Reasoning on Data Words over Numeric Domains

Diego Figueira
diego.figueira@cnrs.fr
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800
Talence, France

Anthony W. Lin
awlin@mpi-sws.org
TU Kaiserslautern
Max Planck Institute for Software Systems
Kaiserslautern, Germany

ABSTRACT

We introduce parametric semilinear data logic (pSDL) for reasoning about data words with numeric data. The logic allows parameters, and Presburger guards on the data and on the Parikh image of equivalence classes (i.e. data counting), allowing us to capture data languages like: (1) each data value occurs at most once in the word and is an even number, (2) the subset of the positions containing data values divisible by 4 has the same number of a's and b's, (3) the data value with the highest frequency in the word is divisible by 3, and (4) each data value occurs at most once, and the set of data values forms an interval. We provide decidability and complexity results for the problem of membership and satisfiability checking over these models. In contrast to two-variable logic of data words and data automata (which also permit a form of data counting but no arithmetics over numeric domains and have incomparable inexpressivity), pSDL has elementary complexity of satisfiability checking. We show interesting potential applications of our models in databases and verification.

CCS CONCEPTS

• **Theory of computation** → **Transducers**; **Automata over infinite objects**; **Logic and verification**; **Modal and temporal logics**; **Regular languages**; **Complexity theory and logic**.

KEYWORDS

Data words, logic and automata, Presburger arithmetic, counting, complexity

ACM Reference Format:
Diego Figueira and Anthony W. Lin. 2022. Reasoning on Data Words over Numeric Domains. In *37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (LICS '22), August 2–5, 2022, Haifa, Israel.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3531130.3533354

1 INTRODUCTION

A data word is a word, each of whose positions contains a label drawn from a finite alphabet (just like a normal word in formal language theory), and a data value from some infinite domain. An example of data word over the alphabet $\mathbb{A} = \{a, b\}$ and data domain $\mathcal{D} = \mathbb{Z}$ is $(a, 7)(b, 10)(a, 3)(a, 100)$. The study of automata and logics over data words has spanned across nearly three decades, starting from the study of register automata [22] with a decidable emptiness problem. In addition to this basic register automata model, there is nowadays a plethora of variants of register automata and other different (and mostly incomparable) models of automata and logics over data words with a decidable emptiness problem including automata with pebbles [31], deterministic memory automata over ordered data [4], data automata and two-variable first-order logic [6, 39], alternating 1-register automata and LTL with freeze quantifiers [15], single-use register automata [8], nominal automata [7], streaming data-string acceptors [2] and its variant over rationals [10], and symbolic finite automata [14] and their extension with registers [13].

Most of the automata models and logics over data words with a decidable emptiness problem impose a severe restriction on the operations that can be performed on the data values, *i.e.*, mostly only comparing data equalities is permitted. In practice, however, we are interested in a specific domain theory like the set of integers and permit operations like those that are allowed in the theory of integer linear arithmetic. For example, consider the (SMT) theories of arrays (*e.g.* see [9, 26]). Structurally, arrays can be construed as data words without a finite alphabet (or equivalently with a unary finite alphabet) and integers as the data domain. However, theories of arrays permit the full integer linear arithmetic to express relationship among the data stored in the arrays, for which there is only a very limited support by any automata model and logic over data words. As we shall soon see, certain types of arithmetic reasoning are also not supported by array theories.

The main goal of this paper is to initiate an investigation of how integer arithmetic reasoning can be incorporated into automata models and logics over data words. In doing so, our hope is to bring automata/logic over data words closer to applications, e.g., in databases and verification.

What type of arithmetic reasoning? In the literature of logic and automata, many types of integer arithmetic reasoning have been considered, which include the following:

- **(i)** Integer arithmetic constraints on the data values in the input word, e.g., two positions $i < j$ in the word $w = w_1 \cdots w_n$ satisfy $data(w_i) > data(w_j)$, $data(w_i) \geq 100$ and $data(w_j) = 0 \mod 2$.
- **(ii)** Letter counting and length, *e.g.*, accept only words whose numbers of *a*s and *b*s coincide.
- **(iii)** Data counting, e.g., every data value occurs at most once in the input word.
- **(iv)** Aggregation, *e.g.*, the k-th largest (or most frequent) data value is even.

Existing models supporting arithmetic reasoning usually permit one but not other types of arithmetic reasoning. In practice, we

LICS '22, August 2–5, 2022, Haifa, Israel

ACM ISBN 978-1-4503-9351-5/22/08.
https://doi.org/10.1145/3531130.3533354

are often interested in combining two such types of reasoning, as explicated in Example 1.1 and Example 1.2.

Example 1.1. We have a daily log file containing a sequence of events of the form (a, i), where i is the user ID and $a \in \{-1, +1\}$ denoting that a dollar has been either spent (-1) or earned $(+1)$. Suppose that we want to ensure that each person earns at least as much as he spends. Such a property combines (ii) and (iii), and is to the best of our knowledge not expressible in any existing model with decidable satisfiability/emptiness on data words.

Example 1.2. We have a log file contining a sequence of pairs of the form $(id, height) \in \mathbb{N}^2$, where id is an id of a person in a group and height the integer round-off of the height of the person. For example, we want to check that each id appears exactly once and that the median of the heights in the sequence is between 170-180. This property makes use of (i), (ii), and (iv) and, to the best of our knowledge, is not expressible in existing decidable models.

State-of-the-art. As we mentioned, most existing models for reasoning over data words do not support arithmetic reasoning over numeric data domains. For example, guards over linear arithmetic (i.e. (i) above) are not allowed in models like two-variable logics $\text{FO}^2(<, +1, \sim)$ and data automata [5, 6]; this is FO^2 over data words with the order $(<)$, successor $(+1)$ and equal data-value $(\sim)$ binary relations. Here, one can talk about two positions $i < j$ in the input word having the same data value $data(i) = data(j)$, but for example not $data(i) < data(j)$. This limitation is partially lifted by Schwentick and Zeume [39], in that two data values can now be checked for inequality in their logic (e.g. $data(i) < data(j)$), at the expense of disallowing the successor relation $+1$ over positions in the input word (e.g. one cannot say now that $j = i + 1$, which can be done in [6]). The strengths of these formalisms lie in data counting, e.g., every datum occurs at most once in the word, or an even number of times; the latter can be done in data automata, but not in FO^2.

Relaxing the ability (iii) to perform data counting, more models can come into consideration. Array Property Fragment (APF) [9, 26] supports a full integer linear arithmetic reasoning on the array indices as well as the elements (i.e. (i)). In an APF formula, universal quantifiers are restricted, so as to allow decidability of satisfiability. APF can express, for instance, that an array is ordered. Array Folds Logic (AFL) [12] addresses the limitations of APF in performing length reasoning and aggregation (i.e. (iv)) at the expense of disallowing universal quantification. Unlike APF, however, AFL cannot express properties like an array is ordered. We also mention the model of *nondeterministic looping word automata with arithmetic* [18], which input ω-words and consider the theory of rational linear arithmetic. If one considers instead finite words and the theory of integer linear arithmetic, this model is strictly subsumed by AFL. Another noteworthy model is that of Symbolic Register Automata (SRA), which are an extension of symbolic automata [14] by registers that can be checked for equality. Such a model is a one-way automata model allowing Presburger guards on the currently seen data value, and can for example express that all seen data are even, and that two data in every two consecutive positions are different (which is not expressible in AFL). We finally mention the register automata model of [10] over rational linear arithmetic (inspired by the streaming transducer model in [3]), extending the original model [22] of Kaminski and Francez. Here, the registers are separated into control registers (on which guards comprising order comparisons can be applied) and data registers (allowing general arithmetic operations). The model supports rational arithmetic operations (i) and aggregation (iv), but not data counting (iii).

In summary, existing logic and automata models on data words still have limited support of arithmetic reasoning. In particular, models that support data counting (e.g. two-variable data logic and data automata [5, 6, 39]) typically do not permit arithmetics on numeric data domain, letter counting and length reasoning, and aggregation. Our goal is to identify a model that supports these four features, while admitting decidable emptiness with elementary complexity (unlike the case of $\text{FO}^2(<, +1, \sim)$ and data automata) and interesting potential applications in databases and verification.

Contributions. We propose in this paper Parametric Semilinear Data Logic (pSDL), which is an extension of Linear Temporal Logic (LTL) for reasoning about data words with numeric data (i.e. the data domain is the set of integers). Aiming to address the four types of arithmetic reasoning (i)–(iv), we extend the standard LTL with four features: (a) Presburger formulas, which serve two purposes, namely to check the data value located at a certain position, as well as to perform letter/data counting and length reasoning, (b) parameters (a form of read-only variables), which can be used in the Presburger formulas, (c) additional modalities of the form $\langle = \rangle_\beta$ (resp. $\langle \neq \rangle_\beta$) with $\langle = \rangle_{\beta(y_1,\ldots,y_n)}(\varphi_1, \ldots, \varphi_n)$ (resp. $\langle \neq \rangle_{\beta(y_1,\ldots,y_n)}(\varphi_1, \ldots, \varphi_n)$) carrying the meaning that one can jump to precisely y_i different positions (other than the current position) of the same data value satisfying φ_i (resp. different data value to the current position), where the integer linear arithmetic constraint $\beta(y_1, \ldots, y_n)$ holds. The resulting logic strictly extends LTL and the modal logic fragment of $\text{FO}^2(<, +1, \sim)$ (essentially, an extension of unary temporal logic [17] with the $\langle = \rangle$ and $\langle \neq \rangle$ modalities). More concretely, pSDL can express the property in Example 1.1. Moreover, in the process of proving decidability for pSDL satisfiability, we introduce the automata counterpart called Parametric Semilinear Data Automata (pSDA), whose expressivity strictly subsumes pSDL, as well as Parikh automata [23], symbolic automata [14], and nondeterministic looping word automata with integer linear arithmetic [18].

The following is the main result of the paper:

Theorem 1.3. *Satisfiability for pSDL is in 2-NEXP and is NEXP-hard. Satisfiability for the fragment* SDL_{MNF} *of pSDL without parameters and linear arithmetic constraints on data values in minterm normal form is NEXP-complete.*

Note that k-NEXP means "k-fold nondeterministic exponential time". The decidability and the complexity results go through a translation to pSDA, whose decidability and complexity of emptiness we also determine in this paper. Here, SDL denotes the fragment of pSDL without parameters. The restriction to minterm normal form (MNF) is one that is often applied in the literature of symbolic automata [14] — which enforces constraints on data values to be the same if they intersect — and does *not* decrease the expressivity of the model. For example, the constraints $p(x) := x > 7$ and $q(x) := x \equiv 3 \pmod 4$ have common solutions, but they can be turned into four constraints in MNF of the form $(\neg)p(x) \land (\neg)q(x)$.

Our theorem also implies that the aforementioned modal logic fragment of $\mathrm{FO}^2(<, +1, \sim)$ is decidable in elementary time (more precisely, in NEXP), unlike the case of $\mathrm{FO}^2(<, +1, \sim)$. This is the modal logic on data words having the successor, predecessor, future, and past binary relations as modalities, as well as the "equal data value" and the "distinct data value" relations. As an aside, our proofs establish interesting connections to Presburger Arithmetic with star operations [21, 34] and unary counting quantifiers [38].

Our logic pSDL has NP-complete membership (since satisfiability of quantifier-free Presburger formulas can be reduced to it), though it becomes solvable in polynomial-time when we restrict to SDL. We believe that these complexity classes could still allow efficient query evaluation (e.g. on our log file examples) with the help of SMT-solvers.

Last but not least, our results can be lifted to the data domain $\mathbb{Z}^k$ and $\mathbb{N}^k$ using a standard "flattening trick", e.g., $(a, 7, 8)(b, 7, 9)(a, 3, 100)$ over the alphabet $\mathbb{A} = \{a, b\}$ can be mapped to $(a_1, 7)(a_2, 8), (b_1, 7)(b_2, 9)(a_1, 3)(a_2, 100)$ over the alphabet $\mathbb{A}' = \{a_1, a_2, b_1, b_2\}$. This allows us to encode the property in Example 1.2. More generally, this allows us to reason about a sequence of events with applications (e.g. querying/static analysis over a time series data), and verifying invariants of array-manipulating programs.

Organization. We provide a more detailed exposition of SDL through examples and potential applications in Section 2. We fix notation and basic terminologies in Section 3. For readability, we start with the simpler fragment, i.e., SDL with 1-ary modalities, i.e., $\langle = \rangle_{\beta(\bar{y})}$ and $\langle \neq \rangle_{\beta(\bar{y})}$ with $|\bar{y}| = 1$. We define this logic in Section 4, provide the automata counterpart (called SDA), for which decidability and complexity of nonemptiness are proven in Section 5. Translation from SDL to SDA is in Section 6. We then provide the extensions to the general case — with parameters, and k-ary modalities — in Section 7. We conclude in Section 8.

