparison with Other Models

The aim in this chapter is the comparison of Progressive Grid Word (Büchi) Automata with other automata models, namely Strong (Büchi) Automata and Register Automata, that can recognize languages over infinite alphabets – this includes closure properties, decidability results as well as the sets of languages recognized by these models. Again we will focus on Progressive Grid Word Automata and extend our results to Progressive Grid Word Büchi Automata afterwards.

5.1 Comparison with Strong Automata

Strong (Büchi) Automata are introduced and studied in [STW11]. Clearly, we cannot argue properly about them without a definition. Therefore, we start with the definition of Strong (Büchi) Automata and their semantics which we borrow from [Czy13].

Throughout this chapter let $\mathcal{M}$ always denote a structure with universe M and $\mathcal{L}$ a logic. For example, $M = \mathbb{N}$, $\mathcal{M} = (\mathbb{N}, +)$ and $\mathcal{L}$ is the FO-Logic. Furthermore, the tuple $(\mathcal{M}, \mathcal{L})$ is called an *alphabet frame*.

Definition 15 (Strong Automaton)**.** Let $(\mathcal{M}, \mathcal{L})$ be an alphabet frame. A *Strong Automaton* over $(\mathcal{M}, \mathcal{L})$ with look-back $s \in \mathbb{N}$ is a tuple $\mathfrak{A} := (P, M^n, p_0, \Delta, F)$ where

- P is a finite set of states,
- M^n is the input alphabet,
- $p_0 \in P$ is the initial state,
- $\Delta \subseteq P \times \left(\bigcup_{i=1}^{s+1} \Psi_{i*n} \right) \times P$ is the finite transition relation where the Ψ_{i*n} are finite sets of $(\mathcal{M}, \mathcal{L})$-formulae with $i \cdot n$ free variables,
- and $F \subseteq P$ is the set of accepting states.

A *run* of a Strong Automaton $\mathfrak{A}$ on an input word $w = a_1 \dots a_m$, $a_i \in M^n$ $\forall 1 \leq i \leq n$ is a sequence of states $p_0 p_1 \dots p_m \in P^*$ (p_0 is the initial state) such that for all $1 \leq i \leq m$:

1) If $i \leq s$ then there is a $\psi_{p_{i-1},p_i} \in \Psi_{i*n}$ with $(p_{i-1}, \psi_{p_{i-1},p_i}, p_i) \in \Delta$ and
$$\mathcal{M} \models \psi_{p_{i-1},p_i}[a_{1,1}, \dots, a_{1,n}, a_{2,1}, \dots a_{i,n}].$$

2) Otherwise, there is a $\psi_{p_{i-1},p_i} \in \Psi_{(s+1)*n}$ with $(p_{i-1}, \psi_{p_{i-1},p_i}, p_i) \in \Delta$ and
$$\mathcal{M} \models \psi_{p_{i-1},p_i}[a_{i-s,1}, \dots, a_{i-s,n}, \dots a_{i,n}].$$

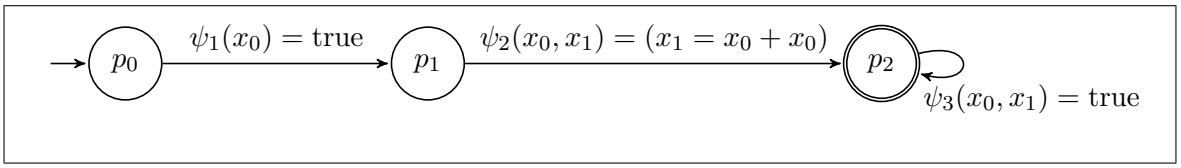


Figure 5.1: Graphical representation of a Strong (Büchi) Automaton with look-back 1 over $(\mathbb{N}, +)$ and FO-logic

In other words, a transition labeled with ψ can be taken if $\mathcal{M}$ models ψ under the constraint that the last s letters (or less if the automaton has not read s letters yet) and the current letter of w are assigned to the free variables of ψ. Analogous to Progressive Grid Word Automata we say that $\mathfrak{A}$ *accepts* w if $p_m \in F$. Moreover, $L(\mathfrak{A}) = \{w \in (M^n)^* \mid \mathfrak{A} \text{ accepts } w\}$ is the language *recognized* by $\mathfrak{A}$.

Definition 16 (Strong Büchi Automaton)**.** Let $(\mathcal{M}, \mathcal{L})$ be an alphabet frame. A *Strong Büchi Automaton* over $(\mathcal{M}, \mathcal{L})$ with look-back $s \in \mathbb{N}$ is a tuple $\mathfrak{A} := (P, M^n, p_0, \Delta, F)$ where the single components are exactly the same as for Strong Automata (cf. definition 15).

Like Progressive Grid Word Büchi Automata a Strong Büchi Automaton accepts infinite words. A *run* of a Strong Büchi Automaton $\mathfrak{A}$ on an infinite input word $v = a_1 a_2 \ldots$ with $a_i \in M^n \ \forall i \geq 1$ is an infinite sequence of states $p_0 p_1 \ldots \in P^\omega$ (p_0 is the initial state) such that the conditions 1) and 2) from the definition of a run of a Strong Automaton hold for all $i \geq 1$. Moreover, $\mathfrak{A}$ *accepts* v if there are infinitely many $i \in \mathbb{N}$ such that $p_i \in F$. This acceptance condition corresponds precisely to the acceptance condition of Progressive Grid Word Büchi Automata (cf. definition 6). At last, the language *recognized* by $\mathfrak{A}$ is $L(\mathfrak{A}) = \{w \in (M^n)^\omega \mid \mathfrak{A} \text{ accepts } w\}$.

Note that in contrast to our definition Strong (Büchi) Automata defined in [Czy13] have always a look-back of $s = 1$. Strong (Büchi) Automata with arbitrary look-back are called extended Strong (Büchi) Automata. Here we will use the more general version without the term "extened" and state the look-back explicitly, if necessary.

Example 9. The language $L' := \{n_0 n_1 u \mid n_0, n_1 \in \mathbb{N}, n_1 = 2n_0, u \in \mathbb{N}^*\}$ can be recognized by a Strong Automaton with look-back 1 using the structure $(\mathbb{N}, +)$ and FO-logic. A graphical representation of a Strong Automaton recognizing L' is given in figure 5.1. If we understand the illustrated graph as Strong Büchi Automaton it recognizes the language $L'_\omega = \{n_0 n_1 u \mid n_0, n_1 \in \mathbb{N}, n_1 = 2n_0, u \in \mathbb{N}^\omega\} \subseteq \mathbb{N}^\omega$.

