University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Aleksander Wiśniewski
Student no. 451468

Sequences defined by weighted logics

Master's thesis
in COMPUTER SCIENCE

Supervisor:
dr Filip Mazowiecki

Warsaw, October 2024

Abstract

This work investigates sequences defined by unrestricted QMSO logic over a one-letter alphabet, introducing the concept of **1Q** sequences. We analyze their properties, focusing on their asymptotic growth and behavior modulo a number. Our key contributions include proving that 1Q sequences exhibit at most doubly exponential growth and that they are eventually periodic when considered modulo m. We also provide novel proof techniques to establish these results, differentiating our approach from existing methods based on weighted automata theory. Additionally, we highlight open questions and future directions for exploring the nature of 1Q sequences.

Keywords

automata theory, weighted logic, monadic second order logic, sequences

Thesis domain (Socrates-Erasmus subject area codes)

11.0 Matematyka, Informatyka

Subject classification

TODO

Tytuł pracy w języku polskim

Ciągi definiowane przez logiki ważone

Contents

Chapter 1

Introduction

It is an important result in computer science that the class of languages defined by finite automata coincides with the languages defined using MSO logic [3]. Finite automata allow us to recognize languages, i.e., to accept or reject words over a given alphabet. Schützenberger [11] extended the model of finite automata to make it possible to calculate quantitative properties of words. He introduced the model of weighted automata, which have richer semantics than finite automata. In weighted automata, transitions are supplied with weights (values from a semiring) and weights along a fixed run are multiplied using the semiring product. A value corresponding to a given input word is the semiring sum of values over all runs. Weighted automata calculate what we call a formal power series: a function from words to the semiring domain, $S : \Sigma^* \to K$.

Manfred Droste and Paul Gastin introduced a logic that coincides with weighted automata [6], in the same spirit as MSO logic coincides with finite automata. Their logic is called *weighted logic*, and its semantics allows defining formal power series like weighted automata do. Unfortunately, the full "natural" form of weighted logic is richer than weighted automata, so the authors had to restrict the class of weighted logic expressions in order to capture the same expressiveness as weighted automata. The restrictions are both at the syntactic and semantic levels.

A follow up work on weighted logics is the work of Stephan Kreutzer and Cristian Riveros [8]. In their work, they introduced QMSO logic - Quantitative Monadic Second Order logic, which fulfills similar goals to the weighted logics of Manfred Droste and Paul Gastin but with easier definitions and a clearer distinction between the logical level (MSO) and the semiring level (addition/multiplication). They also had to restrict their logic to achieve the same power of expression as weighted automata, but their restriction is only at the syntactic level.

In this work, we explore the world of unrestricted QMSO expressions. More precisely, we will look into the properties of formal power series defined by unrestricted QMSO over a one-letter alphabet. Assume our alphabet is $\Sigma = \{a\}$. In this case words can be identified with their length. Thus the formal power series $S : \Sigma^* \to K$ can be seen as a sequence $S : \mathbb{N} \to K$, by mapping words a^n to their length n. Restricted QMSO expressions coincide with weighted automata, and it is folklore that formal power series defined by weighted automata over a one-letter alphabet are exactly linear recursive sequences (see e.g. [2]). It can be easily shown that sequences achieved using unrestricted QMSO form a richer class than linear recursive sequences, which was already demonstrated in [4], but this class has not been investigated beyond that.

Our contributions In this work, we introduce the notation of **1Q** sequences - sequences defined by unrestricted QMSO expressions over a one-letter alphabet. We provide examples of such sequences, including 2^{2^n} and n^n.

In Chapter 3, we investigate the asymptotic behavior of 1Q sequences. The objective is to precisely characterize the class of functions that bound every 1Q sequence from above. Since the sequence 2^{2^n} is a 1Q-sequence, this class is at least doubly exponential. We prove that it is exactly doubly exponential and that every sequence of the form $c^{2^{kn}}$, where $c, k \in \mathbb{N}$, is 1Q-definable.

In Chapter 4, we explore the behavior of 1Q sequences taken modulo m, where $m \in \mathbb{N}$. It is common for complex classes of sequences to exhibit regular, easily characterizable behavior when considered modulo a number. This phenomenon is often observed with recursive sequences, but 1Q sequences are more "iterative" in nature than recursive.

The goal of Chapter 4 is to prove that 1Q sequences modulo are eventually periodic. Eventually periodic sequences exhibit periodic behavior from a certain point onward, continuing indefinitely. Many sequences modulo, such as linear recursive and polynomial recursive sequences, have this property. We prove that 1Q sequences modulo can be reduced to *simple recognizable step functions*, which are essentially mappings from regular languages to semiring values. A crucial aspect of this reduction is the elimination of semiring quantifiers. To achieve this, we employ techniques from finite automata theory, MSO, and the properties of modulo semirings. Specifically, we construct an automaton that can count the number of runs of another nondeterministic automaton modulo a given value.

Manfred Droste and Paul Gastin in [6] have already investigated QMSO-definable series over locally finite semirings, which include modulo semirings. Their results demonstrate that series definable in QMSO over locally finite semirings coincide with those definable by weighted automata over the same semirings. From this, it is possible to deduce that 1Q sequences are eventually periodic modulo $m \in \mathbb{N}$. However, we present different proof methods in this work, notably without introducing the notion of weighted automata.

In Chapter 5, we present additional research topics that were not fully explored but are worth investigating further to better understand the characteristics of 1Q sequences.

Related work In [4], a class of polynomial recursive sequences is defined. It is an extension of linear recursive sequences, and this class enables defining sequences with doubly exponential growth. In [4], it is shown that 1Q sequences can define a sequence that is undefinable as a polynomial recursive sequence: n^n. However, whether the converse inclusion holds is not clear, namely if every polynomial recursive sequence can be defined as a 1Q sequence?

In [12], the class of polynomial recursive sequences is characterised by pushdown automata of level 3.

There are other classes of sequences researched that extend linear recursive sequences, such as rational recursive sequences [5] or holonomic sequences [7]. However, these classes can define Catalan sequences, while 1Q sequences cannot.

Chapter 2

Preliminaries

In this section we present the syntax and semantics for Quantitative Monadic Second Order Logic (QMSO). It was introduced in [8] by Stephen Kreutzer and Cristian Riveros. Our presentation is based directly on this work. Then we will focus on 1Q sequences and provide some examples.

2.1. Linear Recursive Sequences

A sequence over a set $\mathbb{D}$ is a function $a : \mathbb{N} \to \mathbb{D}$.

Linear recursive sequences (LRS) are sequences satifsfying following recurrence relation:

$$a(n) = c_1 a(n-1) + c_2 a(n-2) + \ldots + c_k a(n-k),$$

where $c_1, \ldots, c_k$ are constants. In this work we focus mostly on natural numbers, so we assume that $c_1, \ldots, c_k \in \mathbb{N}$.

Example 2.1.1 (Fibonacci sequence)**.** One of the most poular linear recursive sequences is Fibonacci sequence, satisfying following reccurence:

$$a(n) = a(n-1) + a(n-2),$$

starting from $a(1) = 0$, $a(2) = 1$.

2.2. Monadic Second Order Logic

Let Γ be a finite alphabet. The syntax of MSO over a finite alphabet Γ is given by:

$$\varphi := P_a(x) \mid x \leq y \mid x \in X \mid (\varphi \vee \varphi) \mid \neg\varphi \mid \exists x.\varphi \mid \exists X.\varphi,$$

where: $a \in \Gamma$; x, y are first-order variables; and X is a second-order variable. Universal quantification can be obtained from existential quantification and negation. We can also use syntactic sugar $\wedge$ and $\implies$ as usual.

Let $w = w_1 \ldots w_n \in \Gamma^*$ be a word of length $|w| = n$. We represent w as a structure $(\{1, \ldots, n\}, \leq, (P_a)_{a \in \Gamma})$ where $P_a = \{i \mid w_i = a\}$. We denote by $Dom(w) = \{1, \ldots, n\}$ the domain of w as a structure. Given a finite set V of first-order and second-order variables, a (V, w)-assignment σ is a function that maps every first-order variable in V to $Dom(w)$ and every second-order variable in V to $2^{Dom(w)}$. Furthermore, we denote by $\sigma[x \to i]$ the extension of the (V, w)-assignment σ such that $\sigma[x \to i](x) = i$ and $\sigma[x \to i](y) = \sigma(y)$

for all variables $y \neq x$. The assignment $\sigma[X \to I]$, where X is a second-order variable and $I \subseteq Dom(w)$, is defined analogously. Consider an MSO-formula φ and a (V, w)-assignment σ where V is the set of free variables of φ. We write $(w, \sigma) \models \varphi$ if (w, σ) satisfies φ using the standard MSO-semantics.

