Extending Temporal Logics with Data Variable Quantifications*

Fu Song[1] and Zhilin Wu[2,3]

1 **Shanghai Key Laboratory of Trustworthy Computing and National Trusted Embedded Software Engineering Technology Research Center, East China Normal University, P. R. China**
fsong@sei.ecnu.edu.cn

2 **State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, P. R. China**
wuzl@ios.ac.cn

3 **LIAFA, Université Paris Diderot, France**

Abstract

Although data values are available in almost every computer system, reasoning about them is a challenging task due to the huge data size or even infinite data domains. Temporal logics are the well-known specification formalisms for reactive and concurrent systems. Various extensions of temporal logics have been proposed to reason about data values, mostly in the last decade. Among them, one natural idea is to extend temporal logics with variable quantifications ranging over an infinite data domain. In this paper, we focus on the variable extensions of two widely used temporal logics, Linear Temporal Logic (LTL) and Computation Tree Logic (CTL). Grumberg, Kupferman and Sheinvald recently investigated the extension of LTL with variable quantifications. They defined the extension as formulas in the prenex normal form, that is, all the variable quantifications precede the LTL formulas. Our goal in this paper is to do a relatively complete investigation on this topic. For this purpose, we define the extensions of LTL and CTL by allowing arbitrary nestings of variable quantifications, Boolean and temporal operators (the resulting logics are called respectively variable-LTL, in brief VLTL, and variable-CTL, in brief VCTL), and identify the decidability frontiers of both the satisfiability and model checking problem. In particular, we obtain the following results: 1) Existential variable quantifiers or one single universal quantifier in the beginning already entails undecidability for the satisfiability problem of both VLTL and VCTL, 2) If only existential path quantifiers are used in VCTL, then the satisfiability problem is decidable, no matter which variable quantifiers are available. 3) For VLTL formulas with one single universal variable quantifier in the beginning, if the occurrences of the non-parameterized atomic propositions are guarded by the positive occurrences of the quantified variable, then its satisfiability problem becomes decidable. Based on these results of the satisfiability problem, we deduce the (un)decidability results of the model checking problem.

1998 ACM Subject Classification F.4.1 Mathematical Logic, F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases Temporal logics with variable quantifications, satisfiability and model checking, alternating register automata, data automata

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2014.253

* Fu Song is supported by the NSFC under Grant No. 61402179, STCSM Pujiang Talent Project under Grant No. 14PJ1403200, SHMEC&SHEDF ChenGuang Project under Grant No. 13CG21, the Open Project of Shanghai Key Laboratory of Trustworthy Computing under Grant No. 07dz22304201301, and Shanghai Knowledge Service Platform for Trustworthy Internet of Things under Grant No. ZF1213. Zhilin Wu is supported by the NSFC under Grant No. 61100062, 61272135, and 61472474, and partially supported by the visiting researcher program of China Scholarship Council.

34th Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2014).
Editors: Venkatesh Raman and S. P. Suresh; pp. 253–265

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Context. Data values are ubiquitous in computer systems. To see just the tip of the iceberg, we have the following scenarios: data variables in sequential programs, process identifiers in concurrent parameterized systems where an unbounded number of processes interact with each other, records in relational databases, attributes of elements of XML documents or nodes of graph databases. On the other hand, reasoning about data values is a very challenging task. Either their sizes are huge, e.g. one single 4-byte integer variable in C programs may take values from $-2,147,483,64$ to $2,147,483,647$, or they are even from an infinite domain, e.g. process identifiers in parameterized systems.

Temporal logics are the widely used specification formalisms. They were originally introduced to formally specify and reason about the behaviors of concurrent and reactive systems. But later on, they also became popular in reasoning about the behaviors of sequential programs. Linear temporal logic (LTL) and computation tree logic (CTL) are the two most widely used temporal logics. Since temporal logics are targeted to specify the behaviors of finite state systems, various extensions of temporal logics have been proposed to deal with the data values, mostly in the last decade. Hodkinson et al. initiated the investigation of first-order extensions of temporal logics, that is, first-order logic over relational structures extended with temporal operators ([14, 13]). Their motivation is theoretical and they focused on the decidability issues. Vianu and his coauthors used the first-order extensions of LTL to specify and reason about the behaviors of database-driven systems ([22, 3]). Demri and Lazic extended LTL with freeze quantifiers which can store data values into registers and compare the data values with those stored in the registers ([6]). Schwentick et al., Demri et al. , and Decker et al. considered extensions of LTL with navigation mechanisms for data values or tuples over multi-attributed data words, that is, data words where each position carries multiple data values ([16, 5, 4]).

Another natural idea is to extend temporal logics with variable quantifications over an infinite data domain, which is our focus in this paper. There have been some work on this topic. Grumberg et al. investigated the extension of LTL with variable quantifications. They defined the extension as the formulas in prenex normal form, that is, all the variable quantifications are in the beginning and are followed by LTL formulas, and investigated the decidability of the satisfiability and the model checking problem over Kripke structures extended with data variables ([10, 11]). Figueira proposed an extension of LTL with freeze quantifiers of only one register, by quantifications over the set of data values occurring before or after the current position of data words ([8]).

Contribution. Our goal in this paper is to do a relatively complete investigation on this topic. For this purpose, we consider the extensions of both LTL and CTL with variable quantifications (denoted by VLTL and VCTL) where the variable quantifiers can be nested arbitrarily with temporal and Boolean operators, and the quantifications are not over the set of data values occurring before or after the current position, but over the full data domain. In addition, we investigate the decidability and complexity of both the satisfiability and the model checking problem. More specifically, we obtain the following results.

1. Existential variable quantifiers or one single universal quantifier in the beginning already entails undecidability for the satisfiability problem of both VLTL and VCTL.
2. If the existential variable quantifiers are not nested, then the satisfiability problem becomes decidable for both VLTL and VCTL. The proof is obtained by a reduction to the nonemptiness problem of alternating one register automata ([8]).

Table 1 Summary of the results: U: Undecidable, D: Decidable, T: Theorem, C: Corollary.