First of all, we should be aware of the fact that Strong (Büchi) Automata can even recognize languages over uncountable alphabets, e.g. $\mathcal{M} = (\mathbb{R}, +, <, 0, 1)$, while Progressive Grid Word (Büchi) Automata are restricted to alphabets isomorphic to $\mathbb{N}$. With this in mind, we start – analogous to the structure of the earlier chapters – by comparing the closure properties of (ω-)grid recognizable languages and languages recognized by Strong (Büchi) Automata.

Proposition 3. *The languages recognized by Strong Automata are closed under complement, union and intersection as well as concatenation. Moreover, the languages recognized by Strong Büchi Automata are closed under complement, union and intersection.*

	(ω-)grid recognizable				Strong (Büchi)		quasi-
closed under	det.		non-det.		Automata		regular
Complement	Yes ✓	(No)	No	(No)	Yes ✓	(Yes ✓)	No
Intersection	No	(No)	No	(No)	Yes ✓	(Yes ✓)	Yes ✓
Union	No	(No)	Yes ✓	(Yes ✓)	Yes ✓	(Yes ✓)	Yes ✓
Concatenation	No	(-)	Yes ✓	(-)	Yes ✓	(-)	Yes ✓

Table 5.1: Overview: Comparison of closure properties. The properties of the Büchi variants are written in brackets.

Since Strong Automata are non-deterministic the closure under union and concatenation can be deduced intuitively and is similar to the construction given earlier for Non-deterministic Progressive Grid Word Automata (cf. the proofs of theorems 1 and 2). The closure under intersection and complement needs more arguments. Basically, one can construct a "product" automaton by using the logical "and" to concatenate the formulae of two Strong Automata to conclude the closure under intersection. For Strong Büchi Automata the constructions are the same (note that the concatenation is not defined in this case). Regarding the closure under complement with respect to Strong Automata it is possible to construct a deterministic version and swap the accepting states. This can be achieved by some kind of "powerset construction". Suppose a state of a Strong Automaton has two outgoing transitions labeled with ψ and ϑ. Then in the deterministic automaton their counterparts are $\psi \wedge \vartheta$, $\psi \wedge \neg\vartheta$ and $\neg\psi \wedge \vartheta$ (assuming $\mathcal{L}$ contains the usual propositional conjunction and negation). The proof for the closure under complement with respect to Strong Büchi Automata is significantly costlier and not effective. Detailed proofs may be found in [Czy13].

Remember that deterministically grid recognizable languages are exclusively closed under complement while non-deterministically grid recognizable languages are exclusively closed under union and concatenation. Hence, Progressive Grid Word Automata are clearly at a disadvantage here. For ω-grid recognizable languages nearly the same situation emerges. The only difference is that deterministically ω-grid recognizable languages are not closed under complement, too, which is even worse. A complete overview of all closure properties of the compared models is given in table 5.1.

Next, we consider decidability results regarding the two automata models. The emptiness problem is of particular interest since we know that it is decidable for Progressive Grid Word (Büchi) Automata (cf. chapter 4). Contrariwise, the universality problem is undecidable for Non-deterministic Progressive Grid Word (Büchi) Automata. The emptiness problem for Strong (Büchi) Automata is defined analogous to the one of Progressive Grid Word (Büchi) Automata.

Proposition 4. *Let $\mathcal{M}$ be an arbitrary structure. Then it holds that*

1) *for Strong (Büchi) Automata over $(\mathcal{M}, \text{MSO-Logic})$ with look-back $s < 2$ the emptiness problem is decidable if the MSO-Logic of $\mathcal{M}$ is decidable,*

2) *and for Strong (Büchi) Automata over $(\mathcal{M}, \text{FO-Logic})$ with look-back $s \geq 2$ the emptiness problem is undecidable.*

The proof approach of the first part is a Turing-Reduction to the decidability result of the MSO-theory of $\mathcal{M}$. The basic idea is essentially the same as for Progressive Grid Word (Büchi) Automata (cf. section 4.2.1). That is, given a Strong (Büchi) Automaton a sentence over ($\mathcal{M}, MSO$-Logic) is constructed such that the sentence is valid if and only if there is a run of the Strong (Büchi) Automaton on some word. For that purpose a second order variable is required to "save" the last letter if $s = 1$. The undecidability result uses the same idea and encoding as we did in section 4.2.2 to show that the emptiness problem of General Grid Word (Büchi) Automata is undecidable. Again we refer to [Czy13] for detailed proofs.

We have shown that the emptiness problem for Progressive Grid Word (Büchi) Automata is decidable but for General Grid Word (Büchi) Automata it is undecidable. Intuitively, a Progressive Grid Word (Büchi) Automaton can memorize a single letter it has seen previously and compare it to the current letter by maintaining the same vertical position during the run on a (ω-)Grid Word. A General Grid Word (Büchi) Automaton can look up several letters it has seen before. In particular, it can compare the current letter with a fixed number of direct predecessors. Note that in the proof of the undecidability of the emptiness problem for General Grid Word (Büchi) Automata it suffices to use two movements to the left for each letter of the input word (cf. section 4.2.2). In other words, it suffices to compare the current letter with its two direct predecessors to render the emptiness problem undecidable. This observation corresponds to the result for Strong (Büchi) Automata. Hence, Strong (Büchi) Automata and Grid Word (Büchi) Automata differ marginally with respect to the decidability of the emptiness problem. However, note that the universality problem for Strong (Büchi) Automata is decidable since the recognized languages are closed under complement. This property does only apply to Deterministic Progressive Grid Word Automata (for Deterministic Progressive Grid Word Büchi Automata the problem remained open). Furthermore, the equivalence problem for Strong (Büchi) Automata is decidable, too because of the closure under intersection and complement. This is unknown for Deterministic Progressive Grid Word (Büchi) Automata and even undecidable for Non-deterministic Progressive Grid Word (Büchi) Automata (cf. corollary 4).

Lastly, we will consider the set of languages recognized by Strong (Büchi) Automata and Progressive Grid Word (Büchi) Automata, respectively. From the previous comparison we can already conclude that they are not the same. And indeed, they are even incomparable.

Theorem 12. *The set of (ω-)grid recognizable languages and the set of languages recognized by Strong (Büchi) Automata are incomparable.*

We will show this theorem by providing a language that is recognized by a Progressive Grid Word (Büchi) Automaton but not by a Strong (Büchi) Automaton and vice versa. Since the proof is rather long but can be split in two independent parts it makes sense to do so. More precisely, we will prove two lemmata and combine them afterwards to obtain the theorem.

