Sequences definable by 1-letter quantitative logics

Aleksander Wisniewski

August 18, 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 [2]. Finite automata allow us to recognize languages, i.e., to accept or reject words over a given alphabet. Schützenberger [9] 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 [5], 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 [7]. 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. [1]). It can be easily shown that sequences achieved using unrestricted QMSO form a richer class than linear recursive sequences, which was already demonstrated in [3], 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 [3], 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 [3], 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 [4] or holonomic sequences [6]. However, these classes can define Catalan sequences, while 1Q sequences cannot.

In the original paper by Manfred Droste and Paul Gastin [5], 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 [7] 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:

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

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 [7, 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's 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. [1]). 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:

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

For given n it works as following: x_1 iterates over $1, \ldots, n$. For fixed $x_1 = k$ the expression $\Sigma_{x_2}\ (x_2 \le x_1) \cdot 1$ 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 $L_i \subseteq A^*$ $(i = 1, \ldots, n)$. 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.

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).

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

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 $c^{2^{kn}}$.

□

Filip: ogolne uwagi: za duzo "can"; nie powinno sie uzywac nieformalnego jezyka; nierownosci to czesc zdania (powinny sie konczyc kropka lub przecinkiem); podobnie jak cos wymieniamy jako itemy; kazde zdanie przeczytac i skrocic

4 1Q sequences over finite semirings

Behavior of 1Q sequences modulo can help characterizing them and comparing with other classes of sequences. In this section we will show that 1Q sequences taken modulo $m \in \mathbb{N}\backslash\{0\}$, are eventually periodic.

Definition 4.1. Let $a(n)$ be a sequence. $a(n)$ is *eventually periodic* if there exists $N, p \in \mathbb{N}$, such that $\forall_{n \geq N}\ a(n) = a(n+p)$.

This property shows limitations of 1Q sequences: they start behaving regularly from some point on. It is an effective method of checking if a sequence is 1Q-definable. For example, sequence of Catalan numbers is not eventually periodic, so it will be shown that it is not 1Q-definable. The goal of this section is to prove the following lemma:

Theorem 4.1. 1Q sequences are eventually periodic modulo $m \in \mathbb{N} \setminus \{0\}$.

First, we will characterize 1Q sequences modulo as 1Q sequences on modulo semiring:

Lemma 4.2. 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 $\mathbb{Z}_m = (\{0, \ldots, m-1\}, +_{mod}, \cdot_{mod}, 0, 1)$,
2. every constant k appearing in φ is replaced by $k \mod m$ in φ'.

Proof. Semantics of 1Q expressions consist of addition, multiplication, repeated addition and repeated multiplication (repeated in case of semiring quantifiers). This lemma is a direct consequence of properties of operations modulo:

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

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

□

We will now prove a lemma about finite automata construction that will be useful later. It can be used to count number of runs of nondeterministic automaton. Let A be a nondeterministic finite automaton. Define function $nRuns_A : \Sigma^* \to \mathbb{N}$ to be a function calculating number of accepting runs of automaton A over words $w \in \Sigma^*$. We will also use function $nRuns_A : \Sigma^* \times Q \to \mathbb{N}$, where Q is a set of states of automaton A. $nRuns_A(w, q)$ calculates the number of accepting runs of automaton A over word w which finish in state q.

Lemma 4.3. Given a nondeterministic finite automaton A recognizing the language L_A and given natural numbers m, k, $k < m$, it is possible to create a finite automaton C (as for **c**ounting automaton) recognizing the language $L_C := \{w : w \in L_A \wedge nRuns_A(w) \mod m = k\}$.

Proof. We will create 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, Q_C are functions $f : Q_A \to \{0, \ldots, m-1\}$. There are m^n such functions, so is the number of states in Q_C. For state $q \in Q_C$ we will denote its corresponding function as f_q. State $q \in Q_C$ represented by function f_q will be interpreted as following:

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

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

The initial state of C is a function mapping the initial state of A to 1 and every other state to 0.

Suppose the set of final states of automaton A is F_A. Let $q_C \in Q_C$. Define $sumFin(q_C) = \sum_{q \in F_A} f_{q_C}(q)$. $sumFin$ returns sum of values of function corresponding to given state q_C over all final states of A. Now, final states of C, $F_C \subseteq Q_C$, are states q_C for which $sumFin(q_C) = k \mod m$. Intuitively, final states of C are states in which we accept a word $k \mod m$ times.

Lastly, we need to define transitions of C. Suppose the set of transitions of automaton A is δ_A and set of transitions of automaton C is δ_C. Let q_C^1, q_C^2 be two states of C, f_1, f_2 their corresponding functions. $q_C^1 \xrightarrow{a} q_C^2 \in \delta_C$ 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 count the number of runs ending in given state q_i by summing over all states that have a transition to this state over given letter from previous state.