	$\exists^*$-VLTL	$\forall^*$-VLTL	NN-$\exists^*$-VLTL	NN-$\forall^*$-VLTL	$\exists^*$-VLTL$_{pnf}$
SAT	U (T. 3)	U (T. 6)	D (T. 15)	U (T. 6)	D (T. 22, [10])
MC	U (C. 7)	U (C. 4)	U (C. 7)	D (C. 16)	U (C. 7)
	$\forall^*$-VLTL$_{pnf}$	$\exists$-VLTL$_{pnf}$	$\forall$-VLTL$_{pnf}$	$\exists$-VLTL$^{gdap}_{pnf}$	$\forall$-VLTL$^{gdap}_{pnf}$
SAT	U (T. 6)	D (T. 22, [10])	U (T. 6)	D (T. 22, [10])	D (T. 26)
MC	D (T. 22, [10])	U (C. 7)	D (T. 22, [10])	D (C. 27)	D (T. 22, [10])
	$\exists\forall$-VLTL$^{noap}_{pnf}$	$\forall\exists$-VLTL$^{noap}_{pnf}$	$\exists\exists$-VLTL$^{noap}_{pnf}$	$\forall\forall$–VLTL$^{noap}_{pnf}$	
SAT	U (T. 10)	U (T. 10)	D (T. 22, [10])	?	
MC	U (C. 12)	U (C. 12)	U (T. 9)	D (T. 22, [10])	
	$\exists^*$-VCTL	$\forall^*$-VCTL	NN-$\exists^*$-VCTL	NN-$\forall^*$-VCTL	$\exists^*$-VLTL$_{pnf}$
SAT	U (C. 5)	U (C. 8)	D (T. 17)	U (Cor. 8)	D (T. 23)
MC	U (T. 9)	U (T. 9)	U (T. 9)	D (Cor. 19)	U (T. 9)
	$\forall^*$-VCTL$_{pnf}$	$\exists$-VCTL$_{pnf}$	$\forall$-VCTL$_{pnf}$	$\exists$-VCTL$^{gdap}_{pnf}$	$\forall$-VCTL$^{gdap}_{pnf}$
SAT	U (C. 8)	D (T. 23)	U (C. 8)	D (T. 23)	?
MC	D (T. 24)	?	D (T. 24)	?	D (T. 24)
	$\exists\forall$-VCTL$^{noap}_{pnf}$	$\forall\exists$-VCTL$^{noap}_{pnf}$	$\exists\exists$-VCTL$^{noap}_{pnf}$	$\forall\forall$-VCTL$^{noap}_{pnf}$	EVCTL
SAT	U (C. 11)	U (C. 11)	D (T. 23)	?	D (T. 20)
MC	U (T. 9)	U (T. 9)	U (T. 9)	D (T. 24)	U (T. 9)

3. If only existential path quantifiers are used in VCTL, then the satisfiability problem is decidable (NEXPTIME), no matter whatever variable quantifiers are available. This result is proved by a small model property.
4. For the fragment of VLTL with one single universal variable quantifier in the beginning, if the occurrences of the non-parameterized atomic propositions are guarded by the positive occurrences of the universally quantified variable, then the satisfiability problem becomes decidable. The proof is obtained by a reduction to the nonemptiness of extended data automata ([1]). This decidability result is tight in the sense that adding one more existential variable quantifier before or after the universal one implies undecidability.
5. Based on the above results of the satisfiability problem, we also deduce the (un)decidability results of the model checking problem.

The results obtained in this paper are summarized into Table 1 (where $\exists, \forall$ mean existential and universal variable quantifier, pnf means prenex normal form, NN means non-nested, the question mark means that the decidability is open). The reader can refer to Section 2 for the definitions of the fragments of VLTL and VCTL.

Related Work. Emerson and Namjoshi proposed indexed CTL$^*\backslash X$ to specify and reason about parameterized systems ([7]). The indexed CTL$^*\backslash X$ formulas they considered are in prenex normal form and their main goal is to prove the "cutoff" results. Song and Touili considered some extensions of LTL and CTL to detect the malware where the variable quantifications can be nested arbitrarily with the other operators, but the variables range over a finite domain ([20, 19]). The temporal logics extended with variable quantifications ranging over an infinite data domain are the formalisms over infinite alphabets. Researchers have proposed many such formalisms. To cite a few, nondeterministic register automata ([15]), first-order logic with two variables ([2]), XPath with data value comparisons ([9]).

Very recently, independently of this work, Sheinvald et al. considered the characterization of simulation pre-order in VCTL* ([18]). VCTL* extends CTL* by unrestricted data variable quantifications, thus subsumes VLTL and VCTL considered in this paper. But they have not investigated the satisfiability and model checking problem for VCTL* yet.

The rest of this paper is organized as follows: Preliminaries are given in the next section, Section 3 and 4 present respectively the undecidability and decidability results. All the missing proofs will appear in the journal version of this paper.

2 Preliminaries

Let D be an infinite set of data values, AP a finite set of (non-parameterized) atomic propositions, and T with $AP \cap T = \emptyset$ a finite set of parameterized atomic propositions, where each of them carries one parameter (data value). Let Var be a countable set of data variables which range over D. Let $[k]$ denote the set $\{0, \ldots, k-1\}$, for all $k \in \mathbb{N}$.

A *word* (resp. *data word*) w over AP (resp. $AP \cup T$) is a finite sequence from $(2^{AP})^*$ (resp. $(2^{AP \cup T \times D})^*$). Given $k \geq 1$, a k-ary *tree* is a set $Z \subseteq [k]^*$ s. t. for all $zi \in Z$: $z \in Z$ and $zj \in Z$ for all $j \in [i]$. The node ϵ is called the *root* of the tree. For every $z \in Z$, the nodes $zi \in Z$ for $i \in [k]$ are called the *successors* of z, denoted by $suc(z)$. Let $Leaves(Z)$ denote the set of leaves of Z. A *path* π of a tree Z is a set $\pi \subseteq Z$ s. t. $\epsilon \in \pi$ and $\forall z \in \pi$, either z is a leaf, or there is a unique $i \in [k]$ s. t. $zi \in \pi$. A k-ary *labeled tree* (resp. *data tree*) t over AP (resp. $AP \cup T$) is a tuple (Z, L), where Z is a k-ary tree and $L : Z \to 2^{AP}$ (resp. $L : Z \to 2^{AP \cup T \times D}$) is the labeling function. Given a labeled or data tree $t = (Z, L)$, let $z \in Z$ and π be a path of t, then t_z denotes the labeled or data subtree t rooted at z, and w_π denotes the word or data word on the path π of t. For $z \in Z$ in a tree or data tree $t = (Z, L)$, define the *tree type* of z in t, denoted by $type_t(z)$, as the set $\{l_0, \ldots, l_{k-1}\}$ s. t. for every $j \in [k]$, if $zj \in Z$, then $l_j = \nabla_j$, otherwise $l_j = \overline{\nabla_j}$($\nabla_j$ means the j-th child exists).

A *variable Kripke structure*[1] (VKS) $\mathcal{K}$ is a tuple $(AP \cup T, X, S, R, S_0, I, L, L')$, where AP and T are defined as above, X and S are finite sets of variables and states respectively, $R \subseteq S \times S$ is the set of edges, $S_0 \subseteq S$ is the set of initial states, I is the invariant function that assigns to each state a formula which is a positive Boolean combination of $x_i = x_j$ and $x_i \neq x_j$ for $x_i, x_j \in X$, $L : S \to 2^{AP \cup T \times X}$ is the state labeling function, $L' : R \to 2^{\{reset\} \times X}$ is the edge labeling function. Intuitively, if $(reset, x) \in L'((s, s'))$, then the value of the variable x is reset (to any value) when going from s to s'.

The set of (finite) computation traces and computation trees can be defined similar to Kripke structures. The difference is that the data values are added to positions or nodes of computation traces or trees, while respecting the state invariants and the edge labelings. Let $\mathcal{L}(\mathcal{K})$ and $\mathcal{T}(\mathcal{K})$ denote the set of computation traces and computation trees of $\mathcal{K}$ respectively.

