Sequences definable by 1-letter quantitative logics

Aleksander Wisniewski

August 20, 2024

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 [10] extended the model of finite automata to make it possible to calculate quantitative properties of words. He introduced a 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. In Section 3, we prove that the rate of growth of such sequences is at most doubly exponential, and this bound can be achieved. Then, in Section 4, we prove results about 1Q sequences over finite semirings - namely, that it is possible to simplify corresponding QMSO expressions by removing semiring quantifiers. From this, we deduce that 1Q sequences are eventually periodic modulo, which in particular implies that the Catalan sequence is not a 1Q-definable sequence.

Filip: To musi byc dluzsze. Trzeba jakos opisac co bylo wiadomo, co dokladnie zrobiles itp. Ale moze wrocimy do tego na koniec

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?

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.

In the original paper by Manfred Droste and Paul Gastin [6], it is shown that weighted logics (unrestricted) over finite semirings coincide with weighted automata. From this, it is possible to deduce that 1Q sequences are eventually periodic modulo, but we present different proof methods. In particular, we do not introduce the notation of weighted automata.

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 $a : \mathbb{N} - \{0\} \to \mathbb{D}$.

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 (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 ourselves 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:

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

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 2.2. 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 2.1. 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 won't go into details 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 2.3 (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 $1, \dots, n$. 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 2.4 (Super exponential)**.** Similarly as in Example 2.3 the sequence $a(n) = n^n$ can be defined with the following 1Q expression:

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

Example 2.5 (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}.

3 1Q sequences rate of growth

Let $a(n)$ be a 1Q sequence over natural numbers. We would like to find an upper bound for the asymptotical behavior of $a(n)$ depending only on n and constants appearing in the 1Q expression defining it. We will identify the class of sequences bounding 1Q sequences as the *rate of growth* of these sequences. So a 1Q sequence $a(n)$ will have a rate of growth in 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). It is known that linear recursive sequences have an exponentially bounded rate of growth. More formally, if $a(n)$ is an LRS then $|a(n)|\leq 2^{O(n)}$. In Example 2.3 we saw that in 1Q one 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 a sequence not definable in LRS. As mentioned in previous section (2.1), linear recursive sequences are contained in class of 1Q-definable sequences. From this follows that class of 1Q-definable sequences is strictly richer than LRS.

By Example 2.4 and Example 2.5, it is clear that it is possible to define 1Q sequences with a rate of growth even larger than $n!$. We want to provide an upper bound for the rate of growth for all 1Q sequences. We will prove that the rate of growth of 1Q sequences is bounded by $2^{2^{O(n)}}$. First, we will prove a lemma, which will be useful to simplify QMSO expressions without semiring quantifiers.

Definition 3.1 (Recognizable step function)**.** Fix a semiring K. A series $S: A^*\to K$ is a *recognizable step function*, if $S=\sum_{i=1}^{n}\ \varphi_{L_i}\cdot k_i$ for some $n\in\mathbb{N}$, $k_i\in K$ and regular languages $L_i\subseteq A^*$ $(i=1,\dots,n)$. φ_{L_i} is an MSO formula recognizing language L_i.

Filip: suma w sensie polpierscienia? (alek) tak

Lemma 3.1. 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 3.2 (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 $a_n = c^{2^{kn}}$.

Filip: chcialbym defnicje growth rate w preliminaries (alek) poki co napisalem cos na poczatku sekcji, nadal troche nie wiem jak to dobrze napisac

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

For base case, consider quantifier-free expression φ. Thanks to lemma 3.1 we know that such an expression can be simplified to recognizable step function $\varphi = \sum_{i=1}^{n}\ \varphi_{L_i} \cdot k_i$. Each of φ_{L_i} is 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 will help with further steps.

Filip: tu już ustalamy polpierscien na liczby naturalne? (alek) tak bedzie wygodniej, ale teoretycznie nie musimy

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

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

which is true assuming $c \geq 2$ (which we did in induction base).

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

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

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

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

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

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

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

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

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

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.

Definition 4.1. Let $a(n)$ be 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 4.1. Given a sequence $a(n)$ defined by 1Q expression φ over natural semiring $(\mathbb{N}, +, \cdot, 0, 1)$, a sequence $a(n) \mod m$, $m \in \mathbb{N} \setminus \{0\}$, can be defined by a 1Q expression φ', such that:

1. φ' is a 1Q expression over the semiring $\mathbb{Z}_m = (\{0, \dots, m-1\}, +_{mod}, \cdot_{mod}, 0, 1)$,
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 4.2. 1Q sequences are eventually periodic modulo $m \in \mathbb{N} \setminus \{0\}$.