2.3. Semirings

A semiring signature $\xi := (\oplus, \odot, 0, 1)$ is a tuple containing two binary function symbols $\oplus, \odot$, where $\oplus$ is called the addition and $\odot$ the multiplication, and two constant symbols 0 and 1. A semiring over the signature ξ is a ξ-structure $\mathbb{S} = (S, \oplus, \odot, 0, 1)$, where $(S, \odot, 0)$ is a commutative monoid, $(S, \odot, 1)$ is a monoid, multiplication distributes over addition, and $0 \odot s = s \odot 0 = 0$ for each $s \in S$. If the multiplication is commutative, then we say that $\mathbb{S}$ is commutative.

Semirings we will be interested in in this paper include:

- semiring of natural numbers: $\mathbb{N} = (\mathbb{N}, +, \cdot, 0, 1)$
- finite semirings of numbers modulo arbitrary k: $\mathbb{Z}_k = (\{0, 1, \ldots, k-1\}, +_{mod}, \cdot_{mod}, 0, 1)$ where operations are executed modulo k

We restrict mostly to sequences over natural numbers, although 1Q sequences can be defined over arbitrary semirings.

2.4. Quantitative Monadic Second-Order Logic

(Syntax) The formulas of Quantitative Monadic Second-Order Logic (QMSO) over a semiring $\mathbb{S}$ and a finite alphabet Γ (QMSO[$\mathbb{S}, \Gamma$]) are defined by the following grammar:

$$\theta := \varphi \mid s \mid (\theta \oplus \theta) \mid (\theta \odot \theta) \mid \Sigma_x\ \theta \mid \Pi_x\ \theta \mid \Sigma_X\ \theta \mid \Pi_X\ \theta,$$

where: $\varphi \in MSO[\leq, (P_a)_{a \in \Gamma}]$; $s \in \mathbb{S}$; x is first-order variable; and X is a second-order variable.

(Semantics) Let $w = w_1 \ldots w_n \in \Gamma^*$ where $n = |w|$. For the Boolean level φ, the semantics is the usual semantics of MSO, i.e. for any assignment σ,

$$[\![\varphi]\!](w, \sigma) = \begin{cases} 1 & \text{if } (w, \sigma) \models \varphi \\ 0 & \text{otherwise} \end{cases}$$

The semantics of the semiring level is defined as follows:

$$[\![s]\!](w, \sigma) := s$$
$$[\![(\theta_1 \oplus \theta_2)]\!](w, \sigma) := [\![\theta_1]\!](w, \sigma) \oplus [\![\theta_2]\!](w, \sigma)$$
$$[\![(\theta_1 \odot \theta_2)]\!](w, \sigma) := [\![\theta_1]\!](w, \sigma) \odot [\![\theta_2]\!](w, \sigma)$$
$$[\![\Sigma_x\ \theta]\!](w, \sigma) := \oplus_{i=1}^{n} [\![\theta]\!](w, \sigma[x \to i])$$
$$[\![\Pi_x\ \theta]\!](w, \sigma) := \odot_{i=1}^{n} [\![\theta]\!](w, \sigma[x \to i])$$
$$[\![\Sigma_X\ \theta]\!](w, \sigma) := \oplus_{I \subseteq [1,n]} [\![\theta]\!](w, \sigma[X \to I])$$
$$[\![\Pi_X\ \theta]\!](w, \sigma) := \odot_{I \subseteq [1,n]} [\![\theta]\!](w, \sigma[X \to I])$$

We will call sum (Σ) and product (Π) quantifiers *semiring quantifiers*, as opposed to quantifiers at MSO level $\exists$, $\forall$. Semiring quantifiers can be thought of in following way: we iterate over all valuations of variable bound by quantifier (first-order or second-order) and calculate value of inner expression for given valuation. Then we sum (for Σ) or multiply (for Π) those values.

Example 2.4.1. A simplest example of semiring quantifier usage is an expression that counts the number of occurrences of letter a in a word:

$$\Sigma_x \ P_a(x)$$

We iterate over all positions (x) in word, check if there is letter a on given position ($P_a(x)$). Inner expression returns 1 if there is, 0 otherwise.

2.5. QMSO over one letter alphabet

In this work, we focus on expressions of QMSO logic over a one-letter alphabet. We call these expressions *1Q expressions*. In this case, we only use a fragment of MSO logic; in particular, we do not need letter predicates $(P_a)_{a\in\Gamma}$, as there is only one such predicate and it is true for every element of the structure. For simplicity we assume that our one-letter alphabet is always $\Gamma = \{a\}$.

1Q expressions with no free variables (1Q sentences) generate sequences of numbers. Suppose we have a 1Q sentence φ, then we can generate corresponding sequence $a(n) = [\![\varphi]\!](a^n, \sigma)$, where σ is an empty valuation function (there are no free variables). The class of sequences definable 1Q expressions is called *1Q sequences*.

In [8, Section IV], Stephan Kreutzer and Cristian Riveros introduced a fragment of QMSO logic called Quantitative Iteration Logic (QIL). This syntatic fragment imposes restrictions on quantifier alternations: there can be an arbitrary number of Σ quantifiers used, and then Π quantifier over first-order variable. After Π there can only be quantifier-free expression. They prove following theorem, which will be an important result for us:

Theorem 2.5.1. A function $f : \Gamma^* \to S$ is definable by a weighted automaton over S and Γ iff f is definable by a formula in QIL, and this translation is effective.

We will not go into details of defining weighted automata. What is important is that it is folklore that formal power series defined by weighted automata over a one-letter alphabet are exactly linear recursive sequences (see e.g. [2]). From this follows that all LRS are 1Q-definable.

Below we present some examples of 1Q sequences.

Example 2.5.1 (Factorial)**.** The sequence $a(n) = n!$ can be defined with the following 1Q expression:

$$\Pi_{x_1}\Sigma_{x_2} \ (x_2 \leq x_1) \cdot 1.$$

For given n it works as following: x_1 iterates over $1, \ldots, n$. For fixed $x_1 = k$ the expression $\Sigma_{x_2} \ (x_2 \leq x_1) \cdot 1$ evaluates to k. So the whole expression evaluates to $1 \cdot 2 \cdot \ldots \cdot n = n!$.

Example 2.5.2 (Super exponential)**.** Similarly as in Example 2.5.1 the sequence $a(n) = n^n$ can be defined with the following 1Q expression:

$$\Pi_{x_1}\Sigma_{x_2} \ 1.$$

Example 2.5.3 (Doubly exponential)**.** The sequence $a(n) = 2^{2^n}$ can be defined with 1Q expression:

$$\Pi_{X_1}\ 2.$$

For a fixed n, there are 2^n possible choices of X_1. So 2 is multiplied 2^n times, which gives results in 2^{2^n}.

Chapter 3

1Q sequences rate of growth

Let $a(n)$ be a 1Q sequence over the natural numbers. We aim to find an upper bound for the asymptotic behavior of $a(n)$, depending only on n and constants appearing in the 1Q expression that defines it. We define the *rate of growth* of these sequences as the class of sequences that bound 1Q sequences. Specifically, a 1Q sequence $a(n)$ will have a rate of growth in a class of sequences B if $\exists_{b\in B}\exists_{N\in\mathbb{N}}\exists_{k\in\mathbb{N}}\forall_{n\geq N}\ a(n) \leq k \cdot b(n)$.

We start by discussing linear recursive sequences (LRS). Linear recursive sequences have an exponentially bounded rate of growth. More formally, if $a(n)$ is an LRS then $|a(n)| \leq 2^{O(n)}$.

Lemma 3.0.1. Let $a(n)$ be a linear recursive sequence. The sequence $|a(n)|$ can be bounded above by a function in the class $2^{O(n)}$.

Proof. Assume that the order of $a(n)$ is d. Every linear recursive sequence can be rewritten in matrix form, i.e. $a(n) = uA^nv$, where u is a $1 \times d$ vector, A is a $d \times d$ matrix, v is a $d \times 1$ vector.

Consider the absolute values of constants appearing in A. Let c be the largest of these values. Let B be a $d \times d$ matrix consisting only of the constant c. It is clear that:

$$a(n) \leq uB^nv.$$

We will prove that the values appearing in matrix B^n grow exponentially with n. From this, it follows that uB^nv is bounded above by a function in the class $2^{O(n)}$, which completes the proof.

