# Set Automata and Limits of Decidability of Two-Variable Logic on Data Words

**Shibashis Guha**
Tata Institute of Fundamental Research

**Amaldev Manuel**
Indian Institute of Technology Goa

**S P Rishal**
Sarva Labs

## Abstract

We extend the two-variable logic on data words [4] with guarded regular binary predicates of the form $\widetilde{L}(x, y)$ that is true if positions $x$ and $y$ have the same data value and the factor strictly between $x$ and $y$ is in the regular language $L$. We characterise the class of monoids for which the extension of the two-variable logic with guarded predicates recognised by the monoid is decidable, namely the class of idempotent monoids whose two-sided ideals are linearly ordered, called linear bands. For this, we introduce an automata formalism, set automata, that is equivalent to the class automata of Bojańczyk and Lasota and thus has an undecidable emptiness problem. The set updates used in the automaton form a semigroup of relations. We identify a subclass of set automata called ordered quasi-normal set automata that has a decidable emptiness problem by reduction to the emptiness problem of ordered multicounter automata. We show that the two-variable logic extended with guarded regular predicates recognised by a monoid $M$ is expressively equivalent to a quasi-normal set automaton with the monoid of relations $M$. In particular, if $M$ is a linear band then the resulting automaton is ordered, and the decidability result follows.

**2012 ACM Subject Classification** Theory of computation → Logic and verification; Theory of computation → Formal languages and automata theory

**Keywords and phrases** Data words, Two-variable FO, Class automata, Finite semigroups, Decidability

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2026.194

**Category** Regular Paper

**Related Version** *Full Version*: http://arxiv.org/abs/2605.09077

**Funding** *Shibashis Guha*: Partially supported by CEFIPRA project no. 7302-I and by the Department of Atomic Energy, Government of India, under project no. RTI4014

## 1 Introduction

A *data word* is a finite sequence of pairs $(a_i, d_i) \in \Sigma \times \mathcal{D}$ where $\Sigma$ is a finite alphabet and $\mathcal{D}$ is an infinite domain of *data values* that can be tested only for equality. A set of data words $L$ is called a *data language*. In most cases (including this work) $L$ is assumed to be closed under permutations of the set $\mathcal{D}$, that is, every occurrence of some data value $d_i$ can be replaced with another data value $d_j \neq d_i$ as long as such a mapping is a permutation. Thus the specific data values considered are not important to the study and we only capture properties involving equality of data values. For our purposes, $L$ can be represented as a collection of relational first-order structures of the form $([n], (a)_{a\in\Sigma}, <, +1 \sim, \pm 1)$ where $[n]$ and $<$ denote the set $\{1, \ldots, n\}$ and the natural order on it, the unary predicates $(a)_{a\in\Sigma}$ denotes the labelling by $\Sigma$, and $\sim$ is the equivalence relation on positions given by data values, i.e., $i \sim j$ if $d_i = d_j$. The relation $\pm 1$ denotes the *class successor* relation, i.e., $i \pm 1 = j$, if


$i \sim j$ and the positions strictly between $i$ and $j$ are not equivalent to them. The *class* of $i$ is the equivalence class of $i$ under the relation $\sim$, i.e., the set of all positions labelled with the data value $d_i$.

The primary direction in the study of data words has been identifying suitable notions of automata and logics for them (See [2, 21] and the survey [25]). The models and logics in the literature can broadly be classified into two families: temporal logics (correspondingly, the register automata family) and first-order logics (data automata family). The prominent member of the first family is the freeze-LTL (i.e., LTL with registers) of [10], while the most important logic of the latter is the two-variable first-order logic of [4]. In this work we present a decidable, strict extension of the two-variable logic that incorporates some aspects of temporal logics. First we recall the two-variable logic in some detail.

For a vocabulary $\tau$ of unary and binary predicates let $\mathrm{FO}^2[\tau]$ denote the first-order formulas using the predicates from $\tau$ and the variables $x$ and $y$. The logic $\mathrm{EMSO}^2[\tau]$ consists of formulas of the form $\exists X_1 \dots \exists X_n\, \varphi$ with $\varphi \in \mathrm{FO}^2[\overline{X}, \tau]$, where $\overline{X}$ stands for the monadic second-order predicates $X_1, \dots, X_n$ that quantify over sets of positions. Certainly, it is natural to consider the first-order logic (FO) with the vocabulary $(\Sigma, <, \sim)$ on data words. However, as one would expect, the satisfiability problem of the logic $\mathrm{FO}[\Sigma, <, \sim]$ is undecidable [4]. The undecidability persists even if the number of variables is restricted to three. This prompted the study of the two-variable fragment of the logic. It is shown in [4] that the satisfiability problem of $\mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1]$ is decidable over data words in non-elementary time. The proof is by translating the formulas into an automaton formalism called data automata and showing the decidability of the emptiness problem of data automata. As far as the study of logics over data words is concerned this result is a milestone and despite nearly two decades of active research, $\mathrm{FO}^2$ remains as one of the yardstick logics with which any new formalism is measured.

Next we look at some examples. The *string projection* of a data word $w$ is the word $\mathrm{str}(w) = a_1 \cdots a_n$. Let $1 \le i_1 < i_2 < \cdots < i_k \le n$ be a class of $w$. The string projection of this class is the word $u = a_{i_1} \cdots a_{i_k}$. We say $u$ is a class of $w$ to mean a class whose string projection is $u$.

▶ **Example 1.** Let $\Sigma = \{\iota, \delta, z\}$. We can think of $\iota$ and $\delta$ as increments and decrements of a counter and $z$ as a zero-test.

1. Let $L_1$ be the set of all data words where each class is of the form $\iota\delta$ or $z$. This language is defined by the formula $\varphi_1$ where

$$\varphi_1 := \forall x \forall y\, (x < y \wedge x \sim y \to \iota(x) \wedge \delta(y)) \wedge ((x \sim y \to x = y) \to z(x)) \tag{1}$$

2. Let $L_2$ consist of all data words satisfying the property: there is no position labelled by $z$ between each $\iota$ and $\delta$ of the same class. Note that $z$ could be in a different class than that of the $\iota$ and $\delta$.

The language $L_2$ is not definable in $\mathrm{FO}^2$ as noted in [25]. In fact it is not recognised by the data automata of [4], that is equivalent to the logic $\mathrm{EMSO}^2[\Sigma, <, x+1, \sim, \pm 1]$. However, $L_2$ is definable in freeze-LTL with only forward-looking modalities (Next and Until), a logic with a decidable satisfiability problem (as a trade-off this logic cannot define $L_1$, see [25]).

To define $L_2$ in $\mathrm{FO}^2$, we extend the two-variable logic with the below predicates.

▶ **Definition 2** (Regular Predicate). *A* regular predicate *over the alphabet $\Sigma$ is a binary relation of the form $L(x, y)$ where $L \subseteq \Sigma^*$ is a regular language. A data word $w$ with the string projection $str(w) = a_1 \cdots a_n \in \Sigma^*$ at the pair of positions $(i, j)$, for $i, j \in [n]$, satisfies*

*the predicate* $L(x,y)$ *if* $i<j$ *and the factor* $a_{i+1}\cdots a_{j-1}$ *given by the interval* $(i,j)$ *is in* $L$. *A* guarded *regular predicate* $\widetilde{L}(x,y)$ *is equivalent to the formula* $x\sim y\wedge L(x,y)$.

For a regular expression $r$, let $[\![r]\!]$ denote the language defined by it. Using regular predicates, the language $L_2$ can be defined in $\mathrm{FO}^2$ by the below formula.

$$\forall x\forall y\left((\iota(x)\wedge\delta(y)\vee\delta(x)\wedge\iota(y))\wedge x\sim y\to\neg\widetilde{[\![\Sigma^*z\Sigma^*]\!]}(x,y)\right). \tag{2}$$

The below example illustrates that adding regular predicates to the vocabulary leads to undecidability even when relatively simple regular predicates are used.

▶ **Example 3.** Let $\Sigma'=\{\iota_1,\delta_1,z_1,\iota_2,\delta_2,z_2\}$. We can think of the alphabet as encoding the increments, decrements, and zero-tests of a two-counter machine.

1. Let $L_3$ be the set of all data words satisfying the following property: each class is of the form $\iota_i\delta_i$ or $z_i$, $i\in\{1,2\}$. Clearly $L_3$ is in $\mathrm{FO}^2[\Sigma,<,+1,\sim]$.
2. Let $L_4$ consist of all data words satisfying the following property: for each $i\in\{1,2\}$, there is no position labelled by $z_i$ between each $\iota_i$ and $\delta_i$ of the same class.

From Formula 2, it is not difficult to see that $L_4$ is definable in $\mathrm{FO}^2$ using the guarded regular predicates defined by the languages $\Sigma'^*z_1\Sigma'^*$ and $\Sigma'^*z_2\Sigma'^*$. This is sufficient to encode the run of a two-counter machine as a data word.

The algebraic theory of regular languages where the semigroup recognising a language is considered, allows for properties of regular languages to be understood through the structure of the semigroup recognising it. Motivated by this, in our logic we consider families of regular predicates defined by a semigroup recognising it.

A *semigroup* $S$ is a set with an associative binary operation. It is a *monoid* if the operation has an identity, denoted by 1. All semigroups we consider in this paper are either finite or free semigroups of the form $\Sigma^*$. Let $M$ be a finite monoid. A morphism $h:\Sigma^*\to M$ is a map satisfying $h(ab)=h(a)h(b)$ and $h(\varepsilon)=1$. A language $L\subseteq\Sigma^*$ is *recognised* by a morphism $g:\Sigma^*\to M$, where $M$ is a finite monoid, if there is a finite subset $P\subseteq M$ such that $L=h^{-1}(P)$. We also consider semigroups recognising languages. Let $S$ be a finite semigroup and $S^1$ be the monoid obtained by adjoining an identity 1 to $S$. The monoid morphism $h:\Sigma^*\to S^1$ is *unit-reflecting* if $h^{-1}(1)=\varepsilon$. We say a language $L\subseteq\Sigma^*$ is recognised by $S$ if it is recognised by a unit-reflecting morphism into $S^1$. A monoid $M$ *recognises* a family $\mathcal{F}$ of languages over $\Sigma$ if there is a morphism $h:\Sigma^*\to M$ that recognises each language in the family. Definitions of other semigroup-theoretic notions required for our purposes can be found in Appendix A.

