Reasoning about Data Repetitions with Counter Systems

Stéphane Demri
NYU & LSV, CNRS

Diego Figueira
University of Edinburgh

M. Praveen
LaBRI, Univ. Bordeaux, France

***Abstract*—We study linear-time temporal logics interpreted over data words with multiple attributes. We restrict the atomic formulas to equalities of attribute values in successive positions and to repetitions of attribute values in the future or past. We demonstrate correspondences between satisfiability problems for logics and reachability-like decision problems for counter systems. We show that allowing/disallowing atomic formulas expressing repetitions of values in the past corresponds to the reachability/coverability problem in Petri nets. This gives us 2EXPSPACE upper bounds for several satisfiability problems. We prove matching lower bounds by reduction from a reachability problem for a newly introduced class of counter systems. This new class is a succinct version of vector addition systems with states in which counters are accessed via pointers, a potentially useful feature in other contexts. We strengthen further the correspondences between data logics and counter systems by characterizing the complexity of fragments, extensions and variants of the logic. For instance, we precisely characterize the relationship between the number of attributes allowed in the logic and the number of counters needed in the counter system.**

I. Introduction

Words with multiple data: Finite data words [4] are ubiquitous structures that include timed words, runs of counter automata or runs of concurrent programs with an unbounded number of processes. These are finite words in which every position carries a label from a finite alphabet and a data value from some infinite alphabet. More generally, structures over an infinite alphabet provide an adequate abstraction for objects from several domains: for example, infinite runs of counter automata can be viewed as infinite data words, finite data trees model XML documents with attribute values [9] and so on. A wealth of specification formalisms for data words (or slight variants) has been introduced stemming from automata, see e.g. [26], [29], to adequate logical languages such as first-order logic [1], [5] or temporal logics [23], [19], [9], [17], [10] (see also a related formalism in [12]). Depending on the type of structures, other formalisms have been considered such as XPath [9] or monadic second-order logic [3]. In full generality, most formalisms lead to undecidable decision problems and a well-known research trend consists of finding a good trade-off between expressiveness and decidability. Restrictions to regain decidability are protean: bounding the models (from trees to words for instance), restricting the number of variables, see e.g. [1], limiting the set of the temporal operators or the use of the data manipulating operator, see e.g. [11], [6]. Moreover, interesting and surprising results have been exhibited about relationships between logics for data words and counter automata [1], [7], [2], leading to a first classification of automata on data words [2]. This is why logics for data words are not only interesting for their own sake but also for their deep relationships with data automata or with counter automata. Herein, we pursue further this line of work and we work with words in which every position contains a *vector* of data values.

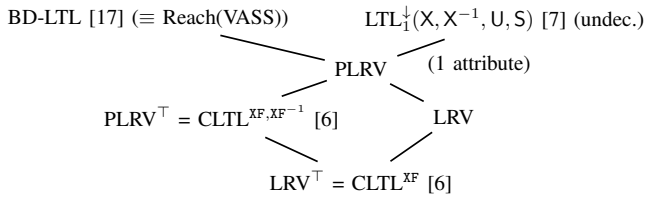
Work supported by project REACHARD ANR-11-BS02-001, and FET-Open Project FoX, grant agreement 233599. M. Praveen was supported by the ERCIM "Alain Bensoussan" Fellowship Programme.

Motivations: In [6], a decidable logic interpreted over (finite or infinite) sequences of variable valuations (understood as words with multiple data) is introduced in which the atomic formulae are of the form either $\mathtt{x} \approx \mathsf{X}^i\mathtt{y}$ or $\mathtt{x} \approx \langle\top?\rangle\mathtt{y}$. The formula $\mathtt{x} \approx \mathsf{X}^i\mathtt{y}$ states that the current value of variable $\mathtt{x}$ is the same as the value of $\mathtt{y}$ i steps ahead (local constraint) whereas $\mathtt{x} \approx \langle\top?\rangle\mathtt{y}$ states that the current value of $\mathtt{x}$ is repeated in a future value of $\mathtt{y}$ (future obligation). Such atomic properties can be naturally expressed with a *freeze* operator that stores a data value for later comparison, and in [6], it is shown that the satisfiability problem is decidable with the temporal operators in $\{\mathsf{X}, \mathsf{X}^{-1}, \mathsf{U}, \mathsf{S}\}$. The freeze operator allows to store a data value in a register and then to test later equality between the value in the register and a data value at some other position. This is a powerful mechanism but the logic in [6] uses it in a limited way: only repetitions of data values can be expressed and it restricts very naturally the use of the freeze operator. The decidability result is robust since, for instance, it holds with the addition of atomic formulas of the form $\mathtt{x} \approx \langle\top?\rangle^{-1}\mathtt{y}$ stating that the current value of $\mathtt{x}$ is repeated in a past value of $\mathtt{y}$ (past obligation). Decidability can be shown either by reduction into $\mathrm{FO}^2(\sim, <, +\omega)$, a first-order logic over data words introduced in [1] or by reduction into the verification of fairness properties in Petri nets, shown decidable in [16]. In both cases, an essential use of the decidability of the reachability problem for Petri nets is made, for which no primitive recursive algorithm is known, see e.g. [21]. Hence, even though the logics shown decidable in [6] poorly use the freeze operator (the only properties about data are related to controlled repetitions), the complexity of their satisfiability problems is unknown. Moreover, it is unclear whether the reductions into the reachability problem for Petri nets are really needed; this would be the case if reductions in the other direction exist. Note that in [17], a richer logic has been introduced and it has been shown that satisfiability

is equivalent to the reachability problem for Petri nets.

Our main motivation is to investigate logics that express repetitions of values, revealing the correspondence between expressivity of the logic and reachability problems for counter machines, including well-known problems for Petri nets. This work can be seen as a study of the precision with which counting needs to be done as a consequence of having a mechanism for demanding "*the current data value is repeated in the future/past*" in a logic. Hence, this is not the study of yet another logic, but of a natural feature shared by most studied logics on data words [6], [1], [7], [11], [17], [9], [10]: the property of demanding that a data value be repeated. We consider different ways in which one can demand the repetition of a value, and study the repercussion in terms of the "precision" with which we need to count in order to solve the satisfiability problem. Our measurement of precision here distinguishes the reachability versus the coverability problem for Petri nets and the number of counters needed as a function of the number of variables used in the logic.

Our contribution: We introduce the logic LRV ("Logic of Repeating Values") interpreted over finite words with multiple data, equipped with atomic formulas of the form either $\mathtt{x} \approx \mathsf{X}^i\mathtt{y}$ or $\mathtt{x} \approx \langle\phi?\rangle\mathtt{y}$ [resp. $\mathtt{x} \not\approx \langle\phi?\rangle\mathtt{y}$], where $\mathtt{x} \approx \langle\phi?\rangle\mathtt{y}$ [resp. $\mathtt{x} \not\approx \langle\phi?\rangle\mathtt{y}$] states that the current value of $\mathtt{x}$ is repeated [resp. is not repeated] in some future value of $\mathtt{y}$ in a position where ϕ holds true. When we impose $\phi = \top$, the logic introduced in [6] is obtained and it is denoted by $\text{LRV}^\top$. The syntax for future obligations is freely inspired from PDL with its test operator '?'. Even though LRV contains the past-time temporal operators X^{-1} and S, it has no past obligations. We write PLRV to denote the extension of LRV with past obligations of the form $\mathtt{x} \approx \langle\phi?\rangle^{-1}\mathtt{y}$ or $\mathtt{x} \not\approx \langle\phi?\rangle^{-1}\mathtt{y}$. Below, we illustrate how LRV and variants are compared to existing data logics.



Our main results are listed below.

1. We begin where [6] stopped: the reachability problem for Petri nets is reduced to the satisfiability problem of PLRV (i.e., the logic with past obligations).
2. Without past obligations, the satisfiability problem is much easier: we reduce the satisfiability problem of $\text{LRV}^\top$ and LRV to the control-state reachability problem for VASS, via a detour to a reachability problem on gainy VASS. But the number of counters in the VASS is exponential in the number of variables used in the formula. This gives us a 2EXPSPACE upper bound.
3. The exponential blow up mentioned above is unavoidable: we show a polynomial-time reduction in the converse direction, starting from a linear sized counter machine (without zero tests) that can access exponentially many counters. This gives us a matching 2EXPSPACE lower bound.
4. Several augmentations to the logic do not alter the complexity: we show that complexity is preserved when MSO-definable temporal operators are added or when infinite words with multiple data are considered.
5. The power of nested testing formulas: we show that the complexity of the satisfiability problem for $\text{LRV}^\top$ reduces to PSPACE-complete when the number of variables in the logic is bounded by a constant, while the complexity of the satisfiability of LRV does not reduce even when only one variable is allowed. Recall that the difference between $\text{LRV}^\top$ and LRV is that the later allows any ϕ in $\mathtt{x} \approx \langle\phi?\rangle\mathtt{y}$ while the former restricts ϕ to just $\top$.
6. The power of pairs of repeating values: we show that the satisfiability problem of $\text{LRV}^\top$ augmented with $\langle\mathtt{x},\mathtt{y}\rangle \approx \langle\top?\rangle\langle\mathtt{x}',\mathtt{y}'\rangle$ (repetitions of pairs of data values) is undecidable, even when $\langle\mathtt{x},\mathtt{y}\rangle \approx \langle\top?\rangle^{-1}\langle\mathtt{x}',\mathtt{y}'\rangle$ is not allowed (i.e., even when past obligations are not allowed).
7. Implications for classical logics: we show a 3EXPSPACE upper bound for the satisfiability problem for *forward*-$\text{EMSO}^2(+1,<,\sim)$ over data words, using results on LRV.

