1Q Sequences

Aleksander Wisniewski

July 29, 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 [8] 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 some semiring) and a run multiplies those weights. A value corresponding to a given input word is the sum of values over all runs. Weighted automata calculate what we call a formal power series: a function from words to semiring values, $S : \Sigma^* \to K$.

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 allow defining formal power series in the same way weighted automata do. Unfortunately, the full, "natural" form of weighted logic is richer than weighted automata, so the authors had to introduce a restricted class of weighted logic expressions in order to have the same power as weighted automata. The restrictions are both at the syntactic and semantic levels.

Later works on weighted logics include, among others, 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.

Now, in this work, we would like to explore the world of unrestricted QMSO expressions. In particular, we want to 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, the formal power series $S : \Sigma^* \to K$ can be also seen as a sequence $S : \mathbb{N} \to K$ if we map words from our domain a^n to their length n. We focus on the unrestricted case, as restricted QMSO expressions coincide with weighted automata, and it is known that formal power

series defined by weighted automata over a one-letter alphabet are exactly linear recursive sequences [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 hasn't been researched 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.

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 this paper, it is shown that 1Q sequences can define a sequence that is undefinable as a polynomial recursive sequence: n^n. However, it is not clear if all polynomial recursive sequences are also 1Q sequences.

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'll summarize syntax and semantics for Quantitative Monadic Second Order Logic (QMSO). It was introduced in [7] by Stephen Kreutzer and Cristian Riveros. Then we'll introduce 1Q sequences with some examples.

2.1 Monadic Second Order Logic

Let Γ be a finite alphabet. The syntax of MSO over 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, X is a second order variable. Universal quantification can be obtained from existential quantification and negation. We can also use $\wedge$ and $\implies$ as usual.

Let $w = w_1 \dots w_n \in \Gamma*$ be a word such that $|w| = n$. We represent w as a structure $(\{1, \dots, n\}, \leq, (P_a)_{a \in \Gamma})$ where $P_a = \{i \mid w_i = a\}$. We denote by $Dom(w) = \{1, \dots, n\}$ 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.

2.2 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'll 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: $\mathbb{Z}_k = (\{0, 1, \dots, k-1\}, +_{mod}, \cdot_{mod}, 0, 1)$ where operations are executed modulo k

We'll restrict ourselves to working with sentences of natural numbers, although 1Q sequences can be defined over arbitrary semirings.

2.3 Quantitative Monadic Second-Order Logic

(Syntax) The formulas of Quantitative Monadic Second-Order Logic (QMSO) over semiring $\mathbb{S}$ and 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 second-order variable.

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

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

The semantics of the semiring level is defined as follows:

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

2.4 QMSO over one letter alphabet

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

In this work, we focus on expressions of QMSO logic over a one-letter alphabet. Let's call these expressions **1Q expressions**. In this case, we only use a small fragment of MSO logic; in particular, we don't care about 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 simplification, let's assume that our one-letter alphabet will always be $\Gamma = \{a\}$.

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**.

Let's look at some examples of 1Q sequences that we'll refer to in following sections:

Example 2.1 (Factorial)**.** Sequence $a(n) = n!$ can be defined with 1Q expression:

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

For given n it works as following: x_1 iterates over $1, \ldots, n$. For some $x_1 = k$ expression $\Sigma_{x_2} \ . \ (x_2 \leq x_1) \cdot 1$ will have value exactly k. So the whole expression will be $1 \cdot 2 \cdot \ldots \cdot n$.

Example 2.2 (Super exponential)**.** Sequence $a(n) = n^n$ can be defined with 1Q expression:

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

Example 2.3 (Doubly exponential)**.** Sequence $a(n) = 2^{2^n}$ can be defined with 1Q expression:

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

For given n, there are 2^n valuations of X_1. So the value is 2 multiplied by itself 2^n times - 2^{2^n}.

3 1Q sequences rate of growth

In this section, we're interested in the bound of the rate of growth of 1Q sequences. Let $a(n)$ be a 1Q sequence - we'd like to find an upper bound for the asymptotical behavior of $a(n)$ depending only on n and constants appearing in 1Q expression defining $a(n)$. It will also be interesting to compare this result to linear recursive sequences (LRS).

