Logics of Repeating Values on Data Trees and Branching Counter Systems

Sergio Abriola[1,2(✉)], Diego Figueira[3], and Santiago Figueira[1,2]

[1] University of Buenos Aires, Buenos Aires, Argentina
sabriola@dc.uba.ar
[2] ICC-CONICET, Buenos Aires, Argentina
[3] CNRS, LaBRI, Talence, France

Abstract. We study connections between the satisfiability problem for logics on data trees and Branching Vector Addition Systems (BVAS). We consider a natural temporal logic of "repeating values" (LRV) featuring an operator which tests whether a data value in the current node is repeated in some descendant node.

On the one hand, we show that the satisfiability of a restricted version of LRV on ranked data trees can be reduced to the coverability problem for Branching Vector Addition Systems. This immediately gives elementary upper bounds for its satisfiability problem, showing that restricted LRV behaves much better than downward-XPath, which has a non-primitive-recursive satisfiability problem.

On the other hand, satisfiability for LRV is shown to be reducible to the coverability for a novel branching model we introduce here, called Merging VASS (MVASS). MVASS is an extension of Branching Vector Addition Systems with States (BVASS) allowing richer merging operations of the vectors. We show that the control-state reachability for MVASS, as well as its bottom-up coverability, are in 3ExpTime.

This work can be seen as a natural continuation of the work initiated by Demri, D'Souza and Gascon for the case of data words, this time considering *branching* structures and counter systems, although, as we show, in the case of data trees more powerful models are needed to encode satisfiability.

1 Introduction

Logics for data trees. Finite data trees are ubiquitous structures that have attracted much attention lately. A *data tree* is a finite tree whose every position carries a *label* from a finite alphabet and a collection of *data values* from some infinite domain.[1] This structure has been considered in the realms of semi-structured data as a simple abstraction of XML documents, timed automata,

We thank STIC AmSud, ANPCyT-PICT-2013-2011, UBACyT 20020150100002BA, and the Laboratoire International Associé "INFINIS".

[1] Other works have considered different simplifications of these structures, either having only one data value per node (*e.g.*, [2]) or ignoring the label (*e.g.*, [7]).

J. Esparza and A.S. Murawski (Eds.): FOSSACS 2017, LNCS 10203, pp. 196–212, 2017.
DOI: 10.1007/978-3-662-54458-7_12

program verification, and generally in systems manipulating data values. Finding decidable logics or automata models over data trees is a fundamental quest when reasoning on data-driven systems.

A wealth of specification formalisms on these structures (either for data trees or its 'word' version, data words) have been introduced, stemming from automata [25,28], first-order logic [2,4,16,19], XPath [13–15,18,20], or temporal logics [7,9,11,21,22,24]. In full generality, most formalisms lead to undecidable reasoning problems and a well-known research trend consists of finding a good trade-off between expressiveness and decidability.

Interesting and surprising results have been exhibited about relationships between *logics* for data trees and *counter automata* [18–20]. This is why logics for data trees are not only interesting for their own sake but also for their deep relationships with counter systems.

This work. The aim of this work is to study the basic mechanism of "data repetition", common to many logics studied on data trees. For this, we study a basic logic that can navigate the structure of the tree through the use of CTL-like modalities, and on the other hand can make "data tests", by asking whether a data value is repeated in the subtree. More concretely, the data tests are formulas of the form $u \approx \mathrm{EF}v$, stating that the data value stored in attribute (also called *variable* here) u of the current node is equal to the data value stored in attribute v of some descendant. This *logic of repeating values*, or LRV, has been the center of a line of investigation studied in [6,7] on *data words*, evidencing tight correspondences between reachability problems for Vector Addition Systems and the satisfiability problem. The current work pursues this question further, exhibiting connections between the satisfiability problem of LRV *over data trees* and the bottom-up coverability problem for branching counter systems. In order to obtain connections with branching Vector Addition Systems with States, or branching VASS [29], we also introduce a restriction where tests of the form $u \approx \mathrm{EF}v$ are only allowed when $u = v$. We denote this restriction by $\mathrm{LRV}^{\mathrm{D}}$. This symbiotic relation between counter systems and logics leads us to consider some natural extensions of both the logic and the branching counter systems. In particular, we introduce a new model of branching counter system of independent interest, with decidable coverability and control-state reachability problems, that captures LRV.