2 PSDL: EXAMPLES AND APPLICATIONS

We provide here an overview of our logic pSDL by means of examples, and discuss potential applications thereof. In the sequel, we work with the data domain $\mathbb{N}$ of natural numbers, but our results easily extend to the data domain $\mathbb{Z}$ of all integers.

Querying log files. We now discuss Example 1.1 and Example 1.2. We first show how to express the property in Example 1.1. This example can already be done in SDL with 2-ary modalities. In particular, the formula expressing it is

$$\mathsf{G}(-1 \to \langle = \rangle_{y_2 > y_1}(-1, +1)).$$

Intuitively, the formula says that it is globally the case that if a user (say with a user ID *id*) spends \$1 (i.e. -1) at a particular time point on the day, then the user earns \$ y_2 on that day, which is at least the total spending (i.e. \$ $y_1 + 1$). In particular, y_2 here counts the number of occurrences of positions labeled by $(+1, id)$, while y_1 counts the number of positions (other than the current position, which is labeled by $(-1, id)$) labeled by $(-1, id)$. The above formula is in fact in $\mathrm{SDL}_{\mathrm{MNF}}$ because no parameters are used and that no arithmetic constraints on the current data values are applied.

We now proceed to the property in Example 1.2, which is a simple reasoning over a relational table. For simplicity, we will assume that only one person has the median height; this is easily extendable to the case when there are more persons with the median height, but will make the formula messier. Using the flattening trick, we consider the finite alphabet $\mathbb{A} = \{1, 2\}$ indicating the first/second arguments in the tuple $(id, height)$. Thus, we ensure that the input word is of the form $((1, ?), (2, ?))^*$, where ? can indicate any number. This can be enforced easily in LTL, e.g., $\mathsf{G}((1 \to \mathsf{X}2) \wedge (2 \wedge \mathsf{X}\top \to \mathsf{X}1))$. Next, we enforce that each ID occurs uniquely in the sequence. This can be enforced by the formula

$$\mathsf{G}(1 \to \neg \langle = \rangle_{y \geq 1} 1)$$

which says that globally one cannot jump to another tuple whose first argument has the same ID as the current one. Indeed, when parameterized with $y \geq 1$, the construct $\langle = \rangle_{y \geq 1} \psi$ can be regarded as the modality "jump to a position with the same data value satisfying ψ". Finally, we use the parameter p_{med} to determine the median

$$\mathsf{F}(2 \wedge 170 \leq x = p_{med} \leq 180 \wedge \langle \neq \rangle_{y_1 = y_2}(x < p_{med}, x > p_{med})).$$

The formula first finds the second argument of a tuple in the table. Here, x denotes the current data value that is "saved" into p_{med}. [In the sequel, x is mostly used to denote the current data value.] This is required since our modality "forgets" the current data value, which has to then be alleviated by the use of parameters. The final conjunct simply says that there are the same number $y_1 = y_2$ of people who are shorter than the person with the median height and those who are taller than the person with the median height. Observe that linear arithmetic constraints are used for two purposes in the above formula: as counting constraints (e.g. $y_1 = y_2$), as well as for limiting the values that certain locations in the input word can take (e.g. $x < p_{med}$).

We show that the first query above can be checked in polynomial-time. The second query, on the other hand, can be written in pSDL, whose membership problem is NP-complete (*cf.* Theorem 7.3). We leave it for future work to determine whether SMT-solvers could be used to effectively perform such a query evaluation for pSDL. On the side of static analysis, Theorem 1.3 implies that vacuity of our queries can be automatically checked.

Array-manipulating programs. We now show a simple application of pSDL for verifying that the bubble sort preserves the invariant *Inv* that "every value occurs precisely once". We will model the bubble sort algorithm as a repeated nondeterministic application of swapping the element x_i at position i and the element x_j at position j such that $i < j$ and $x_i > x_j$. To treat this more formally, we need to model a transduction T for this swap relation.

We model T as the data language over the boosted alphabet $\mathbb{A} = \{a, b, c\}$ containing all words w obtained by replacing the ith position (a, d_i) (resp. jth position (a, d_j)) in the data word $(a, d_1) \cdots (a, d_n)$ by $(b, d_i)(c, d_j)$ (resp. $(b, d_j)(c, d_i)$), for some $i < j$ and $d_i > d_j$. Observe that the subsequence w_1 of w whose first components are a or b represents the initial array content, while the subsequence w_2 of w whose first components are a or c represents the result of applying T.

Example 2.1. Suppose T is to swap the 2nd and 4th elements in the array $[4, 7, 1, 2, 0]$. We represent this array the word $w = (a, 4)(b, 7)(c, 2)(a, 1)(b, 2)(c, 7)(a, 0)$.

Thus, $w_1 = (a,4)(b,7)(a,1)(b,2)(a,0)$ gives the original array, while $w_2 = (a,4)(c,2)(a,1)(c,7)(a,0)$ represents the array obtained after applying the swap. □

Note that we can express T quite easily in pSDL. First we express that the projection to the first components is in $a^*bca^*bca^*$, which is easily expressible in LTL (and so in pSDL). The following formula φ expresses that the swap takes place:

$$\mathsf{F}(b \wedge p = x \wedge \mathsf{X}((p' = x \wedge p > p') \wedge \mathsf{F}(b \wedge p' = x \wedge \mathsf{X}(p = x)))).$$

Note that x is used to record the current data value, while the parameter p (resp. p') is used to save d_i (resp. d_j).

To disprove that *Inv* is an invariant, we need to show that, there exists an input data word w such that w_1 satisfies *Inv* but w_2 satisfies $\neg Inv$. The following SDL formula ψ expresses this:

$$\mathsf{G}((a \vee b) \wedge \neg \langle = \rangle_{y \geq 1}(a \vee b)) \wedge \neg \mathsf{G}((a \vee c) \wedge \neg \langle = \rangle_{y \geq 1}(a \vee c)).$$

The final formula is $\varphi \wedge \psi$, which is unsatisfiable since *Inv* is an invariant under T. The decidability of pSDL implies that this satisfiability can be algorithmically checked.

Other properties. We conclude this section by collecting a few examples that can be expressed in pSDL. As far as we are aware, these cannot be expressed in other formalisms with decidable satisfiability/emptiness problem.

- **(P1)** Each data value occurs at most once in the word and is an even number.
- **(P2)** Property (P1) and the subset of the positions containing data values divisible by 4 has the same number of a's and b's.
- **(P3)** Each data value occurs an even number of times, and a most frequent data is even.
- **(P4)** Each data value occurs at most once, and the set of data values forms an interval.
- **(P5)** Each data occurs at most once, and the k-th biggest value is the length of the word.
- **(P6)** Each data value occurs the same number of times.

For example, (P3) can be expressed in pSDL as the conjunction of

$$\mathsf{G}(\langle = \rangle_{1 \leq y < p \wedge y \equiv 1 \pmod 2} \top).$$

and

$$\mathsf{F}(x \equiv 0 \pmod 2) \wedge \langle = \rangle_{p-1=y \geq 0} \top)$$

(Recall that $\langle = \rangle$ is 'strict', in the sense that it only counts occurrences different from the current position's.) Note that the parameter p is used as a placeholder for the most frequent data value in the input word. As another example, assuming that each data value occurs in the input at most once (which we saw is expressible in pSDL), (P4) can be expressed as a conjunction of $\mathsf{F}(x = p_{max}) \wedge \mathsf{F}(x = p_{min})$ and

$$\mathsf{G}(p_{min} \leq x \leq p_{max}) \wedge \langle \neq \rangle_{y = p_{max} - p_{min}} \top.$$

Here, we save the maximum and minimum data values into parameters, and say that there are precisely $p_{max} - p_{min} + 1$ data values in the input word. Because of uniqueness of data values in the input word, we are guaranteed to have every data value between $[p_{min}, p_{max}]$ in the input word. Note, however, that this trick does not apply when we allow each data value to occur more than once.

3 PRELIMINARIES

Basic notation. Let $\mathbb{N} = \{0, 1, 2, \dots\}$. We write $\underline{k}$ to denote the set $\{1, \dots, k\}$. The set of finite words over a domain A is denoted by A^*. We will often work with finite words over the cartesian product of pairwise disjoint alphabets, *e.g.*, $w \in (A \times B \times C)^*$. We use letters $\mathbb{A}$, $\mathbb{B}$ to denote *finite* alphabets. For $w \in (\mathbb{A} \times \mathbb{N})^*$, we write $data(w)$ and $lab(w)$ to denote the projection of w onto $\mathbb{N}$ and $\mathbb{A}$ respectively. Given a word $w \in A^*$ and a set $I \subseteq \{1, \dots, |w|\}$, we write $w[I]$ to denote the subword of w given by the indices in I (*e.g.*, $w[\{1, \dots, |w|\}] = w$, $w[\emptyset] = \varepsilon$). We write $w[i]$ as short for $w[\{i\}]$. We write $|w|$ to denote the length of w.

Parikh images, semilinear sets, Presburger arithmetic. The **Parikh image** of a word $w \in \mathbb{A}^*$ over a finite alphabet $\mathbb{A}$, is a function $\Pi(w) : \mathbb{A} \to \mathbb{N}$ assigning to each $a \in \mathbb{A}$ the number of appearances of a in w. The Parikh image of a language $L \subseteq \mathbb{A}^*$ is $\Pi(L) = \{\Pi(w) : w \in L\} \subseteq \mathbb{N}^A$.

A **linear set** is a subset of $\mathbb{N}^k$ that can be described as an arithmetic progression $\{\bar{v}_0 + \alpha_1 \bar{v}_1 + \cdots + \alpha_n \bar{v}_n \mid \alpha_1, \dots, \alpha_n \in \mathbb{N}\}$ for some $n \in \mathbb{N}$ and $\bar{v}_0, \dots, \bar{v}_n \in \mathbb{N}^k$. A **semilinear set** is a finite union of linear sets. Linear sets are represented by the **offset** $\bar{v}_0$ and the **generators** $\bar{v}_1, \dots, \bar{v}_n$, where numbers are represented in binary. **Presburger arithmetic** refers to first-order logic in the language of addition (+), inequality (≤), and modulo k (mod k) operators for every $k > 1$, evaluated over the natural numbers (this is sometimes called *extended* Presburger arithmetic). For example, $\exists x\, (x \geq y + y) \wedge ((y + x) \mod 19 = y)$ is a Presburger formula with one free variable. Each Presburger formula $\varphi(x_1, \dots, x_k)$ with k free variables denotes a set $[\![\varphi]\!] \stackrel{\text{def}}{=} \{(n_1, \dots, n_k) \in \mathbb{N}^k : (n_1, \dots, n_k) \models \varphi\}$. It is well-known that semilinear sets correspond precisely to Presburger arithmetic [19] and to Parikh images of context free (or regular) languages by Parikh's Theorem [32]. A **quantifier-free Presburger formula** is any Presburger formula with no quantifiers. Presburger formulas admit quantifier elimination [20, 35]: for every Presburger formula there exists an equivalent quantifier-free formula. An existential formula is a Presburger formula of the form $\exists x_1, \dots, x_n \varphi$, where φ is quantifier-free.

We extend now Presburger Arithmetic with the *star* operator *. For any formula $\varphi(x_1, \dots, x_n)$ and $m \geq 0$, we permit formulas $\varphi^{\leq m}$ and φ^* with semantics $[\![\varphi^{\leq m}]\!] := \{\bar{t}_1 + \cdots + \bar{t}_{m'} : m' \leq m \text{ and } \bar{t}_i \models \varphi \text{ for every } i\} \subseteq \mathbb{N}^n$ (or $\emptyset$ if $m = 0$), and $[\![\varphi^*]\!] := \bigcup_{m \geq 0} [\![\varphi^{\leq m}]\!]$. We define, in an analogous way, sets S^* and $S^{\leq m}$ for any set $S \subseteq \mathbb{N}^n$. It is known that, for every existential Presburger formula φ, φ^* is also expressible by an existential Presburger arithmetic formula of at most exponential size [34], and hence that S^* is semilinear assuming S is too. Observe that $\varphi^{\leq m}$ can be expressed as follows:

$$\varphi^{\leq m}(\bar{x}) = \exists y\, \psi^*(\bar{x}y) \wedge y \leq m, \text{ where } \psi(\bar{x}y) = \varphi(\bar{x}) \wedge y = 1. \quad (\star)$$

Note that variable y is used to 'count' the number of applications of ψ^*. Observe that in the translation above, the resulting formula is of size logarithmic in m. Piskac and Kuncak [34] have shown that existential Presburger formulas with star is NP-complete, so long as they are of star-height 1 (i.e. no nesting of the star operator is allowed). As recently shown in [21], this NP upper bound can be generalized to any fixed star-height.