We claim the following: every element of B^n is equal to $d^{n-1} \cdot c^n$. We will prove this by induction.

Base case: Every element of B^1 is equal to $d^0 \cdot c^1 = c$, which follows from definition.

Induction step: Assume that every element in B^{n-1} is equal to $d^{n-2} \cdot c^{n-1}$. We know that $B^n = B^{n-1} \cdot B$. The result follows from matrix multiplication: each element in B^n is given by $d^{n-2} \cdot c^{n-1} \cdot c + \ldots + d^{n-2} \cdot c^{n-1} \cdot c = d^{n-2} \cdot c^n + \ldots + d^{n-2} \cdot c^n$. The matrix has dimensions $d \times d$, so this simplifies to $d^{n-2} \cdot c^n \cdot d = d^{n-1} \cdot c^n$, which completes the proof. □

In Example 2.5.1, we observed that 1Q expressions can define the sequence $n!$ for which, by Stirling's approximation, we know that $n! = \Omega((\frac{n}{e})^n)$, exceeding the $2^{O(n)}$ bound for LRS. This shows that the class of 1Q sequences contains sequences not definable within LRS. As mentioned in the previous section (2.5.1), linear recursive sequences are a subset of 1Q-definable sequences. Therefore, it follows that the class of 1Q-definable sequences is strictly richer than LRS.

By Example 2.5.2 and Example 2.5.3, it is clear that it is possible to define 1Q sequences with a rate of growth even larger than $n!$. Our goal is to establish an upper bound for the

rate of growth of all 1Q sequences. Specifically, we will prove that the rate of growth of 1Q sequences is bounded by $2^{2^{O(n)}}$. We will use the notion of a *recognizable step function*, which is a power series defined for an arbitrary semiring (although we are primarily interested in natural numbers):

Definition 3.0.1 (Recognizable step function)**.** Fix a semiring K. A series $S : A^* \to K$ is a *recognizable step function*, if $S = \sum_{i=1}^{n} \varphi_{L_i} \cdot k_i$ (semiring sum) for some $n \in \mathbb{N}$, $k_i \in K$ and regular languages $L_i \subseteq A^*$ $(i = 1, \ldots, n)$. φ_{L_i} is an MSO formula recognizing language L_i.

First, we will prove a lemma, which will be useful to simplify QMSO expressions without semiring quantifiers.

Lemma 3.0.2. Let θ be a QMSO expression without semiring quantifiers. Then θ can be expressed as a recognizable step function.

Proof. We prove it by induction on the structure of expressions. First, consider base cases:

1. QMSO expression $\theta = \varphi$, where φ is a MSO expression. The corresponding recognizable step function is $\theta = \varphi \cdot 1$;

2. QMSO expression $\theta = k$, where $k \in K$. The corresponding recognizable step function is $\theta = \varphi \cdot k$, where φ recognizes the whole language (for example, $\varphi = \exists_x\ x = x$).

We proceed with the induction step. We consider two cases: addition and multiplication.

1. Let θ_1, θ_2 be recognizable step functions. Then $\theta_1 + \theta_2$ is a recognizable step function. It is immediate from definition.

2. Let $\theta_1 = \sum_{i=1}^{n_1} \varphi_{L_{1,i}} \cdot k_{1,i}, \theta_2 = \sum_{i=1}^{n_2} \varphi_{L_{2,i}} \cdot k_{2,i}$ be recognizable step functions. Then $\theta = \theta_1 \cdot \theta_2$ is a recognizable step function. Namely, θ is a sum of following expressions: $\varphi_{1,i} \cdot k_{1,i} \cdot \varphi_{2,j} \cdot k_{2,j}$. It can be rewritten in following way: $(\varphi_{1,i} \wedge \varphi_{2,i}) \cdot (k_{1,i} \cdot k_{2,j})$, from which it is immediate that θ is a recognizable step function.

This concludes the proof. □

Lemma 3.0.3 (1Q growth rate class)**.** The growth rate class of 1Q sequences is doubly exponential, i.e. $2^{2^{O(n)}}$. Moreover, for every $k, c \in \mathbb{N}$ it is possible to define a 1Q sequence such that $a_n = c^{2^{kn}}$.

Proof. Every sequence generated by a 1Q expression can be bounded by a sequence defined by a 1Q expression of the following form: $\Pi_{X_1}\Pi_{X_2}\ldots\Pi_{X_k}\ c$, where X_i are second-order quantifiers and $k \geq 0$ (for $k = 0$, there are no quantifiers, only constant c). We will prove this by induction on the structure of 1Q expressions.

For the base case, consider a quantifier-free expression φ. By Lemma 3.0.2, we know that such an expression can be simplified to a recognizable step function $\varphi = \sum_{i=1}^{n} \varphi_{L_i} \cdot k_i$, where each of φ_{L_i} evaluates to either 0 or 1. Set $c = \sum_{i=1}^{n} k_i$. Clearly, $\sum_{i=1}^{n} \varphi_{L_i} \cdot k_i \leq \sum_{i=1}^{n} k_i = c$. Additionally, let us assume that c is not smaller than 2, i.e., $c = max(2, \sum_{i=1}^{n} k_i)$. This assumption will be helpful in the subsequent steps.

Let $\varphi_1 = \Pi_{X'_1}\ldots\Pi_{X'_k}\ c_1$ and $\varphi_2 = \Pi_{X''_1}\ldots\Pi_{X''_k}\ c_2$ be two 1Q expressions. In general, an expression of the form $\Pi_{X_1}\ldots\Pi_{X_k}\ c$ defines a sequence $a(n) = c^{2^{nk}}$. Denote by $a_1(n)$ and $a_2(n)$ the sequences defined by the expressions φ_1 and φ_2, respectively. Then $a_1(n) = (c_1)^{2^{k'n}}$,

$a_2(n) = (c_2)^{2^{k''n}}$. The sequences defined by both $\varphi_1 + \varphi_2$ and $\varphi_1 \cdot \varphi_2$ can be bounded by a sequence $a_3(n)$ defined by a 1Q expression $\varphi_3 = \Pi_{X1'} \dots \Pi_{Xk'} \Pi_{X1''} \dots \Pi_{Xk''}\ c_1 \cdot c_2$:

$$a_1(n) + a_2(n) = (c_1)^{2^{k'n}} + (c_2)^{2^{k''n}} \leq (c_1 \cdot c_2)^{2^{(k'+k'')n}} = a_3(n),$$

and

$$a_1(n) \cdot a_3(n) = (c_1)^{2^{k'n}} \cdot (c_2)^{2^{k''n}} \leq (c_1 \cdot c_2)^{2^{(k'+k'')n}} = a_3(n),$$

this holds true assuming $c \geq 2$, as established in the induction base.

Let $\varphi = \Pi_{X_1} \dots \Pi_{X_k}\ c$ be a 1Q expression. The corresponding sequence is defined as follows: $a(n) = c^{2^{nk}}$. There are four possible types of quantification: Σ_x, Σ_X, Π_x, Π_X. Define the sequences corresponding to 1Q expressions using these quantifiers as follows:

$$a_1(n) = n \cdot c^{2^{nk}} \text{ for } \Sigma_x\ \varphi,$$

$$a_2(n) = 2^n \cdot c^{2^{nk}} \text{ for } \Sigma_X\ \varphi,$$

$$a_3(n) = c^{2^{nk} \cdot n} \text{ for } \Pi_x\ \varphi,$$

$$a_4(n) = c^{2^{n(k+1)}} \text{ for } \Pi_X\ \varphi.$$

It is clear that $c^{2^{n(k+1)}} \geq c^{2^{nk} \cdot n}$, $c^{2^{n(k+1)}} \geq 2^n \cdot c^{2^{nk}}$ and $c^{2^{n(k+1)}} \geq n \cdot c^{2^{nk}}$. From this, it follows that any quantifier can be replaced by a Π_X quantifier to achieve a sequence growing at least as fast. This concludes the proof by induction, as we have handled every form of QMSO expression.