For any VKS $\mathcal{K}$ with the set of variables X, it holds that in every computation trace w or computation tree t of $\mathcal{K}$, the number of data values occurring in each position of w or each node of t is at most $|X|$. Since we are interested in reasoning about the computations of variable Kripke structures, we restrict our attention in this paper to the language of data words (or data trees) s. t. there is a bound K satisfying that each position or node of data words or data trees carries at most K data values.

For the convenience of discussions, we represent data words with at most K data values in each position as K-attributed data words from $\left(2^{AP} \times (2^T \times D)^K\right)^*$ (If the number of data

[1] Variable Kripke structure defined here is the same as that in [10], except that the global invariants are replaced by local state invariants.

values at some position is less than K, then we just copy them). For a K-attributed data word $w = (A_0, ((B_{0,0}, d_{0,0}), \ldots, (B_{0,K-1}, d_{0,K-1}))) \ldots (A_n, ((B_{n,0}, d_{n,0}), \ldots, (B_{n,K-1}, d_{n,K-1})))$, let $prj(w)$ denote the sequence $(A_0, (B_{0,0}, \ldots, B_{0,K-1})) \ldots (A_n, (B_{n,0}, \ldots, B_{n,K-1}))$. Similarly, we represent data trees with at most K data values in each node as K-attributed data trees. From now on, when we say data words or data trees, we always mean K-attributed data words or data trees. Let $\eta = (A, ((B_1, d_1), \ldots, (B_K, d_K))) \in 2^{AP} \times (2^T \times D)^K$, $p \in AP$ and $(\tau, d) \in T \times D$. By abuse of notations, we use $p \in \eta$ to mean $p \in A$, and use $(\tau, d) \in \eta$ to denote $(\tau, d) \in \cup_{1 \leq i \leq K} B_i \times \{d_i\}$.

The syntax of *Variable-LTL* (VLTL)[2] is defined by the following rules,

$$\varphi := p \mid \neg p \mid \tau(x) \mid \neg\tau(x) \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid X\varphi \mid \overline{X}\varphi \mid \varphi U \varphi \mid \varphi R \varphi \mid \exists x \varphi \mid \forall x \varphi,$$

where $p \in AP, \tau \in T, x \in Var$, and $\overline{X}$ is the dual operator of X such that $\overline{X}\varphi \equiv \neg X \neg \varphi$.

Let $var(\varphi)$ and $free(\varphi)$ denote respectively the set of variables occurring in φ and the set of free variables of φ. VLTL formulas without free variable are called *sentences*.

VLTL formulas are interpreted over K-attributed data words. Let $w = w_0 \ldots w_n \in \left(2^{AP} \times (2^T \times D)^K\right)^*$, φ a VLTL formula, $\lambda : free(\varphi) \to D$, and for every $i : 0 \leq i \leq n$, $w^i = w_i \ldots w_n$. We define the satisfaction relation as $w \models_\lambda \varphi$ in an obvious way. In particular, $w \models_\lambda \forall x \varphi_1$ if for all $d \in D$, $w \models_{\lambda[d/x]} \varphi_1$, where $\lambda[d/x]$ is the same as λ, except assigning d to x; $w \models_\lambda \exists x \varphi_1$ if there is $d \in D$ s. t. $w \models_{\lambda[d/x]} \varphi_1$. If φ is a VLTL sentence, we will drop λ from $\models_\lambda$. Let $\mathcal{L}_K(\varphi)$ denote the set of K-attributed data words w satisfying φ.

Let $\mathcal{K}$ be a VKS and φ a VLTL sentence. Then $\mathcal{K}$ satisfies φ, denoted by $\mathcal{K} \models \varphi$, if for every computation trace w of $\mathcal{K}$, $w \models \varphi$.

Let $\exists^*$-VLTL (resp. $\forall^*$-VLTL) denote the set of VLTL formulas without using $\forall$ (resp. $\exists$) quantifier, NN-$\exists^*$-VLTL denote the set of $\exists^*$-VLTL formulas where no $\exists$ quantifiers are nested (in a strict sense), more precisely, for any pair of subformulas $\exists x \psi_1$ and $\exists y \psi_2$ s. t. $x \neq y$, if $\exists y \psi_2$ is a subformula of ψ_1, then $x \notin free(\psi_2)$. NN-$\forall^*$-VLTL is defined similarly. Let VLTL_{pnf} denote the set of VLTL formulas in prenex normal form $\theta_1 x_1 \ldots \theta_k x_k \ \psi$, where $\theta_1, \ldots, \theta_k \in \{\exists, \forall\}$, and ψ is a quantifier-free VLTL formula. Suppose $\theta = \theta_1 \ldots \theta_k \in \{\forall, \exists\}^*$, let θ-VLTL_{pnf} denote the set of VLTL_{pnf} formulas of the form $\theta_1 x_1 \ldots \theta_k x_k \psi$. Note that in general VLTL formulas cannot be turned into equivalent prenex normal forms[3]. Suppose $\Theta \subseteq \{\forall, \exists\}^*$, let Θ-$\text{VLTL}_{pnf} = \cup_{\theta \in \Theta} \theta$-$\text{VLTL}_{pnf}$. For $\theta \in \{\exists, \forall\}^*$ (resp. $\Theta \subseteq \{\exists, \forall\}^*$), let θ-VLTL_{pnf}^{noap} (resp. Θ-VLTL_{pnf}^{noap}) denote the set of θ-VLTL_{pnf} (resp. Θ-VLTL_{pnf}) formulas without using atomic propositions from AP. Let $\theta \in \{\exists, \forall\}$. Then θ-VLTL_{pnf}^{gdap} denote the set of θ-VLTL_{pnf} formulas where each occurrence of the atomic propositions from AP is *guarded* by the positive occurrences of the (unique) quantified variable. More specifically, θ-VLTL_{pnf}^{gdap} is the set of θ-VLTL_{pnf} formulas $\theta x \psi$ s. t. for every $p \in AP$, each occurrence of p (resp. $\neg p$) in ψ is in the formula $p \wedge \tau(x)$ (resp. $\neg p \wedge \tau(x)$) for some $\tau \in T$. For example, $\forall x(open(x) \to close(x))$ is a $\forall$-VLTL_{pnf}^{gdap} formula, while $\forall x(open(x) \to (p \wedge \neg write(x)) \ U \ close(x))$ is not.

The syntax of variable-CTL (VCTL) is defined similarly to VLTL, by adding the path quantifiers A and E before every temporal operator in the syntax rules of VLTL.

VCTL formulas are interpreted over K-attributed data trees. The semantics of VCTL are defined similarly to VLTL. The syntactic fragments of VCTL can also be defined similarly to

[2] VLTL defined in [10] is a fragment of VLTL defined here. In addition, we disallow explicit data variable comparisons (e.g. equality and inequality). We believe that VLTL without any data variable comparison is a kind of first-order extensions of temporal logics of the minimum first-order feature.

[3] For instance, we conjecture that there are no VLTL_{pnf} formulas equivalent to the VLTL formula $G(\exists x \tau(x))$. But at present we do not know how to prove this.

VLTL, with the additional distinction between EVCTL and AVCTL, that is, the fragment of VCTL using only E and A respectively. Let $\mathcal{L}_K(\varphi)$ denote the set of K-attributed data trees t satisfying φ.