It is known that linear recursive sequences have a rate of growth bounded by $2^{O(n)}$. In 1Q we can define the sequence $n!$ (2.1), which has a greater rate of growth than $2^{O(n)}$ - by Stirling's approximation, $n! \sim \sqrt{2\pi n}(\frac{n}{e})^n$, which grows faster than 2^n. This shows that the class of 1Q sequences contains a sequence undefinable in LRS.

This result can be easily extended to show that the class of 1Q sequences is richer than LRS, as Stephan Kreutzer and Cristian Riveros [7] proved that there's a subclass of QMSO expressions defining the whole class of power series definable by weighted automata. It is known that formal power series over a one-letter alphabet definable by weighted automata are precisely LRS [1]. From this, we conclude that there's a subclass of 1Q sequences coinciding with LRS, but the whole class of 1Q sequences is richer (as it's possible to define $n!$).

Now, from examples 2.2 and 2.3, it's clear that it's possible to define 1Q sequences with a rate of growth greater than $n!$. The question we want to answer is what is the upper bound for the rate of growth of all 1Q sequences.

We'll argue that the rate of growth of 1Q sequences is bounded by $2^{2^{O(n)}}$. First, we'll prove a lemma, which will be useful to simplify QMSO expressions without semiring quantifiers.

Definition 3.1 (Recognizable step function)**.** 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 and K is an arbitrary semiring.

Lemma 3.1. Let θ be a QMSO expression without semiring quantifiers. Then θ can be expressed as a recognizable step function.

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

1. QMSO expression $\theta = \varphi$, where φ is a MSO expression. Corresponding recognizable step function is $\theta = \varphi \cdot 1$
2. QMSO expression $\theta = k$, where k is a constant. Corresponding recognizable step function is $\theta = \varphi \cdot k$, where φ recognizes whole language (for example, $\varphi = \exists_x \ . \ x = x$)

Let's move on to induction step. We have to 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. 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's immediate that θ is a recognizable step function.

□

Lemma 3.2 (1Q growth rate class)**.** Growth rate class of 1Q sequences is doubly exponential - $2^{2^{O(n)}}$. Also, for every $k, c \in \mathbb{N}$ it's possible to generate 1Q sequence $c^{2^{kn}}$.

Proof. We will show that every sequence generated by 1Q expression can be bounded by a sequence with doubly exponential growth rate, which is also generated by 1Q expression. In order to do this we'll specify how to build 1Q expressions generating sequences with maximal possible growth rate. We'll analyze 1Q expressions of form:

$$Q_{1Y_1} Q_{2Y_2} \dots Q_{kY_k} \ . \ c$$

Where $Q_i \in \{\Sigma, \Pi\}$ and each Y_i is either first-order or second-order variable. There are k semiring quantifiers, and then an expression without semiring quantification. Innermost expression is just a constant c - it doesn't make sense to have any MSO expressions here, as those only filter out some variable evaluations. More precisely, let φ be a quantifier-free 1Q expression. Then there exists c such that:

$$Q_{1Y_1} Q_{2Y_2} \dots Q_{kY_k} \ . \ \varphi \leq Q_{1Y_1} Q_{2Y_2} \dots Q_{kY_k} \ . \ c \tag{1}$$

c can be obtained by looking at φ as a recognizable step function $\varphi = \sum_{i=1}^{n} \ . \ \varphi_{L_i} \cdot k_i$ - which is justified thanks to lemma 3.1. Now, each of φ_{L_i} is either 0 or 1. To satisfy inequality 1 we can set $c = \sum_{i=1}^{n} \ . \ |k_i|$. Clearly $\sum_{i=1}^{n} \ . \ \varphi_{L_i} \cdot k_i \leq \sum_{i=1}^{n} \ . \ |k_i|$, from which follows inequality 1. From now on let's assume that c is not smaller than 2, so $c = max(2, \sum_{i=1}^{n} \ . \ |k_i|)$.