Let $h:\Sigma^*\to M$ be a morphism. The logic $\mathrm{FO}^2[\Sigma,<,+1,\sim,\pm1,\mathcal{F}_h]$ is the extension of the logic with the family $\mathcal{F}_h$ of all regular predicates recognised by $h$. For a monoid $M$, by $\mathrm{FO}^2[\Sigma,<,+1,\sim,\pm1,\mathcal{F}_M]$ we denote the set of formulas $\varphi\in\mathrm{FO}^2[\Sigma,<,+1,\sim,\pm1,\mathcal{F}_h]$ for some morphism $h:\Sigma^*\to M$. The logics $\mathrm{FO}^2[\Sigma,<,+1,\sim,\pm1,\widetilde{\mathcal{F}}_h]$ and $\mathrm{FO}^2[\Sigma,<,+1,\sim,\pm1,\widetilde{\mathcal{F}}_M]$ are defined analogously where the predicates $\widetilde{\mathcal{F}}_h$ used are guarded. The above definitions extend naturally to semigroups and unit-reflecting morphisms. Now, we introduce the $\mathrm{EMSO}^2$ extension of the above logics. For instance, a formula is in $\mathrm{EMSO}^2[\Sigma,<,+1,\sim,\pm1,\mathcal{F}_h]$ if it is of the form $\exists X_1\ldots\exists X_n\,\varphi$ where $\varphi\in\mathrm{FO}^2[\overline{X},\Sigma,<,+1,\sim,\pm1,\mathcal{F}_h]$ and $\overline{X}$ stands for the monadic second-order predicates $X_1,\ldots,X_n$. Note that the morphism $h:\Sigma\times\{0,1\}^n\to M$, for some finite monoid $M$, maps words over the extended alphabet that indicates the interpretations of the $n$ variables in $\overline{X}$.

▶ **Example 4.** The language family $\{\Sigma^*z\Sigma^*\}$ from Equation (2) is recognised by the two element monoid with a zero $U_1=\{1,0\}$ under the usual product operation, with the

morphism $z \mapsto 0$ and $\iota, \delta \mapsto 1$ and accepting set $P = \{0\}$. However, the family $\mathcal{F} = \{\Sigma'^* z_1 \Sigma'^*, \Sigma'^* z_2 \Sigma'^*\}$ is not recognisable by the monoid $U_1$ as both languages cannot be recognised by the same morphism. However $\mathcal{F}$ is recognised by the product monoid $U_1^2 = U_1 \times U_1 = \{(1,1),(1,0),(0,1),(0,0)\}$. under the morphism $z_1 \mapsto (0,1)$, $z_2 \mapsto (1,0)$ with the accepting sets $\{(0,1),(0,0)\}$ and $\{(1,0),(0,0)\}$ respectively.

A key difference in the structure of the two monoids above is that in $U_1$, the two-sided ideals (See Definition 45) are linearly ordered while it is not the case in $U_1^2$ (See Figure 2 and Example 29). Motivated by this, we consider the class of *linear bands*, namely the class of idempotent monoids $M$ whose two-sided ideals are linearly ordered, that is, for each $x, y \in M$, we have $MxM \subseteq MyM$ or $MyM \subseteq MxM$. The idempotency restriction follows from another such language that leads to undecidability (See Proposition 37).

The below example illustrates how guarded regular predicates defined by a semigroup can be used to capture a previously known decidable logic given in [4].

▶ **Example 5** (Nilpotent Predicates)**.** Consider the monoid $\{0, 1, \ldots, n-1, \infty\}, n \geq 1$ with the below operation where $x + y$ is the usual addition over natural numbers.

$$x \cdot y = \begin{cases} x + y & \text{if } x, y \in \{0, 1, \ldots, n-1\} \text{ and } x + y < n, \\ \infty & \text{otherwise.} \end{cases}$$

The unique morphism given by the map $g : \Sigma \mapsto 1$ (and $g : \Sigma \mapsto \infty$ if $n = 1$) recognises the family of languages $\{\Sigma^i \mid 0 \leq i \leq n-1\} \cup \{\Sigma^{\geq n}\}$. Thus the logic $\mathrm{FO}^2[\Sigma, <, \sim, \mathcal{F}_g] \equiv \mathrm{FO}^2[\Sigma, <, +1, +2, +3, \ldots, +n, \sim]$ where $+k$ is a generalisation of the successor relation $+1$ on positions. The decidability of the latter logic is shown in [4].

### Our Contributions

Our main result is as follows.

▶ **Theorem 6.** *Satisfiability of* $\mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \widetilde{\mathcal{F}}_M]$ *formulas over data words is decidable if and only if* $M$ *is a linear band.*

Our decidable fragment is a strict extension of the $\mathrm{FO}^2$ of [4] as the language $L_2$ of Example 1 is definable in the decidable fragment of Theorem 6 but not in the $\mathrm{FO}^2$ of [4]. The guardedness of the regular predicates does not impose a restriction on the expressibility in the following sense: for every formula in $\mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \mathcal{F}_M]$ with regular predicates recognised by a monoid $M$ there is an equivalent formula in $\mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \widetilde{\mathcal{F}}_{M'}]$ with guarded regular predicates recognised by another monoid $M'$. We state our results for the guarded fragment.

To show decidability of the logic, we employ a new automaton formalism called set automata. A *set automaton* $\mathcal{A}$ is a finite state automaton with a fixed number of sets $X = \{X_1, \ldots, X_k\}$ that can store data values. During the run, the automaton can add or remove the current data value from the sets as well as assign a union of sets to each set (i.e. of the form $X_i = \bigcup_{j \in J} X_j$ for $J \subseteq [k]$). The latter kind of updates can be represented as an element of a semigroup of relations on the sets. The acceptance of a run is determined by the state as well as the sets in which the data values are present at the end of the run. The set automaton model is equivalent to the class automata of [6] and is thus undecidable. Furthermore, for each data automaton there is an equivalent set automaton where the only set update used is the identity relation.

We identify a subclass of set automata with decidable emptiness, called ordered quasi-normal set automata. The decidability of emptiness of ordered quasi-normal set automata is shown by a reduction to the emptiness problem of ordered multicounter automata which is known to be decidable [24]. The complexity of the decision procedures for the logic $\mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \widetilde{\mathcal{F}}_M]$ and set automata are non-elementary and is inherited from $\mathrm{FO}^2[\Sigma, <, +1, \sim]$ and data automata.

We obtain decidability of the logic by converting $\mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \widetilde{\mathcal{F}}_M]$ formulas where $M$ is a linear band to ordered quasi-normal set automata. To show undecidability of the logic when $M$ is not a linear band, we use a reduction from the halting problem of two-counter machines which is known to be undecidable [20].

## Related Work

**Automata on Data Words.** There exists numerous extensions of finite state automata with registers [15], pebbles [21], hash tables [2], counters [19], sets [1, 14], etc. to handle data values (See the survey [25]). The bibliography is extensive and we mention only those relevant to our work. Register automata [15] and its variants [5, 10, 21] equip finite state automata with a fixed number of registers for storing data values and compare them by equality. Set augmented finite automata [1], register set automata [14], and history register automata [11] are generalisations of register automata with unbounded registers (or sets in our terminology). The latter two are equivalent to set automata except for the acceptance conditions. Class automata [6] are closely related to data automata and are expressively equivalent to set automata. A subclass of class automata captures the alternating one register automata of [10]. The decidable subclass presented in our work was previously unknown.

**Logics on Data Words.** As for $\mathrm{FO}^2$ on data words, the results in [4] are already mentioned. The related, but weaker logic $\mathrm{EMSO}^2(\Sigma, \sim, +1)$ was studied in [16]. It was shown to be equivalent to a semantic restriction of data automata, called weak data automata. A weakening of MSO logic with data equality tests called rigidly guarded MSO was introduced in [7]. It was shown that it is as expressive as orbit-finite data monoids [3] and its satisfiability is decidable. A $\mu$-calculus that has modalities corresponding to successor, predecessor, class successor and class predecessor relations was introduced in [8].

**Regular Predicates.** Extending logics with regular predicates or modalities is an idea that dates back to [26] and has been used in a number of contexts ever since, see [9, 18]. The relationship between $\mathrm{FO}^2$ with regular predicates and unary regular temporal logic on words is studied in [23].

## Organisation of the Paper

In Section 2 we introduce set automata and its various restrictions namely normal, quasi-normal, and ordered set automata. We also prove the necessary closure properties required to translate $\mathrm{EMSO}^2$ formulas with guarded regular predicates to quasi-normal set automata. Subsequently, in Section 3 we establish the automata-logic connection. Section 4 details the decidability and undecidability results on $\mathrm{EMSO}^2$ with guarded predicates. Section 5 discusses comparisons of set automata with data automata and class automata, and provides a logical characterisation of set automata. In Section 6 we conclude with some open questions and future directions. Standard facts from the theory of Green's relations on finite semigroups, and definitions of basic relations like preorder, partial order, and other related notions are

given in Appendix A. To improve readability, we have omitted some proofs that are provided in the full version [13].

## 2 Set Automata

In this section, we introduce set automata, and the subclasses normal, quasi-normal, and ordered set automata. As our main result, we show that the emptiness problem of ordered quasi-normal set automata is decidable.

### 2.1 Relations and Transformations

For $k > 0$, let $[k]$ denote the set $\{1, \ldots, k\}$. The set of Boolean values is denoted by $\mathbb{2} = \{0, 1\}$. Let $Y$ be a finite ordered set. The power set of $Y$ is denoted as $\mathcal{P}(Y)$. A Boolean column vector $\overline{u} = (u_y)_{y \in Y}$ indexed by $Y$ is an element of $\mathbb{2}^Y$. For a subset $Y' \subseteq Y$, the restriction of $\overline{u}$ to $Y'$ is given by $\overline{u} \restriction_{Y'} = (u_y)_{y \in Y'}$. The set of column vectors over $\mathbb{2}$ indexed by $Y$ is denoted by $\mathbb{2}^Y$. The Boolean operations can be extended to elements of $\mathbb{2}^Y$ pointwise. For $u \in \mathbb{2}^Y$, we denote by $u^c = (1 - u_y)_{y \in Y}$ the complement of $u$. For $y \in Y$, let $e_y \in \mathbb{2}^Y$ denote the vector whose $y$-component is 1 and all other components are 0. The vectors $\{e_y \mid y \in Y\}$ are called unit vectors.

Let $S$ be a set. For a subset $X \subseteq S$, let $\mathbf{1}_X : S \to \mathbb{2}$ denote the characteristic function given by $\mathbf{1}_X(x) = 1$ if $x \in X$ and $\mathbf{1}_X(x) = 0$ otherwise. Extending this notation, for $\overline{X} = (X_y)_{y \in Y} \in \mathcal{P}(S)^Y$ a vector of subsets of $S$ indexed by $Y$, let $\mathbf{1}_{\overline{X}} : S \to \mathbb{2}^Y$ denote the function $x \mapsto (\mathbf{1}_{X_y}(x))_{y \in Y}$. The vector $\mathbf{1}_{\overline{X}}(x)$ is called the characteristic vector of $x$ in $\overline{X}$.

A *binary relation* $\rho$ on a set $Y$ is a subset of the cartesian product $Y \times Y$. We simply write relation instead of binary relation. The *converse* of a relation $\rho$ is the relation $\rho^{-1} = \{(y, x) \mid (x, y) \in \rho\}$ where the order is switched. For a subset $Y' \subseteq Y$, the restriction of $\rho$ to $Y'$ is $\rho \restriction_{Y'} = \rho \cap (Y' \times Y')$. The *image* of an element $x \in Y$ under the relation $\rho$ is the set $\rho(x) \subseteq Y$ given by $\rho(x) = \{y \in Y \mid (x, y) \in \rho\}$. Similarly, the image of a subset $S$ of $Y$ under the relation is given by $\rho(S) = \bigcup_{x \in S} \rho(x)$. A *transformation* on a set is a function from the set to itself. A bijective transformation is called a *permutation*. Given relations $\rho$ and $\sigma$ on the set $Y$, their *product* $\rho\sigma$ is given by $\rho\sigma = \{(x, z) \in Y \times Y \mid \exists y \in Y, (x, y) \in \rho, (y, z) \in \sigma\}$.