From this, it follows that for an arbitrary 1Q expression Ψ, it is possible to create an expression Φ of the form:

$$\Pi_{X_1} \Pi_{X_2} \dots \Pi_{X_k}\ c,$$

for some constant c depending only on Ψ, such that the sequence generated by Ψ is bounded by the sequence generated by Φ. The sequence generated by Φ is precisely $a(n) = c^{2^{kn}}$, which has a doubly exponential rate of growth. This also shows that for arbitrary c and k, it is possible to achieve the sequence $c^{2^{kn}}$. □

Chapter 4

Eventual periodicity of 1Q sequences modulo

The behavior of 1Q sequences modulo can help characterizing these sequences and comparing them with other classes of sequences. In this section we will demonstrate that 1Q sequences, when taken modulo $m \in \mathbb{N} \setminus \{0\}$, are eventually periodic. As stated in Chapter 1, this result directly follows from the work of Manfred Droste and Paul Gastin [6], who demonstrated that QMSO expressions over finite semirings are equivalent to weighted automata. We use a different proof method, avoiding the need to introduce weighted automata.

Definition 4.0.1. A sequence $a(n)$ is *eventually periodic* if there exists $N, p \in \mathbb{N}$, $p > 0$, such that $\forall_{n>N}\ a(n) = a(n+p)$.

This property reveals a limitation of 1Q sequences modulo: they start behaving regularly after a certain point. This characteristic provides an effective method for determining whether a sequence is 1Q-definable. For instance, the sequence of Catalan numbers is not eventually periodic when considered modulo sufficiently large prime numbers, which will demonstrate that it is not 1Q-definable.

We will characterize 1Q sequences modulo as 1Q sequences over the modulo semiring:

Lemma 4.0.1. Fix a sequence $a(n)$ defined by 1Q expression φ over the natural semiring $(\mathbb{N}, +, \cdot, 0, 1)$. The sequence $a(n) \mod m$, where $m \in \mathbb{N} \setminus \{0\}$, can be defined by a 1Q expression φ', such that:

1. φ' is a 1Q expression over the semiring $\mathbb{Z}_m = (\{0, \ldots, m-1\}, +_{mod}, \cdot_{mod}, 0, 1)$,
2. every constant k appearing in φ is replaced by $k \mod m$ in φ'.

Proof. The semantics of 1Q expressions consist of addition, multiplication, repeated addition and repeated multiplication (repeated in case of semiring quantifiers). This lemma follows directly from the properties of operations modulo:

$$(a+b) \mod m = (a \mod m + b \mod m) \mod m,$$

$$(a \cdot b) \mod m = (a \mod m \cdot b \mod m) \mod m.$$

□

The goal of this section is to prove the following theorem:

Theorem 4.0.2. 1Q sequences are eventually periodic modulo every $m \in \mathbb{N} \setminus \{0\}$.

To prove this theorem, we aim to simplify the 1Q expressions defining these sequences into a form that is sufficiently straightforward for the remainder of the proof to follow naturally. Specifically, our goal is to reduce 1Q expressions over the modulo semiring to *simple recognizable step functions*. Such functions are eventually periodic modulo, as will be shown later.

Definition 4.0.2 (Simple recognizable step function)**.** A *simple recognizable step function* is a recognizable step function $S = \sum_{i=1}^{n} \varphi_{L_i} \cdot k_i$ in which languages $\{L_i : i \in \{1, \ldots, n\}\}$ form a partition of the whole language $\Sigma^* = \dot{\bigcup}_i L_i$.

Simple recognizable step functions are essentially a more explicit form of recognizable step functions; however, in essence, they represent the same functions:

Lemma 4.0.3. Recognizable step functions and simple recognizable step functions define the same classes of functions.

Proof. Suppose we have a recognizable step function $S = \sum_{i=1}^{n} \varphi_{L_i} \cdot k_i$. We will construct a simple recognizable step function that defines the same function. Let P be the powerset of $N = \{1, \ldots, n\}$. Then a simple recognizable step function can be defined as follows:

$$S = \sum_{I \in P} (\bigwedge_{i \in I} \varphi_{L_i} \bigwedge_{j \in (N-I)} \neg \varphi_{L_j}) \cdot (\sum_{i \in I} k_i).$$

Each expression $(\bigwedge_{i \in I} \varphi_{L_i} \bigwedge_{j \in (N-I)} \neg \varphi_{L_j})$ is true for exactly one $I \in P$ for given input w - it follows from the law of excluded middle. To calculate the value corresponding to this intersection, we sum the constants associated with the true expressions. This results in a simple recognizable step function that computes the same values as the original recognizable step function.

By definition, a simple recognizable step function is also a recognizable step function, so no further proof is required. □

4.1. Counting runs of automaton

We already know that semiring quantifier-free 1Q expressions are equivalent to recognizable step functions (Lemma 3.0.2). The complexity arises when we introduce semiring quantifiers into 1Q expressions, creating expressions like $\Pi_{X_1} \Sigma_{X_2} \varphi(X_1, X_2) \cdot 3 + \Sigma_{x_3} 2$, where $\varphi(X_1, X_2)$ is an MSO expression with two free variables. The primary focus of this section will be to demonstrate that when working modulo, these quantifiers can indeed be eliminated, resulting in a simple recognizable step function. To achieve this, we will employ finite automata constructions and leverage the equivalence between MSO expressions and finite automata to count the number of valuations for which an MSO expression φ with free variables holds true.

We will now prove a lemma concerning the construction of finite automata, which will be used later to count the number of valuations of free variables in MSO expressions. Let A be a nondeterministic finite automaton. Define the function $nRuns_A : \Sigma^* \to \mathbb{N}$ as the function that calculates the number of accepting runs of automaton A over words $w \in \Sigma^*$. Additionally, we will use the function $nRuns_A : \Sigma^* \times Q \to \mathbb{N}$, where Q is the set of states of automaton A. The function $nRuns_A(w, q)$ calculates the number of runs of automaton A over word w that finish in state q.

Lemma 4.1.1. Given a nondeterministic finite automaton A that recognizes the language L_A, and given natural numbers m and k with $k < m$, it is possible to construct a finite automaton

C (as for **c**ounting automaton) that recognizes the language $L_C := \{w : w \in L_A \wedge nRuns_A(w) \mod m = k\}$.

Proof. We will construct such an automaton C using an extended powerset construction.

Suppose the original automaton A has n states, $Q_A = \{q_1, \ldots, q_n\}$. States of the new automaton C, denoted Q_C, are functions $f : Q_A \to \{0, \ldots, m-1\}$. Since there are m^n such functions, the number of states in Q_C is m^n. For each state $q \in Q_C$, its corresponding function will be denoted f_q. The state $q \in Q_C$, represented by the function f_q, is interpreted as follows:

$$f_q(q_i) = nRuns_A(w, q_i) \mod m$$

Now, let us define the initial state, final states, and transitions of the automaton C.

The initial state of C is the function that maps the initial state of A to 1 and all other states to 0.

Let the set of final states of automaton A be F_A. For $q_C \in Q_C$, define $sumFin(q_C) = \sum_{q \in F_A} f_{q_C}(q)$. The function $sumFin$ returns the sum of the values of the function corresponding to the given state q_C over all final states of A. The final states of C, $F_C \subseteq Q_C$, are those states q_C for which $sumFin(q_C) = k \mod m$. Intuitively, the final states of C are the states where a word is accepted exactly $k \mod m$ times.

Lastly, we need to define the transitions of C. Let the set of transitions of the automaton A be δ_A and the set of transitions of automaton C be δ_C. Consider two states of C, q_C^1, q_C^2, with corresponding functions f_1 and f_2, respectively. The transition $q_C^1 \xrightarrow{a} q_C^2$ is in δ_C if:

$$f_2(q_i) = (\sum_{q \in Q_A} val(q)) \mod m,$$

where

$$val(q) = \begin{cases} f_1(q) & \text{if } q \xrightarrow{a} q_i \\ 0 & \text{otherwise} \end{cases}$$

Intuitively, we determine the number of runs ending in a given state q_i by summing over all states that have a transition to q_i on the given letter from the previous state.

Now, let us prove the correctness of this construction. We will demonstrate that after reading the input word w, the automaton C reaches a state q_C such that the function f_{q_C} correctly calculates the number of runs over the word w that end in each given state of automaton A (modulo m). From this and the construction of the final states of C, the correctness of the construction will be evident.

We will prove this by induction on the length of the word w. In the base case, consider the empty word ϵ. Automaton C accepts this word only if the following conditions are met:

- Some final state of A is also an initial state of A.
- $k = 1$.

This follows directly from the definitions of the initial and final states of C.

For the induction step, assume that the construction is correct for a word of length $n-1$, $w_1 \ldots w_{n-1}$. That is, after reading the first $n-1$ letters, we end up in a state $q_C \in Q_C$ represented by a function f_{q_C} that correctly assigns the number of runs (modulo m). We want to show that after reading the next letter (thus forming a word of length n), automaton C reaches a state that also correctly assigns numbers of runs.

