Two-variable first order logic with counting quantifiers: complexity results

Kamal Lodaya[1] and A. V. Sreejith[2]

[1] The Institute of Mathematical Sciences, CIT Campus, Chennai *600113*⋆
[2] Chennai Mathematical Institute, Siruseri, Kelambakkam *603103*,
sreejithav@cmi.ac.in

Abstract. Etessami, Vardi and Wilke [5] showed that satisfiability of two-variable first order logic $\mathrm{FO}^2[<]$ on word models is NEXPTIME-complete. We extend this upper bound to the slightly stronger logic $\mathrm{FO}^2[<, succ, \equiv]$, which allows checking whether a word position is congruent to r modulo q, for some divisor q and remainder r. If we allow the more powerful modulo counting quantifiers of Straubing, Thérien and Thomas [22] (we call this two-variable fragment $\mathrm{FOMOD}^2[<, succ]$), satisfiability becomes EXPSPACE-complete. A more general counting quantifier, $\mathrm{FOUNC}^2[<, succ]$, makes the logic undecidable.

1 Introduction

It is well known that first order logic cannot express counting properties like modulo counting. Two-variable logic cannot express threshold counting. One option is to add counting quantifiers, but there are many ways of doing this, see for example Paris and Wilkie [14], Cai, Fürer and Immerman [3], Straubing, Thérien and Thomas [22], Schweikardt [16]. In this work, we look at extensions of first order logic with a few counting quantifiers.

In the most general setting FOUNC, we have counting terms and allow their comparison. This logic can define addition [9] and is equivalent to the well studied *majority logic* [7]. Hence, by an old result of Robinson [15], in the presence of the unary predicates (for $|\Sigma| \geq 2$), satisfiability is undecidable.

In the more restricted FOMOD, counting can only be done modulo a number. From Büchi's theorem [2], the satisfiability problem can be decided by building an automaton and checking for its emptiness. This is, like FO, nonelementary on the quantifier depth.

So it is of interest to study the counting quantifiers in a weaker framework, such as the two-variable sublogic, studied in Grädel, Otto and Rosen [6], Pacholski, Szwast and Tendera [13], Straubing and Thérien [21]. Etessami, Vardi and Wilke [5], showed that satisfiability of $\mathrm{FO}^2[<]$ over finite words is NEXPTIME-complete. This was further extended to words over arbitrary linear orderings in [11]. On the other hand satisfiability of $\mathrm{FO}^2[<]$ over constant alphabet

⋆ Affiliated to Homi Bhabha National Institute, Anushaktinagar, Mumbai *400094*.

is NP-complete as shown by Weis and Immerman [26]. With only two variables, having successor as an atomic formula increases expressiveness and the logic $\mathrm{FOMOD}^2[<, succ]$ is a well studied class. Straubing, Tesson and Thérien [20] give an algebraic characterisation of this logic. In another paper Straubing and Thérien [21] show that any formula in this logic is equivalent to a formula with all modulo quantifiers inside existential quantifiers. Moreover they show that $\mathrm{FOMOD}^2[<, succ] = \Sigma_2\mathrm{MOD}[<, succ] \cap \Pi_2\mathrm{MOD}[<, succ]$. Interestingly the language $(ab)^*$ can be expressed in the logic $\mathrm{FOMOD}^2[<]$ and $\mathrm{FO}[<]$, but not in $\mathrm{FO}^2[<]$ [21]. Tesson and Thérien [23] review the connections between $\mathrm{FOMOD}^2[<, succ]$, algebra and circuit complexity.

Logic	$\lvert\Sigma\rvert \geq 2$
$\mathrm{FO}^2[<]$	NEXPTIME-complete [5] (over fixed alphabet, NP-complete [26])
$\mathrm{FO}^2[<, succ]$	NEXPTIME-complete [5]
$\mathrm{FO}^2[<, \equiv]$	NEXPTIME-complete
$\mathrm{FO}^2[<, succ, \equiv]$	NEXPTIME-complete
$\mathrm{FOMOD}^2[<]$	EXPSPACE-complete
$\mathrm{FOMOD}^2[<, succ, \equiv]$	EXPSPACE-complete
$\mathrm{FOUNC}^2[<]$	undecidable

Table 1. Complexity of satisfiability of various fragments of two variable logics. Those not cited are the results of this paper.

Our contribution. What is known about the satisfiability problems for these logics over word models? The complexities in the table are tight. In this paper we show that the Etessami, Vardi, Wilke upper bound for $\mathrm{FO}^2[<, succ]$ [5] extends to the slightly stronger $\mathrm{FO}^2[<, succ, \equiv]$, and also that their lower bound holds for $\mathrm{FO}^2[<, \equiv]$ for a constant alphabet size; recall that $\mathrm{FO}^2[<]$ is NP-complete for a constant alphabet [26]. Secondly, we show that $\mathrm{FOMOD}^2[<, succ]$ is EXPSPACE-complete. Our upper bound results assume that the integers in modulo quantifiers and modulo predicates are in binary, whereas our lower bound results assume they are in unary. Thus the complexity does not depend on the representation of integers. Our third contribution is to show that the two variable fragment of counting logic $\mathrm{FOUNC}^2[<]$ is undecidable. The PhD thesis [19] contains many of these results. It also shows that *two-variable* Presburger arithmetic (where $y = x + 1$ and $y = x + x$ are definable) in the presence of unary predicates is undecidable.