▶ **Example 7.** Let $\rho = \{(0,0), (0,1), (1,1)\}$ and $\sigma = \{(0,0), (1,0)\}$ be relations on the set $\{0, 1\}$. Then $\rho\sigma = \{(0,0), (1,0)\}$ and $\sigma\rho = \{(0,0), (0,1), (1,0), (1,1)\}$.

The omission of $\circ$ to denote the product operation is intentional. When $\rho$ and $\sigma$ are functions then their *function composition* $\rho \circ \sigma$ is the function $\rho \circ \sigma = \sigma\rho$ that is different from their product $\rho\sigma$. The product of relations is an associative operation and the relations on a set $Y$ form a monoid with the identity function $\mathrm{id}_Y = \{(y, y) \mid y \in Y\}$ as the identity element. Let $\mathbf{R}_Y$ and $\mathbf{T}_Y$ denote the monoid of relations and transformations on the set $Y$, respectively.

A relation $\rho$ over $Y$ is naturally viewed as a Boolean matrix $M(\rho) = (m_{ij})_{(y_i, y_j) \in Y \times Y}$ where $m_{ij} = 1$ if $(y_i, y_j) \in \rho$ and 0 otherwise. Conversely, for every Boolean matrix $M$ there is a corresponding relation $\rho(M)$ such that $M(\rho(M)) = M$. It is easy to verify that $M(\rho\sigma) = M(\rho) \cdot M(\sigma)$. We denote by $M^T$ the transpose of a matrix $M$.

### 2.2 Set Automaton

▶ **Definition 8** (Set automaton). *A* set automaton $\mathcal{A}$ *is a tuple* $(Q, Y, \Sigma, \Delta, I, F, C)$ *where* $Q$ *is a finite set of states,* $Y$ *is a finite set of names for sets, and* $\Sigma$ *is the input alphabet. The transition relation is given by* $\Delta \subseteq Q \times \Sigma \times \mathbb{2}^Y \times \mathbf{R}_Y \times \mathbb{2}^Y \times \mathbb{2}^Y \times Q$*. The initial and final*

*sets of states are respectively $I \subseteq Q$ and $F \subseteq Q$. Finally, $C \subseteq 2^Y$ is the family of accepting vectors of membership of data values.*

A *configuration* of the automaton $\mathcal{A}$ is a pair $\gamma = (q, \overline{X})$ where $q \in Q$ is a state and $\overline{X} = (X_y)_{y \in Y} \in \mathcal{P}(\mathcal{D})^Y$ is a vector of subsets of data values. A data value $d$ is said to be *present* in $\overline{X}$ if it is in $X_y$ for some $y \in Y$. A configuration is *initial* if $q \in I$ and $X_y = \emptyset$ for each $y \in Y$.

When the automaton is in a configuration $\gamma$, on the input pair $(a, d) \in \Sigma \times \mathcal{D}$, the transition $(p, \ell, \overline{z}, \rho, \overline{u}, \overline{v}, p') \in \Delta$ is applicable if $p = q$, $\ell = a$, and $\overline{z} = \mathbf{1}_{\overline{X}}(d)$, that is, the characteristic vector of membership of the data value $d$ is $\overline{z}$. The *set* or *global update* is given by the relation $\rho \in \mathbf{R}_Y$. The global update $\rho$ transfers the contents of each set $x$ to each set $y \in \rho(x)$. This results in the vector of subsets of data values stored in the sets of $\mathcal{A}$, $\overline{X'} = (X'_y)_{y \in Y}$ where $X'_y = \bigcup_{x \in \rho^{-1}(y)} X_x$. We observe that for each data value $d' \in \mathcal{D}$, the characteristic vector of membership $\mathbf{1}_{\overline{X'}}(d') = M(\rho)^T \cdot \mathbf{1}_{\overline{X}}(d')$ after the global update. The *local updates* are given by the vectors $\overline{u}, \overline{v} \in 2^Y$. The local updates are applied on the contents of the sets given by $\overline{X'}$, with the current data value $d$ added to the sets given by $\overline{u}$ and removed from the sets given by $\overline{v}$. Let $\overline{U}, \overline{V} \in \mathcal{P}(\{d\})^Y$ be such that $\overline{u} = \mathbf{1}_{\overline{U}}(d)$ and $\overline{v} = \mathbf{1}_{\overline{V}}(d)$. This results in the configuration $(p', \overline{X''})$ where $\overline{X''} = (X''_y)_{y \in Y}$ is given by $X''_y = (X'_y \cup U_y) \setminus V_y$ for each $y \in Y$.

A configuration $(q, \overline{X})$ is *accepting* if $q \in F$ and the characteristic vector of each data value present in $\overline{X}$ is in $C$. A *successful* run of $\mathcal{A}$ on a data word $w$ is a sequence of applicable transitions taking the automaton from an initial to an accepting configuration. The *language* of $\mathcal{A}$, denoted as $L(\mathcal{A})$, is the set of all data words $w$ on which $\mathcal{A}$ has a successful run.

Let $\mathcal{U}(\mathcal{A})$ denote the set of global updates used in the transitions of the set automaton $\mathcal{A}$. Let $S_{\mathcal{A}} = \langle \mathcal{U}(\mathcal{A}) \rangle$, called the *update semigroup of* $\mathcal{A}$, be the subsemigroup of relations on $Y$ generated by $\mathcal{U}(\mathcal{A})$. Clearly $S_{\mathcal{A}}$ is a subsemigroup of $\mathbf{R}_Y$.

In the rest of the paper, when dealing with set automata, we do not distinguish between the name of a set and its contents and we directly use the name of the set to refer to its contents (i.e., a subset of data values).

▶ **Example 9.** Consider the language $L_{12} = L_1 \cap L_2$ over the alphabet $\Sigma = \{\iota, \delta, z\}$ where $L_1$ and $L_2$ are the languages described in Example 1. This language is accepted by a set automaton $\mathcal{A}_{12}$ with the sets $Y = \{I, D, Z, S\}$. Initially all of them are empty. Whenever the position is labelled by an $\iota$ the automaton checks if the data value is not present in any of the sets and it is then placed in the set $I$ by local updates and the global update is the identity relation, i.e., it leaves the sets unchanged. Whenever a $\delta$ is read, the data value is ensured to be present in the set $I$ and not in any other set and is then moved to $D$ by local updates. Again, the global update is the identity relation. Whenever a $z$ is read, the data value is ensured to not be present in any of the sets and is added to the set $Z$ by local updates. Further, the global update copies each set to itself, and also $I$ to $S$. This is given by the relation $\rho = \mathrm{id}_Y \cup \{(I, S)\}$. This ensures that whenever a $z$ is encountered the set $I$ is empty provided $S$ is empty at the end of the run. In turn, this ensures that the data word is in $L_2$. At the end the automaton accepts only if the sets $I$ and $S$ are empty. This ensures that all classes are of the form $\iota\delta$ or $z$ and there is no $z$ present between a pair of $\iota$ and $\delta$ of the same class.

## 2.3 Normal Set Automaton

A set automaton is *normal* if it maintains the following invariant throughout the run on each data word: each data value is present in at most one set. This is captured by the following

syntactic restriction.

▶ **Definition 10** (Normal Set Automaton). *A* set automaton *is normal if each of its transitions* $(p, \ell, \overline{z}, \rho, \overline{u}, \overline{v}, q)$ *satisfies the following properties:*

1. *the set update* $\rho$ *is a transformation, and,*
2. $\overline{u}$ *is a unit vector and if* $\overline{u} \neq M^T(\rho) \cdot \overline{z}$ *then* $\overline{v} = M^T(\rho) \cdot \overline{z}$*, otherwise* $\overline{v} = \overline{0}$.

In a normal set automaton $\mathcal{A}$ with sets $Y$, since each global update is a transformation, the update semigroup is a transformation semigroup. Further, for the same reason, data values in each set move to exactly one set. This, in combination with restricted local updates where the current data value can be added to at most one set ensures that each data value is present in at most one set throughout the run of a normal set automaton. Note that in the above definition, the characteristic vector $\overline{z}$ can be assumed to be a unit or zero vector without loss of generality where the transition is not enabled otherwise.

For every relation on a set $Y$, there exists a corresponding transformation on the set $2^Y \setminus \{\bar{0}\}$. A consequence of this is that every set automaton can be normalised.

▶ **Example 11.** Consider the relation $\rho$ on the family of sets $Y = \{Y_1, Y_2\}$ given by $\rho = \{(Y_1, Y_2)\} \cup \mathrm{id}_Y$. This relation corresponds to an equivalent transformation $\rho'$ on the sets $2^Y \setminus \{\bar{0}\}$ where each set represents a subset of sets in $Y$. For instance, the set $Y_{(10)}$ tracks the data values present in the set $Y_1$ but not in $Y_2$ and the set $Y_{(11)}$ tracks the data values present in both the sets $Y_1$ and $Y_2$. The corresponding transformation $\rho'$ on the sets $2^Y \setminus \{\bar{0}\}$ is given as follows:

$$\begin{aligned}\rho' &= \{(Y_{\bar{s}}, Y_{\bar{s}'}) \mid \bar{s}, \bar{s}' \in 2^Y \setminus \bar{0}, \bar{s}' = M^T(\rho) \cdot \bar{s}\} \\ &= \{(Y_{(01)}, Y_{(01)}), (Y_{(10)}, Y_{(11)}), (Y_{(11)}, Y_{(11)})\}.\end{aligned}$$

▶ **Proposition 12.** *For each set automaton* $\mathcal{A}$ *with the family of sets* $Y$ *there is an equivalent normal set automaton* $\mathcal{N}(\mathcal{A})$ *with the family of sets* $2^Y \setminus \{\bar{0}\}$.

We now provide a generalisation of normal set automata.

A set $y$ of a set automaton is said to be *stable* if $\rho(y) = \{y\} = \rho^{-1}(y)$ for each global update $\rho$ used in the transitions of the automaton.

Let $\mathcal{A} = (Q, Y, \Sigma, \Delta, I, F, C)$ be a set automaton. The *restriction* of $\mathcal{A}$ to the sets $Y' \subseteq Y$ denoted as $\mathcal{A} \restriction_{Y'} = (Q, Y', \Sigma, \Delta', I, F, C')$ is given as follows: $(p, \ell, \overline{z}', \rho', \overline{u}', \overline{v}', q) \in \Delta'$ if $(p, \ell, \overline{z}, \rho, \overline{u}, \overline{v}, q) \in \Delta$ where $\overline{z}' = \overline{z} \restriction_{Y'}$, $\rho' = \rho \restriction_{Y'}$, $\overline{u}' = \overline{u} \restriction_{Y'}$, and $\overline{v}' = \overline{v} \restriction_{Y'}$, and $C' = \{\overline{u} \restriction_{Y'} \mid \overline{u} \in C\}$.