Suppose we have quantification of depth k. There are four possible kinds of quantification: Σ_x, Σ_X, Π_x, Π_X. We can argue that it only makes sense to consider Π_X quantifications. No matter which Q_{iY_i} quantifier we consider, its result will be sum (for Σ) or multiplication (for Π) of n (for first-order variable) or 2^n (for second-order variable) identical subexpressions, each having value at least 2. Namely:

$$\Sigma_x \ . \ c = n \cdot c$$

$$\Sigma_X \ . \ c = 2^n \cdot c$$

$$\Pi_x \ . \ c = c^n$$

$$\Pi_X \ . \ c = c^{2^n}$$

Clearly $c^{2^n} \geq c^n \wedge c^{2^n} \geq 2^n \cdot c \wedge c^{2^n} \geq n \cdot c$.

From now on we'll look at simplified expressions of form $\Pi_{X_1} \Pi_{X_2} \dots \Pi_{X_k} \ . \ c$.

Our simplified expression is in prenex normal form with regards to semiring quantification. We should justify why every 1Q sequence can be bounded by sequence defined by an expression of such form. Let $\theta_1 = \Pi_{X1'} \dots \Pi_{Xk'} \ . \ c_1$, $\theta_2 = \Pi_{X1''} \dots \Pi_{Xk''} \ . \ c_2$ where $c_1, c_2 \geq 2$.

Let $pref = \Pi_{X1}\Pi_{X2}\dots\Pi_{Xk}$ be some quantifiers prefix, possibly empty. Then it can be seen that:

$$pref\ .\ \theta_1 + \theta_2 \leq pref\ .\ \Pi_{X1'}\dots\Pi_{Xk'}\Pi_{X1''}\dots\Pi_{Xk''}\ .\ c_1 \cdot c_2$$

$$pref\ .\ \theta_1 \cdot \theta_2 \leq pref\ .\ \Pi_{X1'}\dots\Pi_{Xk'}\Pi_{X1''}\dots\Pi_{Xk''}\ .\ c_1 \cdot c_2$$

It follows from the fact that

$$\theta_1 + \theta_2 = (c_1)^{2^{k'n}} + (c_2)^{2^{k''n}} \leq (c_1 \cdot c_2)^{2^{(k'+k'')n}} = \Pi_{X1'}\dots\Pi_{Xk'}\Pi_{X1''}\dots\Pi_{Xk''}\ .\ c_1 \cdot c_2$$

and

$$\theta_1 \cdot \theta_2 = (c_1)^{2^{k'n}} \cdot (c_2)^{2^{k''n}} \leq (c_1 \cdot c_2)^{2^{(k'+k'')n}} = \Pi_{X1'}\dots\Pi_{Xk'}\Pi_{X1''}\dots\Pi_{Xk''}\ .\ c_1 \cdot c_2$$

This shows that we can bound expressions with addition and multiplication using greater quantification depth and we finish with prenex normal form expressions.

From this we see that for arbitrary 1Q expression Ψ we can 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's possible to achieve sequence $c^{2^{kn}}$. □

4 1Q sequences over finite semirings

In order to compare 1Q sequences with other classes of sequences, it's useful to consider what is the behavior of these sequences modulo arbitrary natural numbers.

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

1. φ' is a 1Q expression over 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. Semantics of 1Q expressions consist of addition, multiplication, repeated addition and repeated multiplication (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'll now prove a lemma about finite automata construction that will be useful later:

Lemma 4.2. Given a nondeterministic finite automaton A recognizing language L_A and natural numbers m, k, $k < m$, it's possible to create a finite automaton C (as for **c**ounting automaton) recognizing language $L_C := \{w : w \in L_A \wedge nRuns(A, w) \mod m = k\}$. $nRuns(A, w)$ returns number of runs of automaton A over word w.

Proof. We'll show how exactly to create such an automaton C. We'll do an extended powerset construction.

Suppose original automaton A had n states, $Q_A = \{q_1, \ldots, q_n\}$. States of new automaton C, Q_C will represent functions $f : Q_A \to \{0, \ldots, m-1\}$. There are m^n such functions, so is the number of states of Q_C. For state $q \in Q_C$ we'll denote its corresponding function as f_q.

Let $nRuns(A, w, q_i)$ be a function returning a number of runs of automaton A over word w finishing in state $q_i \in Q_A$. 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's introduce initial state, final states and transitions of automaton C.