For a given state $q \in Q_A$, to determine the number of runs finishing in q after reading the word $w = w_1 \ldots w_{n-1} w_n$, we must consider all the states $q_{prev} \in Q_A$ such that ($q_{prev} \xrightarrow{w_n}$

$q) \in \delta_A$ - that is, all the states from which q can be reached via the last letter in the word. By the induction hypothesis, we know the number of runs ending in those states q_{prev} after reading $w_1 \dots w_{n-1}$. Therefore, the number of runs finishing in state q after reading w can be expressed by the following formula:

$$(\sum_{q \in Q_A} val(q)) \mod m,$$

where

$$val(q) = \begin{cases} f_{q_C}(q) & \text{if } q \xrightarrow{a} q_i \\ 0 & \text{otherwise} \end{cases}$$

which is exactly how we defined transition function of C. □

4.2. Counting valuations of free variables in MSO expressions

To better understand the concept of counting valuations of free variables in an MSO expression, it is essential to specify the encoding of words accepted by MSO expressions with free variables. We will follow the method described in [8]. The "base" alphabet remains a single letter alphabet, $\Sigma = \{a\}$. Now, let φ be an MSO expression with a non-empty set of free variables (both first- and second-order) V. A word accepted by φ is a word over the alphabet $\Sigma_V = \Sigma \times \{0,1\}^{|V|}$.

Let w be a word over Σ, and let σ be a (V, w)-assignment. A word $(w, \sigma) \in \Sigma_V^*$ *encodes* a (V, w)-assignment if w is the projection of (w, σ) onto Σ, and for every variable $v \in V$, we have $\sigma(v) = \{i \in \{1, \dots, |w|\} \mid (w, \sigma)[v]_i = 1\}$, where $(w, \sigma)[v]_i$ denotes the i-th letter of the projection of (w, σ) onto variable v. For first-order variables $v_f \in V$, this set $\sigma(v_f)$ contains exactly one element, while for second-order variables, it may contain any number of positions in the word.

Example 4.2.1. Suppose we have an expression with one free first-order variable:

$$\varphi(y) = \forall x \; y \leq x.$$

In this case, y can only correspond to the first position in the word. The language recognized by this expression can be represented with the following regular expression:

$$\epsilon + \binom{a}{1} \binom{a}{0}^* .$$

In the following steps, it will be convenient to work with finite automata rather than MSO expressions. It is well-known ([3]) that for every MSO expression, there exists a finite automaton that accepts the same language, and conversely, for every finite automaton, there exists an equivalent MSO expression.

Example 4.2.2 (Fibonacci)**.** Consider the following expression with a free variable X (a set). This expression accepts only those sets where no two consecutive positions are included in the set:

$$\varphi(X) = \neg(\exists_{x_1} \exists_{x_2} \; x_1 \in X \wedge x_2 \in X \wedge succ(x_1, x_2)),$$

where $succ(x, y) = x > y \; \wedge \neg(\exists_z \; z > y \wedge z < x)$. The following word is not part of the language recognized by this expression:

$$\begin{pmatrix} a & a & a & a \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

This is because the set represented by the valuation of the free variable X (shown in the second row) contains two consecutive positions - specifically, positions 2 and 3.

The following word is part of the language recognized by this expression:

$$\begin{pmatrix} a & a & a & a \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

The interesting aspect of the property expressed by $\varphi(X)$ is that the number of valuations of X for a given word length n, where $n > 2$, is exactly $F(n+2)$, where F denotes the Fibonacci sequence. This result follows from a combinatorial interpretation of the Fibonacci sequence, specifically: $F(n+2)$ represents the number of all binary strings of length n that do not contain two consecutive ones.

An automaton recognizing the same language as $\varphi(X)$ is shown in Fig. 4.1.

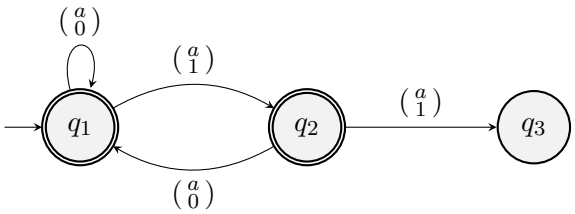

Figure 4.1: Deterministic automaton recognizing the same language as expression $\varphi(X)$

The method we intend to use to prove that 1Q sequences modulo are eventually periodic involves simplifying the 1Q expressions that define such sequences. The primary source of complexity in 1Q expressions is the use of semiring quantifiers. For instance, to calculate the values of a sequence defined by an expression like $\Sigma_X\ \varphi(X)$, it is necessary to determine how many valuations of X satisfy the expression $\varphi(X)$ for words of varying lengths $1, 2, \ldots$.

Fortunately, by manipulating the automata corresponding to expressions with free variables, such as $\varphi(X)$, we can construct nondeterministic automata that effectively "count" the number of valuations for which $\varphi(X)$ holds true.

Let φ_V be an MSO expression with a non-empty set of free variables $V = \{v_1, \ldots, v_n\}$. Let A_V be the corresponding deterministic finite automaton, i.e., the automaton that recognizes the same language as φ. Both φ_V and A_V recognize the language $L_V \subseteq \Sigma_V^*$. A word $(w, \sigma) \in L_V$ can equivalently be represented as $(w, \sigma(v_1), \ldots \sigma(v_n))$.

Our objective is to eliminate a variable $v \in V$ to obtain a language $L_{V\setminus\{v\}}$ - a projection that ignores the dimension corresponding to the variable v - and to count the number of valuations of v. Without loss of generality, assume that we want to remove the variable v_n.

Let $w_p = (w, \sigma(v_1), \ldots, \sigma(v_{n-1}))$ be a word in $L_{V\setminus\{v_n\}}$. We seek to determine $|w_p^{-1}| = |\{(w, \sigma(v_1), \ldots, \sigma(v_{n-1}), x) \in L_V \ : \ x \in \{0,1\}^*\}|$ - the cardinality of the inverse image of w_p under this projection. This cardinality corresponds to the number of valuations of v_n.

Rather than counting this number explicitly, we will construct a nondeterministic automaton whose number of runs corresponds to this count. By applying the method presented in Lemma 4.1.1, we can create automata that accept only languages of words for which the number of valuations modulo $m \in \mathbb{N} \setminus \{0\}$ is congruent to specified numbers. We will use this approach to eliminate semiring quantifiers from 1Q expressions later on.

Lemma 4.2.1. Let φ_V be an MSO expression with a non-empty set of free variables $V = \{v_1, \ldots, v_n\}$, and let A_V be its corresponding deterministic finite automaton recognizing the language $L_V \subseteq \Sigma_V^*$. Consider the projection of L_V onto $L_{V\setminus\{v_n\}}$.

It is possible to construct an automaton $A_{V\setminus\{v_n\}}$ that recognizes the language $L_{V\setminus\{v_n\}}$. Moreover, for any word $(w, \sigma) \in L_V$, the following holds:

$$|\{(w, \sigma(v_1), \ldots, \sigma(v_{n-1}), x) \in L_V \ : \ x \in \{0,1\}^*\}| = nRuns_{A_{V\setminus\{v_n\}}}((w, \sigma(v_1), \ldots, \sigma(v_{n-1})))$$

Proof. The construction of $A_{V\setminus\{v_n\}}$ will be based on A_V.

Consider a word $(w, \sigma(v_1), \ldots, \sigma(v_{n-1}), x) \in L_V$. The automaton $A_{V\setminus\{v_n\}}$ will ensure that for every possible valuation of x that keeps us in a language L_V (while keeping all other variables unchanged), there is a corresponding unique accepting run.

In order to achieve this, for every state $q \in Q_{A_V}$, there will be two corresponding states in $Q_{A_{V\setminus\{v_n\}}}$: q_0 and q_1. If q is an initial state in A_V, then q_0 will be an initial state in $A_{V\setminus\{v_n\}}$. If q is a final state in A_V, then both q_0 and q_1 will be final states in $A_{V\setminus\{v_n\}}$.