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.*

Definition 4.2 (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 $\{L_i : i \in \{1, \dots, n\}\}$ 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 4.3. Recognizable step functions and simple recognizable step functions are equivalent.

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

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

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. $\square$

We already know that semiring quantifier-free 1Q expressions are equivalent to recognizable step functions (Lemma 3.1). 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 4.4. 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 $L_C := \{w : w \in L_A \wedge nRuns_A(w) \bmod m = k\}$.

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 as 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 $w = w_1 \dots w_{n-1} w_n$, 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. □

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 aan 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 4.1. 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 (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:

$$\varphi(X) = \neg(\exists_{x_1}\exists_{x_2}\ x_1 \in X \wedge x_2 \in X \wedge x_2 > x_1 \wedge \neg(\exists_{x_3}\ x_3 > x_1 \wedge x_3 < x_2))$$

Following word is not a part of language recognized by this expression:

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

because the set represented by free variable X valuation (second row) contains two consecutive positions - positions 2 and 3, to be exact.

Following word is a part of language recognized by this expression:

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

The interesting thing about property expressed by $\varphi(X)$ is that the number of valuations of X for given word length $n, n > 2$ is exactly $F(n+2)$, where F is a Fibonacci sequence.. It follows from one of combinatorial interpretations of Fibonacci sequence, namely: that $F(n+2)$ number is the number of all binary strings of length n that do not have two consecutive ones.

An automaton recognizing the same language as $\varphi(X)$ is presented in Fig. 1.

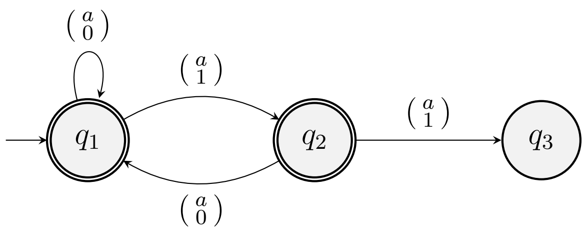

Figure 1: Deterministic automaton recognizing the same language as expression $\varphi(X)$

The method we want to use to prove that 1Q sequences modulo are eventually periodic is to simplify 1Q expressions defining such sequences. The main complexity of 1Q expressions is usage of semiring quantifiers. To calculate values of sequence defined by expression like $\Sigma_X\ \varphi(X)$, it is necessary to know for how many valuations of X the expressions $\varphi(X)$ is true, for words of length $1, 2, \ldots$.

Fortunately, manipulations of automata corresponding to expressions with free variables like $\varphi(X)$ allow us to create nondeterministic automata that "count" the number of valuations for which $\varphi(X)$ is true.

Lemma 4.5. Let φ be an MSO expression with a non-empty set of free variables $V = \{v_1, \ldots, v_n\}$. Let A be its corresponding deterministic finite automaton, i.e. automaton recognizing the same language as φ. φ and A recognize language $L \subseteq \Sigma_V^*$. A word $(w, \sigma) \in L$ can equivalently be denoted in following way: $(w, \sigma(v_1), \ldots \sigma(v_n))$. Let $v_i \in V$. Define following language over reduced alphabet (by "ignoring" variable v_i):

$$L' = \{(w, \sigma(v_1), \ldots, \sigma(v_{i-1}), \sigma(v_{i+1}), \ldots, \sigma(v_n)) \ : \ (w, \sigma(v_1), \ldots \sigma(v_n)) \in L\}$$

Now, following holds:

1. L' can be recognized by automaton A' identical to A, but defined over reduced alphabet $\Sigma_{V \setminus \{v_i\}} = \Sigma \times \{0,1\}^{|V-1|}$. Transitions of A' differ from transitions of A by simply ignoring variable v_i,

2. automaton A' can turn nondeterministic and for arbitrary word

$$w_{\setminus\{v_i\}} = (w, \sigma(v_1), \ldots, \sigma(v_{i-1}), \sigma(v_{i+1}), \ldots, \sigma(v_n)) \in L',$$

   and

$$w^{-1}_{\setminus\{v_i\}} = \{(w, \sigma(v_1), \ldots, \sigma(v_{i-1}), x, \sigma(v_{i+1}), \ldots, \sigma(v_n)) \in L \ : \ x \in \{0,1\}^*\},$$

   it is true that $nRuns_{A'}(w_{\setminus\{v_i\}}) = |w^{-1}_{\setminus\{v_i\}}|$. This means that the number of accepting runs of automaton A' for some valuation of variables (ignoring removed variable v_i) corresponds to the number of accepting valuations of this variable with given valuation of remaining variables in original language L.