Complexity classes We use standard notations for complexity classes [25], including NP, PSPACE, k-NEXP, #P, PP, $P^{\#P}$, and NP^{NP}. For example, 2-NEXP is the class of problems solvable by a nondeterministic Turing machine in double exponential time. The class #P is the class of counting problems, whose solutions correspond to the number of accepting paths of a nondeterministic polynomial-time Turing machine. Some of these classes have also oracle access. For example, $P^{\#P} = P^{PP}$ (e.g. see [41]) corresponds to the class of problems solvable in polynomial time with access to a #P oracle. By Toda's theorem ([40], see also [25]), $P^{\#P}$ contains the entire polynomial hierarchy (PH). The class NP^{NP} corresponds to the second-level of PH. Finally, we use the class $P^{NP[\log]}$ [44] of problems solvable in polynomial-time with logarithmically many calls to an NP oracle. It is known that $P^{NP[\log]}$ contains the entire boolean hierarchy, which in turn contains NP, co-NP, DP, etc.

4 SEMILINEAR DATA LOGIC

We now formally define Semilinear Data Logic (SDL). For readability, we disallow parameters and restrict to 1-ary modalities. This will be generalized in Section 7. SDL has an LTL-navigational flavor, featuring common modalities such as Next, Future, Until, Since, etc. On top of that, it has two kinds of extra modalities. One modality $\langle = \rangle_\beta \, \psi$ which allows to state that β satisfies n, for n the number of positions j different from the current one with the same data value and a certain property ψ. And another modality $\langle \neq \rangle_\beta \, \psi$ which works similarly but for positions with *different* data values. We use quantifier-free Presburger formulas for testing for such properties β. Further, we allow Presburger guards on data values.

The logic can express the following properties:

- "for every a-position there is a b-position with the same value",
- "there are no two a-positions with the same value", or
- "there are no two consecutive positions with the same value".

The first two properties above can be expressed with previously studied logics such as $FO^2(<, \sim)$, and the last one with $FO^2(+1, <, \sim)$ logics of [6], using register automata [22] or freeze-LTL [16]. Further, using the linear arithmetic power, we can 'count' the number of positions with the same data value as the current one. One can then express properties like "for every a-position with an even data value there is an odd number of b-positions with the same value".

Definition. The syntax of Semilinear Data Logic (SDL) over words $w \in (\mathbb{A} \times \mathbb{N})^*$ is given by the following grammar:

$$\varphi ::= a \mid \alpha \mid \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \varphi \,\mathsf{S}\, \varphi \mid \langle = \rangle_\beta \, \varphi \mid \langle \neq \rangle_\beta \, \varphi,$$

where $a \in \mathbb{A}$, and α, β are quantifier-free Presburger formulas with one free variable x. We call a and α **base formulas** of the logic since they correspond to leaves in the grammar derivations. As usual, we write $\bot$ as short for $a \wedge \neg a$ for some $a \in \mathbb{A}$; $\top$ for $\neg\bot$; and $\varphi \vee \psi$ for $\neg(\neg\varphi \wedge \neg\psi)$.

For base formulas, we define the satisfaction relation on a word $w \in (\mathbb{A} \times \mathbb{N})^*$ as $w, i \models \alpha$ if $z \models \alpha$, where $z = data(w[i])$; and $w, i \models a$ if $a = lab(w[i])$. The U, S modalities have the expected LTL semantics, where we define them as 'strict' modalities: $w, i \models \varphi \,\mathsf{U}\, \psi$ (resp. $w, j \models \varphi \,\mathsf{S}\, \psi$) if there is $j > i$ such that $w, j \models \psi$ and for every $i < \ell < j$ we have $w, \ell \models \varphi$. As it is customary, we use the standard LTL modalities as shorthands: $\mathsf{F}\varphi \stackrel{\text{def}}{=} \top \,\mathsf{U}\, \varphi$, $\mathsf{X}\varphi \stackrel{\text{def}}{=} \bot \,\mathsf{U}\, \varphi$, $\mathsf{G}\varphi \stackrel{\text{def}}{=} \neg\mathsf{F}\neg\varphi$, $\mathsf{F}^{-1}\varphi \stackrel{\text{def}}{=} \top \,\mathsf{S}\, \varphi$, $\mathsf{X}^{-1}\varphi \stackrel{\text{def}}{=} \bot \,\mathsf{S}\, \varphi$, $\mathsf{G}^{-1}\varphi \stackrel{\text{def}}{=} \neg\mathsf{F}^{-1}\neg\varphi$. The remaining modalities are the key constructs for testing for data values. Given a word $w \in (\mathbb{A} \times \mathbb{N})^*$, for any position $1 \leq i \leq |w|$, we have $w, i \models \langle = \rangle_\beta \, \varphi$ (resp. $w, i \models \langle \neq \rangle_\beta \, \varphi$) if the number $n \in \mathbb{N}$ of positions $j \in \{1, \ldots, |w|\}$ distinct from i such that (i) $w, j \models \varphi$, (ii) $data(w)[j] = data(w)[i]$ (resp. (i) $w, j \models \varphi$ and (ii) $data(w)[j] \neq data(w)[i]$) is such that $n \models \beta$. Analogously, $w, i \models \langle \neq \rangle_\beta \, \varphi$ if the number $n \in \mathbb{N}$ of positions $j \in \{1, \ldots, |w|\}$ distinct from i such that (i) $w, j \models \varphi$ and (ii) $data(w)[j] \neq data(w)[i]$ is such that $n \models \beta$.

Observe that we have opted for a 'strict' version of the $\langle = \rangle$ modality, in which we count positions *different* from the current one, to be in line with the semantics of U and S. However, a non-strict version $\langle\!\langle = \rangle\!\rangle$ of the modality is definable by $\langle\!\langle = \rangle\!\rangle_{\beta(y)}(\psi) \stackrel{\text{def}}{=} (\neg\psi \wedge \langle = \rangle_{\beta(y)}(\psi)) \vee (\psi \wedge \langle = \rangle_{\beta(y+1)}(\psi))$.

Remark. *Notice that data modalities are closed under taking dual, in the sense: $\neg \langle = \rangle_\beta \, \psi \equiv \langle = \rangle_{\neg\beta} \, \psi$. Observe also that, for $\beta(x) := x \geq 1$, the formula $\langle = \rangle_\beta \, \psi$ evaluated at position i of w tests whether there exists some other position j with the same data value satisfying ψ. In a similar way, we can test that there are at least ℓ (using $\beta(x) := x \geq \ell$) or that there are an even number of such positions j (with $\beta(x) := (x \bmod 2 = 0)$). Indeed,* SDL *allows for counting properties for each data equivalence class. This particular restriction in fact subsumes the modal logic fragment of* $FO^2(<, +1, \sim)$. *As we shall see later, our logic has the advantage of admitting elementary complexity, in contrast to that* $FO^2(<, +1, \sim)$ *being not primitive-recursive. This is because* $FO^2(<, +1, \sim)$ *satisfiability can capture reachability of Petri nets, which is decidable [24, 27, 29, 30] but not primitive-recursive [11, 28].*

Model checking. The model checking problem for this logic, that is, the problem of given a formula φ and a word $w \in (\mathbb{A} \times \mathbb{N})^*$ whether $w, 1 \models \varphi$ is in polynomial time.

Proposition 4.1. *The model checking problem for SDL is in polynomial time.*

Proof. Given $w \in (\mathbb{A} \times \mathbb{N})^*$ and $\varphi \in$ SDL, we use the following standard algorithm to mark each position $1 \leq i \leq |w|$ with the set of subformulas ψ of φ such that $w, i \models \psi$. We proceed by induction: we first treat base formulas, and then formulas containing already treated subformulas.

For each base subformula ψ of φ, we can mark each position i such that $w, i \models \psi$ in linear time (remember that formulas are quantifier-free). For each subformula $\psi \,\mathsf{U}\, \psi'$ or $\psi \,\mathsf{S}\, \psi'$ we can also mark which positions satisfy the formula in linear time, assuming ψ, ψ' have been already treated. Similarly for $\neg\psi$ and $\psi \wedge \psi'$. For a subformula of the form $\langle = \rangle_\beta \, \psi$ we proceed as follows: For each data value d of w, we first count the number n of positions of w having data d and satisfying ψ, and we then mark each position i with data d as satisfying $\langle = \rangle_\beta \, \psi$ iff

- the position is marked as satisfying ψ and $n - 1 \models \beta$, or
- the position is marked as not satisfying ψ and $n \models \beta$.

Observe that this takes quadratic time. Finally, for a subformula of the form $\langle \neq \rangle_\beta \, \psi$ we proceed similarly: For each data value d of w, we count the number n of positions having data different from d and satisfying ψ, and we mark each position i with data d as satisfying $\langle \neq \rangle_\beta \, \psi$ iff $n \models \beta$.

Once all the markings are done, we answer 'yes' if the first position is marked with the input formula φ, and 'no' otherwise. □

Satisfiability. Here we focus on the satisfiability problem, that is, the problem of, given a formula φ whether there is some $w \in (\mathbb{A} \times \mathbb{N})^*$ such that $w, 1 \models \varphi$.

We say that the formula ψ of SDL is in **minterm normal form (MNF)** if for every pair of distinct Presburger base subformulas α, α' thereof, we have that $\alpha(x) \wedge \alpha'(x)$ is unsatisfiable.[1] In particular, $(x \geq 2) \wedge \mathsf{F}(x \leq 5)$ is not in MNF, but $\langle = \rangle_{x \geq 2} \top \wedge \mathsf{F} \langle = \rangle_{x \leq 5} \top$ is. Let $\mathrm{SDL}_{\mathrm{MNF}}$ be the set of formulas in MNF. We will show the following in the next couple of sections.

Theorem 4.2.

(1) The satisfiability problem for SDL *is in 2*NEXP.

(2) The satisfiability problem for SDL_{MNF} *is* NEXP*-complete.*

The gap between 2NEXP and NEXP is due to the cost of bringing the logic to minterm normal form. Closing the gap seems to be a difficult problem, which underlies also the difficulties that are dealt with in practice by symbolic automata algorithms [14]. We leave this as an open problem. The NEXP-hardness proof can be found in the full version of this paper, it follows by a reduction from the *exponential tiling problem* [42]. We will show the upper bound of items 1 and 2 by reduction to an automata model 'SDA', or Semilinear Data Automata, that we introduce in the next section. In fact, SDA is considerably more expressive than SDL. The remaining sections will be dedicated first to defining and showing decidability for SDA, and then to prove the upper bound for the satisfiability of SDL, via an effective language-preserving translation to SDA.

5 SEMILINEAR DATA AUTOMATA

We present an automata model which we call Semilinear Data Automata (SDA) and prove some basic properties (e.g. closures, decidability) about them. We will show in Section 6 that it captures SDL.

5.1 Definition

For a finite alphabet $\mathbb{A}$, we define a language acceptor of words over $\mathbb{A} \times \mathbb{N}$. A **Semilinear Data Automaton (SDA) over** $\mathbb{A}$ is a pair (T, S) where, for a finite alphabet $\mathbb{B}$, we have:

(1) S is a semilinear set over $\mathbb{N}^{\mathbb{B}} \times \mathbb{N}^{\mathbb{B}}$;

(2) $T \subseteq (\mathbb{A} \times \mathbb{N})^* \times \mathbb{B}^*$ is a length-preserving transducer, defined via a regular language $L_T \subseteq (\mathbb{A} \times \Psi \times \mathbb{B})^*$ for a finite set Ψ of quantifier-free Presburger formulas $\varphi(x)$ with one free variable x, and some finite alphabet $\mathbb{B}$. T denotes the set of all pairs $(w, w') \in (\mathbb{A} \times \mathbb{N})^* \times \mathbb{B}^*$ such that there exist a sequence $\psi_1, \ldots, \psi_\ell$ of Ψ-formulas where:
   (i) $|w| = |w'| = \ell$,
   (ii) $data(w)[i] \models \psi_i$ for all $i \in \underline{\ell}$, and
   (iii) $(lab(w)[1], \psi_1, w'[1]) \cdots (lab(w)[\ell], \psi_\ell, w'[\ell])$ is in L_T.
   We call $\mathbb{B}$ the **output alphabet** of T and Ψ its **Presburger alphabet**.

A word $w \in (\mathbb{A} \times \mathbb{N})^*$ is *accepted* by such an SDA if there exists some $w' \in \mathbb{B}^*$ such that

(i) $(w, w') \in T$, and

(ii) for every $n \in \mathbb{N}$, $(\Pi(w'[I_n]), \Pi(w'[\bar{I}_n])) \in S$,

where $I_n = \{1 \leq j \leq |w| : data(w)[j] = n\}$ and $\bar{I}_n = \{1, \ldots, |w|\} \setminus I_n$.