Structure of the paper. In the next section, we formally introduce the counting quantifiers and the various logics we look into. In Section 3, we give the upper bound results, namely the EXPSPACE upper bound for $\mathrm{FOMOD}^2[<, succ]$ and the NEXPTIME algorithm for $\mathrm{FO}^2[<, succ, \equiv]$. Section 4 gives corresponding lower bounds and the undecidability of $\mathrm{FOUNC}^2[<]$.

Acknowledgments. We would like to thank four DLT referees and the DLT program committee for their suggestions to improve this paper.

2 Preliminaries

We denote by $\mathbb{N}$ the set of all natural numbers $\{0, 1, 2, \dots\}$. An alphabet Σ is a finite set of symbols. Each letter a of Σ is also the name of a unary predicate which holds at positions which have that letter. We will use a left end-marker for a word, which is outside Σ. The set of nonempty subsets of Σ is denoted by $\mathcal{P}(\Sigma)$. The set of all finite words over Σ is denoted by Σ^*. The length of a word w is denoted by $|w|$. For a word $w \in \Sigma^*$ the notation $w(i)$ denotes the i^{th} letter in w, i.e. $w = w(0)w(1)w(2)\dots w(|w|)$, here $w(0)$ is the left end-marker, $w(1)\dots w(|w|)$ are the letter positions. Let $\mathcal{V} = \{x_1, x_2, \dots\}$ be a set of variables. A *word model* over $(\Sigma, \mathcal{V})$ is a pair (u, s), where $u \in \Sigma^*$ and $s : \mathcal{V} \to \{0, \dots, |u|\}$.

First order logic (FO[<]) over a finite alphabet Σ is a logic which can be inductively built using the following operations.

$$a(x),\ a \in \Sigma \mid x < y \mid x = y \mid \alpha_1 \vee \alpha_2 \mid \neg\alpha \mid \exists x\ \alpha$$

We will also consider other *regular* relations like (a) *succ* : where $succ(x, y)$ says that $y = x + 1$ and (b) $x \equiv r \mod q$, where $q > 1$.

We use the superscript 2 to denote the sublogics which (perhaps repeatedly) use only two variables. For example, the two-variable fragment of FO[<] is denoted by FO^2. Over finite words, FO^2 can talk about occurrences of letters and also about the order in which they appear [17]. The satisfiability problem for a formula α checks if there is a model (in our case, a word model) for it. Complexity of satisfiability problems for two-variable logics are the focus of this study.

Counting quantifiers. We now introduce the syntax for the counting capabilities which extend FO. In the most general setting FOunC[<], we have counting terms and allow their comparison. The additional syntax is

$$\#x(\alpha) \sim x_j \mid \#x(\alpha) \sim \max \mid \#x(\alpha) \sim n,\ n \in \mathbb{N}$$

Here α is an inductively defined FOunC[<] formula, max denotes the last position of a word, $x, x_j \in \mathcal{V}$ and $\sim$ is in $\{<, =, >\}$. The interpretation of the counting term $\#x(\alpha)$ is $|\{i \mid w, s[x \mapsto i] \models \alpha, 1 \le i \le |w|\}|$. Quantification is over letter positions and not over the end-marker. The positions $0, 1, \dots, |w|$ interpret the counts $0, 1, \dots, |w|$ respectively. Counts greater than $|w|$ do not have an interpretation on this word. Hence in the semantics below we do not require a different sort for numbers, a formal difficulty pointed out to us by Anand Pillay.

$$(w, s) \models \#x(\alpha) \sim x_j \Leftrightarrow |\{i \mid (w, s[x \mapsto i]) \models \alpha, 1 \le i \le |w|\}| \sim s(x_j)$$
$$(w, s) \models \#x(\alpha) \sim \max \Leftrightarrow |\{i \mid (w, s[x \mapsto i]) \models \alpha, 1 \le i \le |w|\}| \sim |w|$$
$$(w, s) \models \#x(\alpha) \sim n \Leftrightarrow |\{i \mid (w, s[x \mapsto i]) \models \alpha, 1 \le i \le |w|\}| \sim n,\ n \in \mathbb{N}$$

Modulo counting quantifiers. In the more restricted FO$^{\text{MOD}}$[<], counting terms cannot be compared with variables, but only compared modulo a number. The extended syntax is:

$$\#x(\alpha) \equiv r \mod q$$

The semantics is given as follows for $r, q \in \mathbb{N},\ q > 1$.

$$(w,s) \models \#x(\alpha) \equiv r \mod q \Leftrightarrow |\{i \mid (w, s[x \mapsto i]) \models \alpha, 1 \leq i \leq |w|\}| \equiv r \mod q$$

For example, if a count in a word is 0, it maps to the position 0, and this is interpreted as even because 0, an even number, maps to this position. For later convenience, we define the abbreviated quantifier **Odd** $y(\alpha)$ to stand for $\#y(\alpha) \equiv 1 \mod 2$, and similarly **Even** $y(\alpha)$. Counting the parity of a letter a requires an FO$^{\text{MOD}}$ formula, such as **Even** $y(a(y))$.