The extension of the logic LRV from words to trees is a very natural one. However, the techniques needed to encode the satisfiability of the logic into a counter system are not simple extensions from the ones provided on data words. The reason for this difficulty is manyfold: (*a*) the fact that now the future is non-linear in addition to the possibility of having a data value repeating at several descendants in *different* variables, makes the techniques of [7] for propagating values of configurations impractical; (*b*) further, this seems to be impossible for the case of data trees, and we could only show a reduction for the fragment $\mathrm{LRV}^{\mathrm{D}}$; (*c*) in order to reduce the satisfiability problem for the full logic we will need to augment the power of branching VASS with the possibility to 'merge'

counters in a more powerful way, somewhat akin to what has been done for encoding the satisfiability for FO^2 [19].

Contributions. The main contributions are the following:

- We show that the satisfiability for $\mathrm{LRV}^{\mathrm{D}}$ on k-ranked data trees is reducible, in exponential space, to the control-state reachability problem for VASS_k (*i.e.*, Branching VASS of rank k) in Sect. 5. Since the control-state reachability problem is decidable [29] in 2ExpTime [8], this reduction yields a decision procedure.
- We consider the addition of an operator $\mathrm{AG}_{\approx v}(\varphi)$ expressing "every descendant with the same v-attribute verifies φ", and we show that the logic resulting from adding positive instances of this operator is equivalent to the control-state reachability for Branching VASS, that is, there are reductions in both directions (Sect. 6).
- We introduce an extension of Branching VASS, called *Merging* VASS or MVASS in Sect. 4.2. This model allows for merging counters in branching rules in a form which is not necessarily component-wise, allowing for some weak form of counter transfers. We show that the bottom-up coverability (and control-state reachability) problem for MVASS is in 3ExpTime (Sect. 4.4). This is arguably a model of independent interest.
- We show that the satisfiability for LRV on k-ranked data trees can be reduced to the control-state reachability for MVASS_k in Sect. 7. As in the case of $\mathrm{LRV}^{\mathrm{D}}$, this yields a decision procedure.

Related work. The most closely related work is the one originated by Demri et al. in [5,6] and pursued in [7]. These works study the satisfiability problem for temporal logics on *data words*, extended with the ability to test whether a data value is repeated in the past/future. Indeed, our current work is motivated by the deep correlations evidenced by these works, between counter systems and simple temporal logics on data words. The present manuscript expands this analysis to *branching* logics and counter systems.

There are several works showing links between reachability-like problems for counter systems and the satisfiability problem of logics on data trees. The first prominent example is that satisfiability for Existential MSO with two variables on data words ($\mathrm{EMSO}^2(+1, <, \sim)$) corresponds precisely to reachability of VASS [3], in the sense that there are reductions in both directions. On the other hand, EMSO^2 over (unranked) *data trees* was shown to have tight connections with the reachability problem for an *extension* of BVASS [19], called 'EBVASS'. This extension has features which are very close to the model we introduce here, MVASS, but it does not capture, nor is captured by, MVASS. One can draw a parallel between the situation of the satisfiablity for EMSO^2 and for LRV: while on data words both are inter-reducible to VASS, the extension to data-trees is non-trivial, and they no longer correspond to BVASS, but to *extensions* thereof.

In the course of the last decade, several logics for data trees have been proposed. Among those that feature navigation in terms of modalities such as

temporal operators, one noticeable logic is that of XPath. Although the satisfiability problem for XPath is undecidable, several fragments have been shown to be decidable through reduction to the reachability or coverability problems for counter systems [9,12,18]. In particular, the satisfiability problem for XPath with strict descendant (usually written $\downarrow_+$) on ranked data trees has already a non-primitive-recursive lower bound in complexity, as can be seen by adapting techniques shown for data words [17].

Modulo a simple coding, our logic LRV is captured by a fragment of regular-XPath, here called reg-XPath$_{\mathrm{LRV}}$, on data trees where path expressions are allowed to use Kleene star on any expression (this what we denote by 'regular' XPath), and data tests are of the form $\langle \varepsilon \star \downarrow^* [\varphi] \rangle$ or $\langle \downarrow^n [\varphi] \star \downarrow^m [\psi] \rangle$ for some $n, m \in \mathbb{N}$ and $\star \in \{=, \neq\}$. There are, however, three provisos for this statement. First, in the aforementioned works on XPath the data model consists of data trees whose every position carries exactly *one* data value. In the present paper, we study 'multi-attributed' data trees where, essentially, each node carries a set of pairs 'attribute:value'. However, by means of a simple coding, such as putting every 'attribute:value' as a leaf of the corresponding node, one can easily translate LRV formulas to XPath formulas. Second, our LRV formulas are of the form $u \approx \mathrm{EF} v$ stating that the current data value under attribute u is repeated in a node x of the subtree under attribute v, but one cannot test that some property ψ further holds at the repeating node x. However, it was shown in [7] that one can extend the logic with this power, obtaining formulas of the form $x \approx \mathrm{EF} y[\psi]$, since this extended logic is PTime-reducible to the logic without these tests. Third, the LRV formulas cannot test for regular properties on the labeling of paths, and thus there is no precise characterization in terms of a natural fragment of regular-XPath, but one could add regular path tests to LRV to match the expressive power of reg-XPath$_{\mathrm{LRV}}$ without changing any of our results.