The transitions in $A_{V\setminus\{v_n\}}$ mimic those in A_V, with one key difference. For a transition $t \in \delta_{A_V}$ such that:

$$t = q_a \xrightarrow{(a, x_1, \ldots, x_{n-1}, 0)} q_b,$$

we create two corresponding transitions in $\delta_{A_{V\setminus\{v_n\}}}$:

$$t_0 = q_{a_0} \xrightarrow{(a, x_1, \ldots, x_{n-1})} q_{b_0},$$

$$t_1 = q_{a_1} \xrightarrow{(a, x_1, \ldots, x_{n-1})} q_{b_0}.$$

This means we can start in an arbitrary copy of q_a, but we can only end in copy of q_b corresponding to the zero valuation (q_{b_0}).

Similarly, for transition $t' \in \delta_{A_V}$ such that:

$$t' = q_a \xrightarrow{(a, x_1, \ldots, x_{n-1}, 1)} q_b,$$

we create two corresponding transitions in $\delta_{A_{V\setminus\{v_n\}}}$:

$$t'_0 = q_{a_0} \xrightarrow{(a, x_1, \ldots, x_{n-1})} q_{b_1},$$

$$t'_1 = q_{a_1} \xrightarrow{(a, x_1, \ldots, x_{n-1})} q_{b_1}.$$

This construction ensures that the number of runs in $A_{V\setminus\{v_n\}}$ for a given word $(w, \sigma(v_1), \ldots, \sigma(v_{n-1}))$ corresponds to the number of accepted valuations of x in $(w, \sigma(v_1), \ldots, \sigma(v_{n-1}), x) \in L_V$. Valuations of x correspond to sequences of subscripts of states; for example, a valuation $x = 0110$ corresponds to a run with subsequent states having subscripts $0, 1, 1, 0$. There can be exactly one such run in $A_{V\setminus\{v_n\}}$, which follows from the fact that A_V is deterministic and the transitions in $A_{V\setminus\{v_n\}}$ directly correspond to the transitions in A_V. □

Example 4.2.3 (Fibonacci, continued)**.** Let us continue working on Example 4.2.2. There, we considered the following expression with a free variable X:

$$\varphi(X) = \neg(\exists_{x_1} \exists_{x_2} \ x_1 \in X \wedge x_2 \in X \wedge succ(x_1, x_2))$$

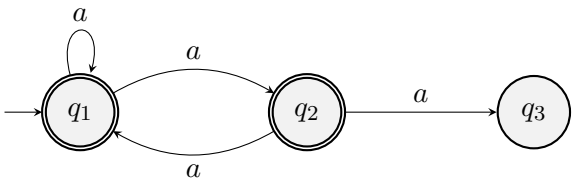

Figure 4.2: Nondeterministic "counting" automaton that skips variable X from $\varphi(X)$

We also constructed a deterministic automaton that recognizes a language over the extended alphabet $\Sigma_V = \{a\} \times \{0, 1\}$. From Lemma 4.2.1, we know that it is possible to build an automaton, potentially nondeterministic, that ignores one of the free variables, with its number of runs corresponding to the number of valuations of the ignored variable. In this case, this results in the automaton presented in Fig. 4.2.

From the explanation in Example 4.2.2 regarding the correspondence between $\varphi(X)$ and Fibonacci numbers, and the results in Lemma 4.2.1, it follows that the number of accepting runs of the automaton shown in Fig. 4.2 for a word of length n (i.e., a^n) should be equal $F(n+2)$. This is indeed the case and can be proven by induction.

4.3. Semiring quantifiers elimination

The primary complexity in simplifying 1Q expressions, as stated at the beginning of this section, lies in handling semiring quantifiers. In this subsection, we will demonstrate how to eliminate these quantifiers while maintaining the simplicity of the expressions. The following lemmas are crucial for the elimination of semiring quantifiers in 1Q expressions:

Lemma 4.3.1. Let $\mathbb{Z}_m$ be a modulo semiring, and let φ be a simple recognizable step function on this semiring. Then $\Sigma_x\ \varphi$ and $\Sigma_X\ \varphi$ are also simple recognizable step functions.

Proof. Since φ is a simple recognizable step function, it can be expressed as $\varphi = \sum_{i=1}^{n} \varphi_{L_i} \cdot k_i$, where φ_{L_i} are MSO expressions and $k_i \in \mathbb{Z}_m$. Now, consider the expression $\Sigma_X\ \varphi$, where X is either a first-order or a second-order variable. The reasoning is the same regardless of whether we work with first-order or second-order variables, so we will handle both simultaneously. Some of the MSO expressions inside φ might contain X as a free variable, while others might not. Nevertheless, in the semantics of QMSO, Σ will iterate over all valuations of X to evaluate this expression. We can assume that every MSO expression inside φ contains X as a free variable, by simply concatenating every expression with the always true expression $X = X$.

Now, for each MSO expression φ_{L_i} that appears within φ, assign a deterministic finite automaton that recognizes the language L_i. These MSO expressions include the free variable X, so the corresponding automata are defined over an extended alphabet Σ_V. By having deterministic automata over this extended alphabet, we can apply Lemma 4.2.1 to construct possibly nondeterministic "counting" automata A_{L_i} over alphabet $\Sigma_{V \setminus \{X\}}$. The number of runs of these new automata corresponds to the number of accepting valuations of free variable X.

To handle addition - the semiring quantifier Σ - it is sufficient to determine how many times (modulo m) each constant k_i appears. To do this, count the number of runs (valuations) for the automaton A_{L_i} using Lemma 4.1.1. For each A_{L_i} and every $k < m$, create an automaton represented by the MSO expression $\varphi_{L_i}^k$ that accepts words over the alphabet $\Sigma_{V \setminus \{X\}}$ for

which the number of accepting runs in the original automaton was equal to $k \mod m$. With these automata in place, the addition quantifiers can be eliminated as follows:

$$\Sigma_X \ \varphi = \sum_{i=1}^{n} \sum_{j=0}^{m-1} \ \varphi_{L_i}^j \cdot (j \cdot k_i \mod m).$$

$\varphi_{L_i}^j$ can be true for at most one j, $0 \leq j \leq m-1$. This j represents the number of successful runs modulo m, meaning the number of valuations of X for which φ_{L_i} was true. In the original expression, k_i would be added for each such valuation, resulting in $j \cdot k_i$ for this part of the expression. Since we are working modulo m, each of these values is taken modulo m. □

Lemma 4.3.2. Let $\mathbb{Z}_m$ be a modulo semiring, and let φ be a simple recognizable step function over this semiring. Then $\Pi_x \ \varphi$ and $\Pi_X \ \varphi$ are also simple recognizable step functions.

Proof. Since φ is a simple recognizable step function, it can be expressed as $\varphi = \sum_{i=1}^{n} \ \varphi_{L_i} \cdot k_i$, where φ_{L_i} are MSO expressions and $k_i \in \mathbb{Z}_m$. The proof for this lemma will use reasoning similar to that in Lemma 4.3.1, but with additional complexity. Again, we need to determine how many times each constant k_i appears across all valuations of the variable bound by the quantifier. However, unlike the addition case, it is insufficient to know how many times this constant appears modulo m, because in the case of exponentiation, different numbers have different periods when taken modulo m.

For example, consider the constant 2 when working modulo 3. We have $2^1 \mod 3 = 2$, $2^2 \mod 3 = 1$, $2^3 \mod 3 = 2$, and so on. For odd exponents, the value will always be 2, and for even exponents, it will be 1. Therefore, in this case, we need to know the number of valuations modulo 2 rather than modulo 3.

The situation is also different for constants that are roots of any order of m. Suppose we are working modulo $m = 16$ and have a constant $k_i = 2$. In this case, $2^1 \mod 16 = 2$, $2^2 \mod 16 = 4$, $2^3 \mod 16 = 8$, $2^4 \mod 16 = 0$, $2^5 \mod 16 = 0$, and so on. Thus, after reaching 2^4, for all subsequent exponents, the value will be 0. In this case, we observe a prefix sequence $2, 4, 8$, followed by a constant sequence of 0.

The general case is as follows. Consider a constant k_i, which we exponentiate modulo m. The sequence $Exp_{k_i} = (k_i)^1 \mod m, (k_i)^2 \mod m, (k_i)^3 \mod m \ldots$ is eventually periodic. This follows from the fact that this sequence can take on at most m different values, and once $(k_i)^{n_i} \mod m = (k_j)^{n_j} \mod m$, it implies that $(k_i)^{n_i+1} \mod m = (k_j)^{n_j+1} \mod m$ (i.e., once a cycle is reached, it cannot be escaped). Since this sequence is eventually periodic, there exist $N_i, p_i \in \mathbb{N}$, with $p_i > 0$ such that $\forall_{n>N_i} \ Exp_{k_i}(n) = Exp_{k_i}(n + p_i)$. Thus, there is a prefix of length N_i (which may be empty if $N_i = 0$) followed by a repeating cycle of length p_i.