Lemma 6. *Let $(\mathcal{M}, \mathcal{L})$ be an alphabet frame and $n \in \mathbb{N}$. Then the following statements hold:*

1) *$L_{\mathcal{M}} = \{a_0 \dots a_p \in (M^n)^* \mid p \in \mathbb{N}$ and $a_0 = a_p\}$ is not recognizable by a Strong Automaton with an arbitrary look-back.*

2) *$L^{\omega}_{\mathcal{M}} = \{a_0 a_1 a_2 \dots \in (M^n)^{\omega} \mid \forall i \in \mathbb{N}\ \exists j > i : a_j = a_0\}$ is not recognizable by a Strong Büchi Automaton with an arbitrary look-back.*

Proof. Assume $L(\mathfrak{A}) = L_{\mathcal{M}}$ for some Strong Automaton $\mathfrak{A} = (P, M^n, p_0, \Delta, F)$ with look-back s. Since $L_{\mathcal{M}}$ contains infinitely many words of length $s+2$ and Δ is finite there is a single run $\pi = p_0 \dots p_m$, $m \geq 2$ that witnesses the membership of two words

$$w_i = a_i a_0^s a_i \in L_{\mathcal{M}},\ i \in 1,2,\ a_0 \neq a_1 \neq a_2 \neq a_0,\ a_0, a_1, a_2 \in M^n$$

by the pigeonhole principle. Moreover, there is a transition $(p_{m-1}, \psi_m, p_m) \in \Delta$ where ψ_m has s free variables and $\mathcal{M} \models \psi[a_0, \dots, a_0, a_i]$ for $i \in \{1,2\}$. Note that we used a_j as a shorthand for $a_{j,1}, \dots, a_{j,n}$ in $\psi[a_0, \dots, a_0, a_i]$ for $j \in \{0,1,2\}$. Thus, $\mathfrak{A}$ accepts $v := a_1 a_0^s a_2 \notin L_{\mathcal{M}}$ because π is a valid run of $\mathfrak{A}$ on v. This contradicts $L(\mathfrak{A}) = L_{\mathcal{M}}$. It follows that $L_{\mathcal{M}}$ is not recognizable by a Strong Automaton with look-back s.

Regarding the second statement suppose there is a Strong Büchi Automaton with look-back s that recognizes $L_{\mathcal{M}}^{\omega}$. Then the same argumentation as above applies to two words

$$w_i = a_i a_0^s a_i u_i \in L_{\mathcal{M}}^{\omega},\ i \in 1,2,\ a_0 \neq a_1 \neq a_2 \neq a_0,\ a_0, a_1, a_2 \in M^n$$

where the $u_i \in (M^n)^{\omega}$ are proper suffixes. That is, u_1 does not contain any a_2 while u_2 does not contain any a_1. □

Intuitively, this result shows that while a Strong (Büchi) Automaton may remember several letters of the word it can not choose which of them (they are always the last s ones). On the contrary, Progressive Grid Word (Büchi) Automata can only remember a single letter but as long as they want (by "keeping the vertical position" in the (ω-)Grid Word).

Lemma 7. *The following two assertions hold:*

1) *The language* $L' := \{n_0 n_1 u \mid n_0, n_1 \in \mathbb{N}, n_1 = 2n_0, u \in \mathbb{N}^*\} \subseteq \mathbb{N}^*$ *is not non-deterministically grid recognizable.*

2) *The language* $L'_{\omega} := \{n_0 n_1 u \mid n_0, n_1 \in \mathbb{N}, n_1 = 2n_0, u \in \mathbb{N}^{\omega}\} \subseteq \mathbb{N}^{\omega}$ *is not non-deterministically ω-grid recognizable.*

Proof. Assume there is a Progressive Grid Word (Büchi) Automaton $\mathcal{A} = (Q, \Delta, q_0, F)$ that recognizes L' (L'_{ω}). Let $w = n_0 n_1 u \in L'$ ($\in L'_{\omega}$) with $n_0 > |Q|$, $n_1 = 2n_0$ and some $u \in \mathbb{N}^*$ ($\in \mathbb{N}^{\omega}$). Consider an accepting run π of $\mathcal{A}$ on w.

Case 1: If $(q, n_1, 2)$ is not in π for some $q \in Q$ then $\mathcal{A}$ accepts $n_0(n_1 - 1)u \notin L'$ ($\notin L'_{\omega}$) because the only difference of $\lambda(n_0 n_1 u)$ and $\lambda(n_0(n_1 - 1)u)$ is the label at position $(n_1, 2)$ which $\mathcal{A}$ does not read. This contradicts the assumption $\lightning$.

Case 2: Otherwise, there is a state repetition in π in a subsequence $\rho = (p, h_1, \ell) \dots (q, h_m, \ell)$ where $\ell \in \{1,2\}$ (that is, ρ is not induced by a move to the right) and $n_1 \geq h_i \geq n_0 + 1$ for all $1 \leq i \leq m$. Note that ρ is either induced only by zeros in the first column or by ones in the second column. It follows that $\mathcal{A}$ accepts $n_0 n_2 u \notin L'$ ($\notin L'_{\omega}$) for some $n_2 > n_1$ because ρ can be repeated several times. However, it is not sufficient to repeat ρ only a single time because $\mathcal{A}$ may check if n_2 is dividable by some $p \leq |Q|$. But the least common multiple with all possible p's and offsets is sufficient. Again this yields a contradiction $\lightning$.

Since none of the two cases are possible we conclude that L' is not non-deterministically grid recognizable as well as L'_{ω} is not non-deterministically ω-grid recognizable □

Proof of the theorem. Let $\mathfrak{N} := (\mathbb{N}, S)$ where S is the successor relation. Then lemma 6 states that $L_{\mathfrak{N}}$ and $L^{\omega}_{\mathfrak{N}}$ are not recognizable by a Strong Automaton and a Strong Büchi Automaton, respectively. But they are deterministically (ω-)grid recognizable (cf. example 4 and 6). Actually, this shows that even the deterministically (ω-)grid recognizable languages are not covered by Strong (Büchi) Automata.

On the other hand, the language L' (L'_{ω}) from lemma 7 is clearly recognizable by a Strong (Büchi) Automaton over $(\mathbb{N}, +)$ (cf. example 9). Furthermore, it is not non-deterministically (ω-)grid recognizable. □

The proof of theorem 12 utilizes a Strong (Büchi) Automaton over the structure $(\mathbb{N}, +)$ to show the existence of a language recognized by a Strong (Büchi) Automaton that is not non-deterministically (ω-)grid recognizable. But intuitively, a Progressive Grid Word (Büchi) Automaton can verify the successor relation S over $\mathbb{N}$. This yields two questions. Firstly, are languages recognized by Strong (Büchi) Automata over $(\mathbb{N}, S)$ with word dimension 1 non-deterministically (ω-)grid recognizable? And if this is the case, is there an extension or variant of Progressive Grid Word (Büchi) Automata that covers the languages recognized by Strong (Büchi) Automata over $(\mathbb{N}, +)$? We will consider the first question here. It is clear, that a Strong (Büchi) Automaton over $(\mathbb{N}, S)$ with look-back $s \geq 2$ can recognize the language $L(\mathcal{A}) = \{(0m)^n \mid m, n \in \mathbb{N}, n > 0\}$ (resp. $L(\mathcal{A}) = \{(0m)^{\omega} \mid m, n \in \mathbb{N}, n > 0\}$) which is not non-deterministically (ω-)grid recognizable (cf. the proof of theorem 5). For this purpose FO-logic suffices. However, the interesting case is $s < 2$ since then the emptiness problem for Strong (Büchi) Automaton is decidable.