Henceforth, we assume that semilinear sets $S \subseteq \mathbb{N}^{\mathbb{B}} \times \mathbb{N}^{B}$ as above are represented as quantifier-free Presburger formulas, using variables $x_b^{=}$ for each $b \in \mathbb{B}$ for the first $\mathbb{B}$-components, and variables $x_b^{\neq}$ for each $b \in \mathbb{B}$ for the last $\mathbb{B}$-components. Similarly, regular languages are represented as non-deterministic finite automata (NFA). We say that an SDA is in **minterm normal form (MNF)** if, for every pair of transitions $(p, (a, \varphi, b), q)$, $(p', (a', \varphi', b'), q')$ of the NFA representing its transducer T, either $\varphi \wedge \varphi'$ is unsatisfiable or $\varphi = \varphi'$.

The definition of SDA is inspired by the Data Automata (DA) model introduced in [6]. However, it is incomparable in expressive power. On the one hand DA work with an abstract infinite domain equipped with an equivalence relation, and the transducer part of DA is just a letter-to-letter transducer $T \subseteq \mathbb{A}^* \times \mathbb{B}^*$. It cannot express, *e.g.*, the SDA property "each datum associated with the letter a is even". On the other hand, DA can test for regular properties of equivalence classes (*e.g.*, "every class is a word in $(ab)^*$") which cannot be expressed by SDA, whereas SDA can test for semilinear constraints on the Parikh-image of equivalence classes (*e.g.*, "for every class there are as many a's as b's") which cannot be expressed by DA. Unfortunately, a generalization of both DA and SDA is infeasible: in the full version of this paper, we show that generalizing both mechanisms (*i.e.*, DA extended with semilinear constraints) would result in a model with undecidable emptiness problem.

Proposition 5.1.

(1) SDA *are effectively closed under union and intersection.*

(2) SDA *are not closed under complement.*

(3) The universality, equivalence and containment problems for SDA *are undecidable.*

Proof of item (1). Assume (T_1, S_1) and (T_2, S_2) are SDA. We assume without any loss of generality that the output alphabet $\mathbb{B}_1$ of T_1 and $\mathbb{B}_2$ of T_2 are disjoint. Let $\mathbb{B}_1 = \{a_1, \ldots, a_{\ell_1}\}$ and $\mathbb{B}_2 = \{b_1, \ldots, b_{\ell_2}\}$.

($\cup$) Assuming S_1 and S_2 are given by Presburger formulas

$$\varphi_{S_1}(x_{a_1}, \ldots, x_{a_{\ell_1}}, y_{a_1}, \ldots, y_{a_{\ell_1}}) \text{ and}$$
$$\varphi_{S_2}(x_{b_1}, \ldots, x_{b_{\ell_2}}, y_{b_1}, \ldots, y_{b_{\ell_2}}),$$

we simply define $S = [\![\varphi_S]\!]$, where $\varphi_S = \varphi_{S_1} \vee \varphi_{S_2}$. Finally, let $T = T_1 \cup T_2$ —whence having an output alphabet $\mathbb{B}_1 \dot\cup \mathbb{B}_2$. It then follows that the language of (T, S) is the union of the languages of the two input SDA.

($\cap$) For intersection, we essentially build the product of both automata, which here implies considering the cartesian product of the alphabets (in the same way as intersection of DFA implies considering the cartesian product of the states).

This time we define S as a set over $\mathbb{N}^{\mathbb{B}_1 \times \mathbb{B}_2} \times \mathbb{N}^{\mathbb{B}_1 \times \mathbb{B}_2}$. Assuming S_1, S_2 are given by formulas $\varphi_{S_1}, \varphi_{S_2}$ as before, we define S as $\varphi_S(\{x_{a_i, b_j}, y_{a_i, b_j}\}_{i \in \underline{\ell_1}, j \in \underline{\ell_2}}) = \varphi_1 \wedge \varphi_2$, where

$$\varphi_\alpha \;=\; \varphi_{S_\alpha}(t^x_{a_1}, \ldots, t^x_{a_{\ell_\alpha}}, t^y_{a_1}, \ldots, t^y_{a_{\ell_\alpha}}),$$

[1] Note that we do not include formulas β of the $\langle = \rangle_\beta$ and $\langle \neq \rangle_\beta$ modalities.

for $\alpha \in \{1,2\}$, $t^x_{a_i} = \sum_{j \in \underline{\ell_{3-\alpha}}} x_{a_i,b_j}$, $t^y_{a_i} = \sum_{j \in \underline{\ell_{3-\alpha}}} y_{a_i,b_j}$ for every $i \in \underline{\ell_\alpha}$. We finally define $T = T_1 \times T_2$, that is, given w it outputs $u \in (\mathbb{B}_1 \times \mathbb{B}_2)^*$ iff $(w, u_{\mathbb{B}_1}) \in T_1$ and $(w, u_{\mathbb{B}_2}) \in T_2$. We then have that (T, S) denotes the intersection of the languages of the two SDA. □

Proof of items (2) and (3). It is easy to see that SDA is effectively equivalent in expressive power to Parikh automata [23] when disregarding the numeric domain $\mathbb{N}$. Since the latter has an undecidable universality problem, it follows that the universality problem for SDA is also undecidable. The undecidability for the equivalence and containment problems follow thus as corollaries. Analogously, the fact that SDA are not closed under complement can be seen also as a consequence of Parikh automata not being closed under complement. □

5.2 The Emptiness Problem for SDA

Theorem 5.2. *The emptiness problem for*

*(1) …*SDA *is decidable in* NEXP.
*(2) …*SDA *in minterm normal form is in* $\mathrm{P}^{\#\mathrm{P}}$ *and* $\mathrm{P}^{\mathrm{NP[log]}}$*-hard.*
*(3) …*SDA *in minterm normal form whose transducers use no modular predicates is* NP*-complete.*

We will first show decidability, and then explain how the bounds follow from the proof.

Lemma 5.3. *The emptiness problem for* SDA *is decidable.*

Proof. Let $\mathcal{A} = (T, S)$ be a SDA. Let Φ be the Presburger alphabet of T. For $P \subseteq \Phi$, we say that $x \in \mathbb{N}$ has **profile** P if it satisfies the formula $\pi_P(x) \stackrel{\text{def}}{=} \bigwedge_{\psi \in P} \psi(x) \wedge \bigwedge_{\psi \in \Phi \setminus P} \neg\psi(x)$. Let $\mathcal{P}_\infty$ be the set of profiles P such that $|[\![\pi_P]\!]| = \infty$ and let $\mathcal{P}_{<\infty}$ be the remaining ones. For $P \in \mathcal{P}_{<\infty}$, let n_P be the number of $x \in \mathbb{N}$ with profile P. Observe that

$$n_P = |[\![\pi_P]\!]|. \qquad (\dagger)$$

Consider T seen as a regular language over $\mathbb{A} \times \Phi \times \mathbb{B}$. By Parikh's Theorem [33], its Parikh image is semilinear, and an existential Presburger formula $\varphi_T(\bar{x})$ representing it can be produced, even in linear time [43]. Assume that $\bar{x}$ (*i.e.*, the free variables of $\varphi_T(\bar{x})$) has a variable $x_{a,\varphi,b}$ for every $a \in \mathbb{A}$, $b \in \mathbb{B}$ and $\varphi \in \Phi$, representing the number of appearances of $(a, \varphi, b) \in \mathbb{A} \times \Phi \times \mathbb{B}$ in the word; and let us assume $\mathbb{B} = \{b_1, \dots, b_m\}$.

We now need to verify whether a given satisfying valuation for $\bar{x}$ in φ_T is such that one can produce a word in the language of $\mathcal{A}$. For this, we will need to first guess how each number denoted by the $x_{a,\varphi,b}$ variables is distributed across profiles. Then we have to check, separately for each profile P, that the guessed number of elements with profile P is in agreement with the bound n_P as defined above. Concretely, for a valuation of $\bar{x}$, consider the property $\mu_\sim$ stating that

(1) there exists a number $x^P_{a,\varphi,b} \in \mathbb{N}$ for every possible $a \in \mathbb{A}$, $b \in \mathbb{B}$, profile P, and $\varphi \in P$, such that $x_{a,\varphi,b} = \sum_{P \ni \varphi} x^P_{a,\varphi,b}$, and
(2) letting $t_i = \sum_{\varphi \in \Phi, a \in \mathbb{A}} x_{a,\varphi,b_i}$ ($i \in \underline{m}$) and $\hat{S} \subseteq \mathbb{N}^{\mathbb{B}}$ be $\{(x_1, \dots, x_m) : (x_1, \dots, x_m, (t_1 - x_1), \dots, (t_m - x_m)) \in S\}$,
   (i) for every $P \in \mathcal{P}_\infty$, we have
   $$\left(\sum_{a \in \mathbb{A}, \varphi \in P} x^P_{a,\varphi,b_1}, \dots, \sum_{a \in \mathbb{A}, \varphi \in P} x^P_{a,\varphi,b_m} \right) \in \hat{S}^*; \text{ and}$$
   (ii) for every $P \in \mathcal{P}_{<\infty}$, we have
   $$\left(\sum_{a \in \mathbb{A}, \varphi \in P} x^P_{a,\varphi,b_1}, \dots, \sum_{a \in \mathbb{A}, \varphi \in P} x^P_{a,\varphi,b_m} \right) \in \hat{S}^{\leq n_P}.$$

The idea is that (1) checks that the partitioning of each $x_{a,\varphi,b}$ into profiles $x^P_{a,\varphi,b}$ is consistent, and (2) guarantees that the numbers are such that: for those profiles for which we can generate as many data tuples as we want, the sum of vectors belongs to $\hat{S}^*$, and for profiles containing only n_P-many data tuples, the sum of vectors belongs to $\hat{S}^{\leq n_P}$. Formally, we obtain the following.

Claim 1. *There exists* $\bar{x} \in \mathbb{N}^{\mathbb{A}\times\Phi\times\mathbb{B}}$ *such that* $\bar{x} \models \varphi_T$ *and* $\bar{x} \models \mu_\sim$ *if, and only if,* $\mathcal{A}$ *has a non-empty language.*

($\Leftarrow$) We first show the right-to-left direction of Claim 1. Assume $w \in (\mathbb{A} \times \mathbb{N})^*$ is in the language of $\mathcal{A}$, let $w' \in (\mathbb{A} \times \Phi \times \mathbb{B})^*$ be the witnessing word of the transducer. We show that $\bar{x} = \Pi(w')$ satisfies both properties. The fact that $\bar{x} \models \varphi_T$ goes by definition. For the satisfaction of $\mu_\sim$, assume $x^P_{a,\varphi,b}$ is the number of positions i of w' such that $w'[i] = (a, \varphi, b)$ and *data*$(w)[i]$ has profile P (i.e. *data*$(w)[i] \models \pi_P$). It follows that it is a partition of $\bar{x}$ satisfying item (1) of $\mu_\sim$. We now proceed to show item (2). Let $w'_{\mathbb{B}}$ be the projection of w' onto its $\mathbb{B}$ component. For $d \in \mathbb{N}$, let $I_d = \{1 \leq i \leq |w| : \mathit{data}(w)[i] = d\}$ and $\bar{I}_d = \{1, \dots, |w|\} \setminus I_d$. For every $j \in \underline{m}$ and $d \in \mathbb{N}$, let x^d_j be the number of indices i such that *data*$(w)[i] = d$ and $w'_{\mathbb{B}}[i] = b_j$; in other words $(x^d_1, \dots, x^d_m) = \Pi(w'_{\mathbb{B}}[I_d])$ for every d. Observe that for every j,

$$\sum_{a \in \mathbb{A}, \varphi \in P} x^P_{a,\varphi,b_j} = \sum_{d \in \mathbb{N} \text{ s.t. } d \models \pi_P} x^d_j. \qquad (\dagger)$$

Now, let us fix some $d \in \mathbb{N}$ appearing in w, and suppose it has profile P. We show that $(x^d_1, \dots, x^d_m) \in \hat{S}$. By definition of $\hat{S}$, this happens if, and only if, $(x^d_1, \dots, x^d_m, (t_1 - x^d_1), \dots, (t_m - x^d_m)) \in S$, where $(t_1, \dots, t_m) = \Pi(w'_{\mathbb{B}})$. Hence, $(t_j - x^d_j)$ is the number of positions $1 \leq i \leq |w|$ such that $w'_{\mathbb{B}}[i] = b_j$ and *data*$(w)[i] \neq d$ (*i.e.*, the number of b_j's in $w'_{\mathbb{B}}[\bar{I}_d]$). In other words, $((t_1 - x^d_1), \dots, (t_m - x^d_m)) = \Pi(w'_{\mathbb{B}}[\bar{I}_d])$. Since w' is a witness for non-emptiness of $\mathcal{A}$, it follows that $(\Pi(w'_{\mathbb{B}}[I_d]), \Pi(w'_{\mathbb{B}}[\bar{I}_d])) \in S$, and by the remarks above $(x^d_1, \dots, x^d_m) \in \hat{S}$. In view of (†), we must then have that

$$\left(\sum_{a \in \mathbb{A}, \varphi \in P} x^P_{a,\varphi,b_1}, \dots, \sum_{a \in \mathbb{A}, \varphi \in P} x^P_{a,\varphi,b_m} \right) \in \hat{S}^{\leq \alpha},$$

for some $\alpha \in \mathbb{N}$ which cannot be greater than the number of elements from $\mathbb{N}$ with profile P. This shows that both conditions (2i) and (2ii) must hold true.