Proof.

1. A' is simply an automaton recognizing projection of Σ_V^* on $\Sigma_{V \setminus \{v_i\}}$ - ignoring one of the dimensions (corresponding to variable v_i).

2. We can show that A' can turn nondeterministic in following way. Let δ_A be the set of transitions of automaton A. Let $q_1, q_2, q_3 \in Q_A$ be some three states of automaton A such that

$$q_1 \xrightarrow{(a, v_1, \ldots, v_{i-1}, 0, \ldots v_n)} q_2 \in \delta_A,$$
$$q_1 \xrightarrow{(a, v_1, \ldots, v_{i-1}, 1, \ldots v_n)} q_3 \in \delta_A.$$

   From construction it follows that in A' following situation happens:

$$q_1 \xrightarrow{(a, v_1, \ldots, v_{i-1}, \ldots v_n)} q_2 \in \delta_{A'},$$
$$q_1 \xrightarrow{(a, v_1, \ldots, v_{i-1}, \ldots v_n)} q_3 \in \delta_{A'},$$

   from which follows that A' is nondeterministic. If there are no such states in A then A' remains deterministic. Now we have to prove that $nRuns_{A'}(w_{\setminus\{v_i\}}) = |w_L|$, for $w_{\setminus\{v_i\}} \in L'$. It follows directly from construction of A', namely: to every accepting path of $w_{\setminus\{v_i\}}$ in automaton A' corresponds some valuation of v_i, for which automaton A must had accepted a word $(w, \sigma(v_1), \ldots, \sigma(v_{i-1}), \sigma(v_i), \sigma(v_{i+1}), \ldots, \sigma(v_n))$.

□

Example 4.3 (Fibonacci, continued)**.** Let us continue working on Example 4.2. There, we had following expression with a free variable X:

$$\varphi(X) = \neg(\exists_{x_1}\exists_{x_2}\ x_1 \in X \wedge x_2 \in X \wedge x_2 > x_1 \wedge \neg(\exists_{x_3}\ x_3 > x_1 \wedge x_3 < x_2))$$

We also constructed deterministic automaton recognizing language over extended alphabet $\Sigma_V = \{a\} \times \{0,1\}$. From Lemma 4.5 we know that it is possible to build an automaton, possibly nondeterministic, that ignores one of the free variables and its number of runs corresponds to number of valuations of ignored variable. In this case, we achieve automaton presented in Fig. 2.

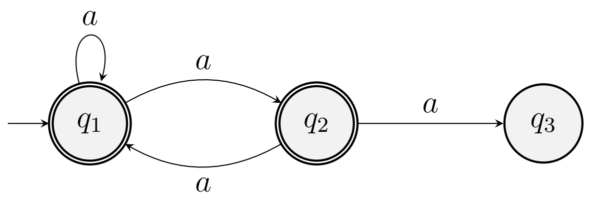

Figure 2: Nondeterministic "counting" automaton that skips variable X from $\varphi(X)$

From explanation in Example 4.2 of correspondence of $\varphi(X)$ and Fibonacci numbers, and from results in Lemma 4.5 it should follow that the number of accepting runs of automaton shown in Fig. 2 for word of length n (a^n) should equal $F(n+2)$. This is indeed true and can be proven by induction.

The following lemmas are crucial in semiring quantifiers elimination in 1Q expressions:

Lemma 4.6. Let $\mathbb{Z}_m$ be a modulo semiring and φ a simple recognizable step function on this semiring. Then $\Sigma_x\ \varphi$ and $\Sigma_X\ \varphi$ are also simple recognizable step functions.

Proof. φ is a simple recognizable step function, so 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 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 every MSO expression φ_{L_i} that appears inside φ, assign a deterministic finite automaton recognizing language L_i. These MSO expressions contain free variable X, so corresponding automata are defined over an extended alphabet. Having deterministic automata over extended alphabet we can use Lemma 4.5 to build possibly nondeterministic "counting" automata. The number of runs of these automata corresponds to number of accepting valuations of free variable X.