We will use MOD[<] when first order quantifiers are not allowed. Also, we use FO$^{\text{MOD}}(q)$[<] (respectively MOD(q)[<]) and FO$^{\text{MOD}}(D)$[<] (resp. MOD(D)[<]) when the divisors in the modulo counting is restricted to be q or from the set of divisors D.

Modulo counting positions. The logic FO[$<, \equiv$] extends FO[<] with the following unary relations.

$$x_i \equiv r \mod q,\ r, q \in \mathbb{N},\ q > 1$$

which is true iff $s(x_i)$ when divided by q leaves a remainder of $r \mod q$. This allows comparison of positions.

Examples. Let us look at some example languages definable in the logics. Even length words over alphabet $\{a, b\}$ can be expressed in FO^2[$<, \equiv$] by $max \equiv 0 \mod 2$, and in MOD$^2(2)$[<] by $\#x(a(x) \vee b(x)) \equiv 0 \mod 2$. By refining this sentence with the one below, the language $(ab + ba)^*$ can also be described in FO^2[$<, \equiv$].

$$\forall x \forall y \Big(\big(succ(x,y) \wedge x \equiv 1 \mod 2 \big) \Rightarrow \big((a(x) \Rightarrow b(y)) \wedge (b(x) \Rightarrow a(y)) \big) \Big)$$

The regular language which allows at most k more a's than b's in every prefix can be defined in first order logic [24]. The simple FO$^{\text{UNC}}$[$<, +k$] sentence below can be written using the successor relation and one additional variable. We do not know whether the language can be defined in FO$^{\text{UNC}^2}$[$<, succ$]. A similar example was brought to our notice by Diego Figueira.

$$\forall x \exists y \Big(\big(\#y\big(y \leq x \wedge a(y)\big) = y \big) \wedge \big(y \leq \#y\big(y \leq x \wedge b(y)\big) + k \big) \Big)$$

The FO$^{\text{UNC}^2}$[<] sentence below defines the nonregular context-free language $\{a^n b^n \mid n \geq 1\}$.

$$\exists x \exists y \Big(\big(y = \#y\big(y \leq x \wedge a(y)\big) \wedge y = \#y\big(y \leq x \wedge (a(y) \vee b(y))\big) \big) \wedge \big(y = \#y\big(y > x \wedge b(y)\big) \wedge y = \#y\big(y > x \wedge (a(y) \vee b(y))\big) \big) \Big)$$

Sizes. The size of a formula is defined inductively as usual. We use binary-size and unary-size to mean that the length of natural number constants is counted as written in binary and unary respectively. For example the number ten (10 in the decimal notation we use) has binary-size 4 and unary-size 10.

3 Upper bounds via linear temporal logic

Temporal logic. The temporal logic UTL over the set of propositions A is the logic with the set of formulas closed under Boolean operations, and including a when a is a letter in A, and $\mathsf{F}\varphi, \mathsf{P}\varphi, \mathsf{X}\varphi, \mathsf{Y}\varphi$ when φ is a formula. To state the semantics fix a word $u \in \mathcal{P}(A)^*$. A position $i \leq |u|$ satisfies the formula a if i is labeled with the letter a, the formula $\mathsf{X}\varphi$ (*resp.* $\mathsf{Y}\varphi$) if position $i+1$ (*resp.* $i-1$) satisfies formula φ, and the formula $\mathsf{F}\varphi$ (*resp.* $\mathsf{P}\varphi$) if there is a position $i \leq j \leq |u|$ (*resp.* $i \geq j$) that satisfies the formula φ. The semantics for Boolean connectives are defined in the usual way. The language of the formula φ is the set of all $u \in \mathcal{P}(A)^*$ that satisfy φ.

Modulo counting temporal logic. The logic UTLmod extends UTL with the following modulo counting operators.

$$MOD^P_{r,q}\varphi \mid MOD^F_{r,q}\varphi$$

For a word $u \in \mathcal{P}(A)^*$, formula $MOD^P_{r,q}\varphi$ (*resp.* $MOD^F_{r,q}\varphi$) is satisfied at a position $i \leq |u|$ if the number of positions $j \leq i$ (*resp.* $j \geq i$) which satisfy φ is $r \mod q$.

The logic UTLmod gives a linear time translation into $\mathrm{FOmod}^2[<, succ]$. There is an *exponential time* translation in the reverse direction.

Lemma 1. *[5, 18] For an* $\mathrm{FOmod}^2[<, \mathrm{succ}]$ *formula* α *of quantifier depth* d *and binary-size* n *there exists an* UTLmod *formula* α' *of operator depth* $2d$ *and binary-size at most* $O(2^n)$*, such that* α *and* α' *accept the same set of word models. Moreover this translation can be done in exponential time.*

From our earlier work, we know that UTLmod satisfiability is in Pspace.

Theorem 2. *[10] Satisfiability of* UTLmod *is* Pspace*-complete.*

Combining Lemma 1 with the above Theorem we get:

Theorem 3. $\mathrm{FOmod}^2[<, succ]$ *satisfiability is in* Expspace.