Let $\mathcal{K}$ be a VKS and φ be a VCTL sentence. Then $\mathcal{K}$ satisfies φ, denoted by $\mathcal{K} \models \varphi$, if for every computation tree t of $\mathcal{K}$, $t \models \varphi$.

For a VLTL or VCTL formula φ, let $\overline{\varphi}$ denote the negation of φ, where $\overline{p} = \neg p$, $\overline{\neg p} = p$, $\overline{\tau(x)} = \neg\tau(x)$, $\overline{\neg\tau(x)} = \tau(x)$, $\overline{X\varphi_1} = \overline{X}\overline{\varphi_1}$, $\overline{A\varphi_1} = E\overline{\varphi_1}$, and so on. Let $|\varphi|$ denote the size of φ, that is, the number of symbols in φ. We will use $\varphi_1 \to \varphi_2$ to mean $\overline{\varphi_1} \vee \varphi_2$. A VLTL or VCTL formula that does not contain subformulas of the form $\psi_1 \to \psi_2$ is called normalized.

We consider the following two decision problems for VLTL and VCTL.

- Satisfiability problem: Given a VLTL (resp. VCTL) sentence φ, decide whether φ is satisfiable, that is, whether there is a data word w (resp. there are $k \geq 1$ and a k-ary data tree t) s. t. $w \models \varphi$ (resp. $t \models \varphi$).
- Model checking problem: Given a VKS $\mathcal{K}$ and a VLTL/VCTL sentence φ, decide whether $\mathcal{K} \models \varphi$.

▶ Remark. We interpret VLTL and VCTL formulas over finite data words and finite data trees, and leave the investigations on infinite data words and infinite data trees as future work. The considerations of temporal logics interpreted over finite words and trees are normally motivated by the verification of safety properties of concurrent systems (cf. e.g. [17]) as well as the verification of properties of sequential programs.

We next define alternating register automata over k-ary 1-attributed data trees by adapting the definition of alternating register automata over unranked data trees in [8].

An *alternating register automaton* over k-ary 1-attributed data trees (ATRA) is a tuple $\mathcal{A} = (AP \cup T, Q, q_0, \delta)$, where AP (resp. T) is a finite set of atomic propositions (resp. parameterized atomic propositions), Q is a finite set of states, $q_0 \in Q$ is the initial state, $\delta : Q \to \Phi$ is the transition function, where Φ is defined by the following grammar[4],

$$p \mid \neg p \mid \tau \mid \neg\tau \mid \nabla_i? \mid \overline{\nabla_i}? \mid eq \mid \overline{eq} \mid q \vee q' \mid q \wedge q' \mid store(q) \mid guess(q) \mid \nabla_i q,$$

where $p \in AP$, $\tau \in T$, $q, q' \in Q$, and $i \in [k]$. Intuitively, $p, \neg p, \tau, \neg\tau$ are used to detect the occurrences of (parameterized) atomic propositions. $\nabla_i?, \overline{\nabla_i}?$ are used to describe the types of nodes in trees, $eq, \overline{eq}$ are used to check whether the data value in the register is equal to the current one, $q \vee q'$ makes a nondeterministic choice, $q \wedge q'$ creates two threads with the state q and q' respectively, $store(q)$ stores the current data value to the register and transfers to the state q, $guess(q)$ guesses a data value for the register and transfers to the state q, $\nabla_i q$ moves to the i-th child of the current node and transfers to the state q.

An ATRA $\mathcal{A} = (AP \cup T, Q, q_0, \delta)$ is called *alternating register automaton* over 1-attributed data words (AWRA) if $k = 1$.

The semantics of ATRA over k-ary 1-attributed data trees are defined in a completely analogous way as those of ATRA over unranked trees in [8]. Let $\mathcal{L}(\mathcal{A})$ denote the set of all k-ary 1-attributed data trees accepted by $\mathcal{A}$.

The closure properties of ATRA and the decidability of the nonemptiness of ATRA can be proved in the same way as the alternating register automata over unranked trees, by utilizing well-structured transition systems (cf. [8]).

▶ **Theorem 1.** *ATRAs are closed under intersection and union. The emptiness problem of ATRAs is decidable and non-primitive recursive.*

[4] The *spread* mechanism in [8] is dropped, since it is not used in this paper.

We also introduce extended data automata ([1]), another automata model over (1-attributed) data words. Extended data automata is an extension of the seminal model of data automata ([2]) over 1-attributed data words.

An *extended data automaton* (EDA) $\mathcal{D}$ is a tuple $(AP \cup T, \mathcal{A}, \mathcal{B})$ s. t. AP and T are as above, $\mathcal{A}$ is a nondeterministic letter-to-letter transducer from the alphabet $2^{AP} \times 2^T$ to some output alphabet Σ, and $\mathcal{B}$ is a finite automaton over $\Sigma \cup \{0\}$ (where $0 \notin \Sigma$). Let $w = (A_0, (B_0, d_0)) \dots (A_n, (B_n, d_n))$ be a 1-attributed data word. Then w is accepted by $\mathcal{D}$ if over $prj(w)$, $\mathcal{A}$ outputs a word $w' = \sigma_0 \dots \sigma_n$ over the alphabet Σ, s. t. for every data value $d \in D$, $cstr_d(w'')$ is accepted by $\mathcal{B}$, where $w'' = (\sigma_0, d_0) \dots (\sigma_n, d_n)$ and $cstr_d(w'')$ is defined as $\sigma'_0 \dots \sigma'_n$, satisfying that for every $i : 0 \leq i \leq n$, $\sigma'_i = \sigma_i$ if $d_i = d$, and $\sigma'_i = 0$ otherwise. Note that for every data value d not occurring in w, $cstr_d(w'') = 0^{n+1}$.

▶ **Theorem 2** ([2, 1]). *EDAs are closed under intersection and union. The nonemptiness problem of EDAs is decidable.*

3 Undecidability

This section is devoted to the various undecidability results for the satisfiability and model checking problem of VLTL and VCTL.

▶ **Theorem 3.** *The satisfiability problem of $\exists^*$-VLTL is undecidable.*

Proof sketch. We show this by a reduction from the PCP problem. We illustrate the proof by considering the special case $K = 1$. Let $(u_i, v_i)_{1 \leq i \leq n}$ be an instance of the PCP problem over an alphabet Σ. A solution of the PCP problem is a sequence of indices $i_1 \dots i_m$ s. t. $u_{i_1} \dots u_{i_m} = v_{i_1} \dots v_{i_m}$.

Let $\Sigma' = \Sigma \cup \{\overline{a} \mid a \in \Sigma\} \cup \{1, \dots, n\} \cup \{\overline{1}, \dots, \overline{n}\} \cup \{\#\}$. Then Σ' can be encoded by $\lceil \log(|\Sigma'|) \rceil$ bits. Let AP be a set of atomic propositions of size $\lceil \log(|\Sigma'|) \rceil$. For each $\sigma \in \Sigma'$, let $atom(\sigma)$ denote the subset of AP corresponding to the binary encoding of σ, and $type(\sigma)$ denote the conjunction of atomic propositions or negated atomic propositions from AP corresponding to the binary encoding of σ. For instance, if the binary encoding of σ is 10, let $AP = \{p_1, p_2\}$, then $atom(\sigma) = \{p_1\}$ and $type(\sigma) = p_1 \wedge \neg p_2$. The definition of $atom(\sigma)$ can be naturally extended to $atom(u)$ for words $u \in (\Sigma')^+$. In addition, let $T = \{\tau\}$. We use $atom'(\sigma)$ to denote $(atom(\sigma), \{\tau\})$. Similarly, the definition $atom'(\cdot)$ can be extended to words $u \in (\Sigma')^+$.