($\Rightarrow$) For the left-to-right direction of Claim 1, suppose $\varphi_T \wedge \mu_\sim$ has a satisfying assignment $\bar{x} \in \mathbb{N}^{\mathbb{A}\times\Phi\times\mathbb{B}}$. We show how to build a word $w \in (\mathbb{A} \times \mathbb{N})^*$ in the language of $\mathcal{A}$. Since $\bar{x} \models \varphi_T$ there must be some $w' = (c_1, \psi_1, c'_1) \cdots (c_\ell, \psi_\ell, c'_\ell) \in L_T$ such that $\Pi(w') = \bar{x}$.

Since $\bar{x} \models \mu_\sim$ each index $1 \leq j \leq |w'|$ can be assigned a profile P_{r_j} so that the word restricted to any fixed profile P satisfies $\sum_{a\in\mathbb{A},\varphi\in P}(x^P_{a,\varphi,b_1},\dots,x^P_{a,\varphi,b_m}) \in \hat{S}^{\leq\alpha}$ for some $\alpha \in \mathbb{N}$ such that $\alpha \leq n_P$ if $|[\![\pi_P]\!]| < \infty$. For any such profile P, we can then take any α-many pairwise distinct elements $d^P_1,\dots,d^P_\alpha \in \mathbb{N}$ with profile P, and assign to each index $j \in \{1 \leq j \leq \ell : P_{r_j} = P\}$ some value $d^P_{i_j}$ so that the Parikh image restricted to each d^P_i is in $\hat{S}$. The final word w in the language of $\mathcal{A}$ is then any word $w = (c_1,d_1)\cdots(c_\ell,d_\ell) \in (\mathbb{A}\times\mathbb{N})^*$ such that $d_j = d^{P_{r_j}}_{i_j}$ for every $1 \leq j \leq \ell$. This concludes the proof of Claim 1.

We finally show that these properties can be expressed in Presburger arithmetic. Since as already discussed $\varphi_T(\bar{x})$ is an existential Presburger formula, it only remains to show:

Claim 2. *$\mu_\sim(\bar{x})$ is expressible by a Presburger formula.*

Let χ_S be a formula having, besides $\bar{x}$, some extra free variables $x_1,\dots,x_m$, defined as

$$\chi_S(x_1,\dots,x_m,\bar{x}) \stackrel{\text{def}}{=} (x_1,\dots,x_m,(t_1-x_1),\dots,(t_m-x_m)) \in S,$$

where $t_i = \sum_{\varphi\in\Phi,a\in\mathbb{A}} x_{a,\varphi,b_i}$ for every $i \in \underline{m}$. For $\alpha \in \{*\}\cup\mathbb{N}$, let $\varphi^{\langle\alpha\rangle}_S$ be the formula expressing that there exist variables $\bar{y}$ (one for each x_{a,φ,b_i}) such that $\chi^{\leq\alpha}_S(x_1,\dots,x_m,\bar{y})$ holds.[2] For the sake of brevity we will henceforth abuse notation writing $\exists_{cond(P,a,\varphi,b)} x^P_{a,\varphi,b}\ \psi$ to denote $\exists x^{P_1}_{a_1,\varphi_1,b_1}\cdots\exists x^{P_z}_{a_z,\varphi_z,b_z}\ \psi$ for all the triples P_i,a_i,φ_i,b_i satisfying the condition *cond*. Now we define $\mu_\sim(\bar{x}) \stackrel{\text{def}}{=} \exists_{a\in\mathbb{A},b\in\mathbb{B},P\subseteq\Phi,\varphi\in P}\ x^P_{a,\varphi,b}\ A\wedge B\wedge C$, where

$$A = \bigwedge_{\substack{a\in\mathbb{A},\\ \varphi\in\Phi,\\ b\in\mathbb{B}}} \left(x_{a,\varphi,b} = \sum_{P\ni\varphi} x^P_{a,\varphi,b} \right),$$

$$B = \bigwedge_{P\in\mathcal{P}_\infty} \varphi^{\langle *\rangle}_S \left(\sum_{a\in\mathbb{A},\varphi\in P} x^P_{a,\varphi,b_1},\dots,\sum_{a\in\mathbb{A},\varphi\in P} x^P_{a,\varphi,b_m},\bar{x} \right),$$

$$C = \bigwedge_{P\in\mathcal{P}_{<\infty}} \varphi^{\langle n_P\rangle}_S \left(\sum_{a\in\mathbb{A},\varphi\in P} x^P_{a,\varphi,b_1},\dots,\sum_{a\in\mathbb{A},\varphi\in P} x^P_{a,\varphi,b_m},\bar{x} \right).$$

It is straightforward to see that $\mu_\sim$ is an existential Presburger formula expressing properties (1) and (2). Hence, decidability follows from decidability of the satisfiability problem for Presburger formulas. ◻

Corollary 5.4. *For every* SDA *recognizable language $L \subseteq (\mathbb{A}\times\mathbb{N})^*$, we have $\Pi(\{lab(w) : w \in L\}) \subseteq \mathbb{N}^{\mathbb{A}}$ is semilinear.*

As a corollary of the previous proof we obtain the bounds of Theorem 5.2.

Proof of Theorem 5.2. First observe that, in the proof of Lemma 5.3, $\mu_\sim$ uses $\hat{S}^*$ in its definition. As already mentioned this *star* operator preserves semilinearity [34], but the equivalent existential Presburger formulas without star may be of exponential size. However, in [21] it is shown that the satisfiability problem for existential Presburger formulas with star operators which happen to be of star-height 1 (as is our case) is decidable in NP. Observe that, in light of the translation (★), $\hat{S}^{\leq n_P}$ can be written as an existential formula of star-height 1 of size logarithmic in n_P and polynomial in the formula expressing $\hat{S}$. On the other hand, counting the number of satisfying assignments of an existential Presburger formula φ is in the counting hierarchy [1], in particular in $\#\mathrm{P}^{\mathrm{NP}}$. This is because if $[\![\varphi]\!]$ is finite, then any satisfying assignment for φ use numbers which are at most exponential [36]; hence an NP Turing machine can guess an assignment $\bar{x} \in \mathbb{N}^k$ and accept iff $\bar{x} \models \varphi$, which necessitates a call to an NP procedure for existential Presburger satisfiability. The number of accepting runs will then correspond to the number of satisfying assignments.

Proposition 5.5 ([36, 45]). *For any quantifier-free formula φ we have the following bounds. $|[\![\varphi]\!]|$ and $\|[\![\varphi]\!]\|_\infty$ are bounded by some singly exponential function [36].[3] Further, if φ does not use modular predicates, $|[\![\varphi]\!]|$ can be computed in polynomial time [45].[4]*

In view of Proposition 5.5, observe that we can compute, in #P, $|[\![\pi_P]\!]|$ for every profile P. Further, if no formula of the transducer uses modular predicates, $|[\![\pi_P]\!]|$ can be computed in polynomial time.

Bearing all this in mind, we now proceed to extract the stated upper bounds.

(1) Let (T,S) be a SDA. We compute, in exponential time, all the n_P's and we produce a singly exponential sized existential formula $\varphi_T \wedge \mu_\sim$ of star-height 1, whose satisfiability can be checked in NEXP (in the size of the automaton).

(2) Let (T,S) be a SDA in MNF. Observe that in this case the nonempty *profiles* are just singleton sets, and hence that π_P can be equivalently expressed as $\pi_{\{\varphi\}}(x) = \varphi(x)$. In this case, we can compute in $\mathrm{P}^{\#\mathrm{P}}$ the n_P's according to (†). Thus, the produced formula $\varphi_T\wedge\mu_\sim$ can be written as a polynomial sized existential formula of star-height 1, which can be tested in NP. This gives an $\mathrm{P}^{\#\mathrm{P}}$ upper bound. The lower bound can be found in the full version of the paper.

(3) As already observed, in this case $n_{\{\varphi\}}$ can be computed in polynomial time, which was the bottleneck of the previous case. Thus, we end up with an NP procedure.

NP-hardness follows by an easy reduction form SAT. Given a Boolean formula φ in n variables $x_1,\dots,x_n$ we produce the semilinear set $S \subseteq \mathbb{N}^{\mathbb{B}}\times\mathbb{N}^{\mathbb{B}}$ for $\mathbb{B} = \{b_1,\dots,b_n\}$, as a quantifier-free formula with free variables $\{y^=_i\}_{i\in\underline{n}}, \{y^{\neq}_i\}_{i\in\underline{n}}$ as the result of replacing each x_i with $y^=_i + y^{\neq}_i > 0$ in φ. We finally let the transducer T be the set of all words $(w,w') \subseteq (\mathbb{A}\times\mathbb{N})^*\times\mathbb{B}^*$ such that $w' \in \mathbb{B}^{|w|}$. It is easy to check that the resulting SDA (T,S) is non-empty if, and only if, φ is satisfiable. Observe that NP-hardness is independent of using or not modular predicates in S, and of data classes. ◻

6 SATISFIABILITY OF SDL

In order to prove decidability for SDL, we show an effective translation from the logic to SDA. We focus here on the upper bounds of items (1) and (2) from Theorem 4.2. The lower bound of item (2)

[2] Recall the definition of $\cdot^*$ and $\cdot^{\leq m}$ of (★).

[3] $\|S\|_\infty$ is the maximum value contained in any of the components of an element of S.

[4] In fact, [45] shows that the number of solutions of an existential Presburger formula with a fixed number of variables is polynomial-time computable.

follows by a reduction from the *exponential tiling problem* [42] and can be found in the full version of this paper.

For a formula ψ, we write $\psi^\neg$ to denote ψ' if ψ is of the form $\neg\psi'$, or $\neg\psi$ otherwise. Given a formula $\varphi \in$ SDL, let $sub(\varphi) = \{\psi, \psi^\neg : \psi$ a subformula of $\varphi\}$. A set $S \subseteq sub(\varphi)$ is a **maximally consistent** set of φ on the alphabet $\mathbb{A}$ if it is $\subseteq$-maximal with respect to the following properties

(1) for every $\psi \in sub(\varphi)$, $\psi \in S$ iff $\psi^\neg \notin S$,
(2) for every $\psi, \psi' \in sub(\varphi)$, $\psi \wedge \psi' \in S$ iff $\psi \in S$ and $\psi' \in S$,
(3) there is $a \in \mathbb{A}$ s.t. $a \in S$ and for every $b \in \mathbb{A} \setminus \{a\}$, $\neg b \in S$.

Let us write MCS(φ) to denote the set of all maximally consistent sets of φ (the alphabet being implicit). Two sets $S, S' \in$ MCS(φ) are **one-step consistent** if they satisfy

(a) $\psi_1 \mathbin{\mathsf{U}} \psi_2 \in S$ iff $\{\psi_1 \mathbin{\mathsf{U}} \psi_2, \psi_1\} \subseteq S'$ or $\psi_2 \in S'$;
(b) $\psi_1 \mathbin{\mathsf{S}} \psi_2 \in S'$ iff $\{\psi_1 \mathbin{\mathsf{S}} \psi_2, \psi_1\} \subseteq S$ or $\psi_2 \in S$.

We define an exponential-sized SDA $\mathcal{A}_\varphi = (T, \mathcal{S})$, whose language consists of all data words that satisfy φ. We define $T \subseteq (\mathbb{A}\times\mathbb{N})^* \times \mathbb{B}^*$ as a transducer over the output alphabet $\mathbb{B} =$ MCS(φ). T is defined as the set of all pairs $((a_1, d_1) \cdots (a_n, d_n), S_1 \cdots S_n)$ such that

(i) $\varphi \in S_1$;
(ii) for every $1 \le i < n$ we have that S_i, S_{i+1} are one-step consistent;
(iii) for every $1 \le i \le n$ we have that $a_i \in S_i$;
(iv) for every $1 \le i \le n$ and Presburger formula $\alpha \in S_i$, we have $d_i \models \alpha$.