Initial state of C is a function mapping 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

$$f_2(q_i) = (\sum_{q \in Q_A} . \text{ if } q \xrightarrow{a} q_i \text{ then } f_1(q) \text{ else } 0) \mod m$$

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's prove correctness of our construction. We'll 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'll prove it by induction on length of word w. In base case we consider an empty word ϵ. In C we'll accept this word only if all of the following occur:

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

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

Now, let's handle 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), we'll also reach a state that 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 focus on all the states $q_{prev} \in Q_A$ such that $(q_{prev} \xrightarrow{w_n} q) \in \delta_A$ - all the states from which we can reach q over last letter in word. Thanks to induction step, 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:

$$\Big(\sum_{q \in Q_A} \text{. if } q \xrightarrow{a} q_i \text{ then } f_{q_C}(q) \text{ else } 0\Big) \mod m$$

Which is exactly how we defined transition function of C.

□

We will make use of this lemma to prove that it's possible to remove quantifiers from 1Q expressions and simplify those expressions.

First, it'll be necessary to explain encoding of words accepted by MSO expressions with free variables. Our '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 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) projection 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.

Let's show a small example. Suppose we have an expression with one free first order variable:

$$\varphi = \forall x \ . \ y \leq x$$

In this case y can only be the first position. Language recognized by this expression can be expressed with following regular expression: $\epsilon + (a,1)(a,0)^*$.

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

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

Definition 4.1 (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 disjoint union of whole language $\Sigma^* = \dot{\bigcup}_i \ . \ L_i$.

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

Lemma 4.3. 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 doesn't require a proof. □

First, we'll prove following lemma, which is crucial in quantifier elimination of 1Q expressions:

Lemma 4.4. Suppose we're working over a modulo semiring $\mathbb{Z}_m$. Let φ be a simple recognizable step function. Then all of: $\Sigma_x \ . \ \varphi$, $\Sigma_X \ . \ \varphi$, $\Pi_x \ . \ \varphi$, $\Pi_X \ . \ \varphi$ 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 $k_i \in \mathbb{Z}_m$. Now, let's consider expression $Q_Y \ . \ \varphi$, where Q is an arbitrary semiring quantifier and Y is a first-order or second-order variable. Some of the MSO expressions inside φ might contain Y as a free variable, other might not - nevertheless in semantics of QMSO, we'll iterate over all valuations of Y to evaluate this expression. We might as well assume that every MSO expression inside φ contains Y as a free variable, even as a part of expression $Y = Y$.

Now, for every MSO expression φ_{L_i} appearing inside φ we can assign a deterministic finite automaton recognizing the same language. These MSO expressions contain free variable Y, so automatons are defined over extended alphabet. Now, we will modify this automaton by changing alphabet: we remove position from alphabet corresponding to valuation of variable Y. This is a similar method to that used in proving that finite automata coincide with MSO expressions. After changing alphabet we turn our deterministic finite automaton into nondeterministic finite automaton which guesses valuations of free variable Y. The number of runs of this new automaton is the same as number of valuations of Y for which φ_{L_i} is true.

To handle addition - semiring quantifiers Σ - we only need to know how many times (modulo m) each constant k_i appeared. We can count the number of runs (valuations) for automatons corresponding to φ_{L_i} using lemma 4.2. For every φ_{L_i} and every $k < m$, we can create automaton $\varphi_{L_i}^k$, accepting words over reduced alphabet (i.e. without valuation of Y) for

which number of accepting valuations in original automaton was equal to $k \mod m$. Now, addition quantifiers can be removed in following way:

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

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

This can be justified in following way. $\varphi_{L_i}^{j}$ can be true for at most one j, $1 \leq j \leq m-1$. This j marks the number of successful runs - that is, number of valuations of x (or X) for which φ_{L_i} was true. In original expression we would add k_i for each such valuation - that gives us $j \cdot k_i$ for this part of expression. We're working modulo m, so we take every such value modulo.

It's a bit more complicated in case of multiplication quantifier Π. Again, we need to know how many times each constant k_i will appear, across all valuations. But it's not enough to know how many times this constant will appear modulo m, as in case of exponentiation modulo - numbers have different periods. For example, consider constant 2 when working modulo 3. $2^1 \mod 3 = 2$, $2^2 \mod 3 = 1$, $2^3 \mod 3 = 2$, etc. For odd exponents the value will be always 2, and for even exponents it will be 1. So in this case we need to know number of valuations modulo 2, not 3.