▶ **Definition 13** (Quasi-normal set automaton). *Let* $\mathcal{A}$ *be a set automaton with the family of sets* $Y$ *and stable sets* $S \subseteq Y$*. The set automaton* $\mathcal{A}$ *is* quasi-normal *if the restriction* $\mathcal{A} \restriction_{(Y \setminus S)}$ *of* $\mathcal{A}$ *to the non-stable sets is a normal set automaton.*

## 2.4 Ordered Normal Set Automaton

In this section, we introduce ordered normal set automata and show that they have a decidable emptiness problem by reduction to the emptiness problem of ordered multicounter automata. Further, we extend this result to ordered quasi-normal set automata.

Let $\mathcal{A}$ be a normal set automaton with the family of sets $Y$. Let $\{\rho_j \mid j \in J\}$ be the set of global updates used by transitions of $\mathcal{A}$ where $J$ is an index set over $\mathbf{R}_Y$. Let $\rho = \bigcup_{j \in J} \rho_j$ be the union of the update relations and $\rho^+$ denote the transitive closure of $\rho$. Consider the graph $(Y, \rho^+)$ and a set $y \in Y$. If there is no vertex $z \in Y$ with $(z, z) \in \rho^+$ that can reach $y$, then the induced subgraph consisting of the vertices $Y' \subseteq Y$ that can reach $y$ is acyclic. For

each vertex $y' \in Y'$, we can show by induction that if the number of vertices from which $y'$ is reachable is $k$, then $y'$ contains at most $k+1$ data values at each point during the run over a data word. Thus, we have the below definition.

▶ **Definition 14** (Bounded set). *Let $\mathcal{A}$ be a normal set automaton with the family of sets $Y$. Let $\rho^+$ be the transitive closure of the union of the update relations used in the transitions of $\mathcal{A}$. A set $y \in Y$ is* bounded *if in the graph $(Y, \rho^+)$, there is no vertex $z$ such that $(z,z) \in \rho^+$ that can reach $y$.*

We now introduce ordered normal set automata. The idea is to impose an order on the non-bounded sets of the automaton and restrict the global updates on the non-bounded sets such that they empty the contents of a prefix of sets and move them to sets higher in the order such that the contents of the higher sets are not further moved anywhere else. Such global updates are simulated by an ordered multicounter automaton using its restricted hierarchical zero-tests on a prefix of counters corresponding to the sets emptied by the global update. The number of data values present in the bounded sets are tracked by the ordered multicounter automaton without using zero-tests by maintaining the number of data values in each set in its finite state space.

Toward this, we recall the notion of an ordered partition. Let $X$ be a set and $\leq$ be a linear order on it. For subsets $Y, Y' \subseteq X$, we write $Y \leq Y'$ to mean that $y \leq y'$ for each $y \in Y$ and $y' \in Y'$. An *ordered partition* of $X$ is a tuple $(X_0, X_1)$ of disjoint subsets that cover $X$ (i.e., $X_0 \uplus X_1 = X$) such that $X_0 \leq X_1$.

▶ **Definition 15** (Ordered normal set automaton). *Let $\mathcal{A}$ be a normal set automaton with family of sets $Y_{\mathcal{A}}$ and bounded sets $Z \subseteq Y$. The automaton $\mathcal{A}$ is* ordered *if there is a linear order $\leq$ on $Y \setminus Z$ such that for each global update transformation $\rho$ used in the transitions there is an associated ordered partition $(Y_0, Y_1)$ of $Y \setminus Z$ such that the tranformation $\rho$ maps the elements of $Y_0$ to $Y_1$, and is the identity transformation on $Y_1$. That is, in a transformation of $\mathcal{A}$ the contents of each set in $Y_0$ is emptied and added to a set in $Y_1$ and the contents of the sets in $Y_1$ are not copied anywhere else. If the automaton $\mathcal{A}$ is quasi-normal with the family of stable sets $S$ then $\mathcal{A}$ is* ordered *if the restriction $\mathcal{A}\restriction_{(Y \setminus S)}$ is ordered.*

Note that there is no restriction on the global update transformations on the bounded sets.

We now present the main result of this section. To this end, we recall the below definition.

▶ **Definition 16** (Ordered multicounter automaton [24]). *An ordered multicounter automaton $\mathcal{M}$ is a tuple $(Q, \Sigma, O, \Delta, J, F)$ where $Q$ is the finite set of states, $\Sigma$ is the finite alphabet, $O$ is the ordered finite set of non-negative counters, $J \subseteq Q$ is the set of initial states, and $F \subseteq Q$ is the set of final states. Let $O = \{c_1, \ldots, c_k\}$. The set of transitions is given by*

$$\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \{I_i, D_i, Z_{\leq i} \mid i \in [k]\} \times Q.$$

*From a given state $p$ on a label $a$, the transition $(p, a, I_j, q)$ increments the counter $c_j$ by one and moves to the state $q$. Similarly, $D_j$ decrements the counter $c_j$ by one and $Z_{\leq j}$ tests the counters $\{c_i \mid 1 \leq i \leq j\}$ for zero value. Whenever the run tries to decrement the value of a counter below zero, the computation halts erroneously and rejects.*

A *configuration* of an ordered multicounter automaton is a pair $(q, \mathbf{v})$ where $q \in Q$ and $\mathbf{v} = (v_1, \ldots, v_k)$ where $v_i$ is the value of the counter $c_i$. An *accepting run* of the automaton over a word is a sequence of transitions from an initial state with all the counters set to zero to a configuration where all counters are zero and the state is final. The language of an ordered multicounter automaton $\mathcal{M}$ is the set of all words on which $\mathcal{M}$ has an accepting run.

The following is a well known result about ordered multicounter automata.

▶ **Proposition 17** (Reinhardt, Theorem 6.1 of [24]). *Emptiness of ordered multicounter automata is decidable.*

We simulate the run of an ordered normal set automaton using an ordered multicounter automaton.

▶ **Proposition 18.** *For each ordered normal set automaton $\mathcal{A}$, an ordered multicounter automaton $\mathcal{M}$ recognising the string projection of $L(\mathcal{A})$ can be constructed.*

**Proof.** Let $\mathcal{A} = (Q, \Sigma, Y, \Delta, I, F, C)$ be an ordered normal set automaton with the family of sets $Y$ and bounded sets $Z \subseteq Y$. Let $\leq$ be the linear order on $Y \setminus Z$ according to the definition. We extend $\leq$ to the family of sets $Y$ by choosing an arbitrary linear order on $Z$ and setting $Y \setminus Z < Z$.

We construct an ordered multicounter automaton $\mathcal{M}$ that simulates the ordered normal set automaton $\mathcal{A}$. The automaton $\mathcal{M}$ has a counter $c_y$ for each $y \in Y$ and the counters are ordered with respect to the order $\leq$. The counter $c_y$ tracks the number of data values in the set $y \in Y$ in a configuration of $\mathcal{A}$ during the simulation. The automaton also remembers the number of data values present in each of the bounded sets using its state space. This memory is updated whenever the bounded sets are updated through global or local updates. This is further elaborated in Item 4 below.

Consider a word $w \in L(\mathcal{A})$ with a successful run $\rho = \delta_1\delta_2 \cdots \delta_n$ of the ordered normal set automaton $\mathcal{S}$. Let $\delta = (p, a, \overline{z}, \rho, e_u, \overline{v}, q)$ be a transition in the run. Note that since the automaton $\mathcal{A}$ is normal, $e_u$ is a unit vector and $\overline{z}$ and $\overline{v}$ are either unit or zero vectors. It is simulated by the ordered multicounter automaton $\mathcal{M}$ as follows:

1. The states $p, q$ and the current label $a$ are handled by the underlying finite state automaton.
2. If $\overline{z}$ is a unit vector $e_z$, the automaton $\mathcal{M}$ tests that the counter $c_z$ is nonzero (by successively decrementing and incrementing), signifying the presence of a data value with the characteristic vector $e_z$. Otherwise if $\overline{z} = \overline{0}$ then the transition is always enabled.
3. Let $(Y_0, Y_1)$ be the ordered partition of $Y \setminus Z$ associated with $\rho$. Let $y \in Y_0$. The automaton $\mathcal{M}$ successively decrements the counter $c_y$ and increments the counter $c_{z'}$ such that $e_{z'} = M^T(\rho) \cdot e_y$ (Note that $z' \in Y_1$). This process is repeated for each set $y \in Y_0$ (on $\varepsilon$-transitions) until the automaton $\mathcal{M}$ guesses the counters $\{c_y \mid y \in Y_0\}$ to be zero. The guess is verified by zero-testing the set of counters $\{c_y \mid y \in Y_0\}$. Note that the sets in $Y_0$ form a prefix that can be zero-tested using the hierarchical zero-tests of $\mathcal{M}$.
4. The global update on the bounded sets is simulated by decrementing the counter $c_x$ for $x \in Z$ and incrementing the counter $c_{x'}$ such that $e_{x'} = M^T(\rho) \cdot e_x$ (Note that $x'$ may not necessarily be in $Z$), until the former become zero. This step is accomplished without the use of zero-tests by remembering in the state space the size of each of the bounded sets. Further, the underlying finite state automaton updates this information after each transition.
5. Finally the local updates $e_u$ and $\overline{v}$ are simulated by incrementing the counter $c_u$ and decrementing the counter $c_v$ corresponding to the unit vector $\overline{v}$ if $\overline{v} \neq \overline{0}$.

The transitions $\delta_1$ and $\delta_n$ are ensured to be initial and final transitions respectively by the underlying finite state automaton. At the end of the run, the automaton $\mathcal{M}$ decrements all counters $c_x$ such that $e_x \in C$ to zero and performs a zero-test on all the counters of the automaton, verifying that the characteristic vector of each data value present in the sets is in the family of accepting vectors of membership $C$. ◀

Thus, by Propositions 17 and 18 we obtain the below result.

▶ **Theorem 19.** *Emptiness of ordered normal set automata is decidable.*

We now extend the above decidability result to ordered quasi-normal set automata.

▶ **Proposition 20.** *For each ordered quasi-normal set automaton there is an equivalent ordered normal set automaton.*

▶ **Corollary 21.** *Emptiness of ordered quasi-normal set automata is decidable.*

## 2.5 Closure Properties

Closure properties of set automata follow from that of class automata (See Proposition 40). We prove closure properties for quasi-normal set automata required for the translation of logical formulas to automata.

▶ **Lemma 22.** *Quasi-normal set automata are closed under letter-to-letter renaming, and union and intersection with data automata.*

**Proof sketch.** It is easy to verify that the standard construction for closure under letter-to-letter renaming preserves quasi-normality. For closure under union and intersection with data automata, we use our result that for each data automaton there is an equivalent set automaton where each set is stable (See Proposition 39). The result follows from the standard constructions using this equivalent set automaton. ◀

# 3 Equivalence of Set Automata and $\mathrm{FO}^2$ with Guarded Regular Predicates

## 3.1 Set Automata to $\mathrm{FO}^2$ with Guarded Regular Predicates

▶ **Proposition 23.** *For each set automaton $\mathcal{A}$ with update semigroup $S_{\mathcal{A}}$ there is a formula $\phi_{\mathcal{A}} \in \mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \widetilde{\mathcal{F}}_{S_{\mathcal{A}}}]$ such that $L(\mathcal{A}) = L(\phi_{\mathcal{A}})$.*

**Proof sketch.** Using standard ideas we write a formula that is satisfied by a data word $w = (a_1, d_1) \cdots (a_n, d_n)$ if and only if there is a successful run of $\mathcal{A}$ on $w$. In this sketch, we focus on how the global and local updates in the transitions of $\mathcal{A}$ are captured by a formula with guarded regular predicates.