We define $\mathcal{S} \subseteq \mathbb{N}^{\mathbb{B}} \times \mathbb{N}^{\mathbb{B}}$ as the set denoted by the quantifier-free formula with variables $\{x_{=,S}\}_{S\in\mathbb{B}} \cup \{x_{\neq,S}\}_{S\in\mathbb{B}}$ consisting on the conjunction of:

(I) $x_{=,S} > 0 \to \alpha((\sum_{S'\in\mathbb{B},\psi\in S'} x_{=,S'}) - r)$ for every α, ψ, S such that $\langle = \rangle_\alpha\, \psi \in S$, where $r = 1$ if $\psi \in S$ or $r = 0$ otherwise;
(II) $x_{=,S} > 0 \to \neg\alpha((\sum_{S'\in\mathbb{B},\psi\in S'} x_{=,S'}) - r)$ for every α, ψ, S such that $\neg\langle = \rangle_\alpha\, \psi \in S$, where $r = 1$ if $\psi \in S$ or $r = 0$ otherwise;
(III) $x_{=,S} > 0 \to \alpha((\sum_{S'\in\mathbb{B},\psi\in S'} x_{\neq,S'}))$ for every α, ψ, S with $\langle \neq \rangle_\alpha\, \psi \in S$; and
(IV) $x_{=,S} > 0 \to \neg\alpha((\sum_{S'\in\mathbb{B},\psi\in S'} x_{\neq,S'}))$ for every α, ψ, S with $\neg\langle \neq \rangle_\alpha\, \psi \in S$.

Observe that $\mathcal{S}$ is a single exponential quantifier-free formula. Therefore $\mathcal{A}_\varphi$ is computable in exponential time. Hence, in the light of Theorem 5.2–(1) we obtain a 2NEXP upper bound as stated in item (1) of Theorem 4.2. Further, if φ is in MNF, then $\mathcal{A}_\varphi$ is too. Observe that the size of base formulas in φ is logarithmic in terms of the size of $\mathcal{A}_\varphi$. This means that the cardinalities $|[\![\pi_P]\!]|$ that need to be computed for every profile P (which are singleton since we are in MNF) are at most polynomial in $\mathcal{A}_\varphi$, and can then be computed in space logarithmic in $\mathcal{A}_\varphi$. With this in mind, following the upper bound proof of Theorem 5.2–(2), we obtain a non-deterministic polynomial time algorithm in the size of $\mathcal{A}_\varphi$ for its non-emptiness. This then yields the NEXP upper bound of Theorem 4.2–(2).

Lemma 6.1. *A word is accepted by $\mathcal{A}_\varphi$ if, and only if, it satisfies φ.*

Proof. ($\Leftarrow$) Suppose first $w, 1 \models \varphi$ and let us show that w is accepted by $\mathcal{A}_\varphi$. Let w' be a word of length $|w|$ whose i-th position is labelled with $\{\psi \in sub(\varphi) : w, i \models \psi\}$, for every i. It is easy to verify that (i) $w' \in \mathbb{B}^*$, (ii) $(w, w') \in T$, and (iii) $(\Pi(w'[I_d]), \Pi(w'[\bar{I}_d])) \in \mathcal{S}$ for every $d \in \mathbb{N}$, $I_d = \{1 \le i \le |w| : data(w)[i] = d\}$, and $\bar{I}_d = \{1, \ldots, |w|\} \setminus I_d$.

($\Rightarrow$) Suppose now that $w \in (\mathbb{A}\times\mathbb{N})^*$ is accepted by $\mathcal{A}_\varphi$, and let us show that $w, 1 \models \varphi$. Let $w' \in \mathbb{B}^*$ be the witnessing word used for the acceptance of w. We will show that for every position i and subformula ψ: $\psi \in w'[i]$ iff $w, i \models \psi$. We show this by induction on the size of ψ. The base case is when ψ is either (a) a letter $a \in \mathbb{A}$ which follows by condition (iii) in the definition of T, or (b) a Presburger formula α which follows by condition (iv). Boolean combinations follow by induction as a direct consequence of the definition of MCS(φ).

The Until modality follows by applying the the one-step-consistency: $\psi \mathbin{\mathsf{U}} \psi' \in w'[i]$ iff there is some $i' > i$ such that $\psi' \in w'[i']$ and for every $i < j < i'$ we have $\psi \in w'[j]$. The Since modality follows analogously.

Consider finally a subformula of the form $\langle = \rangle_\alpha\, \psi \in sub(\varphi)$, and let $d = w_{\mathbb{N}}[i]$. By the semilinear constraint $\mathcal{S}$, we have $\langle = \rangle_\alpha\, \psi \in w'[i]$ if, and only if, the number n of distinct positions of $w'[I_d]$ containing ψ is such that $n \models \alpha$. By induction, this happens iff there are n positions $1 \le i_1 < \cdots < i_n \le |w'|$ such that $w, i_j \models \psi$ for all $j \in \{1, \ldots, n\}$. Hence, $\langle = \rangle_\alpha\, \psi \in w'[i]$ iff $w, i \models \langle = \rangle_\alpha\, \psi$. The case of $\langle \neq \rangle_\alpha\, \psi$ is analogous. ◻

Corollary 6.2 (of Lemma 6.1 and Corollary 5.4). *The spectrum (i.e. the set of sizes of models) of any* SDL *formula is semilinear.*

7 EXTENSIONS

We show in in this section how to extend our results with parameters and k-ary modalities.

7.1 Adding parameters

Adding parameters to SDL. We use pSDL to denote the extension of SDL with parameters. The definition is the same as before but now all Presburger formulas (base formulas and formulas used in modalities) may use some extra free variables $p_1, \ldots, p_t$ which correspond to the *parameters*. Now the satisfaction relation $w, i \models_\sigma \varphi$ is defined relative to some parameter valuation $\sigma : \{p_1, \ldots, p_t\} \to \mathbb{N}$. For any Presburger base formula α, we define $w, i \models_\sigma \alpha$ iff $x, \sigma \models \alpha$, where $x = data(w[i])$. Given a word $w \in (\mathbb{A}\times\mathbb{N})^*$, for any position $1 \le i \le |w|$, we have $w, i \models_\sigma \langle = \rangle_\beta\, \varphi$ (resp. $w, i \models \langle \neq \rangle_\beta\, \varphi$) iff the number $n \in \mathbb{N}$ of positions $1 \le j \le |w|$ such that (i) $w, j \models_\sigma \varphi$, (ii) $data(w[j]) = data(w[i])$ and (iii) $j \neq i$ (resp. (i) $w, j \models \varphi$ and (ii) $data(w[j]) \neq data(w[i])$) is such that $n, \sigma \models \beta$. Finally, $w, i \models \varphi$ holds if there exists some σ such that $w, i \models_\sigma \varphi$.

Adding parameters to SDA. To derive decidability and complexity of pSDL, we extend SDA with parameters, which we call parametric SDA (pSDA). A **parametric SDA (pSDA)** with t parameters, is a tuple (T, S) as before, but the formulas in the transitions of T may also use some parameters $p_1, \ldots, p_t$. Now T is a regular language over $\mathbb{A} \times \Psi \times \mathbb{B}$, where Ψ a finite set of quantifier-free Presburger formulas with free variables $x, p_1, \ldots, p_t$, and S is a semilinear set over $\mathbb{N}^{\mathbb{B}} \times \mathbb{N}^{\mathbb{B}} \times \mathbb{N}^{\{p_1,\ldots,p_t\}}$. Acceptance is defined analogously: A word $w \in (\mathbb{A}\times\mathbb{N})^*$ is *accepted* by (T, S) if for some $w' \in \mathbb{B}^*$ and valuation $\sigma \in \mathbb{N}^{\{p_1,\ldots,p_t\}}$, we have

(i) $(w, w') \in T_\sigma$, where T_σ is the transducer without parameters obtained by replacing each p_i with $\sigma(p_i)$,
(ii) for every $x \in \mathbb{N}$, $(\Pi(w'[I_x]), \Pi(w'[\bar{I}_x]), \sigma) \in S$.

where $I_x = \{1 \le j \le |w| : data(w[j]) = x\}$ and $\bar{I}_x = \{1, \ldots, |w|\} \setminus I_x$.

This model is still closed under union and intersection. The construction is exactly as in Proposition 5.1 (item (1)) assuming, without any loss of generality, that the parameter names used by both automata are disjoint.

We show that the decidability proof of Lemma 5.3 can be adapted to having parameters.

Theorem 7.1. *The emptiness problem for* pSDA *is in* NEXP *and* $\mathrm{NP}^{\mathrm{NP}}$*-hard.*

Proof. For the upper bound, suppose the pSDA automaton $\mathcal{A} = (T, S)$ has t parameters $p_1, \ldots, p_t$ and T is in minterm normal form (that is, in minterm normal form for every possible instantiation of the parameters). We follow closely the proof of Lemma 5.3. The first difference being that now φ_T has some t extra free variables $p_1, \ldots, p_t$. We will need to adjust $\mu_\sim$ to take into account the parameter valuations. Observe that each n_P may depend on the assignment of parameters $p_1, \ldots, p_t$. The crux of the proof will still be to produce a Presburger formula φ such that φ is satisfiable if and only if $\mathcal{A}$ has a non-empty language. But in order to do this, we need to use two constructs in the logic, which preserve semilinearity, and which we describe next.

Given a Presburger formula $\varphi(\bar{x})$ and a fresh variable y let $\varphi^{\le y}(\bar{x})$ be a formula with free variables $\bar{x}y$. Its semantics is such that $\varphi^{\le y}(\bar{x})$ is satisfied by a valuation $\tilde{y}$ of y and $\bar{n}$ of $\bar{x}$ if $\bar{n} \models [\![\varphi]\!]^{\le \tilde{y}}$ where, recall,

$$C^{\le \tilde{y}} = \{\bar{x}_1 + \cdots + \bar{x}_{\tilde{y}'} : \tilde{y}' \le \tilde{y} \text{ and } \bar{x}_i \in C \text{ for every } i\}.$$

Observe that $[\![\varphi^{\le y}]\!]$ is effectively semilinear, definable by the same star-height 1 formula of $(\star)$: $\varphi^{\le y}(\bar{x}) = \exists y' \psi^*(\bar{x}y') \wedge y' \le y$, where $\psi(\bar{x}y') = \varphi(\bar{x}) \wedge y' = 1$. It then follows that $\varphi_S^{\langle y \rangle}(\bar{x})$ is definable as a star-height 1 existential formula.

Consider the following **unary counting quantifier** $\exists^{=x} y \;\; \psi(y, p_1, \ldots, p_t)$ having $x, p_1, \ldots, p_t$ as free variables, which expresses that, for a given assignment $\tilde{p}_1, \ldots, \tilde{p}_t \in \mathbb{N}$ of $p_1, \ldots, p_t$ and $\tilde{x} \in \mathbb{N}$ of x, there are exactly $\tilde{x}$ many different valuations $\tilde{y} \in \mathbb{N}$ of variable y such that $(\tilde{y}, \tilde{p}_1, \ldots, \tilde{p}_t) \models \psi$. It was shown in [38] that such a quantifier preserves semilinearity. Although the complexity was not explicitly mentioned in the paper, the algorithm of [38] could be easily adapted to produce an equivalent existential Presburger formula (without counting quantifiers) in single exponential time. See the full version of the paper for more details. Observe that given an assignment of $p_1, \ldots, p_t$, the number of equivalence classes with profile $P \subseteq \Phi$ is given by the satisfying valuation of y in the formula

$$\rho_P(y, p_1, \ldots, p_t) \overset{\text{def}}{=} \exists^{=y} x \bigwedge_{\varphi \in P} \varphi(x, p_1, \ldots, p_t) \wedge \bigwedge_{\varphi \in \Phi \setminus P} \neg\varphi(x, p_1, \ldots, p_t).$$

Hence, for a given assignment $\sigma \in \mathbb{N}^{\{p_1, \ldots, p_t\}}$ we have that there are finitely many distinct equivalence classes with profile $P \subseteq \Phi$ if, and only if, $\sigma \models \exists y \; \rho_P$.

We also define the infinite version ρ_P^∞ of ρ_P:

$$\rho_P^\infty(p_1, \ldots, p_t) \overset{\text{def}}{=} \exists^\infty x \bigwedge_{\varphi \in P} \varphi(x, p_1, \ldots, p_t) \wedge \bigwedge_{\varphi \in \Phi \setminus P} \neg\varphi(x, p_1, \ldots, p_t).$$

The quantifier $\exists^\infty x \; \psi(x, \bar{z})$ simply says there are infinitely many x's such that $\varphi(x, \bar{z})$ is true. By standard results for quantifier-free and Presburger arithmetic [36], we could replace $\exists^\infty x \; \psi$ with $\exists x \; (x > C \wedge \psi)$ for some constant C that is exponential in the size of ψ (which can therefore be represented in polynomial size in binary).