Length counting temporal logic. For $\mathrm{FO}^2[<, succ, \equiv]$ we can do better. For this, we introduce the logic UTLlen that extends UTL with a restricted modulo counting operator.

$$MOD^P_{r,q}\ true \mid MOD^F_{r,q}\ true$$

Note that in this logic, we can measure the distance of a position (modulo some number) from the start or end of a word.

We next show that a satisfiable UTLLEN formula φ has a model which is exponential in the modality depth and number of propositions (the size of the formula is irrelevant) and polynomial in $lcm(\varphi)$. Here, $lcm(\varphi)$ stands for the least common multiple of the integers q which occur as divisors in a modulo predicate in φ. Following Etessami et al [5] we define an UTLLEN formula φ to be of depth (k, k') if its $\{\mathsf{F}, \mathsf{P}\}$ depth is k and its $\{\mathsf{X}, \mathsf{Y}\}$ depth is k'.

For a word w and a position i, the (k, k', m)-type of i in w is the set of all UTLLEN(m) formulas of depth (k, k') that hold in w at i. The following lemma says that the question of satisfiability can be reduced to counting the number of (k, k', m)-types possible in a word. Let $T(k, k', m)$ be the maximum number of distinct (k, k', m)-types possible in a word. It can be counted inductively.

Lemma 4. *For all k, k', m, $T(k+1, k', m) \leq (2T(k, k', m) + 1)T(0, k', m)$.*

Proof (following [5], Lemmas 2,5). Let w be a word. Then the $(k+1, k', m)$-type at position i in w is uniquely given by the $(0, k', m)$-type at i, the (k, k', m)-types that occur to its right and the (k, k', m)-types that occur to its left. □

A "snipping" lemma gives a small model property for formulas in UTLLEN.

Lemma 5. *Let $m \in \mathbb{N}$ and let φ be an* UTLLEN*(m) formula of depth (k, k'). Then if φ is satisfiable, it is satisfiable in a model of size $T(k, k', m) + 1$.*

Proof (following [5], Lemma 4). Let $w = u_0 u_1 \dots u_n$ be a model for φ and let $n > T(k, k', m) + 1$. Then there exists $i < j \leq n$ such that the (k, k', m)-type at positions i and j are the same. The word $\hat{w} = u_0 u_1 \dots u_i u_{j+1} \dots u_n$ obtained by removing the intervening portion continues to be a model for φ. □

Lemma 6 now shows that a satisfiable UTLLEN formula φ has a model which is exponential in the operator depth and number of propositions (the size of the formula is irrelevant). The divisors contribute a multiplicative factor.

Lemma 6. *Let φ be an* UTLLEN *formula of operator depth d and number of propositions p. If φ is satisfiable, it has a satisfying model of size $O(lcm(\varphi)^{2d^2} 2^{2pd^2})$.*

Proof. Let $m = lcm(\varphi)$ and let w be a word model over p propositions and let the depth of φ be (k, k'). The lemma follows from Lemma 5 if we can show that the number of (k, k', m)-types is bounded by $m^{2d^2} 2^{2pd^2}$. The $(0, k', m)$-type at a position depends on the current position and k' positions to its left and k' positions to its right. Each position satisfies some subset of propositions and $(i \mod m)$ for some $i < m$. Thus $T(0, k', m)$ is bounded by $(m2^p)^{2k'+1}$. Hence $T(k, k', m) \leq (2T(k-1, k', m) + 1)(m2^p)^{2k'+1} = O((m2^p)^{2kk'})$.

The above small model property for UTLLEN gives us that:

Theorem 7. $\mathrm{FO}^2[<, \mathrm{succ}, \equiv]$ *satisfiability is in* NEXPTIME.

Proof. Lemma 1 shows that for every $\mathrm{FO}^2[<, succ, \equiv]$ formula α, there exists an UTLLEN formula α' such that α and α' have the same satisfying models. Moreover, if the quantifier depth of α is d, then the operator depth of α' is $2d$. Lemma 6 shows that every satisfying UTLLEN formula α' of operator depth $2d$ has a satisfying model of size $s = O(lcm(\alpha)^{4d^2} 2^{4pd^2})$. A NEXPTIME machine can guess this model and verify it in time $s^2 \times |\alpha|$.

4 Lower bounds via tiling problems

The lower bound results in this section are shown by reducing from Tiling problems. We define the required Tiling problems now.

Tiling problems. A tiling system is a tuple $\mathfrak{S} = (\mathfrak{T}, \mathfrak{R}, \mathfrak{D})$, where $\mathfrak{T}$ is a finite set of tiles, $\mathfrak{R} \subseteq \mathfrak{T} \times \mathfrak{T}$ and $\mathfrak{D} \subseteq \mathfrak{T} \times \mathfrak{T}$ are, respectively, the right (horizontal) and down (vertical) adjacency relations. A tiling problem is the tuple $(\mathfrak{S}, n, top_1, ..., top_n, bot)$, where $n \in \mathbb{N}$ and $top_1, ..., top_n, bot \in \mathfrak{T}$. A tiling of an $m \times k$ grid $G \subseteq \mathbb{N}^2$ is a mapping $\tau : G \to \mathfrak{T}$ respecting the right and down relations, that is, whenever $(i, j+1)$ or $(i+1, j)$ is in G, we have $\mathfrak{R}(\tau(i, j+1), \tau(i, j))$ or $\mathfrak{D}(\tau(i+1, j), \tau(i, j))$, as the case may be.