In fact, the fragment reg-XPath$_{\mathrm{LRV}}$ extends also the fragment DataGL considered in [1,13] containing only data tests of the form $\langle \varepsilon \star \downarrow^* [\varphi] \rangle$, which is known to be PSpace-complete on unranked data trees [13].

It is not hard to see that the satisfiability problem of LRV on unranked data trees is PSpace-complete following the techniques from [13]. On the other hand, on ranked data trees we know, by the discussion above, that if we would allow intermediate tests in a way to be able to encode the expressive power of XPath($\downarrow_+$) we would have a non-primitive recursive lower bound. It is therefore natural to limit the navigation *disallowing* intermediary tests. This natural fragment was already studied on data words [7], and we now study it on data trees.

2 Preliminaries

Let $\mathbb{N}^+ = \{1, 2, \dots\}$, $\mathbb{N} = \mathbb{N}^+ \cup \{0\}$, and $\underline{n} = \{1, \dots, n\}$ for every $n \in \mathbb{N}$. We use the bar notation $\bar{x}$ to denote a tuple of elements, where $\bar{x}[i]$, for $i > 0$, refers to the i-th element of the tuple. For any pair of vectors $\bar{x}, \bar{y} \in \mathbb{Z}^k$ we write $\bar{x} \leq \bar{y}$

if $\bar{x}[i] \leq \bar{y}[i]$ for all $1 \leq i \leq k$. The constant $\bar{\emptyset}$ refers to the (unique) vector of dimension 0, and the constant $\bar{e}_i$ refers to the vector (whose dimension will always be clear from the context) so that $\bar{e}_i[i] = 1$ and $\bar{e}_i[j] = 0$ for all $j \neq i$. We write $\bar{0}$ for the tuple of all 0's (the dimension being implicit from the context).

A **linear set** of dimension k is a subset of $\mathbb{N}^k$ which is either empty or described as $\{\bar{v}_0 + \alpha_1\bar{v}_1 + \cdots + \alpha_n\bar{v}_n \mid \alpha_1, \ldots, \alpha_n \in \mathbb{N}\}$ for some $n \in \mathbb{N}$ and $\bar{v}_0, \ldots, \bar{v}_n \in \mathbb{N}^k$. Henceforward we assume that linear sets are represented by the **offset** $\bar{v}_0$ and the **generators** $\bar{v}_1, \ldots, \bar{v}_n$, where numbers are represented in binary. For ease of writing we will denote a linear set like the one above by "$\bar{v}_0 + \{\bar{v}_1, \ldots, \bar{v}_n\}^*$".

We fix once and for all an infinite domain of **data values** $\mathbb{D}$. A **data tree** of rank k over a finite set of labels $\mathbb{A}$ and a finite set of attributes $\mathbb{V}$, is a finite tree whose every node x contains a pair $(a, \mu) \in \mathbb{A} \times \mathbb{D}^{\mathbb{V}}$ and has no more than k children. In general, a will be called the **label** of x and $\mu(v)$ will be called the **data value** of attribute $v \in \mathbb{V}$ at x. The i-**ancestor** of a node x of a data tree T is the ancestor at distance i from x (*i.e.*, the 1-ancestor is the parent); while the i-**descendant** of x are all the descendants of x at distance i.

3 Logic of Repeating Values on Data Trees

We will work with a temporal logic using CTL* modalities [10,26] to navigate the tree—although this is not really essential to our results, in the sense that any other MSO definable data-blind operators could also be added to the logic obtaining similar results. The **Logic of Repeating Values** LRV contains the typical modalities from CTL*, such as EF, AF, EU, etc. as well as the possibility to test for the label of the current node, and *data tests*. Data tests are restricted to being very basic, as in [6], of the form "$u \approx \mathrm{EF}v$" stating "the data value of attribute u appears again at the attribute v of some descendant", or "$u \not\approx \mathrm{EF}v$" stating "there is a descendant node whose attribute v contains a different data value from the data value of the attribute u of the current node". Since LRV is closed under Boolean connectives, this means we can also express, for instance, that attribute u of all descendants have the same data value as the current node's: $\neg(u \not\approx \mathrm{EF}u)$.