Then, the final formula is

$$\mu_\sim(\bar{x}, \bar{p}) \overset{\text{def}}{=} \exists_{a \in \mathbb{A}, b \in \mathbb{B}, P \subseteq \Phi, \varphi \in P} \; x^P_{a,\varphi,b} A \wedge \bigwedge_{P \subseteq \Phi} B_P, \text{ where}$$

$$A = \bigwedge_{a \in \mathbb{A}, \varphi \in \Phi, b \in \mathbb{B}} \left(x_{a,\varphi,b} = \sum_{P \ni \varphi} x^P_{a,\varphi,b} \right),$$

$$B_P = \left(\rho_P^\infty \wedge \varphi_S^{\langle * \rangle}(\tau_1, \ldots, \tau_m, \bar{x})\right) \vee \left(\exists y \; \rho_P \wedge \varphi_S^{\langle y \rangle}(\tau_1, \ldots, \tau_m, \bar{x})\right), \text{ and}$$

$$\tau_i = \sum_{a \in \mathbb{A}, \varphi \in P} x^P_{a,\varphi,b_i} \text{ for every } i \in \underline{m}.$$

As before, the language is non-empty if, and only if, $\varphi_T(\bar{x}, \bar{p}) \wedge \mu_\sim(\bar{x}, \bar{p})$ is satisfiable. Note that now any satisfying assignment does not only yield the Parikh image under T of the witnessing word but also the valuation for all parameters. Since the $\exists^{=y}$ quantifier can be eliminated in exponential time, ρ_P can be translated into an equivalent, single-exponential size existential Presburger formula . Thus, $\varphi_T(\bar{x}, \bar{p}) \wedge \mu_\sim(\bar{x}, \bar{p})$ is an exponential sized existential formula of star-height 1, whose satisfiability can be checked in NP. Thus, the upper bound follows.

We now prove that pSDA emptiness is $\mathrm{NP}^{\mathrm{NP}}$-hard. The reduction is from the standard $\mathrm{NP}^{\mathrm{NP}}$-complete problem [25, 37] of satisfiability for quantified Boolean formulas of the form

$$F := \exists y_1, \ldots, y_n \forall z_1, \ldots, z_n G(\bar{y}, \bar{z})$$

where G is a quantifier-free Boolean formula. The corresponding pSDA (T, S) will use the parameter p for encoding assignments to $\bar{y}$, and will only have one state q, which is both initial and final. The assignments to $\bar{y}$ will be stored as the data values of the pSDA. Let $1 < r_1 < \cdots < r_n$ be the first n primes. We use the Gödel encoding techniques for encoding G as a Presburger formula φ_G. Namely by recursive definition:

(1) $\varphi_G := 0 = p \mod r_i$ if $G = y_i$,
(2) $\varphi_G := 0 = x \mod r_i$ if $G = z_i$,
(3) $\varphi_G := \varphi_{G_1} \wedge \varphi_{G_2}$ if $G = G_1 \wedge G_2$, and
(4) $\varphi_G := \neg\varphi_{G_1}$ if $G = \neg G_1$.

To finish the reduction, let $\mathbb{A} = \mathbb{B} = \{a\}$, and $R := \prod_{i=1}^n r_i$. The only transition of T is

$$(q, (a, \varphi_G \wedge 0 < x \le R, a), q)$$

Finally, the semilinear set S is given by the quantifier-free formula $x_a^{=} = 1 \wedge x_a^{\neq} = R - 1$. These enforce that only permutations of the word $(a, 1) \cdots (a, R)$ could be accepted by (T, S), *i.e.*, must contain each of the Gödel encoding of assignments for $\bar{z}$ restricted to the interval $\{1, \dots, R\}$ exactly once as data values. Therefore, F is true iff (T, S) is nonempty. ◻

Satisfiability of SDL. It is easy to see that the reduction of SDL to SDA can be adapted to work also in the case of parameters, which yields decidabilty for the satisfiability problem. We comment more on this adaptation below when discussing extensions with k-ary modalities.

Theorem 7.2. *The satisfiability problem for* pSDL *is in 2*NEXP.

7.2 SDL with k-ary modalities

The logic SDL (with or without parameters) can be also extended with k-ary versions of the (unary) data modalities $\langle = \rangle_{\beta(y,\bar{p})}(\varphi)$ and $\langle \neq \rangle_{\beta(y,\bar{p})}(\varphi)$. We consider now formulas with k-ary modalities of the form $\langle = \rangle_{\beta(y_1,\dots,y_k,\bar{p})}(\varphi_1, \dots, \varphi_k)$ and $\langle \neq \rangle_{\beta(y_1,\dots,y_k,\bar{p})}(\varphi_1, \dots, \varphi_k)$.

Given a parameter valuation $\sigma : \bar{p} \to \mathbb{N}$, a word $w \in (\mathbb{A} \times \mathbb{N})^*$, and a position $1 \le i \le |w|$, we define the satisfaction relation $w, i \models_\sigma \langle = \rangle_{\beta(y_1,\dots,y_k,\bar{p})}(\varphi_1, \dots, \varphi_k)$ (resp. $w, i \models \langle \neq \rangle_{\beta(y_1,\dots,y_k,\bar{p})}(\varphi_1, \dots, \varphi_k)$) iff $n_1, \dots, n_k, \sigma \models \beta$, where each n_ℓ (for $\ell \in \underline{k}$) is the number of positions $1 \le j \le |w|$ such that (i) $w, j \models_\sigma \varphi_\ell$, (ii) $data(w[j]) = data(w[i])$ and (iii) $j \neq i$ (resp. (i) $w, j \models_\sigma \varphi_\ell$ and (ii) $data(w[j]) \neq data(w[i])$). As before, $w, i \models \varphi$ holds if $w, i \models_\sigma \varphi$ for some σ. Let us call SDL$^+$ and pSDL$^+$ the extensions of SDL and pSDL with k-ary modalities (for every k), respectively.

The exponential-time translation from this further extension to SDA can be adapted, and we obtain the following.

Theorem 7.3.

- *Satisfiability for* pSDL$^+$ *is in 2*NEXP.
- *Model-checking for* SDL$^+$ *is in PTIME.*
- *Model-checking for* pSDL$^+$ *and* pSDL *is* NP*-complete.*

Proof. For satisfiability, we can translate in exponential time from the logic to pSDA. The translation is exactly as defined in Section 6 but now the semilinear set $\mathcal{S}$ needs to be updated to take into account the k-ary modalities semantics. That is, we define $\mathcal{S} \subseteq \mathbb{N}^{\mathbb{B}} \times \mathbb{N}^{\mathbb{B}} \times \mathbb{N}^{\bar{p}}$ as the set denoted by the quantifier-free formula with variables $\{x_{=,S}\}_{S \in \mathbb{B}} \cup \{x_{\neq,S}\}_{S \in \mathbb{B}} \cup \bar{p}$ consisting on the conjunction of:

(I) $x_{=,S} > 0 \to \alpha(t_1, \dots, t_k, \bar{p})$ for every $\alpha, \psi_1, \dots, \psi_k, S$ such that $\langle = \rangle_{\alpha(y_1,\dots,y_k,\bar{p})}(\psi_1, \dots, \psi_k) \in S$, where for each $i \in \underline{k}$, $t_i = (\sum_{S' \in \mathbb{B}, \psi_i \in S'} x_{=,S'}) - r_i$ and $r_i = 1$ if $\psi_i \in S$ or $r_i = 0$ otherwise;

(II) $x_{=,S} > 0 \to \neg\alpha(t_1, \dots, t_k, \bar{p})$ for every $\alpha, \psi_1, \dots, \psi_k, S$ such that $\neg \langle = \rangle_{\alpha(y_1,\dots,y_k,\bar{p})}(\psi_1, \dots, \psi_k) \in S$, where for each $i \in \underline{k}$, $t_i = (\sum_{S' \in \mathbb{B}, \psi_i \in S'} x_{=,S'}) - r_i$ and $r_i = 1$ if $\psi_i \in S$ or $r_i = 0$ otherwise;

(III) $x_{=,S} > 0 \to \alpha(t_1, \dots, t_k, \bar{p})$ for every $\alpha, \psi_1, \dots, \psi_k, S$ such that $\langle \neq \rangle_{\alpha(y_1,\dots,y_k,\bar{p})}(\psi_1, \dots, \psi_k) \in S$, where for each $i \in \underline{k}$, $t_i = \sum_{S' \in \mathbb{B}, \psi_i \in S'} x_{\neq,S'}$;

(IV) $x_{=,S} > 0 \to \neg\alpha(t_1, \dots, t_k, \bar{p})$ for every $\alpha, \psi_1, \dots, \psi_k, S$ such that $\neg \langle \neq \rangle_{\alpha(y_1,\dots,y_k,\bar{p})}(\psi_1, \dots, \psi_k) \in S$, where for each $i \in \underline{k}$, $t_i = \sum_{S' \in \mathbb{B}, \psi_i \in S'} x_{\neq,S'}$;

Observe that $\mathcal{S}$ is still a singly-exponential-sized quantifier-free formula. A similar argument as shown in Lemma 6.1 still applies to show that the reduction preserves the language.

Regarding model-checking, the same model-checking algorithm as shown in Proposition 4.1 works for SDL$^+$. To treat a subformula of the form $\langle = \rangle_{\beta(y_1,\dots,y_k)}(\psi_1, \dots, \psi_k)$, we first count, for each data value d of w and $j \in \underline{k}$, the number n_j of positions of w having data d and satisfying ψ_j, and we then mark each position i with data d as satisfying $\langle = \rangle_{\beta(y_1,\dots,y_k)}(\psi_1, \dots, \psi_k)$ iff $(n'_1, \dots, n'_k) \models \beta$, where $n'_j = n_j - 1$ if position i is marked as satisfying ψ, or $n'_j = n_j$ otherwise. Observe that this still takes polynomial time. The treatment of $\langle \neq \rangle$ is similar.

On the other hand, it is easy to see that model-checking of pSDL is NP-hard, by reduction from the satisfiability problem for existential Presburger formulas. Indeed, an existential Presburger formula $\exists p_1, \dots, p_t\ \varphi(p_1, \dots, p_t)$ (where φ is a quantifier-free formula) is satisfiable iff the pSDL formula $\alpha(x, p_1, \dots, p_t)$ with t parameters $p_1, \dots, p_t$ is satisfiable, where α is defined as $(x = x) \wedge \varphi(p_1, \dots, p_t)$.

For the NP upper bound, suppose we are given a word w and a pSDL$^+$ formula φ. We first guess a function $f : \{1, \dots, |w|\} \to \mathrm{MCS}(\varphi)$, where $\mathrm{MCS}(\varphi)$ is the set of maximally consistent sets of subformulas of φ, as defined in Section 6. We verify that the guessing is consistent with the semantics of the logic:

(1) $\varphi \in f(1)$;
(2) for every $1 \le i < |w|$ we have that $f(i)$, $f(i+1)$ are one-step consistent (cf. §6);
(3) for every $1 \le i \le |w|$ we have that $\neg lab(w)[i] \notin f(i)$.

Now we can instantiate non-parametric free variables of Presburger formulas with their corresponding value:

- For every base subformula $\alpha(y, \bar{p})$ and position i, let $\gamma^i_\alpha(\bar{p})$ be the result of replacing y with $data(w)[i]$ in α.
- For every subformula $\psi := \langle = \rangle_{\beta(y_1,\dots,y_k,\bar{p})}(\psi_1, \dots, \psi_k)$ and position i, let $\gamma^i_\psi(\bar{p})$ be the result of replacing in $\beta(y_1, \dots, y_k, \bar{p})$ each y_ℓ with the number of positions $j \neq i$ of w such that $data(w)[j] = data(w)[i]$ and $\psi_\ell \in f(i)$.
- For every subformula $\psi := \langle \neq \rangle_{\beta(y_1,\dots,y_k,\bar{p})}(\psi_1, \dots, \psi_k)$ and position i, let $\gamma^i_\psi(\bar{p})$ be the result of replacing in $\beta(y_1, \dots, y_k, \bar{p})$ each y_ℓ with the number of positions $j \neq i$ of w such that $data(w)[j] \neq data(w)[i]$ and $\psi_\ell \in f(i)$.

Let Ψ be the set of all $\gamma^i_\psi(\bar{p})$ formulas (for every $1 \le i \le |w|$ and $\psi \in f(i)$ of the form above) and all the formulas $\neg\gamma^i_\psi(\bar{p})$ (for every $1 \le i \le |w|$ and $\neg\psi \in f(i)$). Observe that Ψ is of polynomial size. Finally, we check that the quantifier-free Presburger formula $\bigwedge \Psi(\bar{p})$ is satisfiable, which is in NP. ◻

8 CONCLUSIONS

In this paper, we have introduced parametric semilinear data logic (pSDL), which allows different types of arithmetic reasoning (constraints on data values, letter/length counting, data counting, and aggregation) on data words. We have provided decidability and a thorough complexity analysis of the satisfiability problem for the logic, and shown that it can express many interesting properties that cannot be expressed in existing decidable formalisms on data words, potentially leading to interesting applications (e.g., on querying log files and verification of array-manipulating programs). Our proof introduces also the automata counterpart of pSDL called parameteric semilinear data automata (pSDA), which subsume known models like Parikh automata [23], symbolic automata [14], and nondeterministic looping word automata with integer linear arithmetic [18]. We have derived decidability and complexity of emptiness for pSDA, which are of independent interests.