To handle addition - semiring quantifiers Σ - it is only necessary to know how many times (modulo m) each constant k_i appears. Count the number of runs (valuations) for automaton corresponding to φ_{L_i} using Lemma 4.4. For every φ_{L_i} and every $k < m$, create an automaton $\varphi^k_{L_i}$, accepting words over the reduced alphabet (i.e., ignoring variable X) for which the number of accepting valuations in the original automaton was equal to $k \mod m$. Now, addition quantifiers can be removed in following way:

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

$\varphi^j_{L_i}$ can be true for at most one j, $0 \leq j \leq m-1$. This j marks the number of successful runs modulo m, that is, the number of valuations of X for which φ_{L_i} was true. In the original expression, we would add k_i for each such valuation, giving us $j \cdot k_i$ for this part of the expression. Since we work modulo m, every such value is taken modulo m. □

Lemma 4.7. Let $\mathbb{Z}_m$ be a modulo semiring and φ a simple recognizable step function on this semiring. Then $\Pi_x\ \varphi$ and $\Pi_X\ \varphi$ are also simple recognizable step functions.

Proof. φ is a simple recognizable step function, so it can be expressed as $\varphi = \sum_{i=1}^{n}\ \varphi_{L_i} \cdot k_i$, where φ_{L_i} are MSO expressions and $k_i \in \mathbb{Z}_m$. Proof for this lemma will use similar reasoning to that in Lemma 4.6, but with additional complexity. Once again, we need to determine how many times each constant k_i appears across all valuations of variable bound by quantifier. However, it is insufficient to know how many times this constant appears modulo m, as in the case of exponentiation modulo, different numbers have different periods. For example, consider 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, not 3.

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

General case is following. 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, which follows from the fact that this sequence can achieve at most m different values, and $(k_i)^{n_i} \mod m = (k_j)^{n_j} \mod m \implies (k_i)^{n_i+1} \mod m = (k_j)^{n_j+1} \mod m$ (i.e. after reaching a cycle, it is not possible to escape it). As this sequence is eventually periodic, there exists $N_i, p_i \in \mathbb{N}$, $p_i > 0$ such that $\forall_{n>N_i}\ Exp_{k_i}(n) = Exp_{k_i}(n + p_i)$. So there is a prefix of length N_i (empty for $N_i = 0$), and then a repeating cycle of length p_i.

To know what is the influence of constant k_i on the whole expression, we need to introduce $N_i + p_i$ new constants and MSO expressions. First, for $0 < j \leq N_i$, introduce constant $k^j_i = (k_i)^j \mod m$ and MSO expression $\varphi^j_{L_i}$ that expresses that there exist exactly j different valuations of X for which φ_{L_i} is true, i.e. $\varphi^j_{L_i} = \exists^{=j}\ \varphi_{L_i}$. This handles prefix of length N_i.

Next, for $N_i < j \leq N_i + p_i$ introduce constant $k_i^j = (k_i)^j \mod m$ and a MSO expression $\varphi_{L_i}^j$ expressing that there exist more than N_i different valuations of X for which φ_{L_i} is true and this number of valuations is equal to $j \mod p_i$. For second property use automaton counting number of runs modulo from Lemma 4.4, the same way we did in addition case Lemma 4.6.

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

From what we have, the value corresponding to some k_i of the original expression, after quantifier elimination, is $\sum_{j=0}^{N_i+p_i} (\varphi_{L_i}^j \cdot k_i^j)$. The inner MSO expression $\varphi_{L_i}^j$ will be true for exactly one j.

Now, multiplication quantifiers can be removed in following way:

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

□

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

Lemma 4.8. Let $\mathbb{Z}_m$ be a modulo semiring and φ be a 1Q sentence over $\mathbb{Z}_m$. Then a sequence defined by φ can be defined as a simple recognizable step function (i.e. φ can be reduced to a form of simple recognizable step function).

Proof. We use induction on the structure of expression φ. For base case - expressions without quantification at the semiring level - it follows from Lemma 3.1 and Lemma 4.3.

Sum and product of two simple recognizable step functions is a simple recognizable step function - it follows from induction step of proof of Definition 3.1 and from Lemma 4.3.