We intend to encode a solution of the PCP problem, say $i_1 \dots i_m$, as the 1-attributed data word over $2^{AP} \times 2^T$ of the form $w_{i_1} w_{i_2} \dots w_{i_m} (atom'(\#), d)\ \overline{w_{i_1}} \dots \overline{w_{i_m}}$ s. t.

- $prj(w_{i_j}) = atom'(i_j) atom'(u_{i_j})$ and $prj(\overline{w_{i_j}}) = atom'(\overline{i_j}) atom'(\overline{u_{i_j}})$ for every $1 \leq j \leq m$,
- the two sequences of data values corresponding to respectively $atom'(i_1) \dots atom'(i_m)$ and $atom'(\overline{i_1}) \dots atom'(\overline{i_m})$ are the same,
- the two sequences of data values corresponding to respectively $atom'(u_{i_1}) \dots atom'(u_{i_m})$ and $atom'(\overline{u_{i_1}}) \dots atom'(\overline{u_{i_m}})$ are the same.

The data word encodings of the solutions of the PCP problem can be expressed by a $\exists^*$-VLTL formula φ. For instance, during the reduction, we express as the following $\exists^*$-VLTL formula φ_1 the fact that for every two consecutive occurrences of letters from $\{atom'(\sigma) \mid \sigma \in \Sigma\}$, there are two consecutive occurrences of letters from $\{atom'(\overline{\sigma}) \mid \sigma \in \Sigma\}$ with the same letters (by viewing $atom'(\sigma)$ the same as $atom'(\overline{\sigma})$) and the same data values, $\varphi_1 = G \wedge_{\sigma_1, \sigma_2 \in \Sigma} [\psi_0 \rightarrow \exists x \exists y (\psi_1 \wedge X \psi_2 \wedge F(\psi_3 \wedge X \psi_4))]$, where $\psi_0 = type(\sigma_1) \wedge X[type(\sigma_2) \vee \vee_{1 \leq j \leq n} (type(j) \wedge X type(\sigma_2)))]$, $\psi_1 = type(\sigma_1) \wedge \tau(x)$, $\psi_2 = (type(\sigma_2) \wedge \tau(y)) \vee$

$\vee_{1\le j\le n}[type(j) \wedge X(type(\sigma_2) \wedge \tau(y))]$, $\psi_3 = type(\overline{\sigma_1}) \wedge \tau(x)$, $\psi_4 = (type(\overline{\sigma_2}) \wedge \tau(y)) \vee \vee_{1\le j\le n}[type(\bar{j}) \wedge X(type(\overline{\sigma_2}) \wedge \tau(y))]$. ◀

Since a VKS can be defined to accept the set of all 1-attributed data words, we deduce the following result.

▶ **Corollary 4.** *The model checking problem of $\forall^*$-VLTL is undecidable.*

By adding a universal path quantifier A before every temporal operator of φ in the proof of Theorem 3, we get a reduction to the satisfiability problem of $\exists^*$-AVCTL.

▶ **Corollary 5.** *The satisfiability problem of $\exists^*$-AVCTL formulas is undecidable.*

By a similar reduction from the nonemptiness of two-counter machines as that in Theorem 4.1 of [8], we can show the following result.

▶ **Theorem 6.** *The satisfiability problem of $\forall$-$VLTL_{pnf}$ is undecidable.*

▶ Remark. The above theorem demonstrates that the claim in [11] that the satisfiability problem of $\exists^*\forall$-VLTL_{pnf} is decidable is incorrect. We confirmed this observation in an e-mail with Grumberg, Kupferman and Sheinvald ([12]). On the other hand, we will show that the satisfiability problem of $\forall$-VLTL^{gdap}_{pnf}, that is, $\forall$-VLTL_{pnf} formulas where all the occurrences of (negated) atomic propositions are guarded, is decidable (cf. Theorem 26).

▶ **Corollary 7.** *The model checking problem of $\exists$-$VLTL_{pnf}$ is undecidable.*

▶ **Corollary 8.** *The satisfiability problem of $\forall$-$AVCTL_{pnf}$ is undecidable.*

▶ Remark. From the undecidability result of the satisfiability of a fragment of VCTL, we do not immediately deduce the undecidability of the model checking problem of the dual fragment. The reason is that there does not exist a VKS to define the set of all K-attributed computation trees, or define the set of k-ary K-attributed data trees even for a fixed k.

▶ **Theorem 9.** *The model checking problem is undecidable for the following fragments: $\forall^*$-$AVCTL$, $\forall^*$-$EVCTL$, $\exists\exists$-$VCTL^{noap}_{pnf}$, $\exists\forall$-$VCTL^{noap}_{pnf}$, $\forall\exists$-$VCTL^{noap}_{pnf}$, NN-$\exists^*$-$VCTL$.*

Proof sketch. We prove the theorem by reductions from the satisfiability problem of $\exists^*$-VLTL and $\forall$-VLTL over 1-attributed data words.

We illustrate the proof by considering the model checking problem of $\forall^*$-AVCTL and $\exists\exists$-VCTL^{noap}_{pnf}. The arguments for the other cases use the similar idea.

We first consider the model checking problem of $\forall^*$-AVCTL.

Let φ be a $\exists^*$-VLTL formula over $AP \cup T$. We will construct a VKS $\mathcal{K}$ and a $\exists^*$-EVCTL formula φ' s. t. φ is satisfiable iff $\mathcal{K} \not\models \overline{\varphi'}$. Note that $\overline{\varphi'}$ is a $\forall^*$-AVCTL formula.

The idea of the reduction is as follows: We construct a VKS $\mathcal{K}$ which is a single loop (without branchings). Thus each computation tree of $\mathcal{K}$ is in fact a data word. Then we obtain from φ by adding existential path quantifiers E before every temporal operator occurring in φ (plus some other modifications) to obtain φ'. Since $\mathcal{K}$ is a structure without branchings, the satisfaction of φ' over the computation trees of $\mathcal{K}$ mimics the satisfaction of φ over data words.

Suppose $AP = \{p_1, \ldots, p_m\}$, $T = \{\tau_1, \ldots, \tau_{n-m}\}$ (where $m \le n$), and $\tau'_0, \tau'_1 \notin AP \cup T$. Define the VKS $\mathcal{K} = (AP' \cup T', \{x\}, S, R, S_0, I, L, L')$ as follows: $AP' = \emptyset$, $T' = \{\tau'_0, \tau'_1\}$, $S = \{s_0, s_1, \ldots, s_{2n+1}\}$, $R = \{(s_i, s_{i+1 \bmod 2n+2}) \mid 0 \le i \le 2n+1\}$, $S_0 = \{s_0\}$, for every $s_i \in S$, $I(s_i) = true$, $L(s_0) = \{(\tau'_0, x)\}$, and $L(s_i) = \{(\tau'_1, x)\}$ for every $i : 1 \le i \le 2n+1$, $L'((s_i, s_{i+1 \bmod 2n+2})) = \{(reset, x)\}$ for every $i : 0 \le i \le 2n+1$.