For proving the result mentioned in point 3 above, we introduce a new class of counter machines that we call *chain systems* and show a key hardness result for them. This class is interesting for its own sake and could be used in situations where the power of binary encoding needs to be used. We prove the $(k+1)$EXPSPACE-completeness of the control state reachability problem for chain systems of level k (we only use $k = 1$ in this paper but the proof for arbitrary k is no more complex than the proof for the particular case of $k = 1$). In chain systems, the number of counters is equal to an exponential tower of height k but we cannot access the counters directly in the transitions. Instead, we have a pointer that we can move along a chain of counters. The $(k+1)$EXPSPACE lower bound is obtained by a non-trivial extension of the EXPSPACE-hardness result from [22], [8]. Then we show that the control state reachability problem for the class of chain systems with $k = 1$ can be reduced to the satisfiability problem for LRV (see Section V). It was known that data logics are strongly related to classes of counter automata, see e.g. [1], [7], [2] but herein, we show how varying the expressive power of logics leads to correspondence with different reachability problems for counter machines.

II. Preliminaries

We write $\mathbb{N}$ [resp. $\mathbb{Z}$] to denote the set of non-negative integers [resp. integers] and $[i,j]$ to denote $\{k \in \mathbb{Z} : i \leq k \text{ and } k \leq j\}$. For $\mathbf{v} \in \mathbb{Z}^n$, $\mathbf{v}(i)$ denotes the i^{th} element of $\mathbf{v}$ for every $i \in [1,n]$. We write $\mathbf{v} \preceq \mathbf{v}'$ whenever for every $i \in [1,n]$, we have $\mathbf{v}(i) \leq \mathbf{v}'(i)$. For a (possibly infinite) alphabet Σ, Σ^* represents the set of finite words over Σ, Σ^+ the set of finite non-empty words over Σ. For a finite word or sequence $u = a_1 \dots a_k$ over Σ, we write $|u|$ to denote its *length* k. For $0 \leq i < |u|$, $u(i)$ represents the $(i+1)$-th letter of the word, here a_{i+1}.

Logics of Repeating Values: Let $\mathrm{VAR} = \{\mathtt{x}_1, \mathtt{x}_2, \ldots\}$ be a countably infinite set of *variables*. We denote by LRV the logic whose formulas are defined as follows, where $\mathtt{x}, \mathtt{y} \in \mathrm{VAR}$, $i \in \mathbb{N}$.

$$\phi ::= \mathtt{x} \approx \mathsf{X}^i\mathtt{y} \mid \mathtt{x} \approx \langle\phi?\rangle\mathtt{y} \mid \mathtt{x} \not\approx \langle\phi?\rangle\mathtt{y} \mid \phi \wedge \phi \mid \neg\phi \mid \mathsf{X}\phi \mid \phi\mathsf{U}\phi \mid \mathsf{X}^{-1}\phi \mid \phi\mathsf{S}\phi$$

A *valuation* is a map from VAR to $\mathbb{N}$, and a *model* is a finite non-empty sequence σ of valuations. For every model σ and $0 \leq i < |\sigma|$, the satisfaction relation $\models$ is defined:

- $\sigma, i \models \mathtt{x} \approx \mathsf{X}^j\mathtt{y}$ iff $i+j < |\sigma|$ and $\sigma(i)(\mathtt{x}) = \sigma(i+j)(\mathtt{y})$,
- $\sigma, i \models \mathtt{x} \approx \langle\phi?\rangle\mathtt{y}$ iff there exists j such that $i < j < |\sigma|$, $\sigma(i)(\mathtt{x}) = \sigma(j)(\mathtt{y})$ and $\sigma, j \models \phi$.

The semantics $\mathtt{x} \not\approx \langle\phi?\rangle\mathtt{y}$ is defined similarly but asking for a different data value. The temporal operators next (X), previous (X^{-1}), until (U) and since (S) and Boolean connectives are defined in the usual way. We also use the standard derived temporal operators (G, F, F^{-1}, ...) and constants $\top$, $\bot$. We write $\sigma \models \phi$ if $\sigma, 0 \models \phi$. Given a set of temporal operators $\mathcal{O}$, we write LRV($\mathcal{O}$) to denote the fragment of LRV restricted to formulas with operators from $\mathcal{O}$. The *satisfiability problem* for LRV (written SAT(LRV)) is to check for an LRV formula ϕ, whether there is σ such that $\sigma \models \phi$. Let PLRV be the extension of LRV with additional atomic formulas of the form $\mathtt{x} \approx \langle\phi?\rangle^{-1}\mathtt{y}$ and $\mathtt{x} \not\approx \langle\phi?\rangle^{-1}\mathtt{y}$. The satisfaction relation is extended as expected: $\sigma, i \models \mathtt{x} \approx \langle\phi?\rangle^{-1}\mathtt{y}$ iff there is $0 \leq j < i$ such that $\sigma(i)(\mathtt{x}) = \sigma(j)(\mathtt{y})$ and $\sigma, j \models \phi$, and similarly for $\mathtt{x} \not\approx \langle\phi?\rangle^{-1}\mathtt{y}$. We write LRV$^\top$ [resp. PLRV$^\top$] to denote the fragment of LRV [resp. PLRV] in which atomic formulas are restricted to $\mathtt{x} \approx \mathsf{X}^i\mathtt{y}$ and $\mathtt{x} \approx \langle\top?\rangle\mathtt{y}$ [resp. to $\mathtt{x} \approx \mathsf{X}^i\mathtt{y}$, $\mathtt{x} \approx \langle\top?\rangle\mathtt{y}$ and $\mathtt{x} \approx \langle\top?\rangle^{-1}\mathtt{y}$]. In [6], LRV$^\top$ and PLRV$^\top$ were shown to be decidable by reduction into the reachability problem for Petri nets. However, the characterization of their complexity remained open.

Properties: In the table below, we justify our choices for atomic formulae by presenting several abbreviations (with their obvious semantics). By contrast, we include in LRV both $\mathtt{x} \approx \langle\phi?\rangle\mathtt{y}$ and $\mathtt{x} \not\approx \langle\phi?\rangle\mathtt{y}$ when ϕ is an arbitrary formula since there is no obvious way to express one with the other.

Abbreviation	Definition
$\mathtt{x} \not\approx \mathsf{X}^i\mathtt{y}$	$\neg(\mathtt{x} \approx \mathsf{X}^i\mathtt{y}) \wedge \overbrace{\mathsf{X}\cdots\mathsf{X}}^{i \text{ times}}\top$
$\mathtt{x} \approx \mathsf{X}^{-i}\mathtt{y}$	$\overbrace{\mathsf{X}^{-1}\cdots\mathsf{X}^{-1}}^{i \text{ times}}(\mathsf{X}^i\mathtt{x} \approx \mathtt{y})$
$\mathtt{x} \not\approx \mathsf{X}^{-i}\mathtt{y}$	$\neg(\mathtt{x} \approx \mathsf{X}^{-i}\mathtt{y}) \wedge \overbrace{\mathsf{X}^{-1}\cdots\mathsf{X}^{-1}}^{i \text{ times}}\top$
$\mathtt{x} \not\approx \langle\top?\rangle\mathtt{y}$	$(\mathtt{x} \not\approx \mathsf{X}\mathtt{y}) \vee \mathsf{X}((\mathtt{y} \approx \mathsf{X}\mathtt{y})\mathsf{U}(\mathtt{y} \not\approx \mathsf{X}\mathtt{y}))$
$\mathtt{x} \not\approx \langle\top?\rangle^{-1}\mathtt{y}$	$(\mathtt{x} \not\approx \mathsf{X}^{-1}\mathtt{y}) \vee \mathsf{X}^{-1}((\mathtt{y} \approx \mathsf{X}^{-1}\mathtt{y})\mathsf{S}(\mathtt{y} \not\approx \mathsf{X}^{-1}\mathtt{y}))$

Models for LRV can be viewed as finite data words in $(\Sigma \times \mathbb{D})^*$, where Σ is a finite alphabet and $\mathbb{D}$ is an infinite domain. E.g., equalities between dedicated variables can simulate that a position is labelled by a letter from Σ; moreover, we may assume that the data values are encoded with the variable $\mathtt{x}$. Let us express that whenever there are $i < j$ s.t. i and j [resp. $i+1$ and $j+1$, $i+2$ and $j+2$] are labelled by a [resp. a', a''], $\sigma(i+1)(\mathtt{x}) \neq \sigma(j+1)(\mathtt{x})$. This can be stated in LRV by: $\mathsf{G}(a \wedge \mathsf{X}(a' \wedge \mathsf{X}a'') \Rightarrow \mathsf{X}\neg(\mathtt{x} \approx \langle\mathsf{X}^{-1}a \wedge a' \wedge \mathsf{X}a''?\rangle\mathtt{x}))$. This is an example of key constraints, see e.g. [27, Definition 2.1] and the paper contains also numerous examples of properties that can be captured by LRV.

Basics on VASS: A *vector addition system with states* (VASS) is a tuple $\mathcal{A} = \langle Q, C, \delta\rangle$ where Q is a finite set of *control states*, C is a finite set of *counters* and δ is a finite set of *transitions* in $Q \times \mathbb{Z}^C \times Q$. A *configuration* of $\mathcal{A}$ is a pair $\langle q, \mathbf{v}\rangle \in Q \times \mathbb{N}^C$. We write $\langle q, \mathbf{v}\rangle \to \langle q', \mathbf{v}'\rangle$ if there is $(q, \mathbf{u}, q') \in \delta$ such that $\mathbf{v}' = \mathbf{v} + \mathbf{u}$. The *reachability problem* for VASS (written Reach(VASS)) consists in checking whether $\langle q_0, \mathbf{v_0}\rangle \xrightarrow{*} \langle q_f, \mathbf{v_f}\rangle$, given two configurations $\langle q_0, \mathbf{v_0}\rangle$ and $\langle q_f, \mathbf{v_f}\rangle$. The reachability problem for VASS is decidable with non-primitive recursive complexity algorithms [25], [18], [21], but the best known lower bound is only EXPSPACE [22], [8]. The *control state reachability problem* is the following EXPSPACE-complete problem [22], [28]: given a configuration $\langle q_0, \mathbf{v_0}\rangle$ and a control state q_f, check whether $\langle q_0, \mathbf{v_0}\rangle \xrightarrow{*} \langle q_f, \mathbf{v}\rangle$ for some $\mathbf{v} \in \mathbb{N}^C$. We also use two other kinds of computations: with gains ($\to_{\text{gainy}}$) or losses ($\to_{\text{lossy}}$). We write $\langle q, \mathbf{v}\rangle \to_{\text{gainy}} \langle q', \mathbf{v}'\rangle$ [resp. $\langle q, \mathbf{v}\rangle \to_{\text{lossy}} \langle q', \mathbf{v}'\rangle$] if there is a transition $(q, \mathbf{u}, q') \in \delta$ and $\mathbf{w} \in \mathbb{N}^C$ such that $\mathbf{v} \preceq \mathbf{w}$ and $\mathbf{w} + \mathbf{u} \preceq \mathbf{v}'$ [resp. $\mathbf{w} \preceq \mathbf{v}$ and $\mathbf{v}' \preceq \mathbf{w} + \mathbf{u}$].