We give below two versions of the tiling problem $(\mathfrak{S}, n, top_1, ..., top_n, bot)$ corresponding to EXPSPACE and NEXPTIME Turing machines respectively [4].

Rectangle tiling problem. Do there exist an m and a tiling of an $m \times 2^n$ grid such that the first n tiles in the top row are $top_1, ..., top_n$ in order and there exists a tile *bot* in the bottom row?

Proposition 8. *[4] There exists a tiling system $\mathfrak{S} = (\mathfrak{T}, \mathfrak{R}, \mathfrak{D})$, such that its Rectangle tiling problem $(\mathfrak{S}, n, top_1, ..., top_n, bot)$ is* EXPSPACE*-complete.*

Square tiling problem. Does there exist a tiling of a $2^n \times 2^n$ grid, such that the first n tiles in the top row are $top_1, ..., top_n$ in order and there exists a tile *bot* in the bottom row?

Proposition 9. *[4] There exists a tiling system $\mathfrak{S} = (\mathfrak{T}, \mathfrak{R}, \mathfrak{D})$, such that its Square tiling problem $(\mathfrak{S}, n, top_1, ..., top_n, bot)$ is* NEXPTIME*-complete.*

4.1 Modulo counting is Expspace-hard

We show that satisfiability of $\mathrm{FOMOD}^2[<]$ is EXPSPACE-hard by reducing from the EXPSPACE-complete Rectangle tiling problem. The following lemma shows that $x \equiv y \mod 2^n$ is definable by a $\mathrm{MOD}^2[<]$ formula, which allows us to assert the down relation of the tiling system.

Lemma 10. *There is a polynomial time algorithm which given an $n \in \mathbb{N}$ outputs $\mathrm{Cong}_1^n(x, y)$ in $MOD^2[<]$, where $\mathrm{Cong}_1^n(x, y)$ is of binary-size $O(n)$ and quantifier depth 2 such that $\mathrm{Cong}_1^n(x, y)$ is true if and only if $x \equiv y \mod 2^n$.*

There is also a formula $\mathrm{Cong}_2^n(x, y) \in MOD^2(2)[<]$ of unary-size $O(n^2)$ and quantifier depth n^2, such that $\mathrm{Cong}_2^n(x, y)$ is true iff $x \equiv y \mod 2^n$.

Proof. We first give Cong_1^n which is of quantifier depth 2. For all $i \leq n$, we give formulas $lsb_i(x)$ such that $lsb_i(x)$ is true if and only if the i^{th} least significant bit of x is 1. $lsb_1(x)$ is true if x is odd and is given by the formula $\mathbf{Odd}\ y(y \leq x)$. For all $i \geq 2$:

$$lsb_i(x) := \mathbf{Odd}\ y\Big(y < x \wedge \big(\#x(x \leq y) \equiv (2^{i-1}-1) \mod 2^{i-1}\big)\Big)$$

The claim is now proved by induction on the number of positions y which satisfy the conditions

$$y < x \text{ and } y \equiv \big(2^{i-1}-1\big) \mod 2^{i-1}. \tag{1}$$

When this number is 0 there is no y which satisfies property (1). This means the i^{th} lsb is 0. For the induction step, assume the claim to be true for a number k. Consider the first position z where the count is $k+1$. Then we know that $z-1 \equiv \big(2^{i-1}-1\big) \mod 2^{i-1}$. This implies that if we add 1 to $z-1$ the i^{th} bit toggles. Since the claim is true when the count is $k+1$, we have by induction that $lsb_1(x)$ is true if x is odd. Now $\mathrm{Cong}_1^n(x,y) := \bigwedge_{i=1}^n \big(lsb_i(x) \Leftrightarrow lsb_i(y)\big)$. The binary-size of $\mathrm{Cong}_1^n(x,y)$ is $O(n)$.

Note that the unary-size of Cong_2^n is exponential in n, because we need to encode numbers 2^i. This can be reduced as follows. We can replace the subformula $x \equiv (2^{i-1}-1) \mod 2^{i-1}$ in lsb_i by $\mathrm{Cong}_2^{i-1}(x, 2^{i-1}-1)$. An inductive replacement will give us a formula $\mathrm{Cong}_2^n \in \mathrm{MOD}^2(2)[<]$ of size $O(n^2)$ and quantifier depth n^2. □

In the above lemma, Cong_1^n has quantifier depth 2 and in binary notation, and Cong_2^n is in unary notation and has quantifier depth polynomial in n. If we want both unary notation and constant quantifier depth, we need to introduce modulo counting over primes and use Chinese remaindering.