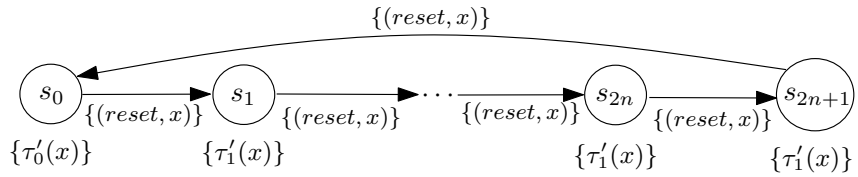

Figure 1 The Variable Kripke Structure.

Notice that in $\mathcal{K}$, T' only contains two parameterized atomic propositions. The set of atomic propositions $AP \cup T$ will be encoded by equalities and inequalities between the data values of two adjacent τ'_1-labeled positions in $\mathcal{K}$. Thus each position (A, B, d) in a 1-attributed data word over $AP \cup T$ will be encoded by a segment of computation traces in $\mathcal{K}$ of length $2n+2$ s. t. the position 0 is labeled by τ'_0, the position $(2i-1)$ and $(2i)$ encode the satisfaction on $A \cup B$ of the i-th atomic proposition from $AP \cup T$, and the last position has the data value d. In addition, x is reset on each edge $(s_i, s_{i+1 \bmod 2n+2})$ $(0 \leq i \leq 2n+1)$ so that an arbitrary data value can be assigned to x on each position $s_0, s_1, \ldots, s_{2n+1}$.

We use an example to illustrate the construction of the $\exists^*$-EVCTL formula φ' from φ. Suppose $AP = \emptyset$ and $T = \{\tau_1, \tau_2\}$ and $n = 2$. Then the $\exists^*$-EVCTL formula corresponding to the $\exists^*$-VLTL formula $\exists x G(\neg\tau_1(x) \vee XF\tau_2(x))$ is $\exists y[\tau'_0(y) \wedge \varphi'_0 \wedge \exists x EG(\psi_1 \vee (EX)^6 EF\psi_2)]$, where $\varphi'_0 = EG(\tau'_0(y) \rightarrow [(E\overline{X})^6\tau'_0(y) \wedge EX\tau'_1(y) \wedge (EX)^3\tau'_1(y)])$ requires that the data value of the position 0 occurs in all the positions i s. t. $i \equiv 0, 1, 3 \bmod 6$, $\psi_1 = \tau'_0(y) \rightarrow [EX(\tau'_1(y) \wedge EX\neg\tau'_1(y))) \vee (EX)^5\neg\tau'_1(x)]$, and $\psi_2 = \tau'_0(y) \wedge (EX)^3(\tau'_1(y) \wedge EX\tau'_1(y)) \wedge (EX)^5\tau'_1(x)$. The formula ψ_1 is satisfied in a position if either $\tau'_0(y)$ does not occur, or $\tau'_0(y)$ occurs and one of the following conditions holds: the next two positions have different data values, or $\tau'_1(x)$ does not occur in the 5th position after the current position. Similarly for ψ_2.

For the model checking problem of $\exists\exists$-VCTL$^{noap}_{pnf}$, we reduce from the satisfiability problem of $\forall$-VLTL over 1-attributed data words. The reduction is similar to that of $\forall^*$-AVCTL. ◀

By using the similar idea as in the proof of Theorem 9, we can get the following result.

▶ **Theorem 10.** *The satisfiability problem of $\exists\forall$-VLTL$^{noap}_{pnf}$ (resp. $\forall\exists$-VLTL$^{noap}_{pnf}$) is undecidable.*

▶ **Corollary 11.** *The satisfiability problem of $\exists\forall$-VCTL$^{noap}_{pnf}$ (resp. $\forall\exists$-VCTL$^{noap}_{pnf}$) is undecidable.*

▶ **Corollary 12.** *The model checking problem of $\exists\forall$-VLTL$^{noap}_{pnf}$ (resp. $\forall\exists$-VLTL$^{noap}_{pnf}$) is undecidable.*

4 Decidability

We first present the encodings of K-attributed data words and data trees into 1-attributed ones, which will be used in the proofs of this section.

Suppose that $w = w_0 \ldots w_n$ is a K-attributed data word over $AP \cup T$ s. t. for every $i : 0 \leq i \leq n$, $w_i = (A_i, ((B_{i,0}, d_{i,0}), \ldots, (B_{i,K-1}, d_{i,K-1})))$. Let $p' \notin AP \cup T$ and $AP' = AP \cup \{p'\}$. A 1-*attributed encoding* of w, denoted by $enc(w)$, is a data word $w' = w'_{0,0} \ldots w'_{0,K-1} \ldots w'_{n,0} \ldots w'_{n,K-1}$ over $AP' \cup T$ s. t. for every $i : 0 \leq i \leq n$, $w'_{i,0} = (A_i \cup \{p'\}, (B_{i,0}, d_{i,0}))$, and for every $j : 1 \leq j \leq K-1$, $w'_{i,j} = (A_i, (B_{i,j}, d_{i,j}))$. The 1-attributed encoding of K-attributed data trees can be defined similarly. The definition of $enc(\cdot)$ can be naturally extended to the languages of K-attributed data words and data trees.

Suppose φ is a normalized VLTL formula. Then $enc(\varphi) = \varphi'_1 \wedge \varphi'_2$, where φ'_1 and φ'_2 are defined as follows.

- φ'_1 is a quantifier free VLTL formula enforcing the following constraints: p' occurs in the first position, for every occurrence of p' in some position, p' will occur in the K-th position after it if there is such a position, but does not occur in between, moreover, for every $p \in AP$, either p occurs in all the positions between two adjacent occurrences of p', or occurs in none of them.
- φ'_2 is obtained from φ by replacing X(resp. $\overline{X}$) by X^K(resp. $\overline{X}^K$), and applying some proper replacements for the (parameterized) atomic propositions. For instance, the occurrence of p_1 in Fp_1 is replaced by $p' \wedge \vee_{0 \le i \le K-1} X^i p_1$.

Here is an example for $enc(\varphi)$: $enc(Fp_1) = \varphi'_1 \wedge F(p' \wedge \vee_{0 \le i \le K-1} X^i p_1)$.

Similarly, $enc(\varphi)$ can be defined for VCTL formulas.