Formally, formulas of LRV are defined by

$$\varphi ::= a \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathrm{EU}(\varphi, \psi) \mid u \approx \mathrm{EF}v \mid u \not\approx \mathrm{EF}v \mid u \approx \mathrm{EX}^i v \mid u \not\approx \mathrm{EX}^i v,$$

where a ranges over a finite set of labels $\mathbb{A}$, u, v range over a finite set of attribute variables $\mathbb{V}$ (also called just 'variables'), and $i \in \mathbb{N}^+$. Given a data tree T and a node x of T, the satisfaction relation $\models$ is defined in the usual way: $T, x \models a$ if a is the label of x; $T, x \models u \approx \mathrm{EF}v$ [resp. $T, x \models u \not\approx \mathrm{EF}v$] if there is a strict descendant y of x so that the u-attribute of x has the same [resp. different] data value as the v-attribute of y; $T, x \models u \approx \mathrm{EX}^i v$ [resp. $T, x \models u \not\approx \mathrm{EX}^i v$] if there exists an i-descendant of x whose v-attribute is equal [resp. distinct] to the u-attribute of x; and $T, x \models \mathrm{EU}(\varphi, \psi)$ if there is some strict descendant y of x so that $T, y \models \varphi$

and every other node z strictly between x and y verifies $T, z \models \psi$. Note that the remaining CTL* modalities (EX, EG, EF, AX, AG, AF, AU) can be expressed using EU[2].

We call $\mathrm{LRV}^{\mathrm{D}}_n$ the logic using at most n attribute variables, whose only admissible data tests are of the form $u \approx \mathrm{EF}u$, $u \not\approx \mathrm{EF}u$, $u \approx \mathrm{EX}^i u$ or $u \not\approx \mathrm{EX}^i u$ (same variable in the left and right sides). Intuitively, this corresponds to the restriction where each attribute variable ranges over a *disjoint* set of data values (hence the letter 'D').

4 Models of Branching Counter Systems

We present the models of counter systems we are going to work with. The first one is a well-known model, usually known as Branching Vector Addition System with States, or "BVASS", while the second one is a useful extension of the first one where the split/merge operation of the counters is controlled by the use of linear sets.

4.1 Branching VASS

A VASS of rank k and dimension n, or $n\mathrm{VASS}_k$, is a tuple $\mathcal{A} = \langle Q, U, B\rangle$, where Q is a finite set of states, $U \subseteq Q \times \mathbb{Z}^n \times Q$ is a set of unary rules, and $B \subseteq Q \times Q^{\leq k}$ is a finite set of branching rules. We notate $q \xrightarrow{\bar{v}} q'$ for a unary rule $(q, \bar{v}, q') \in U$, and $q \to (q_1, \ldots, q_i)$ for a branching rule $(q, q_1, \ldots, q_i) \in B$. A **configuration** is an element from $\mathit{Confs} := Q \times \mathbb{N}^n$. For a configuration $(q, \bar{n})$ we often use the term "counter i" instead of "$n[i]$" (in the case $n = 1$ we speak of *the* counter).

A **derivation tree** [resp. **incrementing derivation tree**] is a finite tree $\mathcal{D}$ whose every node x is either

- labeled with a pair $(p \xrightarrow{\bar{v}} p', (q, \bar{n})) \in U \times \mathit{Confs}$ so that $p \xrightarrow{\bar{v}} p'$ is a unary rule of U, $p = q$ and it has exactly one child, which is labeled $(r_1, (p_1, \bar{n}_1))$ so that $p' = p_1$ and

$$\bar{n} + \bar{v} = \bar{n}_1 \quad [\text{resp. } \bar{n} + \bar{v} \leq \bar{n}_1]; \tag{1}$$

- or labeled with a pair $((p, \bar{q}), (q, \bar{n})) \in B \times \mathit{Confs}$ so that $p \to \bar{q}$, with $\bar{q} \in Q^{k'}$ for some $k' \leq k$, is a branching rule of B, $p = q$ and it has exactly k' children, labeled $(r_1, (p_1, \bar{n}_1)), \ldots, (r_{k'}, (p_{k'}, \bar{n}_{k'}))$ so that $\bar{q} = (p_1, \ldots, p_{k'})$ and

$$\bar{n} = \sum_{i \leq k'} \bar{n}_i \quad [\text{resp. } \bar{n} \leq \sum_{i \leq k'} \bar{n}_i]. \tag{2}$$

Note that leaf nodes are necessarily labeled with rules of the form $q \to \bar{\emptyset} \in B$. Without loss of generality we will assume that the system contains rules $q \to \bar{\emptyset}$ for every state q.