To determine the influence of the constant k_i on the entire expression, we need to introduce $N_i + p_i$ new constants and MSO expressions. First, for $0 < j \leq N_i$, introduce a constant $k_i^j = (k_i)^j \mod m$ and an MSO expression $\varphi_{L_i}^j$ that asserts the existence of exactly j different valuations of X for which φ_{L_i} is true, i.e., $\varphi_{L_i}^j = \exists^{=j} \ \varphi_{L_i}$. This handles the prefix of length N_i.

Next, for $N_i < j \leq N_i + p_i$, introduce a constant $k_i^j = (k_i)^j \mod m$ and an MSO expression $\varphi_{L_i}^j$ that asserts there are more than N_i different valuations of X for which φ_{L_i} is true, with the number of these valuations being congurent to $j \mod p_i$. For the second property, use the automaton that counts the number of runs modulo, as described in Lemma 4.1.1, in the same way we did in the addition case in Lemma 4.3.1.

We also need to address the special case when the constant k_i has no influence on the overall expression, that is, when φ_{L_i} is false for every valuation of X. To handle this case, introduce $k_i^0 = 1$ (neutral element of multiplication) and $\varphi_{L_i}^0 = \neg(\exists_X\ \varphi_{L_i})$.

From the construction described, the value corresponding to a constant k_i in the original expression, after quantifier elimination, is given by $\sum_{j=0}^{N_i+p_i}\ (\varphi_{L_i}^j \cdot k_i^j)$. The inner MSO expression $\varphi_{L_i}^j$ will be true for exactly one j.

Now, the multiplication quantifiers can be removed in the following way:

$$\Pi_X\ \varphi = \prod_{i=1}^{n} \sum_{j=0}^{N_i+p_i}\ \varphi_{L_i}^j \cdot k_i^j.$$

□

4.4. Final steps

Finally, we can characterize 1Q sequences over a modulo semiring as simple recognizable step functions:

Lemma 4.4.1. Let $\mathbb{Z}_m$ be a modulo semiring, and let φ be a 1Q sentence over $\mathbb{Z}_m$. Then the sequence defined by φ can be expressed as a simple recognizable step function (i.e., φ can be reduced to the form of a simple recognizable step function).

Proof. We proceed by induction on the structure of the expression φ. For the base case - expressions without quantification at the semiring level - this follows from Lemma 3.0.2 and Lemma 4.0.3.

The sum and product of two simple recognizable step functions are also simple recognizable step function - this follows from the induction step of proof of Definition 3.0.1 and from Lemma 4.0.3.

Consider an expression $\varphi = Q_Y\ \theta$, where θ is a simple recognizable step function, Q is a semiring quantifier and Y is either a first-order or second-order variable. By lemmas 4.3.1 and 4.3.2, it follows that φ is a simple recognizable step function. This completes the proof. □

From these results, an important characterization of 1Q sequences over modulo semirings follows: 1Q sequences over modulo semirings are eventually periodic.

Lemma 4.4.2. Let $\mathbb{Z}_m$ be a modulo semiring and let φ be a 1Q sentence over $\mathbb{Z}_m$. Then the sequence defined by φ is eventually periodic.

Proof. Thanks to Lemma 4.4.1, we know that φ reduces to a simple recognizable step function. It can be easily seen that the language of words for which the sequence defined by φ achieves a given value k_i is a regular language - it follows directly from the definition of a simple recognizable step function. Since there is a finite number of values this sequence can achieve (at most m). Regular languages on one-letter alphabets are eventually periodic with regard to word length ([10, Theorem 1]).

There are m regular, disjoint languages $L_1, \ldots, L_m$ corresponding to different constants k_i. We assume these languages do not accept the empty word. Each language L_i is eventually periodic with respect to word length, so there exist $N_i, p_i \in \mathbb{N}$ such that $\forall_{n \geq N_i}\ a^n \in L_i \iff a^{n+p} \in L_i$. To find a "global" period that works for all langauges, take $N = \prod_{i=1}^{m}\ N_i$ and $p = \prod_{i=1}^{m}\ p_i$. This period serves as a common period for every L_i since N is a multiple of each N_i and p is a multiple of each p_i. □

From this, the main result of this section, Theorem 4.0.2, follows directly

Proof of Theorem 4.0.2. 1Q sequences modulo are equivalent to 1Q sequences over a modulo semiring (Lemma 4.0.1). Thus, the result follows directly from Lemma 4.4.2. □

Corollary 4.4.1. Catalan numbers are not 1Q-definable

Proof. The reasoning is the same as in [4, Theorem 7, Corollary 8], which uses results about Catalan sequences from [1]. Catalan numbers are not ultimately periodic modulo sufficiently large prime numbers, while 1Q-definable sequences are. Therefore, Catalan numbers are not 1Q-definable. □

Chapter 5

Other Topics of Interest

During the research on 1Q sequences, several other interesting topics emerged, though they were not fully explored.

5.1. Characterization of 1Q sequences

1Q expressions a broad class of sequences, notably including the entire class of linear recursive sequences. An interesting research question is: what precisely is the class of sequences defined by 1Q expressions? Is there a simpler or alternative representation of these sequences?

In the Introduction, it was noted that 1Q expressions can define sequences that are not polynomial recursive. This raises an interesting question: is every polynomial recursive sequence also 1Q-definable? This question is challenging to answer, as expressing recursion using 1Q expressions is not straightforward.

We formulate the following hypothesis: the composition of two Fibonacci sequences is not a 1Q sequence. While the Fibonacci sequence itself is a linear recursive sequence, and hence definable as a 1Q sequence, it seems unlikely that the composition of Fibonacci sequences can be expressed as a 1Q sequence. We denote the composition of two Fibonacci sequences by $F_F(n)$, defined by $F_F(n) = (F \circ F)(n)$, where F is the standard Fibonacci sequence.

We will show that sequence $F_F(n)$ is a polynomial recursive sequence. Consequently, proving that it is not a 1Q sequence would imply that 1Q sequences do not encompass all polynomial recursive sequences. This observation was pointed to us by Geraud Senizurgues [12]. However, we were unable to extract neither the statement nor the proof of this claim. Below we provide a simple proof. Similar reasoning can be used to prove that composition of arbitrary linear recursive sequences over natural numbers is polynomial recursive.

For an introduction to polynomial recursive sequences and the necessary definitions, refer to the work in: [4].

Lemma 5.1.1. The sequence $F_F(n)$ (composition of Fibonacci sequences) is a polynomial recursive sequence.

Proof. We will use the following formulation of the Fibonacci sequence, utilizing matrix exponentiation:

$$F(n) = \begin{pmatrix} 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

From which directly follows that:

$$F_F(n) = \begin{pmatrix} 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{F(n)} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

which can be decomposed into following, recursive definition, based on unfolding of $F(n)$:

$$F_F(n) = \begin{pmatrix} 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{F(n-1)} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{F(n-2)} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Our goal is to generate sequences for the entries in these matrices. To compute new values for these sequences, we require recursive formulas to execute matrix multiplication. Fortunately, this can be achieved using polynomial recursive sequences.