III. The Power of Past: From Reach(VASS) to SAT(PLRV)

While [6] concentrated on decidability results, here we begin with a hardness result. When past obligations are allowed as in PLRV, SAT(PLRV) is equivalent to the very difficult problem of reachability in VASS. Combined with the result of the next section where we prove that removing past obligations leads to a reduction into the control state reachability problem for VASS, this means that reasoning with past obligations is much more complicated.

Theorem 1. There is a polynomial-space reduction from Reach(VASS) into SAT(PLRV).

The proof of Theorem 1 is analogous to the proof of [1, Theorem 16] except that properties are expressed in PLRV instead of being expressed in $\mathrm{FO}^2(\sim, <, +1)$. In Section IV-A, we show how to eliminate test formulas ϕ from $\mathtt{x} \approx \langle\phi?\rangle\mathtt{y}$ and $\mathtt{x} \approx \langle\phi?\rangle^{-1}\mathtt{y}$. Combining this with the decidability proof for PLRV$^\top$ satisfiability from [6], we get that SAT(PLRV) [resp. for SAT(PLRV$^\top$)] is equivalent to Reach(VASS) modulo polynomial-space reductions.

IV. Leaving the Past Behind Simplifies Things: From SAT(LRV) to Control State Reachability

In this section, we show the reduction from SAT(LRV) to the control state reachability problem in VASS. We obtain a 2EXPSPACE upper bound for SAT(LRV) as a consequence. This is done in two steps: (1) simplifying formulas of the form $\mathtt{x} \approx \langle\phi?\rangle\mathtt{y}$ to remove the test formula ϕ (i.e., a reduction from SAT(LRV) into SAT(LRV$^\top$)); and (2) reducing from SAT(LRV$^\top$) into the control state reachability pb. in VASS.

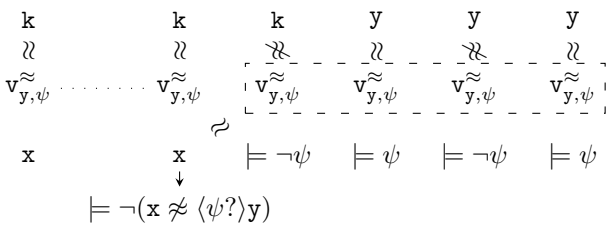
A. Elimination of Test Formulas

We give a polynomial-time algorithm that for every $\varphi \in$ LRV computes a formula $\varphi' \in \text{LRV}^\top$ that preserves satisfiability: there is a model σ s.t. $\sigma \models \varphi$ iff there is a model σ' s.t. $\sigma' \models \varphi'$. We give the reduction in two steps. First, we eliminate formulas with inequality tests of the form $\mathtt{x} \not\approx \langle\varphi?\rangle\mathtt{y}$ using only positive tests of the form $\mathtt{x} \approx \langle\varphi?\rangle\mathtt{y}$. We then eliminate formulas of the form $\mathtt{x} \approx \langle\varphi?\rangle\mathtt{y}$, using only formulas of the form $\mathtt{x} \approx \langle\top?\rangle\mathtt{y}$. Let $\text{LRV}^{\approx}$ be the logic LRV where there are no appearances of formulas of the form $\mathtt{x} \not\approx \langle\varphi?\rangle\mathtt{y}$; and let $\text{PLRV}^{\approx}$ be PLRV without $\mathtt{x} \not\approx \langle\varphi?\rangle\mathtt{y}$ or $\mathtt{x} \not\approx \langle\varphi?\rangle^{-1}\mathtt{y}$.

Proposition 2 (from LRV to $\text{LRV}^{\approx}$)**.** There is a polynomial-time reduction from SAT(LRV) into SAT($\text{LRV}^{\approx}$); and from SAT(PLRV) into SAT($\text{PLRV}^{\approx}$).

Proof sketch: For every $\varphi \in$ LRV, we compute $\varphi' \in \text{LRV}^{\approx}$ in polynomial time, which preserves satisfiability. Besides all the variables from φ, φ' uses a distinguished variable $\mathtt{k}$ (which will have a constant value, different from all the values of the variables of φ) and variables $\mathtt{v}^{\approx}_{\mathtt{y},\psi}$, $\mathtt{v}_{\mathtt{x}\not\approx\langle\psi?\rangle\mathtt{y}}$ for every subformula ψ of φ and variables $\mathtt{x}, \mathtt{y}$ of φ.

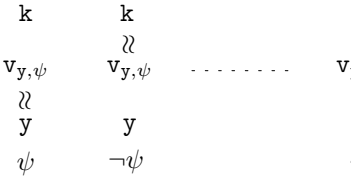
Each subformula $\mathtt{x} \not\approx \langle\psi?\rangle\mathtt{y}$ is replaced by $\mathtt{x} \not\approx \mathtt{v}_{\mathtt{x}\not\approx\langle\psi?\rangle\mathtt{y}} \wedge \mathtt{v}_{\mathtt{x}\not\approx\langle\psi?\rangle\mathtt{y}} \approx \langle\psi?\rangle\mathtt{y}$, where $\mathtt{v}_{\mathtt{x}\not\approx\langle\psi?\rangle\mathtt{y}}$ is a fresh variable. On the other hand, for each subformula $\neg(\mathtt{x} \not\approx \langle\psi?\rangle\mathtt{y})$, we use the variable $\mathtt{v}^{\approx}_{\mathtt{y},\psi}$ in conjunction with $\mathtt{k}$ as shown below.



In the beginning, $\mathtt{v}^{\approx}_{\mathtt{y},\psi} \approx \mathtt{k}$, which is broken at the first position where all future positions where ψ holds have the same value for $\mathtt{y}$. At this position (say $i+1$), $\mathtt{v}^{\approx}_{\mathtt{y},\psi} \not\approx \mathtt{k}$ and $\mathtt{v}^{\approx}_{\mathtt{y},\psi}$ maintains the same value (say n) at all future positions, illustrated as a dashed box above. In all positions after i that satisfy ψ, $\mathtt{y} \approx \mathtt{v}^{\approx}_{\mathtt{y},\psi}$. These conditions can be enforced without using $\neg(\mathtt{x} \not\approx \langle\psi?\rangle\mathtt{y})$. Suppose $\mathtt{x}$ also has value n at position i as shown above. Now in any position after i that satisfies ψ, $\mathtt{y}$ has the value n, which is the value of $\mathtt{x}$ in position i. This is exactly the semantics of $\neg(\mathtt{x} \not\approx \langle\psi?\rangle\mathtt{y})$. Hence, $\neg(\mathtt{x} \not\approx \langle\psi?\rangle\mathtt{y})$ can be replaced by $\neg\mathsf{X}\mathsf{F}\psi \vee \mathtt{x} \approx \mathsf{X}\mathtt{v}^{\approx}_{\mathtt{y},\psi}$. Past obligations are treated in a symmetrical way. ∎

Proposition 3 (from $\text{LRV}^{\approx}$ to $\text{LRV}^\top$)**.** There is a polynomial-time reduction from SAT($\text{LRV}^{\approx}$) into SAT($\text{LRV}^\top$); and from SAT($\text{PLRV}^{\approx}$) into SAT($\text{PLRV}^\top$).

Proof sketch: For every $\varphi \in \text{LRV}^{\approx}$, we compute in polynomial time $\varphi' \in \text{LRV}^\top$ that preserves satisfiability. Besides all the variables from φ, φ' uses a new distinguished variable $\mathtt{k}$, and a variable $\mathtt{v}_{\mathtt{y},\psi}$ for every subformula ψ of φ and every variable $\mathtt{y}$ of φ. We enforce $\mathtt{k}$ to have a constant value different from all values of variables of φ. At every position, we enforce ψ to hold if $\mathtt{v}_{\mathtt{y},\psi} \approx \mathtt{y}$, and ψ not to hold if $\mathtt{v}_{\mathtt{y},\psi} \approx \mathtt{k}$ as shown above. Then $\mathtt{x} \approx \langle\psi?\rangle\mathtt{y}$ is replaced by $\mathtt{x} \approx \langle\top?\rangle\mathtt{v}_{\mathtt{y},\psi}$. ∎



Corollary 4. There is a polynomial-time reduction from SAT(LRV) into SAT($\text{LRV}^\top$).

Since SAT($\text{PLRV}^\top$) is decidable [6], we obtain the decidability of SAT(PLRV).

Corollary 5. SAT(PLRV) is decidable.

B. From $\text{LRV}^\top$ Satisfiability to Control State Reachability

In [6], SAT($\text{LRV}^\top$) is reduced to the reachability problem for a subclass of VASS. Herein, this is refined by introducing *incremental errors* in order to improve the complexity.