We ensure that (1) at each position the sets in which the input data value is present is consistent with the global and local updates of the transition, and (2) in two successive positions of a class, the sets in which the data value is present is consistent with the global updates made in the positions strictly between them.

Let $\Delta$ and $\mathcal{S}$ denote the set of transitions and family of sets of $\mathcal{A}$ respectively. We use the monadic predicates $\overline{X} = \{X_\delta \mid \delta \in \Delta\}$ to indicate the transition used at a position in the run. We use the monadic predicates $\overline{Y} = \{Y_s \mid s \in \mathcal{S}\}$ to indicate the contents of the sets during the run: $Y_s$ is true, for $s \in \mathcal{S}$, at a position $j$ if and only if during the run of $\mathcal{A}$ after executing the transition on $(a_j, d_j)$ we have $d_j$ present in the set $Y_s$. Let $\overline{Z} = \overline{X} \cup \overline{Y}$. Our formula is of the form $\exists \overline{Z}\, \varphi$ where $\varphi \in \mathrm{FO}^2[\Sigma, \overline{Z}, <, +1, \sim, \pm 1, \widetilde{\mathcal{F}}_{S_{\mathcal{A}}}]$ is the conjunction of the formulas detailed in this sketch. To capture that the characteristic vector of membership at position $x$ is $\rho \in 2^{\mathcal{S}}$, we write,

$$\rho(x) := \bigwedge_{\substack{s \in \mathcal{S}, \\ \rho(s)=1}} Y_s(x) \wedge \bigwedge_{\substack{s \in \mathcal{S}, \\ \rho(s)=0}} \neg Y_s(x).$$

First, we ensure that the updates made at each position $x$ with label $(a_i, d_i)$ is consistent with the transition $\delta = (p, a_i, \chi, t, \alpha, \beta, q)$ at position $x$. That is, the current data value $d_i$ with characteristic vector $\chi$ after the application of the transition $\delta$ with global update $t$ and local updates $\alpha$ and $\beta$ has the characteristic vector of membership $\rho_\delta = M^T(t) \cdot \chi + \alpha - \beta$, i.e., $\rho_\delta(x)$ is true. We write

$$\forall x \ (X_\delta(x) \to \rho_\delta(x)).$$

Now, it suffices to ensure that the contents of the sets in two class-successive positions are consistent with the global updates in positions strictly between them. Toward this, we introduce the below notation.

Let $2^{\overline{Z}}$ denote the characteristic vectors of the monadic predicates $\overline{Z}$. Let $V$ denote the subset of $2^{\overline{Z}}$ such that the restriction of $V$ to the set $\overline{X}$ contains only unit vectors, this signifies that at a position exactly one transition is taken. Let $\Sigma' = \Sigma \times V$. Let $\pi_\Delta : \Sigma' \to \Delta$ be the projection function that maps a letter $\ell \in \Sigma'$ to the unique $\delta \in \Delta$ such that $X_\delta$ is true in $\ell$. We extend the map $\pi_\Delta$ to the map $\pi_\Delta^* : \Sigma'^* \to \Delta^*$ in the natural manner.

Let $\pi : \Delta \to S_\mathcal{A}$ be the projection map defined as $\delta = (p, \gamma, \chi, t, \alpha, \beta, q) \mapsto t$ that maps each transition to its update relation given as an element of the update semigroup $S_\mathcal{A}$. We extend the map $\pi$ to the map $\pi^* : \Delta^* \to S_\mathcal{A}^*$ in the natural manner.

Let $\sigma : S_\mathcal{A}^* \to S_\mathcal{A}$ be the product morphism that maps each word $s_1 \cdots s_n$ to the product of $s_1, \ldots, s_n$ in the semigroup $S_\mathcal{A}$. Clearly, the morphism $\sigma \circ \pi^* \circ \pi_\Delta^* : \Sigma'^* \to S_\mathcal{A}$ maps a sequence of letters in $\Sigma'$ to the product of the corresponding set update relations in $S_\mathcal{A}$.

For $\chi \in 2^{\overline{\mathcal{S}}}$, let $\Delta_\chi \subseteq \Delta$ denote all transitions with the characteristic vector of membership $\chi$. Assume that a position $y$ is labelled by a transition $\delta = (p, a_j, \chi, t, \alpha, \beta, p') \in \Delta$. Let position $y$ be the class-successor of position $x$. Assume that the position $x$ satisfies $\rho(x)$, i.e., the characteristic vector of membership of the data value in position $x$ is $\rho$, and $y$ is labelled by a transition in $\Delta_\chi$. Then we ensure that the product of the set update relations in the positions strictly in between $x$ and $y$ is an element $s \in S_\mathcal{A}$ such that $\chi = \rho \cdot s$. We write

$$\forall x \forall y \bigwedge_{\substack{\chi, \rho \in 2^{\mathcal{S}} \\ \delta \in \Delta_\chi}} (x \not\doteq 1 = y \wedge \rho(x) \wedge X_\delta(y) \to \widetilde{L}(x, y))$$

where $L \coloneqq \bigcup_{\substack{m \in M, \\ \chi = \rho \cdot s}} (\sigma \circ \pi^* \circ \pi_\Delta^*)^{-1}(s)$.

◀

## 3.2 $\mathrm{FO}^2$ with Guarded Regular Predicates to Quasi-Normal Set Automata

In this section we translate the guarded logic formulas to a subclass of quasi-normal set automata called *suffix-storing set automata*.

▶ **Definition 24** (Suffix-storing set automaton). *Let $M$ be a monoid and $h : \Sigma^* \to M$ be a morphism. A set automaton over the alphabet $\Sigma$ is* $h$-suffix-storing *if it is a quasi-normal set automaton whose non-stable family of sets is of the form $\{X_m \mid m \in M\}$ and obeys the following property: after reading the input prefix $w \in (\Sigma \times \mathcal{D})^*$, the set $\{m \mid X_m \neq \emptyset\}$ is a subset of images of suffixes of $str(w)$ under $h$.*

The rest of the section is devoted to the proof of Proposition 25.

▶ **Proposition 25.** *For each formula $\psi \in \mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \widetilde{\mathcal{F}}_h]$, there is a h-suffix-storing set automaton $\mathcal{A}_\psi$ such that $L(\psi) = L(\mathcal{A}_\psi)$.*

We proceed by translating the formulas to suffix-storing set automata.

Assume that we are given a formula $\psi = \exists \overline{X} \varphi$ where $\varphi \in \mathrm{FO}^2[\Sigma, \overline{X}, <, +1, \sim, \pm 1, \widetilde{\mathcal{F}}_h]$ and $\overline{X} = \{X_1, \ldots, X_m\}$. Further, assume that $\varphi$ uses guarded predicates defined by the morphism $h : (\Sigma \times 2^{\overline{X}})^* \to M$. Since quasi-normal set automata are closed under letter-to-letter renaming it suffices to show that there is a set automaton recognising $L(\varphi)$. In the following, to transform the formulas we add additional monadic variables to the vocabulary. This requires $h$ to be replaced by the morphism $h \circ \pi$ where $\pi$ is the projection map to the alphabet $\Sigma \times 2^{\overline{X}}$. To keep the discussion simple, we omit referring to the step explicitly.

We convert the formula $\varphi$ to an equivalent formula of size linear in $|\varphi|$ with additional unary predicates $\overline{Y} = \{Y_1, \ldots, Y_n\}$ in Scott Normal Form (see, for instance [12]). Hence, we get an equivalent formula

$$\exists Y_1 \cdots Y_n \left( \forall x \forall y \, \chi \wedge \bigwedge_i \forall x \exists y \, \chi_i \right) \tag{3}$$

where $Y_1, \ldots, Y_n$ are new unary predicates and $\chi$ and $\chi_i$ are quantifier-free with free variables $\overline{X} \cup \overline{Y}$. Again, by appealing to the closure under letter-to-letter renaming of set automata, it suffices to show that there is an equivalent set automaton for formulas of the form

$$\forall x \forall y \, \chi \wedge \bigwedge_i \forall x \exists y \, \chi_i. \tag{4}$$

### Unary and Binary types

Let $x \ll y$ stand for the formula $x < y \wedge x + 1 \neq y$, i.e., $x$ occurs before $y$ but is not its predecessor. Let $O$ denote the set of binary *order-types* on two variables, i.e.,

$$O = \{x \ll y, x + 1 = y, x = y, y + 1 = x, y \ll x\}.$$

No two formulas in $O$ are satisfiable at the same time, and any quantifier-free formula that uses only the predicates $\{<, +1, =\}$ is equivalent to a disjunction of formulas from $O$ (easily verified by converting the formula to DNF).

Let $x \ominus y$, read as '$x$ and $y$ are class-distant', stand for the formula $x \sim y \wedge x \pm 1 \neq y \wedge y \pm 1 \neq x$, i.e., $x$ and $y$ are in the same class but neither is the class successor of the other. Let $E$ denote the set of binary *equivalence-types* on two variables, i.e.,

$$E = \{x \not\sim y, x \ominus y, x \pm 1 = y, y \pm 1 = x, x = y\}.$$

It is easily verified that no two formulas in $E$ are satisfiable at the same time and that any quantifier-free two-variable formula using the predicates $\{\sim, \pm 1\}$ is equivalent to a disjunction of formulas in $E$.

A *literal* is either an atomic formula or a negation of an atomic formula. Let $\Gamma$ be a set of literals. The set $\Gamma$ is *consistent* if $\Gamma$ does not contain a literal and its negation. It is *maximally-consistent* if $\Gamma$ is consistent and no strict superset of $\Gamma$ is consistent.

A *unary-type* over $\Sigma \cup \overline{X}$ is a conjunction of a positive literal from $\Sigma$ and a maximally-consistent set of literals over $\overline{X}$ on the same variable (either $x$ or $y$). For example, let $\Sigma = \{a, b\}$. Then $a(x) \wedge \neg X_1(x)$ is a unary-type over the unary predicates $\{a, b, X_1\}$, whereas $a(x)$ and $a(x) \wedge b(y) \wedge X_1(y)$ are not. Let $U$ denote the set of unary-types.

### Construction of the Quasi-Normal Set Automaton

We first transform the formulas $\forall x \forall y\, \chi$ and $\forall x \exists y\, \chi_i$ into suitable forms given below using standard techniques (see Lemma 12 and 13 of [4]).