Now let us prove correctness of this construction. We will prove that after reading input word w, we reach state q_C in automaton C, for which a function f_{q_C} correctly calculates number of runs over word w ending in given states in automaton A (modulo m). From this and construction of final states of C it will be clear that the construction is correct.

We will prove it by induction on length of word w. In base case consider an empty word ϵ. C accepts this word only if following conditions occur:

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

This follows from definition of initial and final states of C.

For induction step, assume that construction is correct for a word of length $n-1$, $w_1 \dots w_{n-1}$. That is, after reading 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 numbers of runs (modulo m). We want to show that after reading next letter (so having a word of length n), C reaches a state that also correctly assigns numbers of runs. For some state $q \in Q_A$, to find a number of runs finishing in q after reading a word $w = w_1 \dots w_{n-1} w_n$, we need to consider all the states $q_{prev} \in Q_A$ such that $(q_{prev} \xrightarrow{w_n} q) \in \delta_A$ - all the states from which q can be reached over last letter in word. Thanks to induction assumption, we know the numbers of runs ending in those states q_{prev} after reading $w_1 \dots w_{n-1}$. From this it's clear that the number of runs finishing in state q after reading w can be expressed by 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. □

This lemma will be used to prove that it is possible to remove quantifiers from 1Q expressions and simplify those expressions.

First, it will be necessary to specify encoding of words accepted by MSO expressions with free variables. We will use a method explained in [7]. The 'base' alphabet is still 1 letter alphabet $\Sigma = \{a\}$. Now, let φ be a MSO expression with non empty set of free variables (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 some word over Σ and σ a (V, w) assignment. A word $(w, \sigma) \in \Sigma_V^*$ *encodes* a (V, w)-assignment if w is the projection of (w, σ) over 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-letter of the projection (w, σ) over variable v. For first order variables $v_f \in V$ this set $\sigma(v_f)$ has exactly one element. For second order variables it can contain arbitrary number of positions in 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 be the first position. Language recognized by this expression can be expressed with following regular expression: $\epsilon + \binom{a}{1}\binom{a}{0}^*$.

In the following steps it will be convenient to work with finite automata instead of MSO expressions. It is known ([2]) that for every MSO expression it is possible to build a finite state automaton accepting the same language and vice versa.

Example 4.2 (Fibonacci)**.** Consider following expression with a free variable X - a set. This expression only accepts sets in which there are no two consecutive positions contained in this 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 $\varphi(X)$ looks as follows:

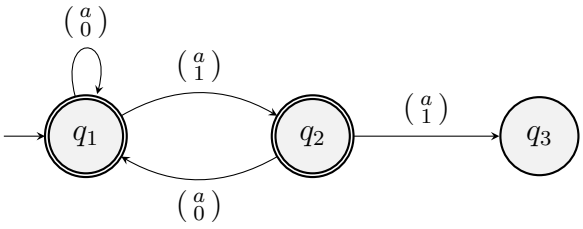

Figure 1: Deterministic automaton recognizing the same language as expression $\varphi(X)$

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 4.4. 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's 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 4.4 we know that it's 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 following automaton:

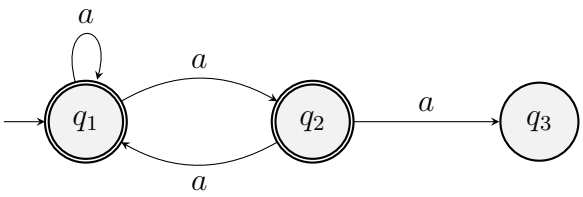

Figure 2: Nondeterministic "counting" automaton that skips variable X from $\varphi(X)$

From explanation in Example 4.2 of correspondence of $\varphi(X)$ and Fibonacci numbers, and from results in Lemma 4.4 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.

We will use notion of simple recognizable step functions, which are an extension of recognizable step functions (3.1).

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 $\{L_i : i \in \{1, \ldots, n\}\}$ form a partition of the whole language $\Sigma^* = \dot{\bigcup}_i\ L_i$.

A recognizable step function can be transformed into simple recognizable step function:

Lemma 4.5. Recognizable step functions and simple recognizable step functions are the same functions.

Proof. Suppose we have a recognizable step function $S = \sum_{i=1}^{n}\ \varphi_{L_i} \cdot k_i$. We can create a simple recognizable step function defining the same function. Let P be a powerset of $N = \{1, \ldots, n\}$. Then a simple recognizable step function can be defined in following way:

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

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 law of excluded middle. Now, to calculate value appearing in this intersection,

we add constants corresponding to true expressions. This gives us a simple recognizable step function which calculates the same values as original recognizable step function.