Theorem 13. *Let $L \subseteq \mathbb{N}^*$ and $L_{\omega} \subseteq \mathbb{N}^{\omega}$. If L (L_{ω}) is recognized by a Strong (Büchi) Automaton with look-back ≤ 1 over $(\mathbb{N}, S)$ and MSO-logic then L (L_{ω}) is non-deterministically (ω-)grid recognizable.*

Proof. First of all, we can restrict the proof to Strong (Büchi) Automaton over $(\mathbb{N}, S)$ and Weak Monadic Second Order logic (WMSO) since every MSO formula over $(\mathbb{N}, S)$ can be rewritten to an equivalent WMSO formula (cf. [Tho80]). In WMSO the quantification of second order variables is restricted to finite sets while in MSO they may be infinite. With this restriction we can use the following well-known result (cf. [Tho90]):

Given a WMSO formula $\psi(x, y)$ over $(\mathbb{N}, S)$ there is a finite state automaton $\mathcal{A}_{\psi}$ over the alphabet $\{[0\ 0], [1\ 0], [0\ 1], [1\ 1]\}$ such that the following equivalence holds for all $n, m \in \mathbb{N}$:

$$(\mathbb{N}, S) \models_{\mathrm{WMSO}} \psi[n, m] \Leftrightarrow \text{ there is a } k \in \mathbb{N} \text{ such that}$$

$$\mathcal{A}_{\psi} \text{ accepts } \begin{cases} [0\ 0]^n\ [1\ 0]\,[0\ 0]^{m-n-1}\,[0\ 1]\,[0\ 0]^k & n < m, \\ [0\ 0]^m\,[0\ 1]\,[0\ 0]^{n-m-1}\,[1\ 0]\,[0\ 0]^k & n > m, \\ [0\ 0]^n\ [1\ 1]\,[0\ 0]^k & n = m. \end{cases} \quad (\star)$$

Moreover, there is a finite state automaton $\mathcal{A}^R_{\psi}$ that recognizes the reverse language of $\mathcal{A}_{\psi}$. That is, $L(\mathcal{A}^R_{\psi}) = \{u_{\ell} \dots u_1 \mid u_1 \dots u_{\ell} \in L(\mathcal{A}_{\psi})\}$.

Now, let $\mathfrak{A} = (P, \mathbb{N}, p_0, \Delta, F)$ be a Strong (Büchi) Automaton with look-back 1 over $(\mathbb{N}, S)$ and WMSO-logic. It suffices to show that $L(\mathfrak{A})$ is recognized by a Non-deterministic

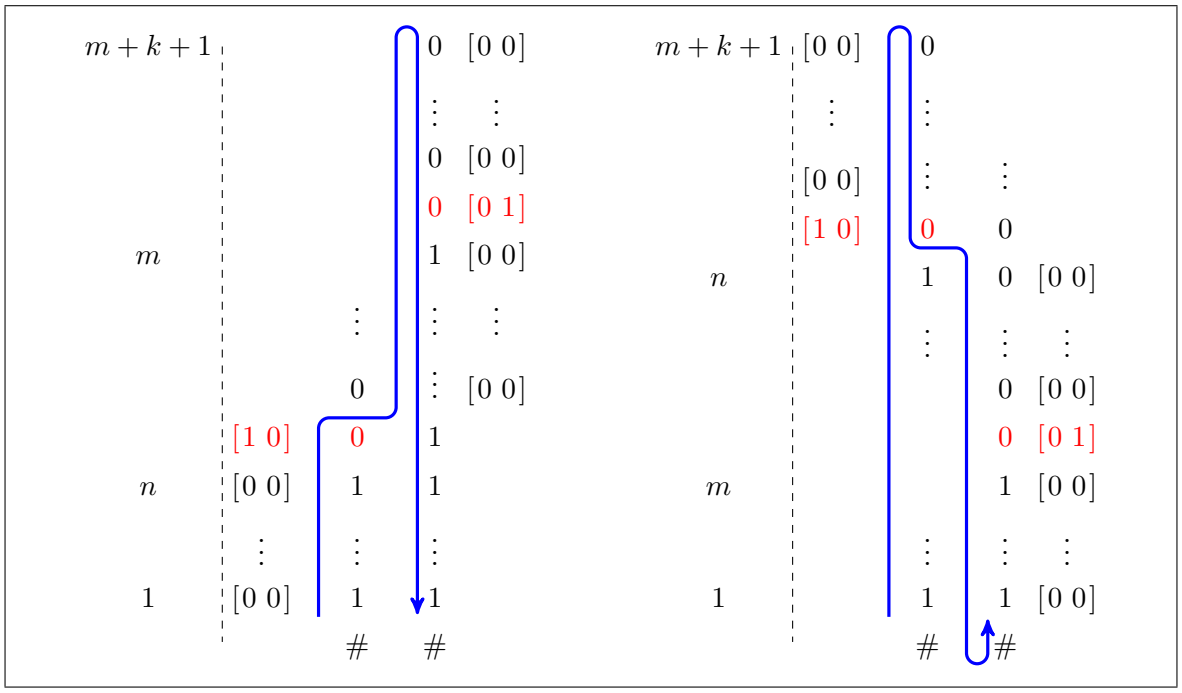


Figure 5.2: Illustration of the simulation of a $\mathcal{A}_\psi$ and $\mathcal{A}_\psi^R$ by $\mathcal{B}$ on the left and right hand side, respectively. The straight (blue) line illustrates the run of $\mathcal{B}$. $\mathcal{B}$ operates on the column on the right side of the line.

Progressive Grid Word (Büchi) Automaton $\mathcal{B}$. The basic idea of the construction of $\mathcal{B}$ is the simulation of the automata $\mathcal{A}_\psi$ as described above to verify that $\mathfrak{A}$ can take a transition labeled with ψ. Note that $\mathcal{B}$ operates on a (ω-)Grid Word where each column encodes the unary representation of a letter of an input word $w \in \mathbb{N}^* (\in \mathbb{N}^\omega)$. Besides, we will denote the label 0 at position (h, i) where $w(h', i) = 1$ for all $0 < h' < h$ as the *first* zero of the i-th column.

$\mathcal{B}$ will maintain the following condition: if $\mathfrak{A}$ reads the letter w_i $\mathcal{B}$ is at position $(1, i-1)$ on the Grid Word. That is, $\mathcal{B}$ trails $\mathfrak{A}$ by one letter. This is required to check if $\mathfrak{A}$ can take a transition ($\mathfrak{A}$ has look-back 1). Moreover, $\mathcal{B}$ can store the current state of $\mathfrak{A}$ in its own state space.