In [6], the standard concept of atoms from the Vardi-Wolper construction of automaton for LTL is used. Refer to the diagram at the top of Figure 1. The formula $\mathtt{x} \approx \langle\top?\rangle\mathtt{y}$ in the left atom creates an obligation for the current value of $\mathtt{x}$

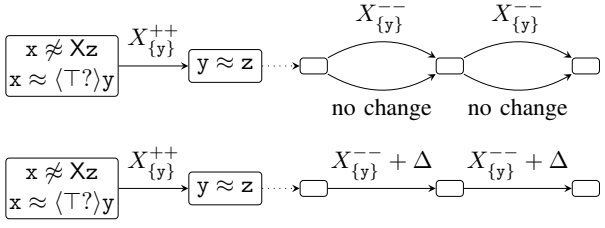


Fig. 1. Automaton constructions from [6] (top) and from this paper (bottom).

to appear some time in the future in $\mathtt{y}$. This obligation cannot be satisfied in the second atom, since $\mathtt{y}$ has to satisfy some other constraint there ($\mathtt{y} \approx \mathtt{z}$). To remember this unsatisfied obligation about $\mathtt{y}$ while taking the transition from the first atom to the second, the counter $X_{\{\mathtt{y}\}}$ is incremented. The counter can be decremented later in transitions that allow the repetition in $\mathtt{y}$. If several transitions allow such a repetition, only one of them needs to decrement the counter (since there was only one obligation at the beginning). The other transitions which should not decrement the counter can take the alternative labelled "no change" in the right part of Figure 1.

The idea here is to replace the combination of the decrementing transition and the "no change" transition in the top of Figure 1 with a single transition with incremental errors as shown in the bottom. After Lemma 6 below that formalises ideas from [6, Section 7], we prove that the transition with incremental errors is sufficient.

Lemma 6 ([6])**.** For a $\text{LRV}^\top$ formula ϕ that uses the variables $\{\mathtt{x}_1, \ldots, \mathtt{x}_k\}$, a VASS $\mathcal{A}_\phi = \langle Q, C, \delta\rangle$ can be defined, along with sets $Q_0, Q_f \subseteq Q$ of *initial* and *final* states resp., s.t.

- the set of counters C consists of all nonempty subsets of $\{\mathtt{x}_1, \ldots, \mathtt{x}_k\}$.
- For all $q, q' \in Q$, either $\{\mathbf{u} \mid (q, \mathbf{u}, q') \in \delta\} = [n_1, m_1] \times \cdots \times [n_{|C|}, m_{|C|}]$ for some $n_1, m_1, \ldots, n_{|C|}, m_{|C|} \in [-k, k]$, or $\{\mathbf{u} \mid (q, \mathbf{u}, q') \in \delta\} = \emptyset$. (We call this property *closure under component-wise interpolation*.)
- If $\delta \cap (\{q\} \times [-k, k]^C \times \{q'\})$ is not empty, then for every $X \in C$ there is $(q, \mathbf{u}, q') \in \delta$ so that $\mathbf{u}(X) \geq 0$. (We call this property *optional decrement*.)
- Let $\mathbf{0}$ be the counter valuation that assigns 0 to all counters. Then ϕ is satisfiable iff $\langle q_0, \mathbf{0}\rangle \xrightarrow{*} \langle q_f, \mathbf{0}\rangle$ for some $q_0 \in Q_0$ and $q_f \in Q_f$.

At the top of Figure 1, only one counter $X_{\{\mathtt{y}\}}$ is shown and is decremented by 1 for simplicity. In general, multiple counters can be changed and they can be incremented/decremented by any number up to k, depending on the initial and target atoms of the transition. If a counter can be incremented by k_1 and can be decremented by k_2, then there will also be transitions between the same pair of atoms allowing changes of $k_1 - 1, \ldots, 1, 0, -1, \ldots - (k_2 - 1)$. This corresponds to the closure under component-wise interpolation mentioned in the lemma above. The optional decrement property corresponds to the fact that there will always be a "no change" transition that does not decrement any counter.

Now, we show that a single transition that decrements all counters by the maximal possible number can simulate the set of all transitions between two atoms, using incremental errors. Let $\mathcal{A}_{\text{inc}} = \langle Q, C, \delta^{min}\rangle$ and $Q_0, Q_f \subseteq Q$, where Q, Q_0, Q_f and C are same as those of $\mathcal{A}_\phi$ and δ^{min} is defined as follows: $(q, minup_{q,q'}, q') \in \delta^{min}$ iff $\delta \cap (\{q\} \times [-k, k]^C \times \{q'\})$ is not empty and $minup_{q,q'}(X) = \min_{\mathbf{u}:(q,\mathbf{u},q')\in\delta}\{\mathbf{u}(X)\}$ for all $X \in C$.

Lemma 7. If $\langle q, \mathbf{v}\rangle \to \langle q', \mathbf{v}'\rangle$ in $\mathcal{A}_\phi$, then $\langle q, \mathbf{v}\rangle \to_{\text{gainy}} \langle q', \mathbf{v}'\rangle$ in $\mathcal{A}_{\text{inc}}$.

Proof sketch: If $\mathcal{A}_\phi$ takes one of the transitions in the top of Figure 1, $\mathcal{A}_{\text{inc}}$ takes the corresponding transition at the bottom, adjusting the incremental error Δ accordingly. ∎

Lemma 8. If $\langle q_1, \mathbf{v}_1\rangle \xrightarrow{*}_{\text{gainy}} \langle q_2, \mathbf{0}\rangle$ in $\mathcal{A}_{\text{inc}}$ and $\mathbf{v}'_1 \preceq \mathbf{v}_1$, then $\langle q_1, \mathbf{v}'_1\rangle \xrightarrow{*} \langle q_2, \mathbf{0}\rangle$ in $\mathcal{A}_\phi$.

Proof sketch: If $\mathcal{A}_{\text{inc}}$ takes a transition at the bottom of Figure 1, $\mathcal{A}_\phi$ takes the corresponding decrementing transition at the top, ignoring any incremental errors. This may result in $\mathcal{A}_\phi$ reaching $\mathbf{0}$ earlier in the run, in which case "no change" transitions are used in the rest of the run. ∎

Theorem 9. SAT(LRV$^\top$) is in 2EXPSPACE.

Proof sketch: The proof is in four steps (standard arguments are used for 3. and 4.).

1 From [6], a LRV$^\top$ formula ϕ is satisfiable iff $\langle q_0, \mathbf{0}\rangle \xrightarrow{*} \langle q_f, \mathbf{0}\rangle$ in $\mathcal{A}_\phi$ for some $q_0 \in Q_0$ and $q_f \in Q_f$.
2 This is the step that requires new insight. From Lemmas 7 and 8, $\langle q_0, \mathbf{0}\rangle \xrightarrow{*} \langle q_f, \mathbf{0}\rangle$ in $\mathcal{A}_\phi$ iff $\langle q_0, \mathbf{0}\rangle \xrightarrow{*}_{\text{gainy}} \langle q_f, \mathbf{0}\rangle$ in $\mathcal{A}_{\text{inc}}$.
3 Let $\mathcal{A}_{\text{dec}}$ be the VASS obtained from $\mathcal{A}_{\text{inc}}$ by "reversing" each transition. By replacing each gainy transition of $\mathcal{A}_{\text{inc}}$ by the reverse lossy transtion of $\mathcal{A}_{\text{dec}}$, we infer that $\langle q_0, \mathbf{0}\rangle \xrightarrow{*}_{\text{gainy}} \langle q_f, \mathbf{0}\rangle$ in $\mathcal{A}_{\text{inc}}$ iff $\langle q_f, \mathbf{0}\rangle \xrightarrow{*}_{\text{lossy}} \langle q_0, \mathbf{0}\rangle$ in $\mathcal{A}_{\text{dec}}$.
4 $\langle q_f, \mathbf{0}\rangle \xrightarrow{*}_{\text{lossy}} \langle q_0, \mathbf{0}\rangle$ iff $\langle q_f, \mathbf{0}\rangle \xrightarrow{*} \langle q_0, \mathbf{v}\rangle$ for some $\mathbf{v}$. Checking the latter condition is precisely the control state reachability problem for VASS.

The number of control states and counters in $\mathcal{A}_\phi$ and in $\mathcal{A}_{\text{dec}}$ is exponential in $|\phi|$ (size of ϕ). Each control state and transition function of $\mathcal{A}_{\text{dec}}$ can be represented in space polynomial in $|\phi|$. Given a control state, testing if it is an initial or a final state can be done in linear time in the size of the state. The automaton $\mathcal{A}_{\text{dec}}$ can be constructed in exponential time in $|\phi|$. Hence, the EXPSPACE upper bound for the control state reachability problem in VASS [28] gives the 2EXPSPACE upper bound for SAT(LRV$^\top$). ∎

Corollary 10. SAT(LRV) is in 2EXPSPACE.

V. Simulating Exponentially Many Counters

In Section IV, the reduction from SAT(LRV) to control state reachability involves an exponential blow up, since we use one counter for each subset of variables. The question of whether this can be avoided depends on whether LRV is powerful enough to reason about subsets of variables or whether there is a smarter reduction without a blow-up. Similar questions are open in other related areas [24].

Here we prove that LRV is indeed powerful enough to reason about subsets of variables. We establish a 2EXPSPACE lower bound, which leverages the power of LRV to access exponentially many counters through binary encoding. Consider the variables and their values shown in the table below. The conditions in the third row are about the equality of variable values in a single position.

Variable	$\mathtt{x}_1$	$\mathtt{x}_2$	$\mathtt{x}_3$	$\mathtt{z}$	inc	dec
Value	20	30	20	20	10	20
Condition	$\mathtt{x}_1 \approx \mathtt{z}$	$\mathtt{x}_2 \not\approx \mathtt{z}$	$\mathtt{x}_3 \approx \mathtt{z}$		inc $\not\approx \mathtt{z}$	dec $\approx \mathtt{z}$
Encodes	1	0	1			decrement

The combination of all the conditions can be thought of as encoding "decrement 101^{th} counter", considering 101 as the binary encoding of 5, where the i-th bit is 1 if $\mathtt{x}_i \approx \mathtt{z}$. A total of 8 counters can be manipulated with $\mathtt{x}_1, \mathtt{x}_2$ and $\mathtt{x}_3$. Using the power of LRV to reason about values that repeat in the future, we can encode the condition "for every counter, an increment is followed by a decrement in the future", which then ensures control state reachability.