We would like to conclude with several open problems. Firstly, the complexity gap between 2-NEXP and NEXP of pSDL should be filled. At the moment, we can only bridge this gap when pSDL is restricted to SDL_{MNF}, which subsumes the modal logic fragment of $\text{FO}^2(<, +1, \sim)$. Similarly, the complexity gap for SDA and pSDA should be filled (e.g. between NP^{NP} and NEXP). This, in turn, raises interesting open questions on the complexity of existential Presburger Arithmetic with unary counting quantifiers and star, which (to the best of our knowledge) is not yet studied in the literature. Secondly, can we adapt pSDL to other infinite domains and other decidable theories, e.g., real linear arithmetic? The answer is far from obvious: our proof exploits heavy machinery on Presburger Arithmetic and semilinear sets, which include closure under star [21, 34] and closure under unary counting quantifiers [38], which does not hold in every decidable quantifier-free theories. Thirdly, we believe that it is highly crucial to understand further the relationships among existing models over data words, as well as array theories, with respect to their expressive power. We conjecture, among others, that our logic (or maybe a slight variant thereof) subsumes Array Folds Logic [12]. Finally, it would be interesting to investigate if the idea of using parameters could further be exploited in other logic/automata models over data words. For example, can one still extend two-variable logics [6, 39] with parameters while preserving decidability of satisfiability?

ACKNOWLEDGMENTS

We are extremely grateful to the anonymous reviewers for their valuable comments. We would also like to thank Christoph Haase, Rupak Majumdar, Alessio Mansutti, and Philipp Rümmer for productive discussions. Diego Figueira is partially supported by ANR QUID, grant ANR-18-CE40-0031. Anthony Lin was supported by the ERC Starting Grant 759969 (AV-SMP) and a Max-Planck Society Fellowship.

REFERENCES

[1] Eric Allender and Klaus W. Wagner. 1993. Counting Hierarchies: Polynomial Time and Constant Depth Circuits. In *Current Trends in Theoretical Computer Science - Essays and Tutorials*, Grzegorz Rozenberg and Arto Salomaa (Eds.). World Scientific Series in Computer Science, Vol. 40. World Scientific, 469–483. https://doi.org/10.1142/9789812794499_0035

[2] Rajeev Alur and Pavol Cerný. 2011. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *Annual Symposium on Principles of Programming Languages (POPL)*, Thomas Ball and Mooly Sagiv (Eds.). ACM Press, 599–610. https://doi.org/10.1145/1926385.1926454

[3] Rajeev Alur and Pavol Cerný. 2011. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *Annual Symposium on Principles of Programming Languages (POPL)*, Thomas Ball and Mooly Sagiv (Eds.). ACM, 599–610. https://doi.org/10.1145/1926385.1926454

[4] Michael Benedikt, Clemens Ley, and Gabriele Puppis. 2010. What You Must Remember When Processing Data Words. In *Proceedings of the Alberto Mendelzon International Workshop on Foundations of Data Management (AMW) (CEUR Workshop Proceedings, Vol. 619)*, Alberto H. F. Laender and Laks V. S. Lakshmanan (Eds.). CEUR-WS.org. http://ceur-ws.org/Vol-619/paper11.pdf

[5] Henrik Björklund and Thomas Schwentick. 2010. On notions of regularity for data languages. *Theoretical Computer Science* 411, 4-5 (2010), 702–715. https://doi.org/10.1016/j.tcs.2009.10.009

[6] Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. 2011. Two-variable logic on data words. *ACM Transactions on Computational Logic* 12, 4 (2011), 27:1–27:26. https://doi.org/10.1145/1970398.1970403

[7] Mikołaj Bojańczyk, Bartek Klin, and Sławomir Lasota. 2014. Automata theory in nominal sets. *Logical Methods in Computer Science (LMCS)* 10, 3 (2014). https://doi.org/10.2168/LMCS-10(3:4)2014

[8] Mikolaj Bojańczyk and Rafal Stefanski. 2020. Single-Use Automata and Transducers for Infinite Alphabets. In *International Colloquium on Automata, Languages and Programming (ICALP) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 168)*, Artur Czumaj, Anuj Dawar, and Emanuela Merelli (Eds.). Leibniz-Zentrum für Informatik, 113:1–113:14. https://doi.org/10.4230/LIPIcs.ICALP.2020.113

[9] Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. 2006. What's Decidable About Arrays?. In *International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI) (LNCS, Vol. 3855)*, E. Allen Emerson and Kedar S. Namjoshi (Eds.). Springer, 427–442. https://doi.org/10.1007/11609773_28

[10] Yu-Fang Chen, Ondrej Lengál, Tony Tan, and Zhilin Wu. 2017. Register automata with linear arithmetic. In *Annual Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society Press, 1–12. https://doi.org/10.1109/LICS.2017.8005111

[11] Wojciech Czerwinski and Lukasz Orlikowski. 2021. Reachability in Vector Addition Systems is Ackermann-complete. *CoRR* abs/2104.13866 (2021). arXiv:2104.13866 https://arxiv.org/abs/2104.13866

[12] Przemyslaw Daca, Thomas A. Henzinger, and Andrey Kupriyanov. 2016. Array Folds Logic. In *International Conference on Computer Aided Verification (CAV)*. 230–248.

[13] Loris D'Antoni, Tiago Ferreira, Matteo Sammartino, and Alexandra Silva. 2019. Symbolic Register Automata. In *International Conference on Computer Aided Verification (CAV) (LNCS, Vol. 11561)*, Isil Dillig and Serdar Tasiran (Eds.). Springer, 3–21. https://doi.org/10.1007/978-3-030-25540-4_1

[14] Loris D'Antoni and Margus Veanes. 2017. The Power of Symbolic Automata and Transducers. In *International Conference on Computer Aided Verification (CAV)*. 47–67. https://doi.org/10.1007/978-3-319-63387-9_3

[15] Stéphane Demri and Ranko Lazić. 2009. LTL with the freeze quantifier and register automata. *ACM Transactions on Computational Logic* 10, 3 (2009), 16:1–16:30. https://doi.org/10.1145/1507244.1507246

[16] Stéphane Demri and Ranko Lazic. 2009. LTL with the freeze quantifier and register automata. *ACM Transactions on Computational Logic* 10, 3 (2009), 16:1–16:30. https://doi.org/10.1145/1507244.1507246

[17] Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. 2002. First-Order Logic with Two Variables and Unary Temporal Logic. *Inf. Comput.* 179, 2 (2002), 279–295. https://doi.org/10.1006/inco.2001.2953

[18] Rachel Faran and Orna Kupferman. 2020. On Synthesis of Specifications with Arithmetic. In *International Conference on Current Trends in Theory and Practice of Informatics (SOFSEM) (LNCS, Vol. 12011)*, Alexander Chatzigeorgiou, Riccardo Dondi, Herodotos Herodotou, Christos A. Kapoutsis, Yannis Manolopoulos, George A. Papadopoulos, and Florian Sikora (Eds.). Springer, 161–173. https://doi.org/10.1007/978-3-030-38919-2_14

[19] Seymour Ginsburg and Edwin H. Spanier. 1966. Semigroups, Presburger formulas, and languages. *Pacific J. Math.* 16, 2 (1966), 285–296.

[20] Christoph Haase. 2018. A survival guide to presburger arithmetic. *ACM SIGLOG News* 5, 3 (2018), 67–82. https://dl.acm.org/citation.cfm?id=3242964

[21] Christoph Haase and Georg Zetzsche. 2019. Presburger arithmetic with stars, rational subsets of graph groups, and nested zero tests. In *Annual Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society Press, 1–14. https://doi.org/10.1109/LICS.2019.8785850

[22] Michael Kaminski and Nissim Francez. 1994. Finite-Memory Automata. *Theoretical Computer Science* 134, 2 (1994), 329–363. https://doi.org/10.1016/0304-3975(94)90242-9

[23] Felix Klaedtke and Harald Rueß. 2003. Monadic Second-Order Logics with Cardinalities. In *International Colloquium on Automata, Languages and Programming (ICALP) (Lecture Notes in Computer Science, Vol. 2719)*. Springer, 681–696. https://doi.org/10.1007/3-540-45061-0_54

[24] S. Rao Kosaraju. 1982. Decidability of Reachability in Vector Addition Systems (Preliminary Version). In *Symposium on Theory of Computing (STOC)*, Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber (Eds.). ACM, 267–281. https://doi.org/10.1145/800070.802201

[25] Dexter C. Kozen. 2006. *Theory of Computation*. Springer.

[26] Daniel Kroening and Ofer Strichman. 2008. *Decision Procedures*. Springer.

[27] Jean-Luc Lambert. 1992. A Structure to Decide Reachability in Petri Nets. *Theor. Comput. Sci.* 99, 1 (1992), 79–104. https://doi.org/10.1016/0304-3975(92)90173-D

[28] Jérôme Leroux. 2021. The Reachability Problem for Petri Nets is Not Primitive Recursive. *CoRR* abs/2104.12695 (2021). arXiv:2104.12695 https://arxiv.org/abs/2104.12695

[29] Ernst W. Mayr. 1981. An Algorithm for the General Petri Net Reachability Problem. In *Symposium on Theory of Computing (STOC)*. ACM, 238–246. https://doi.org/10.1145/800076.802477

[30] Ernst W. Mayr. 1984. An Algorithm for the General Petri Net Reachability Problem. *SIAM J. Comput.* 13, 3 (1984), 441–460. https://doi.org/10.1137/0213029

[31] Frank Neven, Thomas Schwentick, and Victor Vianu. 2004. Finite state machines for strings over infinite alphabets. *ACM Transactions on Computational Logic* 5, 3 (2004), 403–435. https://doi.org/10.1145/1013560.1013562

[32] Rohit Parikh. 1966. On Context-Free Languages. *J. ACM* 13, 4 (1966), 570–581. https://doi.org/10.1145/321356.321364

[33] Rohit Parikh. 1966. On Context-Free Languages. *J. ACM* 13, 4 (1966), 570–581. https://doi.org/10.1145/321356.321364

[34] Ruzica Piskac and Viktor Kunčak. 2008. Linear Arithmetic with Stars. In *International Conference on Computer Aided Verification (CAV) (Lecture Notes in Computer Science, Vol. 5123)*. Springer, 268–280. https://doi.org/10.1007/978-3-540-70545-1_25

[35] Mojżesz Presburger and Dale Jabcquette. 1991. On the completeness of a certain system of arithmetic of whole numbers in which addition occurs as the only operation. *History and Philosophy of Logic* 12, 2 (1991), 225–233.

[36] Bruno Scarpellini. 1984. Complexity of subcases of Presburger arithmetic. *Trans. Amer. Math. Soc.* 284, 1 (1984), 203–218.

[37] Marcus Schaefer and Christopher Umans. 2002. Completeness in the polynomial-time hierarchy: A compendium. *SIGACT News* (2002).

[38] Nicole Schweikardt. 2005. Arithmetic, first-order logic, and counting quantifiers. *ACM Transactions on Computational Logic* 6, 3 (2005), 634–671. https://doi.org/10.1145/1071596.1071602

[39] Thomas Schwentick and Thomas Zeume. 2012. Two-Variable Logic with Two Order Relations. *Logical Methods in Computer Science (LMCS)* 8, 1 (2012). https://doi.org/10.2168/LMCS-8(1:15)2012

[40] Seinosuke Toda. 1991. PP is as Hard as the Polynomial-Time Hierarchy. *SIAM Journal on computing* 20, 5 (1991), 865–877. https://doi.org/10.1137/0220053

[41] Jacobo Torán. 1991. Complexity Classes Defined by Counting Quantifiers. *J. ACM* 38, 3 (1991), 753–774. https://doi.org/10.1145/116825.116858

[42] Peter van Emde Boas. 1997. The convenience of tilings. *Complexity, Logic, and Recursion Theory* (1997), 331–363.

[43] Kumar Neeraj Verma, Helmut Seidl, and Thomas Schwentick. 2005. On the Complexity of Equational Horn Clauses. In *International Conference on Automated Deduction (CADE) (Lecture Notes in Computer Science, Vol. 3632)*. Springer, 337–352. https://doi.org/10.1007/11532231_25

[44] Klaus W. Wagner. 1987. More Complicated Questions About Maxima and Minima, and Some Closures of NP. *Theoretical Computer Science* 51 (1987), 53–80. https://doi.org/10.1016/0304-3975(87)90049-1

[45] Kevin Woods. 2015. Presburger Arithmetic, Rational Generating Functions, and quasi-polynomials. *Journal of Symbolic Logic* 80, 2 (2015), 433–449. https://doi.org/10.1017/jsl.2015.4