It is straightforward to first transform $\chi$ to CNF and distribute the universal quantifiers over the conjunction. Furthermore, by using standard logical equivalences, we write each of the exponentially many conjuncts in $\chi$ obtained from the CNF transformation in the below form:

$$\forall x \forall y (\alpha(x) \wedge \beta(y) \wedge o(x,y) \wedge \epsilon(x,y) \rightarrow \widetilde{\lambda}(x,y)) \tag{5}$$

where $\alpha, \beta \in U$, $o(x,y) \in O$, $\epsilon(x,y) \in E$, and $\widetilde{\lambda}(x,y)$ is either *false* or a guarded regular predicate of the form $\widetilde{L}(x,y)$ or $\widetilde{L}(y,x)$ for some language $L$ defined by $h$. The set automaton constructed ensures that for each $\alpha$ and $\beta$ with the class and order conditions met, the factor bordered by them satisfies the guarded regular predicate $\widetilde{\lambda}$ by storing information about the factors in its sets.

Similarly, for each $\chi_i$, by introducing additional unary predicates, it suffices to consider formulas of the form

$$\forall x \exists y (\alpha(x) \rightarrow \beta(y) \wedge o(x,y) \wedge \epsilon(x,y) \wedge \widetilde{\lambda}(x,y)). \tag{6}$$

Here, the set automaton ensures that each $\alpha$ is witnessed by a $\beta$ satisfying the order, class, and guarded regular predicate constraints.

Thus, it suffices to construct a $h$-suffix-storing set automaton for a formula with conjuncts in either of the above forms. Let $k$ be the number of such conjuncts. For the construction of the set automaton, we use the family of non-stable sets $X = \{X_m \mid m \in M\}$. In addition to the sets in $X$, we use a number of other stable sets for book-keeping purposes. On a letter $\gamma \in U$, the automaton applies the set update transformation $\{X_p \mapsto X_{p \cdot h(\gamma)} \mid p \in M\}$ to the sets in $X$.

We now obtain Proposition 25 from Lemma 26.

▶ **Lemma 26.** *For each formula* $= \bigwedge_{i \in [k]} \phi_i$ *where each* $\phi_i$ *is a formula of form*

$$\forall x \forall y\ (\alpha(x) \wedge \beta(y) \wedge o(x,y) \wedge \epsilon(x,y) \rightarrow \widetilde{\lambda}(x,y)), \text{ or}$$
$$\forall x \exists y\ (\alpha(x) \rightarrow \beta(y) \wedge o(x,y) \wedge \epsilon(x,y) \wedge \widetilde{\lambda}(x,y)),$$

*there is an equivalent $h$-suffix-storing set automaton.*

We illustrate the key ideas of the construction by considering the formula $\phi$ given below.

$$\forall x \forall y\ (\alpha(x) \wedge \beta(y) \wedge x \ll y \wedge x \ominus y \rightarrow \widetilde{L}(x,y))$$

The formula $\phi$ states that if an $\alpha$ is followed by a class-distant $\beta$ that is not its successor, then the word bordered by them is in $L$. Observe that the condition $x \ominus y$ captures that $x + 1 \neq y$. Thus in the construction we do not need to specifically ensure that $x \ll y$ and can instead simply work with $x < y$.

We construct a set automaton $\mathcal{A}$ that in addition to the sets in $X$, has the family of stable sets $A \cup H \cup F$ where $A = \{A_\gamma \mid \gamma \in U\}$, $H = \{H_m \mid m \in M\}$, and $F = \{F_m \mid m \in M\}$.

Let $w$ be the string projection of an input data word. Assume that the automaton is on a position $y$. Let $d$ be any data value occurring in the data word prior to position $y$. The following invariants are maintained during the run of the set automaton $\mathcal{A}$:

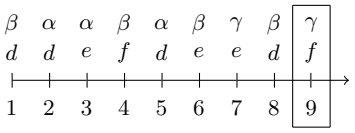
(X) The data value $d$ is in the set $X_m$, where $m \in M$, if and only if the factor bordered by latest occurrence of $d$ and the current position has image $m \in M$. Note that a data value can be present in at most one of the sets in $X$, ensuring quasi-normality of the set automaton.

(A) The data value $d$ is in the set $A_\gamma$ if and only if the latest occurrence of $d$ before $y$ was with the label $\gamma$.

(H) The data value $d$ is in $H_m$ if and only if there is a factor bordered by a non-latest occurrence of $\alpha$ in $d$'s class and the latest occurrence of $\alpha$ in $d$'s class, whose image is $m$.

(F) The data value $d$ is in $F_m$ if and only if there is a factor bordered by the latest occurrence of $\alpha$ in $d$'s class and the latest occurrence of $d$ whose image is $m$.

▶ **Example 27.** The invariants can be understood from the below figure. For ease of presentation, in Figure 1, for a pair of positions $x < y$ we use the notation $X_{(x,y)}$ (respectively $H_{(x,y)}, F_{(x,y)}$) to denote the set $X_m$ (respectively $H_m, F_m$) where $m$ is the image of the factor $w_{(x,y)}$ in the monoid $M$.




$X_{(4,9)} = \{f\}$
$X_{(7,9)} = \{e\}$
$X_{(8,9)} = X_1 = \{d\}$

$A_\beta = \{d, f\}$
$A_\gamma = \{e\}$

$H_{(2,5)} = \{d\}$

$F_{(5,8)} = \{d\}$
$F_{(3,7)} = \{e\}$

**Figure 1** A data word $w$ and the contents of the sets of $\mathcal{A}$ on encountering position 9 of $w$.

On encountering a pair $(\ell, d')$, the automaton performs the following actions.

The set update $\{X_p \mapsto X_{p \cdot h(\ell)} \mid p \in M\}$ is performed, updating the sets in $X$ and maintaining the invariant (X). The data value $d'$ is removed from the unique set in $X$ in which it is present, if any. Further, to maintain the invariant (X) for the upcoming positions in the run, the data value $d'$ is added to the set $X_1$ (here, 1 is the identity of the monoid $M$) by means of local updates since the run of the monoid $M$ starts at the identity element. To maintain the invariant (A), the data value $d'$ is removed from the unique set in $A$, if any, in which it is present and is added to the set $A_\ell$ using local updates.

*If* $\ell = \alpha$: The current data value $d'$ is removed from all the sets in $H$ and is added to the set $H_{u \cdot h(\alpha) \cdot v \cdot w}$ if prior to the set update, the data value was present in $H_u, F_v$ and $X_w$. Further, the data value is added to the set $H_m$ if prior to the set update, the data value was present in $X_m$ and $A$. This is to maintain the invariant (H) in the upcoming positions. To maintain the invariant (F) in the upcoming positions, the data value is removed from the sets in $F$ if present.

*If* $\ell \neq \alpha$: If the data value $d'$ is present in any of the sets in $F$, it is removed from it and added to the set $F_{m \cdot h(\gamma) \cdot n}$ if prior to the set update, the data value was present in the sets $F_m, A_\gamma$, and $X_n$. Further, if the data value $d'$ was present in $X_m$ and $A_\alpha$ prior to the set update then it is added to the set $F_m$. This is to maintain the invariant (F). Further if $\ell = \beta$, the following operations are also performed. To satisfy $\phi$, we need to ensure that (1) the unique set $F_m$ containing $d'$ corresponds to an accepting element of the monoid for the language $L$ if $d'$ is not in $A_\alpha$ (i.e., if the latest element in the class is not an $\alpha$) (2) and all the sets $H_n$ containing $d'$ must satisfy $n \cdot h(\alpha) \cdot m \in P$ where $P \subseteq M$ is the accepting set for the language $L$ used in the guarded regular predicate. Note that this captures only the condition $x \ll y$ as it only considers $x \ominus y$ where $x < y$, thus ensuring $x + 1 \neq y$. Both the conditions can be ascertained from the characteristic vector of $d'$. If it is not, then $\mathcal{A}$ halts erroneously.

Finally, at the end of the run, every configuration of the set automaton is accepting.

# 4 Decidability & Undecidability Results on $\mathrm{FO}^2$ with Guarded Regular Predicates

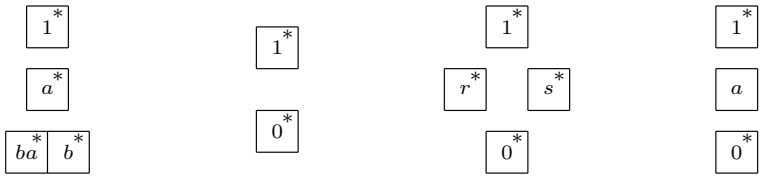
## 4.1 Linear Bands

A *band* is a semigroup in which every element is an idempotent.

▶ **Definition 28** (Linear Band). *A monoid $M$ is a* linear band *if it is a band and the preorder relation $\leq_{\mathcal{J}}$ on $M$ is total. In other words, $M$ satisfies identities $x^2 = x$ and $xyx = x \vee yxy = y$ for all $x, y \in M$.*

▶ **Example 29.** Below are some examples and non-examples of linear bands.

1. The two-element monoid $U_1 = \{1, 0\}$ with a zero is a linear band.
2. Consider the band $U_1^2 = \{1, r, s, 0\}$ with the operation $rs = sr = 0$ given in Figure 2. It is not a linear band since the elements $r$ and $s$ are $\leq_{\mathcal{J}}$-incomparable, i.e., the two-sided ideals generated by $r$ and $s$ are incomparable with respect to inclusion. This example shows that linear bands are not closed under products.
3. Consider the null monoid $N_k = \{1, x_1, \ldots, x_k, 0\}$, with the operation $x_i \cdot x_j = 0$, for each $i, j \in [k]$. The null monoid $N_2$ is given in Figure 2. The monoid $N_k$ is not a linear band.




(a) (b) (c) (d)

**Figure 2** (a) A linear band recognising the language $(a+b)^*b+a^*$ over the alphabet $\{a, b\}$, (b) A linear band $U_1$, (c) A band $U_1^2$, and (d) a linear monoid $N_2$. Here, (c) and (d) are not linear bands. Idempotents are denoted by a star in the above monoids.

A semigroup $S$ is in the class of semigroups **DA** if it satisfies the identity $ese = e$ for each element $s$ and idempotent $e$ such that $e \leq_{\mathcal{J}} s$. The name **DA** comes from an equiavalent definition that all regular $\mathcal{D}$-classes of $S$ are aperiodic subsemigroups. They correspond to positive regular languages (subsets of $\Sigma^+$) recognised by $\mathrm{FO}^2(\Sigma, <)$ formulas.

The below result follows from basic facts about Green's relations.

▶ **Lemma 30.** *Every linear band is in* **DA**.

## 4.2 Decidability with Guarded Regular Predicates recognised by Linear Bands

In this section, we show that the logic $\mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \widetilde{\mathcal{F}}_M]$ has a decidable satisfiability problem when $M$ is a linear band. It suffices to show the below result.

▶ **Proposition 31.** *Let $M$ be a linear band and $h : \Sigma^* \to M$ be a morphism. For each $h$-suffix-storing set automaton there is an equivalent ordered quasi-normal set automaton.*

**Proof.** Let $M$ be a linear band and $\mathcal{A}$ be a $h$-suffix-storing set automaton for some morphism $h : \Sigma^* \to M$. Assume that $X = \{X_m \mid m \in M\}$ are the non-stable sets of $\mathcal{A}$. Since stable

sets can always be added to an ordered quasi-normal set automaton, to simplify the below construction we ignore the stable sets.

First, we introduce some machinery for the proof.