[2] $\mathrm{EX}\varphi = \mathrm{EU}(\varphi, \bot)$, $\mathrm{EF}\varphi = \mathrm{EU}(\varphi, \top)$, $\mathrm{EG}\varphi = \mathrm{EU}(\varphi \wedge \neg\mathrm{EX}\top, \varphi)$, $\mathrm{AU}(\varphi, \psi) = \neg\mathrm{EU}(\neg\psi \wedge \neg\varphi, \neg\psi) \wedge \neg\mathrm{EG}(\neg\varphi)$, etc.

4.2 Merging VASS

We present an extension of the model above where the branching rules, now called *merging rules*, are more powerful: they allow us to reorganize the counters. Whereas in an (incrementing) derivation tree for VASS_k the component i of the configuration of a node depends only on the component i of its children and the rule applied, MVASS_k allows to have transfers *between components*. However, these transfers have some restrictions—otherwise the model would have non-elementary or undecidable coverability/reachability problems [23]. First, transfers between components are 'weak', in the sense that we cannot force a transfer of the whole value of a coordinate i to a distinct coordinate j of a child, we can only make sure that part of it will be transferred to component j and part of it will remain in component i. Second, these weak transfers can only be performed for any pair of coordinates i, j adhering to a partial order, where transfers occur from a bigger component to a smaller one.

A **Merging-VASS** of rank k and dimension n, or $n\mathrm{MVASS}_k$, is a tuple $\mathcal{A} = \langle Q, U, M, \preceq \rangle$, where $\preceq$ is partial order on $\underline{n}$, Q and U are as before, and M is a set of merging rules of the form $(q, S, \bar{q})$ where $q \in Q$, $\bar{q} \in Q^{k'}$ with $k' \leq k$, and $S \subseteq \mathbb{N}^{n\cdot(k'+1)}$ is a linear set of dimension $n\cdot(k'+1)$ of the form $\bar{0} + (B \cup S_0)^*$, where

1. all the elements of B are of the form $(\bar{e}_i, \bar{x}_1, \ldots, \bar{x}_{k'})$, where for each $1 \leq \ell \leq k'$, $\bar{x}_\ell \in \mathbb{N}^n$ is either $\bar{0}$ or $\bar{e}_j$ for some $j \prec i$; and
2. S_0 consists of the following $k' \cdot n$ vectors

$$S_0 = \bigcup_{1 \leq i \leq n} \{(\bar{e}_i, \bar{e}_i, \bar{0}, \bar{0}, \ldots, \bar{0}), (\bar{e}_i, \bar{0}, \bar{e}_i, \bar{0}, \ldots, \bar{0}), \ldots, (\bar{e}_i, \bar{0}, \ldots, \bar{0}, \bar{e}_i)\}. \quad (3)$$

The idea is that in point 1 we allow to transfer contents from component i to components of smaller order. For example, on dimension 3 and rank 2, a vector $\bar{v} = (1,0,0)(0,1,0)(0,0,1)$ in B would imply that during the merge one can transfer a quantity $m > 0$ from component 1 of the father into component 2 of the first child and component 3 of the second child, assuming $2, 3 \prec 1$). On the other hand, point 2 tells us that for every i we can always have some quantity of component i that is *not* transferred to other components, *i.e.*, that stays in component i. Continuing our example, the children configurations $(m, m' + s, t)$ and $(m, s, m' + t)$ can be merged into $(m + m', s, t)$ for every $m, m', s, t \geq 0$, using the vector $\bar{v}$ and S_0.

A derivation tree [resp. incrementing derivation tree] is defined just as before, with the sole difference being that condition (2) is replaced with

$$\begin{aligned} &(\bar{n}, \bar{n}_1, \ldots, \bar{n}_{k'}) \in S \\ &[\text{resp. } (\bar{n}, \bar{n}'_1, \ldots, \bar{n}'_{k'}) \in S \text{ for } (\bar{n}'_1, \ldots, \bar{n}'_{k'}) \leq (\bar{n}_1, \ldots, \bar{n}_{k'})]. \end{aligned} \quad (4)$$

Notice that this is a generalization of VASS_k. Indeed, VASS_k corresponds to the restriction where all the k'-ary merging rules have $S = \bar{0} + S_0^*$ for S_0 as defined in (3). Note that an (incrementing) derivation tree for $n\mathrm{VASS}_k$ is, in particular, an

(incrementing) derivation tree for nMVASS$_k$. As before, we assume that there are always rules $(q, \emptyset, \bar{\emptyset})$ for every state q.