The essential part of the construction is the conversion of the transitions. Suppose that $\mathfrak{A}$ reads the i-th letter w_i of an input word $w \in \mathbb{N}^*$ $(\in \mathbb{N}^\omega)$ by taking a transition labeled with ψ. Then $\mathcal{B}$ guesses this transition of $\mathfrak{A}$. Let $n = w_{i-1}$ and $m = w_i$. $\mathcal{B}$ verifies that $\mathfrak{A}$ can take the guessed transition as follows:

Case 1: $n < m$. Starting at the position $(1, i-1)$ $\mathcal{B}$ simulates $\mathcal{A}_\psi$ on the $(i-1)$-th column (each move from $\mathcal{A}_\psi$ to the right corresponds a move from $\mathcal{B}$ upwards). Each input symbol is interpreted as [0 0] but the first zero is interpreted as [1 0]. The latter case corresponds to the fact that $\mathcal{B}$ has read the unary encoding of n. Hence, $\mathcal{A}_\psi$ should read a [1 0]. Afterwards, $\mathcal{B}$ moves to the right to the next column and interprets all symbols as [0 0] again while still simulating $\mathcal{A}_\psi$. Once it encounters the first zero of the second column $\mathcal{A}_\psi$ should read [0 1]. Finally, $\mathfrak{A}$ could have taken the transition if $\mathcal{B}$ can enter an accepting state of $\mathcal{A}_\psi$ after it has read [0 1]. In this case $\mathcal{B}$ returns to the position $(1, i)$ and starts

over. This situation is illustrated in figure 5.2 on the left hand side. Otherwise, when $\mathcal{A}_\psi$ does never accept $\mathcal{B}$ does not halt and hence, does not accept. This corresponds to the fact that $(\mathbb{N}, S) \not\models_{\text{WMSO}} \psi[n, m]$. That is, $\mathfrak{A}$ cannot take the transition.

Case 2: $n > m$. In this case $\mathcal{A}_\psi$ has to read $[0\ 1]$ first. This corresponds to first zero in the i-th column (cf. the second case of $(\star)$). But $\mathcal{B}$ has to read $[1\ 0]$ first because it cannot return to the $(i-1)$-th column to read $[1\ 0]$ correctly afterwards. However, $\mathcal{A}_\psi^R$ has to read $[1\ 0]$ at first as well. Hence, $\mathcal{B}$ can be constructed as follows. Firstly, it moves upwards to the position $(m + k + 1, i - 1)$. Then $\mathcal{B}$ can simulate $\mathcal{A}_\psi^R$ in the same manner as $\mathcal{A}_\psi$ but by moving downwards. The correct k can be guessed. Moreover, $\mathcal{B}$ can guess and verify if it reads the first zero as the next lower position is labeled with one. Note that in this case $\mathcal{A}_\psi^R$ have to be in an accepting state if $\mathcal{B}$ enters the position $(0, i)$. The simulation of $\mathcal{A}_\psi^R$ is depicted in figure 5.2 on the right hand side.

Case 3: $n = m$. Actually, this can be considered as a special sub case of the first case $n < m$. When $\mathcal{B}$ moves to the right the position is not labeled with a one but with the first zero. Clearly, $\mathcal{B}$ can verify this.

Note that $\mathcal{B}$ can guess and easily verify if $n < m$ (or $m > n$) while simulating $\mathcal{A}_\psi$ (or $\mathcal{A}_\psi^R$). The first transition is a special case but can be handled analogous (the formula has only one free variable). $\mathfrak{A}$ accepts ϵ if and only the initial state of $\mathfrak{A}$ is an accepting state. Hence, the initial state of $\mathcal{B}$ is an accepting if and only if the initial state of $\mathfrak{A}$ is. Eventually, if we consider the case $L \subseteq \mathbb{N}^*$ then $\mathcal{B}$ can guess if it is in the last column, verify and accept if $\mathfrak{A}$ is in an accepting state. This can be achieved by moving to the right and enter an accepting state. If $\mathcal{B}$ has guessed correctly, then it accepts because it is in an accepting state at position $(1, |w| + 1)$. Otherwise, the only possible action of $\mathcal{B}$ is to leave the accepting state, enter a sink state and reject. In the case $L \subseteq \mathbb{N}^\omega$ $\mathcal{B}$ enters and accepting state when it moves to the right if and only if $\mathfrak{A}$ will enter an accepting state by taking the transition that $\mathcal{B}$ guessed. Independent of this $\mathcal{B}$ proceeds with the simulation. Thus, the acceptance behavior from $\mathfrak{A}$ is inherited by $\mathcal{B}$.
Correctness: Let $v_1 \dots v_\ell \in \mathbb{N}^*$ (resp. $v_1 v_2 \dots \in \mathbb{N}^\omega$). Suppose $\mathfrak{A}$ is in state p and consumes $m := v_i$ for some $i > 1$ by taking a transition $(p, \psi(x, y), q) \in \Delta$. Moreover, assume $n := v_{i-1} < v_i = m$. Then $(\mathbb{N}, S) \models_{\text{WMSO}} \psi[n, m]$. Hence, there is a $k \in \mathbb{N}$ such that $\mathcal{A}_\psi$ accepts $[0\ 0]^n\,[1\ 0]\,[0\ 0]^{m-n-1}\,[0\ 1]\,[0\ 0]^k$ due to $(\star)$. Besides, $\mathcal{B}$ can guess this transition and simulate $\mathcal{A}_\psi$ starting at position $(1, i - 1)$ on the column $\lambda(n)$. Then $\mathcal{B}$ reads n ones which are interpreted as $[0\ 0]$ followed by the first zero interpreted as $[1\ 0]$. Afterwards, $\mathcal{B}$ moves to the right to the position $(n + 1, i)$. Since $n < m$ $\mathcal{B}$ reads $m - (n + 1)$ further ones and finally a zero again which are interpreted as $[0\ 0]$ and $[0\ 1]$ respectively. All in all, $\mathcal{B}$ simulates $\mathcal{A}_\psi$ on a word with the prefix $[0\ 0]^n\,[1\ 0]\,[0\ 0]^{m-n-1}\,[0\ 1]$ (cf. figure 5.2). Thus, if $\mathcal{A}$ continues the simulation (it feeds $\mathcal{A}_\psi$ only with $[0\ 0]$ symbols afterwards) then $\mathcal{A}_\psi$ enters an accepting state after k steps. Then $\mathcal{B}$ has verified the transition successfully, returns to the position $(1, i)$ and changes the internal state of $\mathfrak{A}$ to q. Note that $\mathcal{B}$ is in the correct starting position for the next transition.