Let $\sqsubseteq_{\mathcal{L}}$ be a partial order on $M$ extending $\leq_{\mathcal{L}}$, obtained by fixing total orders within each $\mathcal{J}$-class on $\mathcal{L}$-classes such that they are stable on $\mathcal{R}$-classes, and ordering distinct $\mathcal{J}$-classes according to $\leq_{\mathcal{L}}$. More precisely, $\sqsubseteq_{\mathcal{L}}$ satisfies the following conditions. Let $x, y \in M$.

1. If $x \sqsubset_{\mathcal{L}} y$ then $x \leq_{\mathcal{L}} y$.
2. The relation $\sqsubseteq_{\mathcal{L}}$ is antisymmetric, i.e., $x \sqsubseteq_{\mathcal{L}} y$ and $y \sqsubseteq_{\mathcal{L}} x$ then $x = y$.
3. If $x <_{\mathcal{L}} y$ then $x \sqsubset_{\mathcal{L}} y$, and, if $x \mathcal{L} y$ and $x \neq y$ then either $x \sqsubset_{\mathcal{L}} y$ or $y \sqsubset_{\mathcal{L}} x$.
4. If $x \sqsubseteq_{\mathcal{L}} y$, $x \mathcal{R} x'$ and $y \mathcal{R} y'$, then $x' \sqsubseteq_{\mathcal{L}} y'$.

An order $\sqsubseteq_{\mathcal{L}}$ is easily found by fixing an ordering on the columns of each $\mathcal{J}$-class in the eggbox diagram of $M$.

Let $x \sqsupseteq_{\mathcal{L}} y$ and $x \sqsupset_{\mathcal{L}} y$ denote $y \sqsubseteq_{\mathcal{L}} x$ and $y \sqsubset_{\mathcal{L}} x$ respectively. A (strict) $\sqsubseteq_{\mathcal{L}}$*-chain* in $M$ is a sequence $m_1, \ldots, m_n$ of elements from $M$ such that $m_1 \sqsupset_{\mathcal{L}} m_2 \sqsupset_{\mathcal{L}} \cdots \sqsupset_{\mathcal{L}} m_n$. Let $k \geq 1$ be the length of the longest $\sqsubseteq_{\mathcal{L}}$-chain in $M$. The $\sqsubseteq_{\mathcal{L}}$-height of an element $m \in M$, denoted by $k_m$, is the length of the longest $\sqsubseteq_{\mathcal{L}}$-chain starting with $m$.

We observe that a subset $M' \subseteq M$ is an $\leq_{\mathcal{L}}$-total set, if, and only if, it is $\sqsubseteq_{\mathcal{L}}$-total, and equivalently, the elements of $M'$ forms a strict $\sqsubseteq_{\mathcal{L}}$-chain. Moreover, if $m_1 \sqsupset_{\mathcal{L}} m_2 \sqsupset_{\mathcal{L}} \cdots \sqsupset_{\mathcal{L}} m_n$ is a strict $\sqsubseteq_{\mathcal{L}}$-chain then $k_{m_1} > k_{m_2} > \cdots > k_{m_n}$.

▷ Claim 32. If $M' \subseteq M$ is an $\leq_{\mathcal{L}}$-total set. Then for all $m \in M$, $M'm$ is also $\leq_{\mathcal{L}}$-total.

Proof. Assume that $M' \subseteq M$ is $\leq_{\mathcal{L}}$-total. It suffices to prove that for each $m_1, m_2 \in M'$, if $m_1 \leq_{\mathcal{L}} m_2$, then $m_1 m \leq_{\mathcal{L}} m_2 m$, which follows from the fact that $\leq_{\mathcal{L}}$ is stable under right multiplication (Fact 48). ◁

▷ Claim 33. If $m \mathcal{R} m'$ for $m, m' \in M$, then $k_m = k_{m'}$.

Proof. Assume that $m \mathcal{R} m'$. Let $m = m_1 \sqsupset_{\mathcal{L}} m_2 \sqsupset_{\mathcal{L}} \cdots \sqsupset_{\mathcal{L}} m_n$ be a strict $\sqsubseteq_{\mathcal{L}}$-chain starting in $m$. Let $z_i \in M$, for $1 \leq i \leq n-1$, be such that $z_i m_i = m_{i+1}$. The we claim that the $\sqsubseteq_{\mathcal{L}}$-chain $m' \sqsupseteq_{\mathcal{L}} z_1 m' \sqsupseteq_{\mathcal{L}} z_2 z_1 m' \cdots \sqsupseteq_{\mathcal{L}} z_{n-1} \cdots z_1 m'$ is strict. Since $m \mathcal{R} m'$ we have $z_1 m \mathcal{R} z_1 m'$. By induction, we have $z_i \cdots z_1 m \mathcal{R} z_i \cdots z_1 m'$, for $1 \leq i \leq n-1$. Since $\sqsubseteq_{\mathcal{L}}$ is compatible with the $\mathcal{R}$-relation, we conclude that the chain is strict. Thus we infer that $k_m \leq k_{m'}$. The other direction follows symmetrically. ◁

▷ Claim 34. Let $m_1, m_2 \in M$ be two elements such that $m_1 \sqsubset_{\mathcal{L}} m_2$. Then for each $m \in M$, if $k_{m_1 m} < k_{m_1}$ then $k_{m_2 m} < k_{m_1}$.

Proof. Assume that $k_{m_1 m} < k_{m_1}$. We claim that $m_1 m <_{\mathcal{J}} m_1$. Otherwise $m_1 m \mathcal{J} m_1$ and we also have $m_1 m \leq_{\mathcal{R}} m_1$, therefore by Fact 47 we have $m_1 m \mathcal{R} m_1$. Now by Claim 33 we have $k_{m_1 m} = k_{m_1}$, a contradiction. Hence we have $m_1 m <_{\mathcal{J}} m_1$. Since $M$ is linear, we have either $m <_{\mathcal{J}} m_1$ or $m_1 \leq_{\mathcal{J}} m$. If $m_1 \leq_{\mathcal{J}} m$ we have $m_1 m m_1 = m_1$ since $M$ is in **DA**. Here we have $m_1 \leq_{\mathcal{J}} m_1 m$, a contradiction. Thus $m <_{\mathcal{J}} m_1$ and since $m_2 m \leq_{\mathcal{J}} m$ we get $m_2 m <_{\mathcal{J}} m_1$. We deduce that $k_{m_2 m} < k_{m_1}$. ◁

We construct an ordered quasi-normal set automaton $\mathcal{B}$ that has the family of sets $Y \cup Z$, where $Y = \{Y_i \mid i \leq k\}$ and $Z = \{Z_m \mid m \in M\}$. The sets in $Y$ are used to simulate the non-stable sets $X$ of the $h$-suffix-storing set automaton $\mathcal{A}$. At any point during a run of the automaton $\mathcal{B}$, the set $Y_{k_m}$ has the content of a unique nonempty set $X_m$ if it exists. During the run, except for the last transition, the sets in $Z$ remain empty and the identity transformation is applied to them. At the last transition we copy the nonempty sets in $Y$ to

the corresponding set in $Z$ (i.e., $Y_{k_m}$ to $Z_m$). The automaton $\mathcal{B}$ simulates $\mathcal{A}$ in the following way.

Since the automaton $\mathcal{A}$ is $h$-suffix-storing, at any point during the run, the subset $M' = \{m \in M \mid X_m \neq \emptyset\}$ is an $\leq_{\mathcal{L}}$-total set, and hence $\sqsubseteq_{\mathcal{L}}$-total also. The contents of each $X_m$ is stored in the set $Y_{k_m}$ and the automaton $\mathcal{B}$ also remembers in its state the information about the element $m$ that corresponds to the nonempty set $Y_{k_m}$ (this is a partial map $f : [k] \to M$ defined as $f : k_m \mapsto m$). To effect the global update $\rho = \{(X_m, X_{mm'}) \mid m \in M\}$ of $\mathcal{A}$, i.e., right multiplication by an element $m' \in M$, the automaton $\mathcal{B}$ moves the data values in each nonempty set $Y_{k_m}$ to the set $Y_{k_{mm'}}$, i.e., by the global update $\rho' = \{(Y_{k_m}, Y_{k_{mm'}}) \mid m \in M'\} \cup \{(Z_m, Z_m) \mid m \in M\}$, and replaces the map $f$ by the map $f' : k_{mm'} \mapsto mm'$. Note that since $M'm'$ is a $\sqsubseteq_{\mathcal{L}}$-total set, elements of $M'm'$ form a strict $\sqsubseteq_{\mathcal{L}}$-chain, and therefore $k_{m_1m'} \neq k_{m_2m'}$ whenever $m_1m' \neq m_2m'$, for $m_1, m_2 \in M'$. Therefore each nonempty set $Y_i$ corresponds to precisely one nonempty set from $X$. On the last input pair the global update $\rho = \{(X_m, X_{mm'}) \mid m \in M\}$ is simulated by the update $\rho'' = \{(Y_{k_m}, Z_{mm'}) \mid m \in M\}$, i.e., the sets in $Y$ are copied to sets in $Z$ using the map stored in the sets. Finally the automaton $\mathcal{B}$ accepts if a final state is reached and each data value is in an accepting configuration as given by that of $\mathcal{A}$.

Let $\lesssim$ be the order on the family $Y \cup Z$, where $Y \lesssim Z$, and $Y$ is ordered by the relation $\{(Y_i, Y_j) \mid j \leq i\}$, and an arbitrary order is fixed on $Z$. Using Claim 34, it is straightforward to check that global updates of $\mathcal{B}$ are ordered with respect to $\lesssim$. ◀

Thus, by Propositions 25, 31, and Corollary 21, we have the below result.

▶ **Theorem 35.** *Satisfiability of* $\mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \widetilde{\mathcal{F}}_M]$ *formulas over data words is decidable when* $M$ *is a linear band.*

## 4.3 Undecidability with Guarded Regular Predicates not recognised by Linear Bands

▶ **Proposition 36.** *If* $M$ *is a monoid that is not a linear band then* $U_1^2$ *or* $N_2$ *divides* $M$*.*

**Proof.** Let $M$ be a monoid that is not a linear band. If $M$ contains an element $x$ that is not an idempotent then $M$ is not a band. Here $N_2$ divides the submonoid generated by $x$. Hence assume that all elements of $M$ are idempotent. Now since $M$ is not a linear band, there exist two idempotents $e$ and $f$ that are $\leq_{\mathcal{J}}$-incomparable. Here $U_1^2$ divides $M$. ◀

We obtain the below undecidability result by a reduction from the halting problem of two-counter machines [20]. We encode the run of a two-counter machine as a data word and write a formula that is satisfied by a data word if and only if the data word encodes an accepting run of the two-counter machine.

▶ **Proposition 37.** *Satisfiability of* $\mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \widetilde{\mathcal{F}}_M]$ *formulas over data words is undecidable when* $U_1^2$ *or* $N^2$ *divides* $M$*.*

The below result follows from Propositions 36 and 37.

▶ **Theorem 38.** *Satisfiability of* $\mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \widetilde{\mathcal{F}}_M]$ *formulas over data words is undecidable when* $M$ *is not a linear band.*

Our main result restated below follows from Theorem 35, Proposition 36 and Theorem 38.