Jacquemard et al. [19] study an extension of BVASS, 'EBVASS', in relation to the satisfiability of FO$^2(<, +1, \sim)$ over *unranked* data trees. EBVASS has some features for merging counters. While MVASS and EBVASS are incomparable in computational power, it can be seen that without the restriction $j \prec i$ in condition 1, MVASS would capture EBVASS. In fact, this condition is necessary for the (elementary) decidability of the coverability problem for MVASS, while the status of the coverability problem for EBVASS is unknown.

4.3 Decision Problems

Given a counter system $\mathcal{A}$, a set of states $\widehat{Q}$, and a configuration $(q, \bar{n})$ of $\mathcal{A}$, we write $(q, \bar{n}) \rightsquigarrow_{\mathcal{A}} \widehat{Q}$[resp. $(q, \bar{n}) \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}$] if there exists a derivation tree [resp. incrementing derivation tree] for $\mathcal{A}$ with root configuration $(q, \bar{n})$, so that all the leaves have configurations from $\widehat{Q} \times \{\bar{0}\}$. The reachability and incrementing reachability problems are defined as follows.

Problem:	VASS$_k$ **reachability problem** [resp. VASS$_k$ **incrementing reachability problem**]
Input:	an nVASS$_k$ $\mathcal{A}$ with states Q, a set of states $\widehat{Q} \subseteq Q$, and a configuration $(q, \bar{n})$ of $\mathcal{A}$
Output:	'Yes' iff $(q, \bar{n}) \rightsquigarrow_{\mathcal{A}} \widehat{Q}$ [resp. $(q, \bar{n}) \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}$]

Observe that when $k = 1$ this problem is equivalent to the reachability and coverability problems for Vector Addition Systems with States.

The **MVASS$_k$ reachability problem** and **MVASS$_k$ incrementing reachability problem** are defined just as before but considering $\mathcal{A}$ to be an nMVASS$_k$ instead of a nVASS$_k$. We will often refer to these problems as Reach( ) and Reach$^+$( ). We also remark that the incrementing reachability problem is simply a restatement of the *coverability problem*. In particular, it is monotone: if $(q, \bar{n}) \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}$ and $\bar{n}' \leq \bar{n}$ then $(q, \bar{n}') \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}$. We define the **control-state reachability problem** CSReach as the problem of, given $\mathcal{A}, q, \widehat{Q}$, whether $(q, \bar{n}) \rightsquigarrow_{\mathcal{A}} \widehat{Q}$ for some $\bar{n}$. It is easy to see that this problem is equivalent to the problem of whether $(q, \bar{0}) \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}$.

In [8] the coverability problem (or equivalently, the incrementing reachability problem) for a single-state formulation, called BVAS, is studied. A BVAS consists of a tuple $\langle n, R_1, R_2 \rangle$, where R_1 is a set of unary rules, R_2 is a set of binary rules (both rules included in $\mathbb{Z}^n$ which add up a vector). The *size* of a given BVAS is defined as $n\ell$, where ℓ represents the maximum binary size of an entry in $R_1 \cup R_2$.

Proposition 1. *[8] Coverability for* BVAS *is 2ExpTime-complete. If the dimension n is fixed, the problem is in ExpTime.*

4.4 Decidability of Reach$^+$(MVASS)

The arguments used in [8] to prove the previous proposition can be adapted to show a similar result for MVASS: the Reach$^+$ and CSReach problems are in 3ExpTime.

Theorem 2. *Reach$^+$(MVASS$_k$) and CSReach(MVASS$_k$) are in 3ExpTime for every $k \geq 1$. If the dimension n is fixed, the problem is in 2ExpTime.*

Proof (idea). In a somewhat similar way as it was shown in [8, Lemma 6] for the case of VASS$_k$, one can show that if there is an incrementing derivation $\mathcal{D}$ witnessing $(q, \bar{n}) \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}$, where $\mathcal{A} = (Q, U, M, \preceq)$ is an nMVASS$_k$ counter system, then there is a 'contraction' (*i.e.*, the result of the repeated replacing of the subtree at a node x with the subtree at some descendant y while maintaining the property of being a derivation) $\mathcal{D}'$ of $\mathcal{D}$ with height bounded doubly-exponentially in the dimension. One significant difficulty in adapting the proof for VASS$_k$ to MVASS$_k$, is that in a derivation for a VASS$_k$, if the component i of a configuration at a node x is "very big", and the same component i at the root is "small" (say, 0), this means that the distance between x and the root in the derivation tree must be big. This is a crucial ingredient for bounding the height of a minimal derivation and obtaining the 2ExpTime upper bound proof for Reach$^+$(VASS$_k$) in [8]. However, this is no longer the case for MVASS$_k$, since the size of component i at x may come from a transfer of the parent from another component with higher $\preceq$-index. Nevertheless, by using the fact that (i) transfers between components induced by merging rules are $\preceq$-ordered (point 1 of the definition), and (ii) linear sets of merging rules are monotone in the sense that they contain S^*_0 as defined in (3), we can recover bounds for the minimal height of derivations. This can be done by induction on the preorder—note that relative to maximal $\preceq$ coordinates MVASS$_k$ behaves like VASS$_k$.