Consider an expression $\varphi = Q_Y\ \theta$, where θ is a simple recognizable step function, Q a semiring quantifier and Y first order or second order variable. By lemmas 4.6 and 4.7 it follows that φ is a simple recognizable step function. This finishes the proof. □

Fact 4.1. Let S be a power series definable by a simple recognizable step function $S = \sum_{i=1}^{n} \varphi_{L_i} \cdot k_i$. Language of words for which S achieves value k_i is exactly φ_{L_i} (which is a regular language).

Proof. This follows directly from definition of simple recognizable step function and the fact that φ_{L_i} is a MSO expression. □

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

Lemma 4.9. Let $\mathbb{Z}_m$ be a modulo semiring and φ be a 1Q sentence over $\mathbb{Z}_m$. Then a sequence defined by φ is eventually periodic.

Proof. Thanks to Lemma 4.8 we know that φ reduces to simple recognizable step function. Additionaly, by Fact 4.1, we know that language of words for which sequence defined by φ achieves a given value k_i is a regular language. 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 ([9, Theorem 1]).

There are m regular, disjoint languages $L_1, \ldots, L_m$ corresponding to different constants k_i. We assume these languages don't accept empty word. Each language L_i is eventually periodic with regard 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 $N = \prod_{i=1}^{m} N_i$, $p = \prod_{i=1}^{m} p_i$. This period is simultaneously a period for every L_i since N divides each N_i and p divides each p_i. ☐

From this, the main result of this section, Theorem 4.2, directly follows:

Proof of Theorem 4.2. 1Q sequences modulo are the same as 1Q sequences over a modulo semiring (Lemma 4.1). The result then follows directly from Lemma 4.9. ☐

Corollary 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 prime numbers $p > 3$, while 1Q-definable sequences are. Therefore, Catalan numbers are not 1Q-definable. ☐

5 Other subjects

While researching 1Q sequences, there were other interesting subjects to go into, but were not completed.

Characterization of 1Q sequences

1Q expressions allow to define wide class of sequences. In particular, whole class of linear recursive sequences is contained in 1Q sequences. The interesting research question is exactly what class of sequences is defined by 1Q expressions - if there is other, maybe simpler representation of such sequences?

In Introduction it was mentioned that 1Q expressions allow defining a sequence that is not polynomial recursive. From this follows interesting question: is every polynomial recursive sequence also 1Q-definable? It does not seem easy to answer, as it is not trivial to express recursion with 1Q expressions.

Complexity of zero checking

Suppose we work over modulo semiring. What is the time complexity $O(|\varphi|)$ of checking whether sequence defined by such a sentence is constantly equal zero?

Consider naive procedure of zero checking which simply iterates over first n elements of sequence defined by φ and checks whether they are 0. How many elements do we need to check in order to be sure that the sequence is constantly equal to zero? This problem can also be stated in following way: how far in the sequence can first non-zero element appear (with regard to 1Q expression length)?

It turns out that this problem has super-exponential complexity, even without semiring quantifiers. It follows from expressive power of MSO. In MSO, it is possible to define a sentence of length n that does not accept words of length smaller than $tower(n)$ (citation needed). Function $tower$ is defined recursively: $tower(0) = 2$, $tower(n) = 2^{tower(n-1)}$.

1Q on first order logic

The task of characterizing 1Q sequences seems non-trivial, but it might be interesting to look into following restriction of 1Q expressions: we can only use first order variables on semiring level and logical level (i.e. we move from monadic second order logic to first order logic).

Unfortunately, this class of sequences also seems hard to characterize and it is not obvious how to tell if given sequence is or is not definable as 1Q sentence on first order logic.

Instead, it might be worthwhile to again consider sequences modulo like we did in Section 4.

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

$$\Sigma_x \ . \ 1$$

It defines the following sequence modulo 2: $\{1, 0, 1, 0, 1, \ldots\}$, which is not aperiodic. Interesting thing about this sequence is that languages of words for which it achieves given values are not first order definable. For example, it achieves value 1 for words of odd lengths $(1, 3, 5, \ldots)$. Such a language is not first order definable. So, even though we restricted expressions to only use first order logic, there is additional complexity provided by semiring operations.

We might want to ask if it is possible to define following sequence in 1Q modulo 2 using first order logic: $\{0, 0, 1, 0, 0, 1, \ldots\}$, that is: we have ones on positions divisible by 3. It rather seems to not be the case, and it might be provable that the only sequences definable modulo 2 are those with periods being a power of 2.

References

[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] 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`.

[10] 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`.