The rest of this section is divided into three parts. The first part defines chain systems, which are like VASS where transition rules like "increment counter X" are replaced by "increment 101^{th} counter". The second part shows lower bounds for the control state reachability problem for chain systems. The third part shows that LRV can reason about chain systems of level 1.

A. Chain Systems

We introduce a new class of counter systems that is instrumental to show that SAT(LRV$^\top$) is 2EXPSPACE-hard. This is an intermediate formalism between counter automata with zero-tests with counters bounded by triple exponential values (having 2EXPSPACE-hard control state reachability problem) and properties expressed in LRV$^\top$. Systems with chained counters have no zero-tests and the only updates are increments and decrements. However, the systems are equipped with a finite family of counters, each family having an exponential number of counters. Let $\mathtt{exp}(0,n) \stackrel{\text{def}}{=} n$ and $\mathtt{exp}(k+1,n) \stackrel{\text{def}}{=} 2^{\mathtt{exp}(k,n)}$ for every $k \geq 0$.

Definition 11. A *counter system with chained counters* (herein called a *chain system*) is a tuple $\mathcal{A} = \langle Q, f, k, Q_0, Q_F, \delta \rangle$ where (1) $f : [1,n] \to \mathbb{N}$ where $n \geq 1$ is the *number of chains* and $\mathtt{exp}(k, f(\alpha))$ is the number of counters for the chain α where $k \geq 0$, (2) Q is a non-empty finite set of *states*, (3) $Q_0 \subseteq Q$ is the set of *initial states* and $Q_F \subseteq Q$ is the set of *final states*, (4) δ is the set of *transitions* in $Q \times I \times Q$ where

$$I = \{\mathrm{inc}(\alpha), \mathrm{dec}(\alpha), \mathrm{next}(\alpha), \mathrm{prev}(\alpha), \mathrm{first}(\alpha)?, \overline{\mathrm{first}(\alpha)}?, \\ \mathrm{last}(\alpha)?, \overline{\mathrm{last}(\alpha)}? : \alpha \in [1,n]\}.$$

The system $\mathcal{A} = \langle Q, f, k, Q_0, Q_F, \delta \rangle$ is said to be at *level* k. In order to encode the natural numbers from f and the value k, we use a unary representation. We say that a transition containing $\mathrm{inc}(\alpha)$ is α-*incrementing*, and a transition containing $\mathrm{dec}(\alpha)$ is α-*decrementing*. The idea is that for each chain $\alpha \in [1,n]$, we have $\mathtt{exp}(k, f(\alpha))$ counters, but we cannot access them directly in the transitions like we do in VASS. Instead, we have a pointer to a counter that we can move. We can ask if we are pointing to the first counter ($\mathrm{first}(\alpha)?$) or not ($\overline{\mathrm{first}(\alpha)}?$), or the last counter ($\mathrm{last}(\alpha)?$) or not ($\overline{\mathrm{last}(\alpha)}?$), and we can change the pointer to the next ($\mathrm{next}(\alpha)$) or previous ($\mathrm{prev}(\alpha)$) counter.

A *run* is a finite sequence ρ in δ^* such that

1) for every two $\rho(i) = q \xrightarrow{\mathrm{instr}} r$ and $\rho(i+1) = q' \xrightarrow{\mathrm{instr}'} r'$ we have $r = q'$,
2) for every chain $\alpha \in [1,n]$, for every $i \in [1, |\rho|]$, we have $0 \leq c_i^\alpha < \mathtt{exp}(k, f(\alpha))$ where

$$c_i^\alpha = \mathrm{card}(\{i' < i \mid \rho(i') = q \xrightarrow{\mathrm{next}(\alpha)} r\}) \; - \\ \mathrm{card}(\{i' < i \mid \rho(i') = q \xrightarrow{\mathrm{prev}(\alpha)} r\}), \tag{1}$$

3) for every $i \in [1, |\rho|]$ and for every chain $\alpha \in [1,n]$,
   (a) if $\rho(i) = q \xrightarrow{\mathrm{first}(\alpha)?} q'$, then $c_i^\alpha = 0$;
   (b) if $\rho(i) = q \xrightarrow{\overline{\mathrm{first}(\alpha)}?} q'$, then $c_i^\alpha \neq 0$;
   (c) if $\rho(i) = q \xrightarrow{\mathrm{last}(\alpha)?} q'$, then $c_i^\alpha = \mathtt{exp}(k, f(\alpha)) - 1$;
   (d) if $\rho(i) = q \xrightarrow{\overline{\mathrm{last}(\alpha)}?} q'$, then $c_i^\alpha \neq \mathtt{exp}(k, f(\alpha)) - 1$.

A run is *accepting* whenever $\rho(1)$ starts with an initial state from Q_0 and $\rho(|\rho|)$ ends with a final state from Q_F. A run is *perfect* iff for every $\alpha \in [1,n]$, there is some injective function

$$\gamma : \{i \mid \rho(i) \text{ is } \alpha\text{-decrementing}\} \to \{i \mid \rho(i) \text{ is } \alpha\text{-incrementing}\}$$

such that for every $\gamma(i) = j$ we have that $j < i$ and $c_i^\alpha = c_j^\alpha$. A run is *gainy and ends at zero* iff for every chain $\alpha \in [1,n]$, there is some injective function

$$\gamma : \{i \mid \rho(i) \text{ is } \alpha\text{-incrementing}\} \to \{i \mid \rho(i) \text{ is } \alpha\text{-decrementing}\}$$

such that for every $\gamma(i) = j$ we have that $j > i$ and $c_i^\alpha = c_j^\alpha$. In the sequel, we shall simply say that the run is gainy. Below, we define two problems for which we shall characterize the computational complexity.

PROBLEM: Existence of a perfect accepting run of level $k \geq 0$ (Per(k))
INPUT: A chain system $\mathcal{A}$ of level k. QUESTION: Does $\mathcal{A}$ have a *perfect* accepting run?

PROBLEM: Existence of a gainy accepting run of level $k \geq 0$ (Gainy(k))
INPUT: A chain system $\mathcal{A}$ of level k. QUESTION: Does $\mathcal{A}$ have a *gainy* accepting run?

Per(k) is actually a control state reachability problem in VASS where the counters are encoded succinctly. Gainy(k) is a reachability problem in VASS with incrementing errors and the reached counter values are equal to zero. Here, the counters are encoded succinctly too.

Lemma 12. For every $k \geq 0$, Per(k) and Gainy(k) are interreducible in logarithmic space.

Lemma 13. Per(k) is in $(k+1)$EXPSPACE.

The proof of Lemma 13 consists of simulating perfect runs by the runs of a VASS in which the control states record the positions of the pointers in the chains.

B. Hardness Results for Chain Systems

We show that Per(k) is $(k+1)$EXPSPACE-hard. The implication is that replacing direct access to counters by a pointer that can move along a chain of counters does not decrease the power of VASS, while providing access to more counters. To demonstrate this, we extend Lipton's EXPSPACE-hardness proof for the control state reachability problem in VASS [22] (see also its exposition in [8]). Since our pointers can be moved only one step at a time, this extension involves new insights into the control flow of algorithms used in Lipton's proof, allowing us to implement it even with the limitations imposed by step-wise pointer movements.

Lipton's proof starts from the standard result in computability theory that a Turing Machine using space 2^{n^γ} can be simulated by a counter automaton equipped with 4 counters whose values are bounded by $2^{2^{n^\gamma}}$. Lipton's proof shows that such a counter automaton can be simulated with a VASS. Unlike counter automata, VASS does not have zero-test transitions. Each counter $\mathtt{c}$ of the automaton is complemented with an extra counter $\overline{\mathtt{c}}$ in the VASS. The VASS is designed such that the sum of the values in $\mathtt{c}$ and $\overline{\mathtt{c}}$ is $2^{2^{n^\gamma}}$ in any reachable configuration. Now testing $\mathtt{c}$ for zero is equivalent to testing that $\overline{\mathtt{c}}$ is $2^{2^{n^\gamma}}$. This later test is implemented in a VASS as explained next.

Figure 2 illustrates how a counter $\mathtt{s}_i$ is decremented 2^{2^i} times. There are two nested loops indexed by $\mathtt{y}_{i-1}$ and $\mathtt{z}_{i-1}$, initialised to $2^{2^{i-1}}$. When all the iterations are finished, we would have decremented $\mathtt{s}_i$ exactly $2^{2^{i-1}} \times 2^{2^{i-1}} = 2^{2^i}$ times. Testing $\mathtt{y}_{i-1}$ and $\mathtt{z}_{i-1}$ for zero is done by a similar gadget, where $i, i-1$ are replaced by $i-1, i-2$. The resulting VASS is composed of n^γ constant sized VASS, each one implementing the nested loops for an i between 1 and n^γ. The idea behind our $(k+1)$EXPSPACE lower bound is to replace the VASS with a chain system and n^γ with $\exp(k, n^\gamma)$. This will not work as it is, since a chain system composed of $\exp(k, n^\gamma)$ constant sized portions will not give a polynomial-time reduction. We reduce the size of the chain system by observing that the decrementing algorithm for $i-1$ is the same as the one for i, except that $i, i-1$ are replaced by $i-1, i-2$ respectively. We can write a single sequence of instructions for the decrementing algorithm and invoke it for any i by placing the pointers at the appropriate counters. Roughly speaking, $\exp(k, n^\gamma)$ copies of the code for large decrements is replaced by one copy of the code plus a chain of implicit length $\exp(k, n^\gamma)$ equipped with a pointer to refer to a specific copy.