Lemma 11. *For every $n > 1$, there is a number $q > 2^n$ and a formula $\mathrm{Cong}_3^n(x,y)$ in $MOD^2[<]$ of unary-size $O(n^4)$ and quantifier depth 1 such that $\mathrm{Cong}_3^n(x,y)$ is true if and only if $x \equiv y \mod q$.*

Proof. Let $p_1, ..., p_n$ be the first n primes and $q = \prod_{i=1}^n p_i$ be their product. Clearly $q \geq 2^n$. For all numbers $x, y \in \mathbb{N}$, Chinese remaindering says that the vector $(x \mod p_1, \ldots, x \mod p_n) = (y \mod p_1, \ldots, y \mod p_n)$ if and only if $x \equiv y \mod q$. The following formula asserts this

$$\mathrm{Cong}_3^n(x,y) := \bigwedge_{j \leq n} \bigvee_{r_j < p_j} \Big(\big(x \equiv r_j \mod p_j\big) \wedge \big(y \equiv r_j \mod p_j\big)\Big)$$

Note that $(x \equiv r_j \mod p_j)$ can be asserted by the following $\mathrm{MOD}^2[<]$ formula, $\big(\#y(y \leq x) \equiv r_j \mod p_j\big)$. By the prime number theorem, asymptotically there are n primes within the first $n \log n$ numbers and hence one can generate the first n primes in time polynomial in n. Therefore, the unary-size of Cong_3^n is $\sum_{j \leq n} \sum_{i \leq p_j} i \leq \sum_{k \leq q} k \leq n^4$. □

We will now go to our EXPSPACE-hardness result. Assume $|\Sigma| \geq 2$. Below we show the hardness for three classes of logics, each depending on the formula Cong_i^n, where $i \in \{1, 2, 3\}$ we choose from the above Lemmas.

Theorem 12. *The satisfiability problem for the following logics over a constant alphabet is* EXPSPACE*-hard.*

1. $\text{FOMOD}^2(2)[<]$ *formulas (using unary notation).*
2. $\text{FOMOD}^2(D)[<]$ *formulas (using binary notation) of quantifier depth* 3*, where* $D = \{2^i \mid i \in \mathbb{N}\}$.
3. $\text{FOMOD}^2[<]$ *formulas (using unary notation) of quantifier depth* 3.

Proof. The proof of the three claims differ only on the use of Cong^n formula and therefore we follow the same proof for all the three claims. We show EXPSPACE hardness by reducing from the EXPSPACE-complete Rectangle tiling problem $\mathfrak{I} = (\mathfrak{S}, n, top_1, \ldots, top_n, bot)$ where $\mathfrak{S} = (\mathfrak{T}, \mathfrak{R}, \mathfrak{D})$ and $\mathfrak{T} = \{T_1, \ldots, T_t\}$ given by Proposition 8.

We give a polynomial time algorithm which when given the tiling problem $\mathfrak{I}$ outputs the formula $\psi_\mathfrak{I}$ such that there is a tiling for $\mathfrak{I}$ if and only if $\psi_\mathfrak{I}$ is satisfiable. The alphabet for $\psi_\mathfrak{I}$ is $\Sigma = \mathfrak{T} \times \mathcal{P}(\mathfrak{T}^{Dn}) \times \mathcal{P}(\mathfrak{T}^{Rt})$, where $\mathfrak{T}^{Dn} = \{T_1^{Dn}, \ldots, T_t^{Dn}\}$ and $\mathfrak{T}^{Rt} = \{T_1^{Rt}, \ldots, T_t^{Rt}\}$ are two copies of $\mathfrak{T}$. Note that we are overriding the symbol $\mathfrak{T}$ to mean both tiles and part of the alphabet. It will be clear from the context of the proof what we refer to.

We associate a word model $w_\tau \in \Sigma^*$ with a tiling τ such that τ is a tiling for $\mathfrak{I}$ iff $w_\tau \models \psi_\mathfrak{I}$. In fact every position in w_τ contains atmost 2 letters from $\mathfrak{T}^{Dn}$ and atmost 2 letters from $\mathfrak{T}^{Rt}$. We denote by $w_\tau(i, j)$ the letter at the $(i-1)2^n + j^{th}$ position in w_τ. We will ensure that w_τ will satisfy the property $\tau(i, j) = T_l \Leftrightarrow w_\tau(i, j) \in T_l \times \mathcal{P}(\mathfrak{T}^{Dn}) \times \mathcal{P}(\mathfrak{T}^{Rt})$.

The formula ψ_I is written as a conjunction of the formulas $\psi_{init}, \psi_{final}, \psi_{next}$ and $\psi_{constraints}$ describing the initial configuration, the final configuration, the next move, and the tiling constraints respectively. The formula for the initial configuration, ψ_{init} is the conjunction of $\alpha_1, ..., \alpha_n$, where α_i says that the i^{th} cell in the first row contains the tile top_i. This is encoded by saying that the first location x which satisfies $x \equiv i \ (mod\ 2^n)$ is the i^{th} cell in the first row.

$$\alpha_i := \forall x \left(\left(\text{Cong}^n(x, i) \wedge \forall y < x \ \neg \text{Cong}^n(y, i) \right) \implies top_i(x) \right)$$

Cong^n denotes one of Cong_j^n, where $j \in \{1, 2, 3\}$ (comes from either Lemma 10 or Lemma 11). Similarly, ψ_{final} is given by saying $\exists y \ bot(y)$. We also need to ensure that there is exactly one tile $T_k \in \mathfrak{T}$ in a cell. This is asserted by a sentence $\psi_{constraints}$ in $\text{FO}^2[<]$. The hardest part of the reduction is to ensure that the relations down $\mathfrak{D}$ and right $\mathfrak{R}$ are respected in the word model. This is given by the sentence ψ_{next} which is a conjunction of the formulae ψ_{down} and ψ_{right}.