▶ **Proposition 13.** *For every VLTL (resp. VCTL) formula φ, $enc(\mathcal{L}_K(\varphi)) = \mathcal{L}_1(enc(\varphi))$.*

4.1 Non-nested existential variable quantifiers

▶ **Proposition 14.** *Let $\mathcal{K}$ be a VKS . Then $enc(\mathcal{L}(\mathcal{K}))$ (resp. $enc(\mathcal{T}(\mathcal{K}))$) can be defined by an AWRA (resp. ATRA).*

▶ **Theorem 15.** *The satisfiability problem of NN-$\exists^*$-VLTL is decidable and non-primitive recursive.*

Proof sketch. The decidability proof is by a reduction to the nonemptiness problem of AWRAs. Since quantifiers are not nested, w.l.o.g. we assume that there is only one variable, say x, used in φ. Note that the variable x may be reused and existentially quantified for many times. The AWRA $\mathcal{A}_{enc(\varphi)}$ can be constructed by induction on the syntax of NN-$\exists^*$-VLTL formulas similar to the construction of alternating automata from LTL formulas (cf. [21]), by using $guess(q)$ to deal with the existential quantifiers $\exists x$.

The lower bound is obtained by a reduction from the nonemptiness problem of two-counter machines with incrementing errors (cf. e.g. [6]). The reduction is similar to that in Theorem 6, with all the four conditions, except the last one, expressed in NN-$\exists^*$-VLTL. ◀

▶ **Corollary 16.** *The model checking problem of NN-$\forall^*$-VLTL is decidable and non-primitive recursive.*

▶ **Theorem 17.** *The satisfiability problem of NN-$\exists^*$-VCTL is decidable and non-primitive recursive.*

Theorem 17 is proved in the same way as Theorem 15, by utilizing the following result.

▶ **Lemma 18.** *For every $\exists^*$-VCTL sentence φ, if φ is satisfiable over a 1-attributed data tree, then there is a $(2|\varphi|)$-ary 1-attributed data tree satisfying φ.*

Similar to Corollary 16, we deduce the following result from Theorem 17.

▶ **Corollary 19.** *The model checking problem of NN-$\forall^*$-VCTL formulas is decidable and non-primitive recursive.*

4.2 Existential path quantifiers for VCTL

▶ **Theorem 20.** *The satisfiability problem of EVCTL is in NEXPTIME.*

Theorem 20 can be deduced from the following lemma.

▶ **Lemma 21.** *Let φ be an EVCTL formula, t be a data tree, and $\lambda : free(\varphi) \to D$ s. t. $t \models_\lambda \varphi$. Then a data tree t' can be constructed from (t, λ) s. t. $t' \models_\lambda \varphi$ and (t', λ) contains at most $|\varphi|$ data values.*

Proof sketch. The proof is by induction on the syntax of EVCTL formulas. The induction base $\varphi := p, \neg p, \tau(x), \neg\tau(x)$ is trivial. For induction step, we show the cases $\exists x\varphi_1$ and $\forall x\varphi_1$.

$\varphi := \exists x\varphi_1$: Suppose $t \models_\lambda \exists x\varphi_1$. Then there is $d \in D$ s. t. $t \models_{\lambda[d/x]} \varphi_1$. By the induction hypothesis, t_1 can be constructed from $(t, \lambda[d/x])$ s. t. $t_1 \models_{\lambda[d/x]} \varphi_1$ and $(t_1, \lambda[d/x])$ contains at most $|\varphi_1|$ data values. Then (t_1, λ) is the desired pair.

$\varphi := \forall x\varphi_1$: Suppose $t \models_\lambda \forall x\varphi_1$. Then for every $d \in D$, $t \models_{\lambda[d/x]} \varphi_1$. Suppose the range of λ is $\{d_1, \ldots, d_k\}$. Let d_0 be a data value not occurring in t. In addition, if $D(t) \setminus \{d_1, \ldots, d_k\}$ contains at least $|\varphi_1| - k$ data values, let $d_{k+1}, \ldots, d_{|\varphi_1|}$ be a sequence of $|\varphi_1| - k$ distinct data values from $D(t) \setminus \{d_0, d_1, \ldots, d_k\}$; otherwise let $d_{k+1}, \ldots, d_{|\varphi_1|}$ be a sequence of $|\varphi_1| - k$ distinct data values from $D \setminus \{d_0, d_1, \ldots, d_k\}$ that include all the data values in $D(t) \setminus \{d_0, \ldots, d_k\}$. Then from $t \models_{\lambda[d_i/x]} \varphi_1$ (where $i = 0, \ldots, |\varphi_1|$) and the induction hypothesis, we know that t_i can be constructed from $(t, \lambda[d_i/x])$ s. t. $t_i \models_{\lambda[d_i/x]} \varphi_1$, $(t_i, \lambda[d_i/x])$ contains at most $|\varphi_1|$ data values. Since for every $i : 0 \le i \le |\varphi_1|$, t_i contains at most $|\varphi_1|$ data values, we could replace the data values of t_i that are from $D \setminus \{d_0, \ldots, d_{|\varphi_1|}\}$ by data values in $\{d_1, \ldots, d_{|\varphi_1|}\}$, to get t'_i s. t. all the data values of t'_i are from $\{d_1, \ldots, d_{|\varphi_1|}\}$ and $t'_i \models_{\lambda[d_i/x]} \varphi_1$. Note that d_0 does not occur in any of $t'_0, \ldots, t'_{|\varphi|}$. Without loss of generality, we may assume that the roots of $t'_0, \ldots, t'_{|\varphi_1|}$ have the same label. Let t' be the data tree obtained from t'_0 by adding all the subtrees of the roots of $t'_1, \ldots, t'_{|\varphi_1|}$ as the new subtrees of the root of t'_0 (with the original subtrees of the root of t'_0 untouched). We claim that $t' \models_\lambda \forall x\varphi_1$. At first, for every d_i with $i : 0 \le i \le |\varphi_1|$, we have $t'_i \models_{\lambda[d_i/x]} \varphi_1$, thus $t' \models_{\lambda[d_i/x]} \varphi_1$ since φ_1 contains only existential path quantifiers. Let $d \notin \{d_0, \ldots, d_{|\varphi_1|}\}$. Since $t'_0 \models_{\lambda[d_0/x]} \varphi_1$ and neither d nor d_0 occurs in t'_0, assigning d to x has the same impact as assigning d_0 to x for the satisfaction of φ_1 on t'_0. Therefore, $t'_0 \models_{\lambda[d/x]} \varphi_1$. We deduce that $t \models_{\lambda[d/x]} \varphi_1$, since φ_1 contains only existential path quantifiers. From the fact that d is an arbitrary data value not in $\{d_0, \ldots, d_{|\varphi_1|}\}$, we conclude that $t' \models_\lambda \forall x\varphi_1$ and (t', λ) contains at most $|\varphi_1| + 1 \le |\varphi|$ data values. ◀

4.3 Variable quantifications in the beginning

▶ **Theorem 22.** *([10]) The following two problems are PSPACE-complete: The satisfiability problem of $\exists^*$-$VLTL_{pnf}$ and the model checking problem of $\forall^*$-$VLTL_{pnf}$.*[5]

▶ **Theorem 23.** *The satisfiability problem of $\exists^*$-$VCTL_{pnf}$ is EXPTIME-complete.*

The proof of the upper bound is similar to the proof of the satisfiability problem of $\exists^*$-VLTL$_{pnf}$. The lower bound follows from the satisfiability problem of CTL.