It's also different for constants that are root of any order of m. Suppose we work modulo $m = 16$ and we have a constant $k_i = 2$. Now, $2^1 \mod 16 = 2$, $2^2 \mod 16 = 4$, $2^3 \mod 16 = 8$, $2^4 \mod 16 = 0$, $2^5 \mod 16 = 0$ etc. So after reaching 2^4, for all further exponents we'll have value 0. In this case we have some prefix $2, 4, 8$ and then constant 0.

We need to handle both of these cases. Let's start with the second one, i.e. k_i is a root of order p_i of m. We'll define p_i new MSO expressions and constants. Suppose we want to remove quantifier $\Pi_x \,.\, \varphi$ (the reasoning will be the same for set quantification). Then $\varphi_{L_i}^{j}$ for $j < p_i$ will express that there exist exactly j different valuation of x for which φ_{L_i} is true. For $j = p_i$, $\varphi_{L_i}^{j}$ expresses that there are (at least) p_i different valuations of x. Constants will be defined in following way: $k_i^j = 2^j \mod m$, 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 $k_i^j = 0$ for $j = p_i$.

For first case, i.e. k_i is not a root of m, so the values of $(k_i)^j \mod m$ form a cycle, we need to do the following. For every such constant k_i appearing in φ, we need to find its period p_i modulo m with regards to exponentiation. This period will be no greater than m, as $(k_i)^n \mod m$ can achieve at most m different values and cycle. We'll define $k_i^j = (k_i)^j \mod m$, for $1 \leq j \leq p_i$. Now we can repeat reasoning similar to addition case - we create automatons $\varphi_{L_i}^{j}$ for $1 \leq j \leq p_i$. Automaton $\varphi_{L_i}^{j}$ accepts a language of words for which number of runs (valuations) modulo p_i is j. Unfortunately there's

Now, multiplication quantifiers can be removed in following way:

$$\Pi_x \,.\, \varphi = (\prod_{i=1}^{n} \sum_{j=1}^{p_i} \,.\, (\varphi_{L_i}^{j} \cdot k_i^j)) \cdot (\exists_x \,.\, \bigvee_{i=1}^{n} \varphi_{L_i})$$

$$\Pi_X \ . \ \varphi = (\prod_{i=1}^{n} \sum_{j=1}^{p_i} \ . \ (\varphi_{L_i}^j \cdot k_i^j)) \cdot (\exists_X \ . \ \bigvee_{i=1}^{n} \varphi_{L_i})$$

Multiplication quantifier (from 1 to n) is natural here, as we multiply constants appearing in φ. Then, for give i, there will be only one j between 1 and p_i for which $\varphi_{L_i}^j$ is true - that follows from construction. □

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

Lemma 4.5. Suppose we're working over a modulo semiring $\mathbb{Z}_m$. Let φ be a 1Q sentence. 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'll prove this lemma by induction on structure of our expression φ. For base case - expressions without quantification on semiring level - it follows from lemmas 3.1 and 4.3. Now, for induction step, consider 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 lemma 4.4 it follows that φ is a simple recognizable step function. This finishes the proof. □

Lemma 4.6. 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 follows important characterization of 1Q sequences on modulo semirings: 1Q sequences on modulo semirings are eventually periodic.

Definition 4.2. Let $a(n)$ be a sequence. $a(n)$ is *eventually periodic* if there exists $N, k \in \mathbb{N}$, such that $\forall_{n \geq N} \ . \ a(n) = a(n+k)$

Lemma 4.7. Suppose we're working over a modulo semiring $\mathbb{Z}_m$. Let φ be a 1Q sentence. Then a sequence defined by φ is eventually periodic.

Proof. Thanks to lemma 4.5 we know that φ can be reduced to simple recognizable step function. Now, thanks to 4.6 we know that language of words for which sequence defined by φ achieves given value k_i is a regular language. There is a finite amount of values this sequence can achieve (at most m). Regular languages are eventually periodic, so from this follows that these sequences also are. (TODO...) □

From this we have a simple corollary, that 1Q sequences are eventually periodic modulo:

Corollary 4.1. 1Q sequences are eventually periodic modulo

Proof. 1Q sequences modulo are the same as 1Q sequences over modulo semiring (4.1). The result then follows directly from lemma 4.7. □

Corollary 4.2. Catalan numbers sequence is 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] 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.