These results imply that, in order to decide the incrementing reachability problem, it suffices to search for a derivation of doubly-exponential height, whose vectors may contain triply-exponential entries in principle. As a consequence of this, the verification of the existence of such a derivation can be performed in alternating double exponential space, as it is shown in [8, Theorem 8], and thus the incrementing reachability for MVASS is in 3ExpTime.

If n is fixed, the height of the witnessing derivation becomes singly exponential and thus the problem is in 2ExpTime (as explained in [8, Theorem 8]). □

5 Satisfiability of LRV$^{\mathrm{D}}$ on data trees

We call SAT$_k$ the satisfiability problem on finite k-ranked data-trees. The main result of this section is the following.

Theorem 3. *SAT$_k$-LRV$^{\mathrm{D}}_n$ is ExpSpace-reducible to* CSReach(nVASS$_k$).

In the proof of the theorem, the number of attribute variables of the formula will become the dimension of the VASS$_k$. Since the CSReach problem for VASS$_k$

is decidable in 2ExpTime, this yields a decidable procedure for SAT_k-$\mathrm{LRV}^{\mathrm{D}}$ for every k. For the case $k = 1$, *i.e.*, on *data words*, it has been shown [7] that there is a reduction from SAT_1-LRV_n to $\mathrm{CSReach}(2^n\text{-}\mathrm{VASS}_1)$, where the dimension of the VASS_1 is exponential in the number of variables. However, it is easy to see that the proof of [7] also yields a reduction from SAT_1-$\mathrm{LRV}^{\mathrm{D}}_n$ to $\mathrm{CSReach}(n\mathrm{VASS}_1)$. Thus, this theorem has been shown for $k = 1$, and here we generalize it to $k > 1$. However, there are a number of problems that appear if one tries to "extend" the proof of [7] to the branching setup. In particular, the non-linearity of the future in addition to the possibility of having a data value repeating at several descendants in different variables, calls for a non-standard way of propagating the values of configurations, which is not contemplated in VASS_k. This is why we are only able to show the reduction for the 'disjoint' fragment $\mathrm{LRV}^{\mathrm{D}}$, and which leads us to consider the extended model MVASS_k in Sect. 7. This propagation problem does not appear when one only considers that the classes of different values are disjoint, that is, that all formulas of the type $v \star \mathrm{EF} w$ with $\star \in \{\approx, \not\approx\}$ have $v = w$, motivating the study of SAT_k-$\mathrm{LRV}^{\mathrm{D}}_n$.