We will now explain how the down constraints are respected. Let us assume $T_k \in \tau(i, j)$ and the down constraint $\mathfrak{D}(T_k, T_l)$ is true. We need to now assert that $w_\tau(i + 1, j)$ contains T_l. The idea is to count modulo 2, the number of

occurrences of T_l^{Dn} in all cells above i and in the same column. That is, we count the size of the set $\{k \mid T_l^{Dn} \in \tau(k,j), k < i\}$. If this count is even then we force T_l^{Dn} to be true at $w_\tau(i,j)$, otherwise we force T_l^{Dn} to be false. This ensures that the count $\{k \mid T_l^{Dn} \in \tau(k,j), k \leq i\}$ is odd. For other tiles we ensure that the count is even. We preserve this invariant at every cell. Hence the tile at $(i+1,j)$ can be determined by looking at the counts for every tile T_l^{Dn} and setting that tile whose count is odd. The following formula says that in the column strictly above x, there is an even number of occurrences of T_l^{Dn}.

$$\phi_l(x) := \textbf{Even}\ y\Big(T_l^{Dn}(y) \wedge y < x \wedge \mathrm{Cong}^n(x,y)\Big)$$

Now if we want to transfer the information that the cell right below x has to contain letter T_l, we set the count of T_l^{Dn} on this column above and including position x to be odd. This can be asserted by the formula, $\phi_l(x) \Leftrightarrow T_l(x)$. The following formula $\psi_1(x)$ transfers this information by taking into consideration the down constraints.

$$\psi_1(x) := \bigwedge_{k=1}^{t} \left(T_k(x) \implies \left(\bigvee_{(T_k,T_l)\in\mathfrak{D}} \Big(\phi_l(x) \wedge \bigwedge_{j\neq l} \neg\phi_j(x)\Big)\right)\right)$$

Now we need to set the tiles at $w_\tau(i+1,j)$ by looking at the count of T_l^{Dn} strictly above and in the same column as x. The following formula $\psi_2(x)$ says that if you see an odd number of occurrences of the letter $T_l^{Dn} \in \mathfrak{T}^{Dn}$ in the column strictly above x, then we set letter $T_l \in \mathfrak{T}$ to be true at x.

$$\psi_2(x) := \bigwedge_{l=1}^{t} \Big(\textbf{Odd}\ y\big(T_l^{Dn}(y) \wedge y < x \wedge \mathrm{Cong}^n(x,y)\big)\Big) \implies T_l(x)$$

The formula ψ_{down} is a conjunction of the formulas ψ_1 and ψ_2. A similar formula ψ_{right} using the letters $\mathfrak{T}^{Rt}$ can assert that the right relations $\mathfrak{R}$ are ensured. □

4.2 Modulo predicates are harder than linear order

We now show that $\mathrm{FO}^2[<,\equiv]$ is NEXPTIME-hard even for a constant alphabet, as opposed to $\mathrm{FO}^2[<]$ being NP-complete [26].

Theorem 13. $\mathrm{FO}^2[<,\equiv]$ *satisfiability is* NEXPTIME*-hard (constant alphabet size).*

Proof. We reduce from the NEXPTIME-complete Square tiling problem given by Proposition 9. We introduce $2n$ distinct primes, $p_1, ..., p_n$ (for encoding row index), and $q_1, ..., q_n$(for encoding column index). These primes can encode any cell (i,j). One now writes a formula $\alpha_{down}(x)$ which asserts there exists a y such that the row index of y is one more than that of x and the column index of x and y are the same. The formula can also specify that y should satisfy the down constraints. Similarly one can write a formula to force the right constraints. It is easy to write the initial and final conditions. □

4.3 General counting is undecidable

We show that the satisfiability problem of FOUNC$[<, succ]$ with just two variables is undecidable. Grädel, Otto and Rosen [6] had showed this over graphs. Here we show that it is undecidable even over words.

Theorem 14. *Satisfiability of* FOUNC$^2[<, succ]$ *over words is undecidable.*

Proof. Recall that a 2-counter automaton $M = (Q, \to, q_0, q_n)$ is a finite automaton over the alphabet $A = \{inc_k, dec_k, zero_k \mid k = 1, 2\}$ with both counters initially set to zero. Reachability over 2-counter automata is undecidable [12]. A valid run $q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \ldots \xrightarrow{a_n} q_n$ on the word $a_0 a_1 \ldots a_n$ satisfies the property that for all i from 0 to n, there is an a_i-labeled transition from q_{i-1} to q_i and a_i is enabled at q_{i-1}. Here are the enabling conditions for $k = 1, 2$ which enforce the semantics of counters: inc_k is always enabled, $zero_k$ is enabled if the count of inc_k labels for $j < i$ equals the count of dec_k labels for $j < i$, and dec_k is enabled if the count of inc_k labels for $j < i$ exceeds that of dec_k labels for $j < i$. A zero test on counter 1 at position x is written as follows $\exists y(y = \#y(y < x \wedge inc_1(y)) \wedge y = \#y(y < x \wedge dec_1(y)))$. Given these conditions, it is easy to see that reachability from q_0 to q_n can be expressed by an FOUNC$^2[<, succ]$ formula over the monadic predicates $Q \cup A$. □

5 Outlook

In an earlier paper [10], we studied the effect of adding modulo counting to linear temporal logic LTL. In this paper we carried out the same effort for two-variable first order logic FO^2, also in the presence of counting quantifiers and unary predicates.