We define the sequences in following way:

$$\begin{pmatrix} a(n) & b(n) \\ c(n) & d(n) \end{pmatrix} = \begin{pmatrix} a(n-1) & b(n-1) \\ c(n-1) & d(n-1) \end{pmatrix} \cdot \begin{pmatrix} x(n-1) & y(n-1) \\ z(n-1) & k(n-1) \end{pmatrix},$$

$$\begin{pmatrix} x(n) & y(n) \\ z(n) & k(n) \end{pmatrix} = \begin{pmatrix} a(n-1) & b(n-1) \\ c(n-1) & d(n-1) \end{pmatrix}.$$

It can be seen that $c(n)$ is exactly the element we chose as $F_F(n)$. Initial value of these sequences will follow initial values of "base" matrices:

$$\begin{pmatrix} a(1) & b(1) \\ c(1) & d(1) \end{pmatrix} = \begin{pmatrix} a(2) & b(2) \\ c(2) & d(2) \end{pmatrix} = \begin{pmatrix} x(1) & y(1) \\ z(1) & k(1) \end{pmatrix} = \begin{pmatrix} x(2) & y(2) \\ z(2) & k(2) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^1.$$

We wish for following property of sequences a, b, c, d, for $n \geq 2$:

$$\begin{pmatrix} a(n) & b(n) \\ c(n) & d(n) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{F(n)},$$

and following property of sequences x, y, z, k, for $n \geq 2$:

$$\begin{pmatrix} x(n) & y(n) \\ z(n) & k(n) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{F(n-1)}.$$

Let us prove by induction that it is indeed the case. For base case $n = 2$, we have by definition:

$$\begin{pmatrix} a(2) & b(2) \\ c(2) & d(2) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{F(2)},$$

$$\begin{pmatrix} x(2) & y(2) \\ z(2) & k(2) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{F(1)}.$$

Now, assume that this property holds for $k-1$, where $k-1 \geq 2$. Let us prove it for k. By definition:

$$\begin{pmatrix} a(k) & b(k) \\ c(k) & d(k) \end{pmatrix} = \begin{pmatrix} a(k-1) & b(k-1) \\ c(k-1) & d(k-1) \end{pmatrix} \cdot \begin{pmatrix} x(k-1) & y(k-1) \\ z(k-1) & k(k-1) \end{pmatrix},$$

by induction assumption:

$$\begin{pmatrix} a(k) & b(k) \\ c(k) & d(k) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{F(k-1)} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{F(k-2)},$$

which finishes the proof for a, b, c, d. Now, for x, y, z, k, by definition:

$$\begin{pmatrix} x(k) & y(k) \\ z(k) & k(k) \end{pmatrix} = \begin{pmatrix} a(k-1) & b(k-1) \\ c(k-1) & d(k-1) \end{pmatrix},$$

by induction assumption:

$$\begin{pmatrix} x(k) & y(k) \\ z(k) & k(k) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{F(k-1)},$$

which finishes the proof for x, y, z, k.

At last, $F_F(n)$ can be defined as:

$$F_F(n) = c(n).$$

□

5.2. Complexity of Zero Checking

When working over a modulo semiring, what is the time complexity $O(|\varphi|)$ of checking whether a sequence defined by such a sentence is constantly equal to zero?

Consider a naive zero-checking procedure that simply iterates over the first n elements of the sequence defined by φ and checks whether they are all zero. How many elements must be checked to ensure that the sequence is constantly equal to zero? This problem can also be phrased as: how far into the sequence can the first non-zero element appear, relative to the length of the 1Q expression?

It turns out the problem of zero checking has super-exponential complexity, even without the use of semiring quantifiers. This complexity arises from the expressive power of MSO. As stated in [13], [9], the decision problem for the weak monadic second order logic of successor is not elementary-recursive. This is true when considering arbitrary alphabets, although the proofs primarily focus on binary alphabets. While we work with a unary alphabet, the following transformation allows us to construct equisatisfiable sentences over a unary alphabet from sentences over other alphabets:

- Let φ be a sentence over a non-unary alphabet. Without loss of generality, let this alphabet be binary, $\Sigma = \{a, b\}$.
- Let $Part(X, Y) = \forall_x\ (x \in X \vee x \in Y) \wedge \neg(x \in X \wedge x \in Y)$ state that the sets X and Y form a partition of the whole model.
- An equisatisfiable sentence for φ over a unary alphabet can be constructed as follows. For every letter in the alphabet Σ, we create a corresponding set variable. In this case, we introduce two set variables A and B, corresponding to the letters a and b, respectively. The final sentence is $\exists_A \exists_B\ Part(A, B) \wedge \varphi'$, where every appearance of the letter predicate $P_a(x)$ in φ is replaced by expression $x \in A$ in φ', and similarly for the letter b. This sentence asserts that we can partition the set of positions into those containing the letter a and those containing the letter b, such that φ holds.
- This construction can be easily generalized to alphabets of arbitrary size.

5.3. 1Q on first-order logic

The task of characterizing 1Q sequences is non-trivial, but it may be interesting to look into a restriction of 1Q expressions where we only use first-order variables at both the semiring and logical level (i.e., moving from monadic second-order logic to first-order logic).

Unfortunately, this class of sequences also appears challenging to characterize, and it is not obvious how to determine whether a given sequence is definable as a 1Q sentence in first-order logic.

Instead, it might be worthwhile to revisit sequences modulo, as we did in Chapter 4.

As stated in [6], first-order definable series coincide with series definable by weighted automata when working with aperiodic semirings. Unfortunately, the semirings we are working with - modulo semirings - are not aperiodic. This can be illustrated by the sequence defined by the following expression:

$$\Sigma_x \ . \ 1$$

This defines the following sequence modulo 2: $\{1, 0, 1, 0, 1, \ldots\}$, which is not aperiodic. An interesting aspect of this sequence is that the languages of words for which it achieves specific values are not first-order definable. For instance, it achieves the value 1 for words of odd lengths $(1, 3, 5, \ldots)$. Such a language is not first-order definable. Therefore, even though we restrict the expressions to first-order logic, additional complexity is introduced by the semiring operations.

We might want to ask if it is possible to define following sequence in 1Q modulo 2 using first-order logic: $\{0, 0, 1, 0, 0, 1, \ldots\}$, where the ones appear on positions divisible by 3. It seems unlikely, and it might be provable that the only sequences definable modulo 2 are those with periods that are powers of 2.

Bibliography

[1] Ronald Alter and K. K. Kubota. Prime and prime power divisibility of catalan numbers. *J. Comb. Theory A*, 15(3):243–256, 1973. doi:10.1016/0097-3165(73)90072-1.

[2] Corentin Barloy, Nathanaël Fijalkow, Nathan Lhote, and Filip Mazowiecki. A robust class of linear recurrence sequences. *Inf. Comput.*, 289(Part):104964, 2022. URL: https://doi.org/10.1016/j.ic.2022.104964, doi:10.1016/J.IC.2022.104964.

[3] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/malq.19600060105, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/malq.19600060105, doi:10.1002/malq.19600060105.

[4] Michaël Cadilhac, Filip Mazowiecki, Charles Paperman, Michal Pilipczuk, and Géraud Sénizergues. On polynomial recursive sequences. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 117:1–117:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. URL: https://doi.org/10.4230/LIPIcs.ICALP.2020.117, doi:10.4230/LIPICS.ICALP.2020.117.

[5] Lorenzo Clemente, Maria Donten-Bury, Filip Mazowiecki, and Michal Pilipczuk. On rational recursive sequences. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 24:1–24:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. URL: https://doi.org/10.4230/LIPIcs.STACS.2023.24, doi:10.4230/LIPICS.STACS.2023.24.

[6] Manfred Droste and Paul Gastin. Weighted automata and weighted logics. *Theor. Comput. Sci.*, 380(1-2):69–86, 2007. URL: https://doi.org/10.1016/j.tcs.2007.02.055, doi:10.1016/J.TCS.2007.02.055.

[7] George Kenison, Oleksiy Klurman, Engel Lefaucheux, Florian Luca, Pieter Moree, Joël Ouaknine, Markus A. Whiteland, and James Worrell. On positivity and minimality for second-order holonomic sequences. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia*, volume 202 of *LIPIcs*, pages 67:1–67:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. URL: https://doi.org/10.4230/LIPIcs.MFCS.2021.67, doi:10.4230/LIPICS.MFCS.2021.67.

[8] Stephan Kreutzer and Cristian Riveros. Quantitative monadic second-order logic. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New*

Orleans, LA, USA, June 25-28, 2013, pages 113–122. IEEE Computer Society, 2013. doi:10.1109/LICS.2013.16.

[9] Albert R. Meyer. Weak monadic second order theory of succesor is not elementary-recursive. In Rohit Parikh, editor, *Logic Colloquium*, pages 132–154, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg.

[10] Giovanni Pighizzini and Jeffrey O. Shallit. Unary language operations, state complexity and jacobsthal's function. *Int. J. Found. Comput. Sci.*, 13(1):145–159, 2002. doi:10.1142/S012905410200100X.

[11] Marcel Paul Schützenberger. On the definition of a family of automata. *Inf. Control.*, 4(2-3):245–270, 1961. doi:10.1016/S0019-9958(61)80020-X.

[12] Géraud Sénizergues. Word-mappings of level 3. *CoRR*, abs/2301.09966, 2023. URL: https://doi.org/10.48550/arXiv.2301.09966, arXiv:2301.09966, doi:10.48550/ARXIV.2301.09966.

[13] Larry J. Stockmeyer. *The complexity of decision problems in automata theory and logic*. PhD thesis, Massachusetts Institute of Technology, USA, 1974. URL: http://hdl.handle.net/1721.1/15540.