▶ **Theorem 6.** *Satisfiability of* $\mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \widetilde{\mathcal{F}}_M]$ *formulas over data words is decidable if and only if* $M$ *is a linear band.*

# 5 Discussion

## 5.1 Comparison with Other Models

A natural restriction of set automata captures the class of languages recognised by data automata.

▶ **Proposition 39.** *For each data automaton there is an equivalent set automaton where each set is stable, that is, the only set update used in its transitions is the identity relation.*

The following result highlights the robustness of the set automaton formalism.

▶ **Proposition 40.** *Set automata are expressively equivalent to class automata.*

## 5.2 Logical Characterisation of Set Automata

First, we recall a logical characterisation of class automata given in [6].

▶ **Proposition 41** (Bojańczyk and Lasota, [6])**.** *Class automata are expressively equivalent to a restricted fragment of* $\mathrm{MSO}[\Sigma, <, \sim]$ *consisting of formulas of the form*

$$\exists X_1 \cdots \exists X_n \, \forall Y \, \mathsf{class}(Y) \to \phi(X_1, \ldots, X_n, Y) \tag{7}$$

*where* $\mathsf{class}(Y) := \exists y \forall x \, (Y(x) \leftrightarrow y \sim x)$ *and* $\phi$ *is in* $\mathrm{MSO}[\Sigma, <]$.

It is easy to see that the data value equality test, the class successor relation, and regular predicates are captured in this restricted fragment of MSO.

▶ **Proposition 42.** *For each formula* $\varphi \in \mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \mathcal{F}]$*, there is a formula* $\psi$ *belonging to the restricted fragment of* $\mathrm{MSO}[\Sigma, <, \sim]$ *consisting of formulas of the form given in Equation* (7) *such that* $L(\varphi) = L(\psi)$.

By Propositions 23, 40, 41, and 42, we obtain the below result.

▶ **Theorem 43.** *Set automata and* $\mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \mathcal{F}]$ *are expressively equivalent.*

Now by Proposition 23, we obtain the below corollary.

▶ **Corollary 44.** *For each formula in* $\mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \mathcal{F}_M]$*, there is an equivalent formula in* $\mathrm{EMSO}^2[\Sigma, <, +1, \sim, \pm 1, \widetilde{\mathcal{F}}_{M'}]$ *for some monoid* $M'$.

# 6 Conclusion

We have characterised the decidability frontier for the $\mathrm{EMSO}^2$ logic over data words with guarded regular predicates recognised by a monoid. The characterisation in the case of guarded regular predicates recognised by semigroups is yet to be done. The class of regular languages recognised by linear bands has not been studied in the literature and needs to be explored further.

# A Appendix

## Orders Preliminaries

A *preorder* $\lesssim$ on a set $X$ is a binary relation on $X$ that is reflexive and transitive. A preorder $\lesssim$ is *total* if $x \lesssim y$ or $y \lesssim x$ for each $x, y \in X$. We say the subset $Y \subseteq X$ is $\lesssim$-*total* if any two elements in $Y$ is comparable with respect to $\lesssim$, in other words $Y$ is totally preordered by $\lesssim$. A *partial order* is an antisymmetric preorder. A *linear order* is a total partial order.

## Semigroup Preliminaries

We recall some fundamental definitions from the local theory of semigroups. A detailed account can be found in standard textbooks, such as [17, 22].

For a semigroup $S$, its monoidal extension $S^1$ is defined to be $S$ if it is already a monoid and otherwise to be the monoid obtained by adjoining an identity element 1 to it. We denote by $\langle X \rangle$ the semigroup generated by elements in the set $X$. For a semigroup $S$, a subset $T$ of $S$ is a *subsemigroup* if $x, y \in T$ implies that $xy \in T$. A *submonoid* of a monoid is a subsemigroup containing the identity. An element $e$ of a semigroup is said to be an *idempotent* if $e^2 = e$. Let $S$ and $T$ be semigroups. The semigroup $T$ is a *quotient* of $S$ if there exists a surjective morphism from $S$ onto $T$. The semigroup $T$ is said to *divide* $S$ if $T$ is a quotient of a subsemigroup of $S$.

▶ **Definition 45** (Ideals). *Let $S$ be a semigroup. A right ideal of $S$ is a subset $R$ of $S$ such that $RS^1 \subseteq R$, that is, for each $r \in R$ and $s \in S^1$, we have that $rs \in R$. Symmetrically, a left ideal is a subset $L$ of $S$ such that $S^1L \subseteq L$. A (two-sided) ideal $I$ is a subset of $S$ which is both a right and a left ideal, i.e. $S^1IS^1 \subseteq I$.*

▶ **Definition 46** (Green's relations). *Let $s, t$ be elements of a semigroup $S$. The Green's preorders $\leq_{\mathcal{R}}, \leq_{\mathcal{L}}$, and $\leq_{\mathcal{J}}$ are given as follows: $s \leq_{\mathcal{R}} t \triangleq s = ty$ for some $y \in S^1$, $s \leq_{\mathcal{L}} t \triangleq s = xt$ for some $x \in S^1$, and $s \leq_{\mathcal{J}} t \triangleq s = xty$ for some $x, y \in S^1$.*

*Let $\mathcal{K} \in \{\mathcal{R}, \mathcal{L}, \mathcal{J}\}$. The Green's preorders have equivalence relations associated with them that are given as $s\mathcal{K}t \triangleq s \leq_{\mathcal{K}} t$ and $t \leq_{\mathcal{K}} s$ Further, the relation $\mathcal{H}$ is given by $s\mathcal{H}t \triangleq s\mathcal{L}t$ and $s\mathcal{R}t$.*

These relations can be formulated in terms of ideals. We have that $s \leq_{\mathcal{J}} t$ (respectively $s \leq_{\mathcal{R}} t$, $s \leq_{\mathcal{L}} t$) if and only if $S^1sS^1 \subseteq S^1tS^1$ ($sS^1 \subseteq tS^1$, $S^1s \subseteq S^1t$), that is, the ideal (right ideal, left ideal) generated by $s$ is contained in the ideal (right ideal, left ideal) generated by $t$.

The equivalence class of an element $s$ under the relation $\mathcal{K} \in \{\mathcal{R}, \mathcal{L}, \mathcal{J}, \mathcal{H}\}$ is called the $\mathcal{K}$-class of $s$ and is denoted by $\mathcal{K}(s)$. Let $\mathcal{K}' \in \{\mathcal{R}, \mathcal{L}, \mathcal{J}\}$. Two $\mathcal{K}'$-classes are *comparable* when every element in one class is comparable to every element in the other class in the $\leq_{\mathcal{K}'}$ preorder. In finite semigroups the relations $\mathcal{D} = \mathcal{R} \circ \mathcal{L} = \mathcal{L} \circ \mathcal{R}$ and $\mathcal{J}$ coincide, and we use them interchangeably.

▶ **Fact 47** (Prefix/Suffix Lemma). *Let $x, y$ be elements of a* finite *semigroup.*

1. *If $x \leq_{\mathcal{L}} y$ and $x\mathcal{J}y$ then $x\mathcal{L}y$.*
2. *If $x \leq_{\mathcal{R}} y$ and $x\mathcal{J}y$ then $x\mathcal{R}y$.*

▶ **Fact 48** (Stability of $\leq_{\mathcal{L}}$ and $\leq_{\mathcal{R}}$). *The following is true in any semigroup.*

1. *$\leq_{\mathcal{R}}$ is stable on the left, i.e., if $x \leq_{\mathcal{R}} y$ then $zx \leq_{\mathcal{R}} zy$. Hence, if $x\mathcal{R}y$ then $zx\mathcal{R}zy$.*
2. *$\leq_{\mathcal{L}}$ is stable on the right, i.e., if $x \leq_{\mathcal{L}} y$ then $xz \leq_{\mathcal{L}} yz$. Hence, if $x\mathcal{L}y$ then $xz\mathcal{L}yz$.*

## References


1 Ansuman Banerjee, Kingshuk Chatterjee, and Shibashis Guha. Set augmented finite automata over infinite alphabets. In *DLT*, volume 13911 of *LNCS*, pages 36–50. Springer, 2023.

2 Henrik Björklund and Thomas Schwentick. On notions of regularity for data languages. In *Fundamentals of Computation Theory*, pages 88–99. Springer Berlin Heidelberg, 2007.

3 Mikołaj Bojańczyk. Data monoids. In *STACS*, volume 9 of *LIPIcs*, pages 105–116. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011.

4 Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Logic*, 12(4), Jul 2011.

5 Mikołaj Bojańczyk and Rafał Stefański. Single-Use Automata and Transducers for Infinite Alphabets. In *ICALP*, volume 168 of *LIPIcs*, pages 113:1–113:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

6 Mikolaj Bojańczyk and Sławomir Lasota. An extension of data automata that captures xpath. In *LICS*, pages 243–252, 2010.

7 Thomas Colcombet, Clemens Ley, and Gabriele Puppis. On the use of guards for logics with data. In *MFCS*, volume 6907 of *LNCS*, pages 243–255. Springer, 2011.

8 Thomas Colcombet and Amaldev Manuel. Fragments of Fixpoint Logic on Data Words. In *FSTTCS*, volume 45 of *LIPIcs*, pages 98–111. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015.

9 Stephane Demri and Paul Gastin. Specification and verification using temporal logics. In *Modern Applications of Automata Theory*, 2012.

10 Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3):16:1–16:30, 2009.

11 Radu Grigore and Nikos Tzevelekos. History-register automata. *Log. Methods Comput. Sci.*, 12(1), 2016.

12 Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *The Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

13 Shibashis Guha, Amaldev Manuel, and S P Rishal. Set automata and limits of decidability of two-variable logic on data words. *CoRR*, abs/2605.09077, 2026. `arXiv:2605.09077`.

14 Sabína Gulčíková and Ondřej Lengál. Register set automata (technical report), 2022. `arXiv:2205.12114`.

15 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

16 Ahmet Kara, Thomas Schwentick, and Tony Tan. Feasible automata for two-variable logic with successor on data words. In *LATA*, volume 7183 of *LNCS*, pages 351–362. Springer, 2012.

17 Gerard Lallement. *Semigroups and combinatorial applications*. Wiley, New York, 1979.

18 Martin Leucker and César Sánchez. Regular linear temporal logic. In *Theoretical Aspects of Computing – ICTAC 2007*, pages 291–305. Springer Berlin Heidelberg, 2007.

19 Amaldev Manuel and Ramaswamy Ramanujam. Class counting automata on data words. *International Journal of Foundations of Computer Science*, 22, 11 2011.

20 Marvin L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., 1967.

21 Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Logic*, 5(3):403–435, Jul 2004.

22 Jean-Éric Pin. *Mathematical Foundations of Automata Theory*. 2022.

23 Thomas Place and Marc Zeitoun. All about unambiguous polynomial closure. *TheoretiCS*, Volume 2, Jan 2024.

24 Klaus Reinhardt. Reachability in petri nets with inhibitor arcs. *Electron. Notes Theor. Comput. Sci.*, 223:239–264, Dec 2008.

25 Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL*, pages 41–57. Springer Berlin Heidelberg, 2006.

26 Pierre Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1):72–99, 1983.