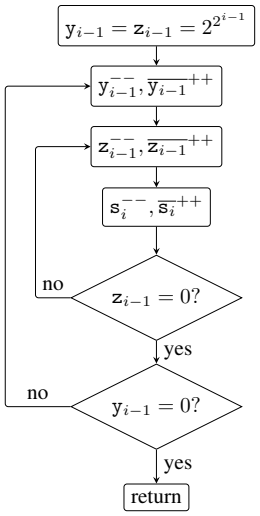


Fig. 2. Control flow of algorithm that decrements $\mathtt{s}_i$ 2^{2^i} times.

The main difficulty in the implementation is to ensure that the multiple invocations of the algorithm return to the correct point. Unlike Lipton's proof that uses VASS, we have to work with chain systems and this introduces some more constructions, the technical details of which are rather tedious.

Theorem 14. Per(k) is $(k+1)$EXPSPACE-hard.

C. Reasoning About Chain Systems with LRV

Given a chain system $\mathcal{A}$ of level 1, we construct a formula in LRV that is satisfiable iff $\mathcal{A}$ has a gainy accepting run. Hence, we get a 2EXPSPACE lower bound for SAT(LRV). The main idea is to encode runs of chain systems with LRV formulas that access the counters using binary encoding, so that the formulas can handle exponentially many counters.

Lemma 15. There is a polynomial-time reduction from Gainy(1) into SAT(LRV(X, F)).

Proof sketch: Let $\mathcal{A} = \langle Q, f, 1, Q_0, Q_F, \delta \rangle$ be a chain system of level 1 with $f : [1, n] \to \mathbb{N}$, having thus n chains of counters, of respective size $2^{f(1)}, \ldots, 2^{f(n)}$.

We encode a word $\rho \in \delta^*$ that represents an accepting run. For this, we use the alphabet δ of transitions. We can simulate the labels $\delta = \{t_1, \ldots, t_m\}$ with variables $\mathtt{t}_0, \ldots, \mathtt{t}_m$, where a node has an encoding of the label t_i iff the formula $\langle t_i \rangle = \mathtt{t}_0 \approx \mathtt{t}_i$ holds true. We build an LRV formula φ so that there is an accepting gainy run $\rho \in \delta^*$ of $\mathcal{A}$ if, and only if, there is a model σ so that $\sigma \models \varphi$ and σ encodes ρ.

The counter-blind conditions to check are: (a) Every position satisfies $\langle t \rangle$ for some $t \in \delta$; (b) the first position satisfies $\langle (q_0, \mathrm{instr}, q) \rangle$ for some $q_0 \in Q_0$, $\mathrm{instr} \in I$, $q \in Q$; (c) the last position satisfies $\langle (q, \mathrm{instr}, q') \rangle$ for some $q \in Q$, $\mathrm{instr} \in I$, $q' \in Q_F$; (d) no two consecutive positions i and $i+1$ satisfy $\langle (q, u, q') \rangle$ and $\langle (p, u', p') \rangle$ respectively, with $q' \neq p$. The difficulty is then in checking that the values of the counters encode indeed a correct gainy run.

We say that a position i is *α-incrementing* [resp. *α-decrementing*] if it satisfies $\langle (q, u, q') \rangle$ for some $q, q' \in Q$ and $u = \mathrm{inc}(\alpha)$ [resp. $u = \mathrm{dec}(\alpha)$]. We use a label (α, i) and variables $\mathtt{x}^\alpha_{inc}$, $\mathtt{x}^\alpha_{dec}$ for every $\alpha \in [1, n]$ and $i \in [1, f(\alpha)]$. We say that a position i operates on the α-counter $\mathtt{c}$, if $\langle (\alpha, j) \rangle$ holds (i.e., position i encodes the label (α, j)) for every position j of the representation of $\mathtt{c}$ in base 2 containing a '1', and $\neg\langle (\alpha, j) \rangle$ for every position j containing a '0'. Note that we can encode every value $0 \leq \mathtt{c} < 2^{f(\alpha)}$.

For every chain α, let us consider the following properties:

- Every two positions of σ have different values of $\mathtt{x}^\alpha_{inc}$ [resp. of $\mathtt{x}^\alpha_{dec}$].
- For every position i of σ operating on an α-counter $\mathtt{c}$ with an instruction '$\mathrm{first}(\alpha)$?' [resp. '$\overline{\mathrm{first}(\alpha)}$?', '$\mathrm{last}(\alpha)$?', '$\overline{\mathrm{last}(\alpha)}$?'] , we have $\mathtt{c} = 0$ [resp. $\mathtt{c} \neq 0$, $\mathtt{c} = 2^{f(\alpha)} - 1$, $\mathtt{c} \neq 2^{f(\alpha)} - 1$].
- For every position i of σ operating on an α-counter $\mathtt{c}$, if the position contains an instruction '$\mathrm{next}(\alpha)$' [resp. '$\mathrm{prev}(\alpha)$'], then the next position $i+1$ operates on the α-counter $\mathtt{c}+1$ [resp. $\mathtt{c}-1$]; otherwise, the position $i+1$ operates on the α-counter $\mathtt{c}$.
- For every α-incrementing position i of σ operating on an α-counter $\mathtt{c}$ there is a future α-decrementing position $j > i$ on the same α-counter, so that $\sigma(i)(\mathtt{x}^\alpha_{inc}) = \sigma(j)(\mathtt{x}^\alpha_{dec})$.

In fact, these properties together with (a)–(d) are sufficient and necessary to encode a gainy and accepting run of $\mathcal{A}$, and they can be all expressed in LRV. Then, we obtain a polynomial-time reduction from Gainy(1) into the satisfiability problem for LRV(X, F). ∎

VI. A Robust Equivalence

We have seen that the satisfiability problem for LRV is equivalent to the control state reachability problem in an exponentially larger VASS. In this section we evaluate how robust is this result with LRV variants or fragments.

A. Infinite Words with Multiple Data

So far, we have considered only finite words with multiple data. It is also natural to consider the variant with infinite words, but it is known that this sometimes leads to undecidability. However, in this case the decidability and complexity results are preserved. Let LRV$_\omega$ be the variant of LRV in which infinite models of length ω are taken into account

instead of finite ones. The logics PLRV_ω, $\text{LRV}_\omega^\top$, etc. are defined accordingly.

Proposition 16. (I) $\text{SAT}(\text{PLRV}_\omega)$ is decidable.
(II) $\text{SAT}(\text{LRV}_\omega)$ is 2EXPSPACE-complete.

B. Adding MSO-Definable Temporal Operators

It is standard to extend LTL with MSO-definable temporal operators (see e.g. [30], [13]), and the same can be done with LRV. A temporal operator $\oplus$ of arity n is *MSO-definable* whenever there is a formula $\phi(\mathtt{x}, \mathtt{P_1}, \ldots, \mathtt{P_n})$ from monadic second-order logic with a unique free position variable $\mathtt{x}$ and with n free unary predicates such that $\sigma, i \models \oplus(\phi_1, \ldots, \phi_n)$ iff $\sigma \models \phi(i, X_1, \ldots, X_n)$ in MSO where each set X_j is equal to the set of positions from σ satisfying the formula ϕ_j, see e.g. [13]. The 2EXPSPACE upper bound is preserved with a fixed finite set of MSO-definable operators.

Theorem 17. Let $\{\oplus_1, \ldots, \oplus_N\}$ be a finite set of MSO-definable temporal operators. Satisfiability problem for LRV extended with $\{\oplus_1, \ldots, \oplus_N\}$ is 2EXPSPACE-complete.

Note that PLRV augmented with MSO-definable temporal operators is decidable too.

C. The Now Operator or the Effects of Moving the Origin

The satisfiability problem for Past LTL with the temporal operator Now is known to be EXPSPACE-complete [20]. The satisfaction relation is parameterized by the current position of the origin and past-time temporal operators use that position. For instance, $\sigma, i \models_o \mathtt{Now}\ \phi \overset{\text{def}}{\Leftrightarrow} \sigma, i \models_i \phi$ and $\sigma, i \models_o \phi_1 \mathsf{S} \phi_2 \overset{\text{def}}{\Leftrightarrow}$ there is $j \in [o, i]$ such that $\sigma, j \models_o \phi_2$ and for all $j' \in [j-1, i]$, we have $\sigma, j' \models_o \phi_1$. The powerful operator Now can be obviously defined in MSO but not with the above definition since it requires *two* free position variables, one of which refers to the current position of the origin.

Theorem 18. $\text{SAT}(\text{LRV} + \mathtt{Now})$ is 2EXPSPACE-complete.

This contrasts with the undecidability results from [17, Theorem 5] in presence of the operator Now.

D. Bounding the Number of Variables

Given the relationship between the number of variables in a formula and the number of counters needed in the corresponding VASS, we investigate the consequences of fixing the number of variables. Interestingly, this classical restriction has an effect only for $\text{LRV}^\top$, i.e., when test formulas ϕ are restricted to $\top$ in $\mathtt{x} \approx \langle \phi ? \rangle \mathtt{y}$. Let LRV_k [resp. $\text{LRV}_k^\top$, $\text{PLRV}_k^\top$] be the restriction to formulas with at most k variables. In [6, Theorem 5], it is shown that $\text{SAT}(\text{LRV}_1^\top)$ is PSPACE-complete by establishing a reduction into the reachability problem for VASS when counter values are linearly bounded. Below, we generalize this result for any $k \geq 1$ by using the proof of Theorem 9 and the fact that the control state reachability problem for VASS with at most k counters is in PSPACE.

Proposition 19. For every $k \geq 1$, $\text{SAT}(\text{LRV}_k^\top)$ is PSPACE-complete.

This does not imply that LRV_k is in PSPACE, since the reduction from LRV into $\text{LRV}^\top$ in Section IV-A introduces new variables. In fact, it introduces a number of variables that depends on the size of the formula. It turns out that this is unavoidable, and that its satisfiability problem is 2EXPSPACE-hard, by the following reduction.