Over words, the logic FOMOD [1] is a strict subset of monadic second order logic; the latter is pleasant to use and has a well-developed theory [2, 25]. The main advantage of the modulo counting logics is that they directly represent numbers using standard binary notation and the two variable fragment provide an elementary decision procedure. This is also the case if we add threshold counting quantifiers [8]. We can also add both kinds of quantifiers at the cost of an extra exponent, but we do not know whether this is necessary. We also leave open the complexity of decidability in modulo counting logic MOD$[<]$.

In this paper we show that once we add unary predicates (in other words, a small alphabet of letters), even over two variables, general counting quantifiers bring undecidability. In the absence of unary predicates, Presburger logic is well known to be decidable, also in the presence of counting quantifiers [16].

References

1. Barrington, D.A.M., Immerman, N., Straubing, H.: On uniformity within NC^1. J. Comp. Syst. Sci. 41(3), 274–306 (Dec 1990)

2. Büchi, J.R.: Weak second-order arithmetic and finite automata. Z. Math. Logik Grundl. Math. 6, 66–92 (1960)
3. Cai, J., Fürer, M., Immerman, N.: An optimal lower bound on the number of variables for graph identification. Combinatorica 12(4), 389–410 (1992)
4. van Emde Boas, P.: The convenience of tilings. In: Sorbi, A. (ed.) Complexity, logic and recursion theory, pp. 331–363. CRC Press (1997)
5. Etessami, K., Vardi, M.Y., Wilke, T.: First-order logic with two variables and unary temporal logic. Inf. Comput. 179(2), 279–295 (2002)
6. Grädel, E., Otto, M., Rosen, E.: Undecidability results on two-variable logics. Archive for Mathematical Logic 38, 213–354 (1999)
7. Krebs, A.: Typed semigroups, majority logic, and threshold circuits. Ph.D. thesis, Universität Tübingen (2008)
8. Krebs, A., Lodaya, K., Pandya, P., Straubing, H.: Two-variable logic with a between relation. In: Proc. 31st LICS (2016)
9. Lautemann, C., McKenzie, P., Schwentick, T., Vollmer, H.: The descriptive complexity approach to LOGCFL. J. Comp. Syst. Sci 62(4), 629–652 (2001)
10. Lodaya, K., Sreejith, A.: LTL can be more succinct. In: Proc. 8th ATVA, Singapore. LNCS, vol. 6252, pp. 245–258 (2010)
11. Manuel, A., Sreejith, A.: Two-variable logic over countable linear orderings. In: 41st MFCS, Kraków. pp. 66:1–66:13 (2016)
12. Minsky, M.L.: Computation: Finite and Infinite Machines. Prentice-Hall (1967)
13. Pacholski, L., Szwast, W., Tendera, L.: Complexity of two-variable logic with counting. In: 12th LICS, Warsaw. pp. 318–327. IEEE (1997)
14. Paris, J., Wilkie, A.: Counting Δ_0 sets. Fundam. Math. 127 pp. 67–76 (1986)
15. Robinson, R.M.: Restricted set-theoretical definitions in arithmetic. Proc. Amer. Math. Soc. 9, 238–242 (1958)
16. Schweikardt, N.: Arithmetic, first-order logic, and counting quantifiers. ACM Trans. Comput. Log. 6(3), 634–671 (2005)
17. Schwentick, T., Thérien, D., Vollmer, H.: Partially-ordered two-way automata: A new characterization of DA. In: Developments in Language Theory, 5th International Conference, DLT. pp. 239–250 (2001)
18. Sreejith, A.V.: Expressive completeness for LTL with modulo counting and group quantifiers. Electr. Notes Theor. Comput. Sci. 278, 201–214 (2011)
19. Sreejith, A.V.: Regular quantifiers in logics. Ph.D. thesis, HBNI (2013)
20. Straubing, H., Tesson, P., Thérien, D.: Weakly iterated block products and applications to logic and complexity. Int. J. Alg. Comput. 20(2), 319–341 (2010)
21. Straubing, H., Thérien, D.: Regular languages defined by generalized first-order formulas with a bounded number of bound variables. Theory Comput. Syst. 36(1), 29–69 (2003)
22. Straubing, H., Thérien, D., Thomas, W.: Regular languages defined with generalized quantifiers. Inf. Comput. 118(3), 389–301 (1995)
23. Tesson, P., Thérien, D.: Logic meets algebra: the case of regular languages. Log. Meth. Comp. Sci. 3(1) (2007)
24. Thomas, W.: Classifying regular events in symbolic logic. J. Comput. Syst. Sci. 25(3), 360–376 (1982)
25. Thomas, W.: Languages, automata and logic. In: Handbook of formal language theory, vol. III, pp. 389–455. Springer (1997)
26. Weis, P., Immerman, N.: Structure theorem and strict alternation hierarchy for FO^2 on words. Log. Meth. Comp. Sci. 5(3) (2009)