On the other hand, if $\mathcal{B}$ can guess a transition labeled with ϑ and can simulate $\mathcal{A}_\vartheta$ successfully on the infix $v_{i-1}v_i = nm$ then it follows by a symmetric argument that there is a transition $(p, \vartheta(x, y), q)$ such that $(\mathbb{N}, S) \models_{\text{WMSO}} \vartheta[n, m]$.

The cases for $n > m$ and $n = m$ are symmetric. Recall that in the case $n > m$ $\mathcal{B}$ simulates $\mathcal{A}_\psi^R$ instead of $\mathcal{A}_\psi$ (cf. the illustration in figure 5.2 on the right hand side).

In conclusion, $\mathcal{B}$ can process a (ω-)Grid Word if and only if $\mathfrak{A}$ can process the corresponding word in $\mathbb{N}^*$ ($\mathbb{N}^\omega$). In particular, $\mathcal{B}$ can always track the internal state of $\mathfrak{A}$. Note that the case $w = \epsilon$ is handled in the construction (by adding the initial state to the set of accepting state, if necessary). Furthermore, the first letter can be similarly verified by $\mathcal{B}$. The only difference is that the formula of $\mathfrak{A}$ has only a single free variable and $\mathcal{B}$ does not need to move to the right.

Finally, we have already discussed that $\mathcal{B}$ can accept if and only if $\mathfrak{A}$ does as part of the construction. Note that in the special case of word length one, $\mathcal{B}$ has to verify the first transition before it is allowed to guess the end of the word. □

5.2 Comparison with Register Automata

The procedure of this section will be the same as for Strong Automata. We will use the notations and some results from Kaminski and Francez who established Register Automata as Finite-memory Automata in [KF94]. Register Automata should not be mixed up with register machines used in chapter 4. Note that we do not define a Büchi variant of Register Automata. Hence, this section is limited to finite words and Progressive Grid Word Automata, respectively.

Definition 17. Let $r \in \mathbb{N}_{>0}$ and Σ be an infinite alphabet. Moreover, let $\perp \notin \Sigma$. Then a Register Automaton with r registers is defined as a tuple $\mathcal{A} := (S, s_0, u, \rho, \Delta, F)$ where

- S is a finite set of states,
- $s_0 \in S$ is the initial state,
- $u = u_1 \dots u_r \in (\Sigma \cup \{\perp\})^r$ is the initial assignment of the registers,
- $\rho : S \to \{1, \dots, r\}$ is the reassignment which is a partial function in general,
- $\Delta \subseteq S \times \{1, \dots, r\} \times S$ is the transition relation,
- and $F \subseteq S$ is the set of accepting states.

A *configuration* of a Register Automaton $\mathcal{A}$ with r registers is an element in $Q \times (\Sigma \cup \{\perp\})^r$. The first component is the current internal state of $\mathcal{A}$ while the second component represents the values stored in the registers of $\mathcal{A}$. That is, given a $v \in (\Sigma \cup \{\perp\})^r$ the letter stored in the i-th register is v_i. $\mathcal{A}$ operates as follows on an input word $w = w_1 \dots w_n \in \Sigma^*$: The initial configuration is (s_0, u). Now, let (s, v) be the current configuration of $\mathcal{A}$ and $1 \leq i \leq n$ the current position on the input word. If $\rho(s)$ is defined then w_i is stored in the $\rho(s)$-th register regardless of the previous value. Afterwards, $\mathcal{A}$ choses a transition $(s, k, s') \in \Delta$ such that the value of the k-th register is w_i, enters the internal state s' and moves to the next position on the input word. If there is no proper transition $\mathcal{A}$ cannot proceed. Note that the order of the previous operations is important. All in all, the next configuration is (s', v') where $v' = v_1 \dots v_{\rho(s)-1} w_i v_{\rho(s)+1} \dots v_r$ if $\rho(s)$ is defined or otherwise $v' = v$. Eventually, $\mathcal{A}$ starts over and repeats the procedure with (s', v') at the next position until it has consumed the last letter w_n of w. If the last state $\mathcal{A}$ entered is an accepting state $\mathcal{A}$ *accepts* w. The sequence of configurations implied by the operation of $\mathcal{A}$ above is an *(accepting) run* of $\mathcal{A}$

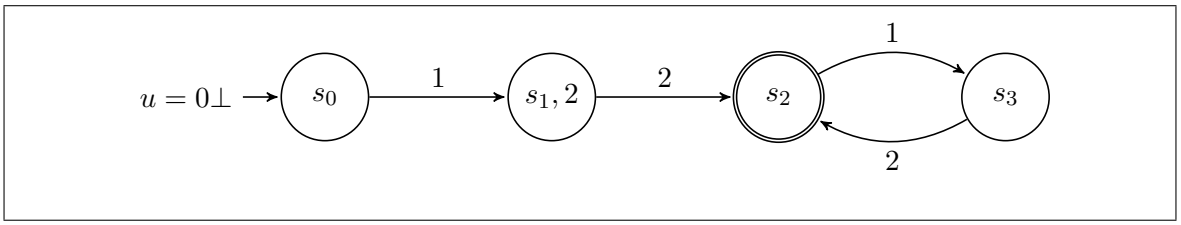


Figure 5.3: Graphical representation of a Register Automaton

on w. Moreover, $\mathcal{A}$ *recognizes* the language $L(\mathcal{A}) := \{w \in \Sigma^* \mid \mathcal{A} \text{ accepts } w\}$. Finally, a language recognized by a Register Automaton is called a *quasi-regular* language.

Example 10. Consider the Register Automaton $\mathcal{A}$ with two registers over the alphabet $\mathbb{N}$ depicted in figure 5.3. The initial assignment is $0\perp$. If $\rho(s)$ is defined then the state s is labeled $s, \rho(s)$. $\mathcal{A}$ operates as follows: Since there is no reassignment defined for s_0 the first letter has to be equal to $u_1 = 0$ to take the only transition $(s_0, 1, s_1)$. Afterwards, the second letter is stored in the second register ($\rho(s_1) = 2$) and the transition to state s_2 is the only choice. Once the automaton has entered state 2 the register values do not change any more and $\mathcal{A}$ can read the value stored in the first and in the second register alternately. All in all, we have that $L(\mathcal{A}) = \{(0m)^n \mid m, n \in \mathbb{N}, n > 0\} \subseteq \mathbb{N}^*$.

Like Strong Automata a Register Automaton can recognize languages over an uncountable alphabet. Again we will dismiss this and restrict the further comparison to countable alphabets.