Lemma 20. There is a polynomial-time reduction from $\text{SAT}(\text{LRV}^\top)$ into $\text{SAT}(\text{LRV}_1)$ [resp. $\text{SAT}(\text{PLRV}^\top)$ and $\text{SAT}(\text{PLRV}_1)$].

Proof sketch: Let $\varphi \in \text{LRV}^\top$ using k variables $\mathtt{x}_1, \ldots, \mathtt{x}_k$ so that $\sigma \models \varphi$. We will encode σ restricted to $\mathtt{x}_1, \ldots, \mathtt{x}_k$ inside a model σ_φ with only one variable, say $\mathtt{x}$. To this end, σ_φ is divided into N segments $s_1 \cdots s_N$ of equal length, where $N = |\sigma|$. A special data value d not used in σ plays the role of *delimiter* between segments and between positions that code values from σ. Suppose that d is a data value that is not in σ. Then, each segment s_i has length $k' = 2k + 1$, and is defined as the data values "$d\ d_1\ d\ d_2\ \ldots d\ d_k\ d$", where $d_j = \sigma(i)(\mathtt{x}_j)$. Figure 3 contains an example. We can force that the model has this shape with LRV_1. With this coding, we can tell that we are between two segments if there are two consecutive equal data values. In fact, we are at a position corresponding to $\mathtt{x}_i$ (for $i \in [1, k]$) inside a segment if we are standing at the $2i$-th element of a segment, and we can test this with the formula $\gamma_i = \mathsf{X}^{k'-2i}\mathtt{x} \approx \mathsf{X}^{k'-2i+1}\mathtt{x} \vee (\mathsf{X}^{k'-2i}\top \wedge \neg \mathsf{X}^{k'-2i+1}\top)$. Using the γ_i's, we translate $\mathtt{x}_i \approx \langle \top ? \rangle \mathtt{x}_j$ in φ into a formula that: moves to the position $2i$ of the segment (the one corresponding to $\mathtt{x}_i$), and tests $\mathtt{x} \approx \langle \gamma_j ? \rangle \mathtt{x}$. We can do this similarly with all formulas. ∎

Corollary 21. For all $k \geq 1$, $\text{SAT}(\text{LRV}_k)$ is 2EXPSPACE-complete.

Corollary 22. For all $k \geq 1$, $\text{SAT}(\text{PLRV}_k)$ is as hard as Reach(VASS).

E. The Power of Pairs of Repeating Values

Let us consider an LRV variant so that repetition of tuples of values is possible: we add formulas of the form $(\mathtt{x}_1, \ldots, \mathtt{x}_k) \approx \langle \phi ? \rangle (\mathtt{y}_1, \ldots, \mathtt{y}_k)$ where $\mathtt{x}_1, \ldots, \mathtt{x}_k, \mathtt{y}_1, \ldots, \mathtt{y}_k \in \text{VAR}$. This extends $\mathtt{x} \approx \langle \varphi ? \rangle \mathtt{y}$ by testing whether the vector of data values from the variables $(\mathtt{x}_1, \ldots, \mathtt{x}_k)$ of the current position coincides with that of $(\mathtt{y}_1, \ldots, \mathtt{y}_k)$ in a future position.

We call this extension LRV_{vec}. Unfortunately, we can show that $\text{SAT}(\text{LRV}_{vec})$ is undecidable, even when only tuples of dimension 2 are allowed. This is proved by reduction from a variant of Post's Correspondence Problem (PCP). In order to code solutions of PCP instances, we adapt a proof technique used in [1] for first-order logic with two variables and two equivalence relations on words. However, our proof uses only *future* modalities (unlike the proof of [1, Proposition 27]) and no past obligations (unlike the proof of [17, Theorem 4]). To prove this result, we work with a variant of the PCP problem in which solutions $u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots v_{i_n}$ have to satisfy $|u_{i_1} \cdots u_{i_j}| \leq |v_{i_1} \cdots v_{i_j}|$ for every j.

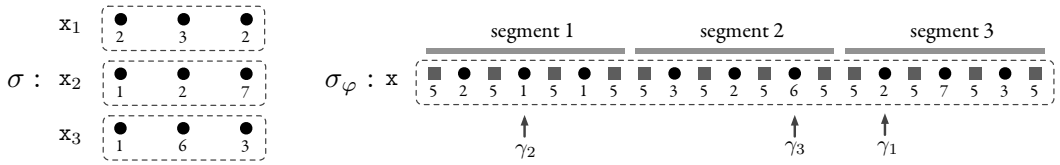


Fig. 3. Example of the reduction from SAT(LRV$^\top$) into SAT(LRV$_1$), for $k = 3$, $N = 3$ and $d = 5$.

Theorem 23. SAT(LRV$_{vec}$(X, U)) is undecidable.

Furthermore, our proof can be adapted to show that SAT(LRV$^\top_{vec}$) is also undecidable.

VII. Implications for Logics on Data Words

A data word is an element of $(\Sigma \times \mathbb{D})^*$, where Σ is a finite alphabet and $\mathbb{D}$ is an infinite domain. We focus here on first-order logic with two variables, and on a temporal logic.

Two-variable Logics: We study a fragment of EMSO$^2(+1, <, \sim)$ on data words, and we show that it has a satisfiability problem in 3EXPSPACE, as a consequence of our results on the satisfiability for LRV. The satisfiability problem for EMSO$^2(+1, <, \sim)$ is known to be decidable, equivalent to the reachability problem for VASS [1], with no known primitive-recursive algorithm. Here we show a large fragment with elementary complexity.

Consider the fragment of EMSO$^2(+1, <, \sim)$ —that is, first-order logic with two variables, with a prefix of existential quantification over monadic relations— where all formulas are of the form $\exists X_1, \ldots, X_n\ \varphi$ with

$$\begin{aligned} \varphi ::= & \ \mathit{atom} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \\ & \exists x \exists y\ \varphi \mid \forall x \forall y\ \varphi \mid \forall x \exists y\ (x \leq y\ \wedge\ \varphi), \text{ where} \\ \mathit{atom} ::= & \ \zeta = \zeta' \mid \zeta \neq \zeta' \mid \zeta \sim \zeta' \mid \zeta < \zeta' \mid \zeta \leq \zeta' \mid \\ & +1(\zeta, \zeta') \mid X_i(\zeta) \mid a(\zeta) \end{aligned}$$

for any $a \in \Sigma$, $i \in [1, n]$, and $\zeta, \zeta' \in \{x, y\}$. The relation $x < y$ tests that the position y appears after x in the word; $+1(x, y)$ tests that y is the next position to x; and $x \sim y$ tests that positions x and y have the same data value. We call this fragment *forward*-EMSO$^2(+1, <, \sim)$. In fact, *forward*-EMSO$^2(+1, <, \sim)$ captures EMSO$^2(+1, <)$ (i.e., all regular languages on the finite labeling of the data word).[1] However, it seems to be strictly less expressive than EMSO$^2(+1, <, \sim)$, since it does not appear to be able to express, for example, *there are exactly two occurrences of every data value*. Yet, it can express *there are at most two occurrences of every data value* (with $\exists X\ \forall x \forall y.\ x \sim y \wedge x < y \rightarrow X(x) \wedge \neg X(y)$), and *there is exactly one occurrence of every data value*. For the same reason, it would neither capture EMSO$^2(+1, \sim)$.

By an exponential reduction into SAT(LRV), we obtain that *forward*-EMSO$^2(+1, <, \sim)$ is decidable in elementary time.

[1]Indeed, note that one can easily test whether a word is accepted by a finite automaton with the help of a monadic relation X_{fst} that holds only at the first position. Further, the property of X_{fst} can be expressed in *forward*-EMSO$^2(+1, <, \sim)$ as $(\exists y\ .\ X_{fst}(y)) \wedge (\forall x \forall y\ .\ x < y \rightarrow \neg X_{fst}(y))$.

Proposition 24. The satisfiability problem for *forward*-EMSO$^2(+1, <, \sim)$ is in 3EXPSPACE.

Proof sketch: Through a standard translation we can bring any formula of *forward*-EMSO$^2(+1, <, \sim)$ into a formula of the form

$$\varphi = \exists X_1, \ldots, X_n\ \left(\forall x \forall y\ \chi\ \wedge\ \bigwedge_k \forall x \exists y\ (x \leq y \wedge \psi_k) \right)$$

that preserves satisfiability, where χ and all ψ_k's are quantifier-free formulas, and there are no tests for labels. Furthermore, this is a polynomial-time translation. This translation is just the Scott normal form of EMSO$^2(+1, <, \sim)$ [1] adapted to *forward*-EMSO$^2(+1, <, \sim)$, and can be done in the same way.

We now give an exponential-time translation tr : *forward*-EMSO$^2(+1, <, \sim) \rightarrow$ LRV. For any formula φ of *forward*-EMSO$^2(+1, <, \sim)$, $tr(\varphi)$ is an equivalent (in the sense of satisfiability) LRV formula.