A simple recognizable step function is a recognizable step function by definition, so it does not require a proof. □

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 4.4 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 4.3. For every φ_{L_i} and every $k < m$, create an automaton $\varphi_{L_i}^k$, 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:

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

$\varphi_{L_i}^{j}$ can be true for at most one j, $0 \leq j \leq m-1$. This j 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. 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.

Both cases need to be handled. Start with the second one, i.e., when k_i is a root of order p_i of m. Define p_i new MSO expressions and constants. Suppose we want to remove the quantifier $\Pi_X\ \varphi$, where X is either a first-order or second-order variable. Then $\varphi^j_{L_i}$ for $1 \leq j < p_i$ will express that there exist exactly j different valuations of X for which φ_{L_i} is true. For $j = p_i$, $\varphi^j_{L_i}$ expresses that there are (at least) p_i different valuations of X. Constants will be defined in following way: $k^j_i = (k_i)^j \mod m$, for $1 \leq j \leq p_i$. It follows from the fact that k_i is a root of order p_i of m that $k^j_i = 0$ for $j = p_i$.

For first case, i.e., k_i is not a root of m and the values $(k_i)^j \mod m$ form a cycle, we need to proceed as follows. For every such constant k_i appearing in φ, we need to find its period p_i modulo m with respect to exponentiation. This period will be no greater than m, as sequence $a(n) = (k_i)^n \mod m$ can achieve at most m different values before repeating. Define $k^j_i = (k_i)^j \mod m$, for $1 \leq j \leq p_i$. Now we can apply reasoning similar to the addition case: we create automata $\psi^j_{L_i}$ for $1 \leq j \leq p_i$. The expression $\psi^j_{L_i}$ accepts a language of words for which number of runs (valuations) modulo p_i is j. However, in this case, we need to handle the special situation where there zero accepting valuations (as opposed to zero modulo p_i). We want $\varphi^j_{L_i}$ to be false in this case, for every j. To achieve this, define the expression: $\varphi^j_{L_i} = \psi^j_{L_i} \wedge \exists_X\ \varphi_{L_i}$, where X is a first-order or second-order variable, depending on the type of quantifier we are eliminating.

From what we already have, the value corresponding to some k_i of the original expression, after quantifier elimination, is $\sum_{j=1}^{p_i} (\varphi^j_{L_i} \cdot k^j_i)$. The inner MSO expression $\varphi^j_{L_i}$ will be true for at most one j. It will not be true for any j if there are zero accepting valuations. In this case, $\sum_{j=1}^{p_i} (\varphi^j_{L_i} \cdot k^j_i)$ will be zero. It will be more convenient if we had one in this case, as one is a neutral element with respect to multiplication. To achieve this, we will use the following expression for a given constant k_i: $(\sum_{j=1}^{p_i}\ \varphi^j_{L_i} \cdot k^j_i) + (\neg\exists_X\ \varphi_{L_i})$

Now, multiplication quantifiers can be removed in following way:

$$\Pi_x\ \varphi = \prod_{i=1}^{n} (\sum_{j=1}^{p_i}\ \varphi^j_{L_i} \cdot k^j_i) + (\neg\exists_x\ \varphi_{L_i}),$$

$$\Pi_X\ \varphi = \prod_{i=1}^{n} (\sum_{j=1}^{p_i}\ \varphi^j_{L_i} \cdot k^j_i) + (\neg\exists_X\ \varphi_{L_i}).$$

□

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 lemmas 3.1 and 4.5.

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 4.5.

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. □

Lemma 4.9. Inverse images of simple recognizable step functions are regular languages

Proof. This follows directly from definition. Let $\varphi = \sum_{i=1}^{n}\ \varphi_{L_i} \cdot k_i$ be a simple recognizable step function. Then $\varphi^{-1}(k_i) = \varphi_{L_i}$, where φ_{L_i} specifies a regular language. □

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

Lemma 4.10. 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 4.9, 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 ([8, 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 4.1, directly follows:

Proof of Theorem 4.1. 1Q sequences modulo are the same as 1Q sequences over a modulo semiring (Lemma 4.2). The result then follows directly from Lemma 4.10. □

Corollary 4.1. Catalan numbers are not 1Q-definable

Proof. Reasoning is the same as in [3, Theorem 7, Corollary 8] - Catalan numbers are not ultimately periodic modulo prime numbers $p > 3$, while 1Q-definable sequences are. Therefore, Catalan numbers are not 1Q-definable. □

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.