Proposition 5. *Quasi-regular languages are closed under union, intersection and concatenation but not under complement.*

A proof of this proposition may be found in [KF94]. It is clear that the language of all words where some letter appears two times is quasi-regular. A Register Automaton may skip all letters with an auxiliary register but store non-deterministically a letter that appears again some time later in another register. But the complement is not quasi-regular because a Register Automaton would have to remember all letters it has read. Since the input words may have arbitrary length and the number of registers is fix this is not possible. This summarizes the proof approach for the claim that quasi-regular languages are not closed under complement. The very basic idea of the proof approach for the closure under union, intersection and concatenation is a power set construction as done in [RS59] for finite state automata. However, it is very technical and includes the introduction of an auxiliary model. Therefore, we will skip the further explanation of the idea here.

The non-closure under complement corresponds to the non-closure under complement we have for Non-deterministic Progressive Grid Word Automata. In fact, the language where a letter occurs twice is also non-deterministically grid recognizable. The automaton given in example 4 can easily be transformed to a Non-deterministic Progressive Grid Word Automaton that recognizes this language. Otherwise, Register Automata are still at an advantage over Progressive Grid Word Automata because quasi-regular languages are closed under intersection. The overview given in table 5.1 covers quasi-regular languages, too. We continue with the comparison of the decidability results. Once again we will focus on the emptiness problem.

Proposition 6. *The emptiness problem for Register Automata is decidable.*

This problem can be reduced to emptiness problem for finite state automata which is decidable (cf. [RS59]). The crucial point in the reduction is the following observation. If a Register Automaton with r registers accepts a word then it accepts a word of the same length that contains at most r different letters of the alphabet Σ. Hence, it suffices to consider a finite subset of Σ. The complete proof is presented in [KF94].

In the comparison with Strong Automata we stated that it suffices to compare a letter with its two direct predecessors to render the emptiness problem undecidable. Register Automata can clearly achieve this but the emptiness problem is still decidable. However, Register Automata can only compare letters by testing for equality while Progressive Grid Word Automata (and Strong Automata) can compare letters with respect to other relationships. For instance, a Progressive Grid Word Automata may verify the successor relation over $\mathbb{N}$. Thus, we can tighten our statement. The emptiness problem is rendered undecidable if the automaton model can compare a letter with its two direct predecessors but an equality check does not suffice. The ability of Register Automata to check for equality but not for other relationships between letters is used in the next theorem, too. In particular, it will show that the ability to remember a fixed number of letters is an advantage and on the other hand it is a disadvantage that the registers can only be used to check for equality.

Theorem 14. *The set of quasi-regular languages and the set of non-deterministically grid recognizable languages are incomparable.*

Proof. "$\not\subseteq$": The language $L = \{(0m)^n \mid m, n \in \mathbb{N}, n > 0\}$ is quasi-regular due to example 10. On the other hand we have shown in the proof of theorem 5 that L is not non-deterministically grid recognizable.
"$\not\supseteq$": Consider the language L_2 which consists of all words over $\mathbb{N}$ where the first symbol of a word is in the subset $2\mathbb{N} := \{2k \mid k \in \mathbb{N}\}$. Clearly, this language is deterministically grid recognizable (cf. the automaton described in example 5 as well as the modification of it in remark 2). Assume there is a Register Automaton $\mathcal{A}$ that recognizes L_2. Then $\mathcal{A}$ can test for equality of the first letter with some value in a register. Since $\mathcal{A}$ has only finitely many registers but the cardinalities of $2\mathbb{N}$ as well as $\mathbb{N} \setminus 2\mathbb{N}$ are infinite $\mathcal{A}$ either accepts a word where the first letter is not in $2\mathbb{N}$ or it rejects a word where the first letter is in $2\mathbb{N}$. In both cases this is contradiction. Note that it suffices to consider words of length one. Hence, it is irrelevant whether $\mathcal{A}$ stores the letter in some register. In conclusion, L_2 is not quasi-regular. □

Note that for the second direction we could have split any infinite but countable alphabet Σ in two infinite disjunct subsets. Besides, we used a Register Automaton with two registers (cf. example 10) to show the first inequality. This naturally raises the question if this result could be tightened by using a Register Automaton with only a single register. We finalize this chapter by showing that this is not possible.

Theorem 15. *Languages over a countable alphabet that are recognized by Register Automata with a single register are non-deterministically grid recognizable.*

Proof. We can restrict ourselves to the alphabet $\Sigma = \mathbb{N}$ since the elements of every infinite but countable alphabet can be identified with natural numbers. Let $\mathcal{A} = (S, s_0, u, \rho, \Delta, F)$

be a Register Automaton with a single register. A Non-deterministic Progressive Grid Word Automaton $\mathcal{P}$ that recognizes $L(\mathcal{A})$ operates as follows: Let $w = w_1 \dots w_n \in \mathbb{N}^*$. Recall that $\mathcal{P}$ operates on $\lambda(w)$. As usual $\mathcal{P}$ keeps track of the current state of $\mathcal{A}$ in its own state space. $\mathcal{P}$ will maintain the following circumstances. $\mathcal{A}$ starts to read the i-th letter of w and in the register the value k is stored if and only if $\mathcal{P}$ is at the position (k, i). Then $\mathcal{P}$ simulates $\mathcal{A}$ as follows:

1) To initialize the situation above $\mathcal{P}$ moves to the position $(u_1, 1)$ (initially $i = 1$).

2) If s is the current state of $\mathcal{A}$ and $\rho(s)$ is defined than $\mathcal{A}$ overwrites the register value with w_i. Hence, $\mathcal{P}$ moves to the position (w_i, i). It is clear that $\mathcal{P}$ can find this position since it is the position below the lowest position labeled with 0 in the current column.

3) The internal state of $\mathcal{A}$ is still s. Now, $\mathcal{A}$ may take an arbitrary transition (all transitions of $\mathcal{A}$ are labeled with 1) if and only the value stored in the register equals w_i. $\mathcal{P}$ can verify this condition easily by checking if the current position is the last one labeled with 1 or #. Or in other words, if the current position is labeled with 1 or # but the position above with 0. If $\mathcal{P}$ asserts that $\mathcal{A}$ can take a transition it choses one non-deterministically and moves to the right. Of course, it adapts the internal state of $\mathcal{A}$ and moves to an accepting state if and only if $\mathcal{A}$ does. Then $\mathcal{P}$ starts over with step 2). Finally, $\mathcal{P}$ rejects if $\mathcal{A}$ cannot take a transition.