The translation makes use of: a distinguished variable $\mathtt{x}$ that encodes the data values of any data word satisfying φ; variables $\mathtt{x}_0, \ldots, \mathtt{x}_n$ that are used to encode the monadic relations $X_1, \ldots, X_n$; and a variable $\mathtt{x}_{prev}$ whose purpose will be explained later on. We give now the translation. To translate $\forall x \forall y\ \chi$, we first bring the formula to a form

$$\bigwedge_{m \in M} \neg\exists x \exists y\ (\ x \leq y\ \wedge\ \chi_m\ \wedge\ \chi^x_m\ \wedge\ \chi^y_m\ ), \qquad (\star)$$

where every χ^x_m (resp. χ^y_m) is a conjunction of (negations of) atoms of monadic relations on x (resp. y); and $\chi_m = \mu \wedge \nu$ where $\mu \in \{x{=}y, +1(x, y), \neg(+1(x, y) \vee x{=}y)\}$ and $\nu \in \{x \sim y, \neg(x \sim y)\}$. As an example, if $\chi = \neg(x \sim y) \vee X(x) \vee X(y)$, then the corresponding formula would be

$$\bigwedge_{\mu, \chi^x, \chi^y} \begin{array}{l} \big((\neg\exists x \exists y\ x \leq y \wedge \mu \wedge \neg X(x) \wedge \chi^y)\ \wedge \\ \ (\neg\exists x \exists y\ x \leq y \wedge \mu \wedge \chi^x \wedge \neg X(y))\big) \end{array}$$

for all $\mu \in \{x{=}y, +1(x, y), \neg(+1(x, y) \vee x{=}y)\}$, $\chi^x \in \{X(x), \neg X(x)\}$, $\chi^y \in \{X(y), \neg X(y)\}$. We can bring the formula into this normal form in exponential time. We define $tr(\chi^x_m)$ as the conjunction of all the formulas $\mathtt{x}_0 \approx \mathtt{x}_i$ so that $X_i(x)$ is a conjunct of χ^x_m, and all the formulas $\neg(\mathtt{x}_0 \approx \mathtt{x}_i)$ so that $\neg X_i(x)$ is a conjunct of χ^x_m; we do similarly for $tr(\chi^y_m)$. If $\mu = +1(x, y)$ and $\nu = x \sim y$ we define $tr\left(\exists y\big(x \leq y \wedge \chi_m \wedge \chi^x_m \wedge \chi^y_m\big)\right)$ as $\mathtt{x} \approx \mathsf{X}\mathtt{x} \wedge tr(\chi^x_m) \wedge \mathsf{X} tr(\chi^y_m)$.

If $\mu = (x = y)$ and $\nu = x \sim y$ we translate

$$tr\left(\exists y\big(x \leq y \wedge \chi_m \wedge \chi^x_m \wedge \chi^y_m\big)\right)\ =\ tr(\chi^x_m) \wedge tr(\chi^y_m).$$

We proceed similarly for $\mu = +1(x,y)$, $\nu = \neg(x \sim y)$; and the translation is of course $\bot$ (false) if $\mu = (x{=}y)$, $\nu = \neg(x{\sim}y)$. The difficult cases are the remaining ones. Suppose $\mu = \neg(+1(x,y) \vee x{=}y)$, $\nu = x{\sim}y$. In other words, x is at least two positions before y, and they have the same data value. Observe that the formula $tr(\chi^x_m) \wedge \mathtt{x} \approx \langle tr(\chi^y_m)?\rangle \mathtt{x}$ does not encode precisely this case, as it would correspond to a weaker condition $x < y \wedge x \sim y$. In order to properly translate this case we make use of the variable $\mathtt{x}_{prev}$, ensuring that it always has the data value of the variable $\mathtt{x}$ in the previous position

$$prev \;=\; \mathsf{G}\big(\; \mathsf{X}\top \;\Rightarrow\; \mathtt{x} \approx \mathsf{X}\mathtt{x}_{prev} \;\big).$$

We then define $tr\big(\exists y \;\big(\; x \le y \;\wedge\; \chi_m \;\wedge\; \chi^x_m \;\wedge\; \chi^y_m \;\big)\big)$ as

$$tr(\chi^x_m) \;\wedge\; \mathtt{x} \approx \langle \mathtt{x}_{prev} \approx \langle tr(\chi^y_m)?\rangle \mathtt{x}?\rangle\, \mathtt{x}_{prev}.$$

Note that by nesting twice the future obligation we ensure that the target position where $tr(\chi^y_m)$ must hold is at a distance of at least two positions. For $\nu = \neg(x{\sim}y)$ we produce a similar formula, replacing the innermost appearance of $\approx$ with $\not\approx$ in the formula above. We then define $tr(\forall x \forall y\ \chi)$ as

$$prev \;\wedge\; \bigwedge_{m \in M} \neg\, \mathsf{F}\; tr\big(\exists y\ (x \le y \;\wedge\; \chi_m \wedge \chi^x_m \wedge \chi^y_m)\big).$$

To translate $\forall x \exists y\ (x \le y \wedge \psi_k)$ we proceed in a similar way. One can show that tr preserves satisfiability. By Corollary 10 we can decide the satisfiability of the translation in 2EXPSPACE, and since the translation is exponential, this gives us a 3EXPSPACE bound for the satisfiability of *forward*-EMSO$^2(+1, <, \sim)$. ∎

Temporal Logics: Our results have also implications for a fragment of the logic LTL extended with one register for storing and comparing data values (called the *freeze* operator), noted $\mathrm{LTL}^{\downarrow}_1$. Its satisfiability problem was shown to be decidable, but with non-primitive-recursive complexity [7].

Our results yield a fragment of $\mathrm{LTL}^{\downarrow}_1$ with elementary 2EXPSPACE upper bound.

VIII. Conclusion

We introduced the logic LRV and variants by allowing data value repetitions thanks to formulas of the form $\mathtt{x} \approx \langle \phi? \rangle \mathtt{y}$. LRV extends the main logic from [6] but it is also a fragment of BD-LTL from [17] whose satisfiability is equivalent to Reach(VASS). We showed that SAT(LRV) is 2EXPSPACE-complete by reduction into the control-state reachability problem for VASS (via a detour to gainy VASS) and by introducing a new class of counter machines (the chain systems) in order to get the complexity lower bound. This new class of counter machines is interesting for its own sake and could be used to establish other hardness results thanks to our results that extend non-trivially the proof from [22], [8]. Correspondences between extensions of LRV (such as PLRV, PLRV$^\top$ and PLRV$_1$) and reachability problem for VASS are also established, strengthening further the relationships between data logics and reachability problems for counter machines. Other results for variants are presented in the paper and a summary can be found below.

$$\mathrm{LRV}^\top_k : \text{PSPACE-complete}$$
$$\mathrm{LRV} \equiv \mathrm{LRV}^\top \equiv \mathrm{LRV}_1 \equiv \mathrm{LRV} + \{\oplus_1, \ldots, \oplus_k\} : \text{2EXPSPACE-complete}$$
$$\mathrm{PLRV} \equiv \mathrm{PLRV}^\top \equiv \mathrm{PLRV}_1 \equiv \text{Reach(VASS)}$$
$$\mathrm{LRV}^\top_{vec} : \text{undecidable}$$

References

[1] M. Bojańczyk, C. David, A. Muscholl, Th. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM ToCL*, 12(4):27, 2011.

[2] M. Bojańczyk and S. Lasota. An extension of data automata that captures XPath. In *LICS'10*, pages 243–252. IEEE, 2010.

[3] B. Bollig, A. Cyriac, P. Gastin, and K. Narayan Kumar. Model checking languages of data words. In *FoSSaCS'12*, volume 7213 of *LNCS*, pages 391–405. Springer, 2012.

[4] P. Bouyer. A logical characterization of data languages. *IPL*, 84(2):75–85, 2002.

[5] C. David. *Analyse de XML avec données non-bornées*. PhD thesis, LIAFA, 2009.

[6] S. Demri, D. D'Souza, and R. Gascon. Temporal logics of repeating values. *JLC*, 22(5):1059–1096, 2012.

[7] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM ToCL*, 10(3), 2009.

[8] J. Esparza. Decidability and complexity of Petri net problems — an introduction. In *Advances in Petri Nets 1998*, volume 1491 of *LNCS*, pages 374–428. Springer, Berlin, 1998.

[9] D. Figueira. *Reasoning on words and trees with data*. PhD thesis, ENS Cachan, 2010.

[10] D. Figueira. A decidable two-way logic on data words. In *LICS'11*, pages 365–374. IEEE, 2011.

[11] D. Figueira and L. Segoufin. Future-looking logics on data words and trees. In *MFCS'09*, volume 5734 of *LNCS*, pages 331–343, 2009.

[12] M. Fitting. Modal logic between propositional and first-order. *JLC*, 12(6):1017–1026, 2002.

[13] P. Gastin and D. Kuske. Satisfiability and model checking for MSO-definable temporal logics are in PSPACE. In *CONCUR'03*, volume 2761 of *LNCS*, pages 222–236. Springer, 2003.

[14] P. Jančar. Undecidability of bisimilarity for Petri Nets and some related problems. *TCS*, 148:281–301, 1995.

[15] A. Kara, Th. Schwentick, and Th. Zeume. Temporal logics on words with multiple data values. In *FST&TCS'10*, pages 481–492. LZI, 2010.

[16] S. Rao Kosaraju. Decidability of reachability in vector addition systems. In *STOC'82*, pages 267–281, 1982.

[17] O. Kupferman and M. Vardi. Memoryful Branching-Time Logic. In *LICS'06*, pages 265–274. IEEE, 2006.

[18] F. Laroussinie, N. Markey, and Ph. Schnoebelen. Temporal logic with forgettable past. In *LICS'02*, pages 383–392. IEEE, 2002.

[19] J. Leroux. Vector addition system reachability problem: a short self-contained proof. In *POPL'11*, pages 307–316, 2011.

[20] R. J. Lipton. The reachability problem requires exponential space. Technical Report 62, Dept. of Computer Science, Yale University, 1976.

[21] A. Lisitsa and I. Potapov. Temporal logic with predicate λ-abstraction. In *TIME'05*, pages 147–155. IEEE, 2005.

[22] A. Manuel and R. Ramanujam. Counting multiplicity over infinite alphabets. In *RP'09*, volume 5797 of *LNCS*, pages 141–153, 2009.

[23] E.W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM Journal of Computing*, 13(3):441–460, 1984.

[24] F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM ToCL*, 5(3):403–435, 2004.

[25] M. Niewerth and Th. Schwentick. Two-variable logic and key constraints on data words. In *ICDT'11*, pages 138–149. ACM, 2011.

[26] C. Rackoff. The covering and boundedness problems for vector addition systems. *TCS*, 6(2):223–231, 1978.

[27] L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL'06*, volume 4207 of *LNCS*, pages 41–57. Springer, 2006.

[28] P. Wolper. Temporal logic can be more expressive. *I&C*, 56:72–99, 1983.