▶ **Theorem 24.** *The model-checking problem of $\forall^*$-$VCTL_{pnf}$ is decidable in $EXPTIME$.*

[5] In [10], only model checking problem of $\forall^*$-VLTL$_{pnf}$ is considered. The result for the satisfiability problem of $\exists^*$-VLTL$_{pnf}$ can be shown by following the same idea.

Theorem 24 can be easily deduced from the following lemma.

▶ **Lemma 25.** *Let $\mathcal{K} = (AP, X, S, R, S_0, I, L, L')$ be a VKS and $\forall x_1 ... \forall x_n \psi$ be a $\forall^*$-VCTL$_{pnf}$ sentence. Then there is a computation tree $t = (Z, L)$ of $\mathcal{K}$ s. t. $t \models \exists x_1 ... \exists x_n \overline{\psi}$ iff there is a computation tree $t' = (Z, L')$ of $\mathcal{K}$ s. t. $t' \models \exists x_1 ... \exists x_n \overline{\psi}$ and t' contains at most $|X| + n$ different values.*

We next consider the satisfiability and model checking problem of $\forall$-VLTL$^{gdap}_{pnf}$.

▶ **Theorem 26.** *The satisfiability problem of $\forall$-VLTL$^{gdap}_{pnf}$ is decidable.*

Proof sketch. Suppose $\varphi = \forall x \psi$ is a $\forall$-VLTL$^{gdap}_{pnf}$ formula over $AP \cup T$.

From the definition of $enc(\cdot)$, we know that $enc(\varphi) = \varphi'_1 \wedge \varphi'_2$ and $\varphi'_2 = \forall x \psi'$ for some quantifier free VLTL formula ψ'. Then $enc(\varphi)$ can be rewritten into $\forall x(\varphi'_1 \wedge \psi')$, since no variables occur in φ'_1. So $enc(\varphi)$ is a $\forall$-VLTL$_{pnf}$ formula over $AP' \cup T$, where $AP' = AP \cup \{p'\}$.

It is not hard to observe that if φ is a $\forall$-VLTL$^{gdap}_{pnf}$ formula, then ψ' can be rewritten into a quantifier free VLTL formula where all the occurrences of p and $\neg p$ for $p \in AP$ are guarded by the positive occurrences of $\tau(x)$ for some $\tau \in T$. For instance, an occurrence of $p \wedge \tau(x)$ in ψ s. t. $p \in AP$ and $\tau \in T$ is transformed into $(p' \wedge \bigvee_{0 \leq i \leq K-1} X^i p) \wedge (p' \wedge \bigvee_{0 \leq i \leq K-1} X^i \tau(x))$, which is equivalent to $p' \wedge \bigvee_{0 \leq i \leq K-1} X^i (p \wedge \tau(x))$, since either none of $X^i p$ holds or all of them hold. By abuse of notations, we still denote the resulting formula by ψ'. Note that the formula $\forall x(\varphi'_1 \wedge \psi')$ is not a $\forall$-VLTL$^{gdap}_{pnf}$ formula since the occurrences of p' are not guarded.

To continue the proof, we introduce the following notation. Suppose $w = w_0 \dots w_n$ is a 1-attributed data word over $AP' \cup T$ s. t. $w_i = (A_i, (B_i, d_i))$ for every $i : 0 \leq i \leq n$. Then $prj_{AP}(w) = w_0|_{AP} \dots w_n|_{AP}$, where for every $i : 0 \leq i \leq n$, $w_i|_{AP} = (A_i \cap AP, (B_i, d_i))$. The definition of $prj_{AP}(\cdot)$ can be naturally generalized to languages of 1-attributed data words.

From Proposition 13, we know that the satisfiability of φ over K-attributed data words is reduced to the nonemptiness of the language $\mathcal{L}_1(enc(\varphi))$. The nonemptiness of $\mathcal{L}_1(enc(\varphi))$ is then reduced to the nonemptiness of $prj_{AP}(\mathcal{L}_1(enc(\varphi)))$.

In addition, it is not hard to show that an EDA $\mathcal{D}_{enc(\varphi)}$ can be constructed from $enc(\varphi)$ s. t. $\mathcal{L}(\mathcal{D}_{enc(\varphi)}) = prj_{AP}(\mathcal{L}_1(enc(\varphi)))$. The decidability then follows from the decidability of the nonemptiness of EDA. ◀

▶ **Corollary 27.** *The model checking problem of $\exists$-VLTL$^{gdap}_{pnf}$ is decidable.*

References

1 R. Alur, P. Cerny, and S. Weinstein. Algorithmic analysis of array-accessing programs. *ACM Trans. Comput. Logic*, 13(3):27:1–27:29, 2012.

2 M. Bojanczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Log.*, 12(4), 2011.

3 E. Damaggio, A. Deutsch, R. Hull, and V. Vianu. Automatic verification of data-centric business processes. In *BPM*, 2011.

4 N. Decker, P. Habermehl, M. Leucker, and D. Thoma. Ordered navigation on multi-attributed data words. In *CONCUR*, 2014.

5 S. Demri, D. Figueira, and M. Praveen. Reasoning about data repetitions with counter systems. In *LICS*, 2013.

6 S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Logic*, 10(3):16:1–16:30, 2009.

7 E. A. Emerson and K. S. Namjoshi. On reasoning about rings. *Int. J. Found. Comput. Sci.*, 14(4):527–550, 2003.

8 D. Figueira. Alternating register automata on finite words and trees. *Logical Methods in Computer Science*, 8(1), 2012.

9 D. Figueira. Decidability of downward XPath. *ACM Trans. Comput. Log.*, 13(4):34, 2012.

10 O. Grumberg, O. Kupferman, and S. Sheinvald. Model checking systems and specifications with parameterized atomic propositions. In *ATVA*, 2012.

11 O. Grumberg, O. Kupferman, and S. Sheinvald. An automata-theoretic approach to reasoning about parameterized systems and specifications. In *ATVA*, 2013.

12 O. Grumberg, O. Kupferman, and S. Sheinvald. Personal communication, June 2014.

13 I. M. Hodkinson, F. Wolter, and M. Zakharyaschev. Decidable and undecidable fragments of first-order branching temporal logics. In *LICS*, 2002.

14 I. M. Hodkinson, F. Wolter, and M. Zakharyaschev. Decidable fragment of first-order temporal logics. *Ann. Pure Appl. Logic*, 106(1-3):85–134, 2000.

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

16 A. Kara, T. Schwentick, and T. Zeume. Temporal logics on words with multiple data values. In *FSTTCS*, 2010.

17 O. Kupferman. Variations on safety. In *TACAS*, 2014.

18 O. Sheinvald, S. Grumberg and O. Kupferman. A game-theoretic approach to simulation of data-parameterized systems. In *ATVA*, 2014.

19 F. Song and T. Touili. LTL model-checking for malware detection. In *TACAS*, 2013.

20 F. Song and T. Touili. Pushdown model checking for malware detection. *STTT*, 16(2):147–173, 2014.

21 M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Conference on Logics for Concurrency : Structure Versus Automata*, pages 238–266, 1996.

22 V. Vianu. Automatic verification of database-driven systems: a new frontier. In *ICDT*, 2009.