The actions described above can clearly be done by an Non-deterministic Progressive Grid Word Automaton. Moreover, the construction is obviously correct. That is, $\lambda(L(\mathcal{A})) = L(\mathcal{P})$. Thus, we have that $L(\mathcal{A})$ is non-deterministically grid recognizable. □

6 Conclusion and Further Prospects

To sum up, we have properly introduced Progressive Grid Word Automata and grid recognizable languages. Furthermore, we have worked out the closure properties of deterministically and non-deterministically grid recognizable languages which turned out to be quite unsatisfactory. Especially, neither deterministically nor non-deterministically grid recognizable languages are closed under intersection. Besides, we have seen that Non-deterministic Progressive Grid Word Automata are strictly more expressive than Deterministic Progressive Grid Word Automata. Moreover, decidability results for the word, the emptiness and the universality problem have been presented. Additionally, we have proved that the emptiness problem is no longer decidable for General Grid Word Automata while the universality and the equivalence problem are not decidable for Non-deterministic Progressive Grid Word Automata. In the last chapter of this thesis we compared Progressive Grid Word Automata with Strong Automata and Register Automata. As a result, the set of grid recognizable languages is incomparable to the set of languages recognized by Strong Automata and the set of quasi-regular languages. In both cases we have identified proper subsets of those sets of languages hat are covered by Progressive Grid Word Automata. However, with respect to closure properties the Progressive Grid Word Automata are clearly at a disadvantage compared to the other models.

Further on, we have transfered nearly all our results to Progressive Grid Word Büchi Automata which recognize languages of infinite words or, more precisely, the ω-grid recognizable languages. The only exceptions are the comparison with a Büchi variant of Register Automata and the universality problem for Deterministic Progressive Grid Word Büchi Automata. The latter one could not be transferred because deterministically ω-grid recognizable are not closed under complement. In fact, this is the only property that is considered in this thesis and differs between ω-grid recognizable and grid recognizable as well as Grid Word Büchi Automata and Grid Word Automata.

All in all, we conclude that Progressive Grid Word (Büchi) Automata are a useful model with a decidable emptiness problem that is not covered by existing models but lacks good closure properties.

Apart from that the results of this thesis could be extended. For instance, all provided decidability results imply an effective decision procedure but we did not consider the complexity of these problems at all. Furthermore, we utilized MSO-logic to prove the decidability of the emptiness problem for Progressive Grid Word Automata. However, the capabilities of MSO-logic are only required to construct transitive closures of sets (e.g. to encode the sets T_c^k in a formula, cf. definition 11). This raises the question if it is possible to strengthen our result by proving the decidability of the emptiness problem with some kind of transitive closure logic that is less expressive than MSO-logic.

Another aspect is that our constructions and examples are huge in comparison with other models (cf. chapter 5). While we have noted in remark 2 that even a minimal Deterministic Progressive Grid Word Automaton is not unique (w.r.t. the recognized language) it is still

desirable to obtain a procedure that reduces the size of a given automaton.

Concerning the lack of closure properties an alternating variant of Progressive Grid Word (Büchi) Automata could be examined like it is done in [NSV04] for Register Automata. That is, we allow the automaton to use universal non-determinism in addition to the existential non-determinism we used here. Then the closure under intersection can be achieved as easy as the closure under union. Moreover, regarding languages of finite words it is reasonable that the languages recognized by those automata are closed under complement since the universal non-determinism is the "negation" of the existential one (at least if a language is recognized by an automaton that halts on every input). On the other hand this extension would probably result in the undecidability of the emptiness problem because the undecidability of the universality problem is inherited from the Non-deterministic Progressive Grid Word Automata. Since we suspect the closure under complement the universality problem could be reduced to the emptiness problem.

Finally, it could be allowed to label the "inner" positions of a (ω-)Grid Word with arbitrary symbols from a finite alphabet Σ instead of the single symbol 1. On the one hand the respective automata model may lose expressiveness since it loses the ability to know the symbols beneath the current position (e.g. if the label at the current position is 1 a Progressive Grid Word Automaton knows that all positions below are 1, too). On the contrary this may provide the ability to recognize languages over uncountable alphabets. Note that we have already addressed this extension in proposition 2 which claims that the emptiness problem is still decidable. Actually, this extension has already been defined by Blum and Hewitt in 1967 (cf. [BH67]) as automata on a two-dimensional (finite) input word. Rosenfeld modified their model later in 1979 (cf. [Ros79]) and yielded three-way automata which correspond to our Progressive Grid Word Automata. However, both papers study this model as an acceptor for two-dimensional words (also referred to as "pictures") but not as an acceptor for languages over infinite alphabets.

Bibliography

[Bès08] Alexis Bès. "An Application of the Feferman-Vaught Theorem to Automata and Logics for Words over an Infinite Alphabet". In: *Logical Methods in Computer Science* 4.8 (2008), pp. 1–23.

[BH67] Manuel Blum and Carl Hewitt. "Automata on a 2-dimensional tape". In: *Switching and Automata Theory, 1967. SWAT 1967. IEEE Conference Record of the Eighth Annual Symposium on*. IEEE. 1967, pp. 155–160.

[Büc62] J. Richard Büchi. "On a decision method in restricted second order arithmetic". In: *Logic, Methodology, and Philosophy of Science: Proceedings of the 1960 International Congress*. Ed. by Ernest Nagel. Stanford University Press, 1962, pp. 1–11.

[Czy13] Christopher Czyba. "Finite Automata with Infinite Structures as Alphabet: Closure Properties and Decidability Results". Bachelor Thesis. RWTH Aachen University, 2013.

[IT80] Katsushi Inoue and Itsuo Takanami. "A note on decision problems for three-way two-dimensional finite automata". In: *Information Processing Letters* 10.4 (1980), pp. 245–248.

[KF94] Michael Kaminski and Nissim Francez. "Finite-memory automata". In: *Theoretical Computer Science* 134.2 (1994), pp. 329–363.

[Min67] Marvin Lee Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall Englewood Cliffs, 1967.

[NSV04] Frank Neven, Thomas Schwentick, and Victor Vianu. "Finite State Machines for Strings over Infinite Alphabets". In: *ACM Transactions on Computational Logic* 5.3 (July 2004), pp. 403–435. ISSN: 1529-3785.

[Ros79] Azriel Rosenfeld. *Picture languages*. Academic Press, 1979. Chap. 3-5.

[RS59] Michael O Rabin and Dana Scott. "Finite automata and their decision problems". In: *IBM journal of research and development* 3.2 (1959), pp. 114–125.

[STW11] Alex Spelten, Wolfgang Thomas, and Sarah Winter. "Trees over Infinite Structures and Path Logics with Synchronization". In: *Proceedings 13th International Workshop on Verification of Infinite-State Systems, INFINITY 2011, Taipei, Taiwan, 10th October 2011*. Ed. by Fang Yu and Chao Wang. Vol. 73. EPTCS. 2011, pp. 20–34.

[Tho80] Wolfgang Thomas. "On the bounded monadic theory of well-ordered structures". In: *The Journal of Symbolic Logic* 45.02 (1980), pp. 334–338.

[Tho90] Wolfgang Thomas. "Automata on infinite objects". In: *Handbook of theoretical computer science*. Ed. by Jan van Leeuwen. Vol. B: Formal Models and Semantics. Elsevier Science Publishers, 1990, pp. 133–192.