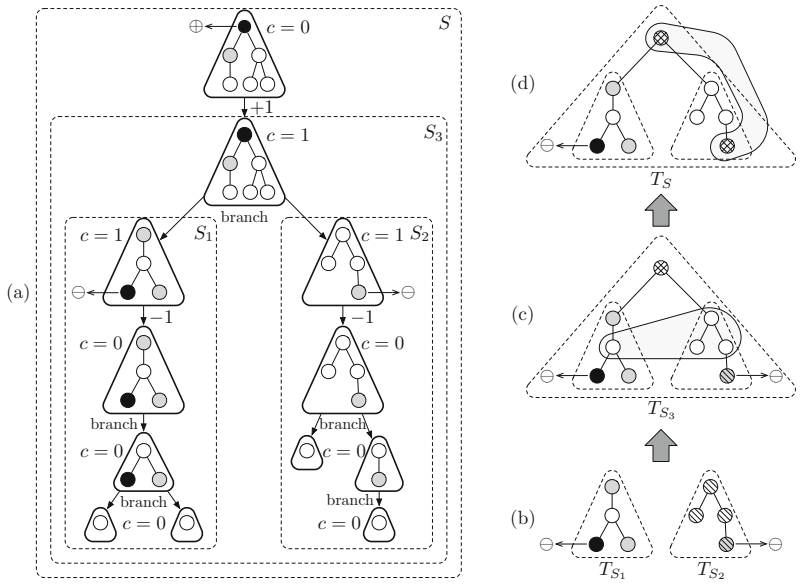
Proof idea. We start by analyzing a restricted case, which serves as building block: the logic $\mathrm{LRV}^{\mathrm{D}-}_1$ whose only formulas are conjuncts of terms of the form $v \star \mathrm{EX}^i v$, $v \star \mathrm{EF} v$, or their negation, where $\star \in \{\approx, \not\approx\}$. We show that for any formula φ of $\mathrm{LRV}^{\mathrm{D}-}_1$, there is a $1\mathrm{VASS}_k$ $\mathcal{A}^k_\varphi = \langle Q, U, B\rangle$, a set of initial states $Q_0 \subseteq Q$, and a set of final states $\widehat{Q} \subseteq Q$ such that $\mathrm{SAT}_k(\varphi)$ iff there is a derivation tree with a starting node in $q_0 \in Q_0$ that is a solution to $\mathrm{CSReach}(\mathcal{A}^k_\varphi, q_0, \widehat{Q})$—it is easy to see that this problem is equivalent to CSReach as stated in Sect. 4.3. We then extend this construction to the automaton $\mathcal{B}^k_\varphi$, enabling a reduction from the full logic $\mathrm{LRV}^{\mathrm{D}}_1$, but still restricted to only one variable. Finally, because of the disjointness of the variables, it is easy to extend these constructions to the full logic $\mathrm{LRV}^{\mathrm{D}}_n$.

Here we only give a brief explanation of the construction of $\mathcal{A}^k_\varphi = \langle Q, U, B\rangle$ for the logic $\mathrm{LRV}^{\mathrm{D}-}_1$. For the sake of simplicity, we assume our logic has no labels; their addition to the construction is straightforward. Since LRV can only deal with data (in)equality and since in this case we consider $n = 1$, we will interchangeably speak of an equivalence relation between the nodes of the tree or of the particular data values.

We define the EX-length of a formula ψ as the maximum i such that ψ contains a subformula of the form $v \star \mathrm{EX}^i v$. Let d be the EX-length of φ. The set Q consists of all valid (d, k)-frames, where a (d, k)-frame is a tree of depth d and rank k, equipped with an equivalence relation, and with some extra attributes (node-labeling functions) to include some special marks and semantic information of future requirements of the form $(\neg)v \approx \mathrm{EF} v$ and $(\neg)v \not\approx \mathrm{EF} v$. The initial states Q_0 are those frames F satisfying the *local* part of φ (that is, subformulas of φ the form $v \star \mathrm{EX}^i v$). *Future requirements* (that is, subformulas of φ of the form $v \star \mathrm{EF} v$) may not be satisfied locally in F. The set $\widehat{Q}$ is the singleton with a frame consisting in a single node. The basic idea is that the counter of $\mathcal{A}^k_\varphi$ keeps track of how many future requirements are not yet satisfied. Some nodes

of the frames may have extra information in the form of labels $\oplus$ or $\ominus$. States F whose root is labeled with $\oplus$ are *points of increment*: $\mathcal{A}^k$ will have unary rule in U that increments the (sole) counter in 1. A point of increment denotes that some subformula $v \approx \mathrm{EF}v$ of φ should hold, but it is not satisfied *locally*, that is, inside F. Leaves with $\ominus$ are those not related to ancestors in the frame with the same data value, they can thus be "joined" into the same equivalence class to a future requirement originated at some distant ancestor. States with leaves $\ominus$-labeled are *points of decrement*: $\mathcal{A}^k$ will have a unary rule in U to decrement the counter depending on the number of equivalence classes of leaves labeled with $\ominus$. The branching rules B of $\mathcal{A}^k$ are of the form $F \to (F_1, \ldots, F_i)$, where F_j overlaps with an adequate part of F.

Example 4. The following figure illustrates a scheme of an incrementing derivation S of the $1\mathrm{VASS}_2$ $\mathcal{A}^2_\varphi$ (a) and some steps (b, c and d) in the bottom up construction of the data tree T_S satisfying φ, for $\varphi = \neg v \approx \mathrm{EX}v \wedge \neg v \approx \mathrm{EX}^2 v \wedge v \approx \mathrm{EF}v$. Triangles represent $(2,2)$-frames. Shades of gray represent the equivalence classes, which only make sense inside any frame. The counter is notated with c, and arrows represent the (unary/branching) transitions of the derivation. Notice that the top branching is 'incremental', and that the local requirements of φ (namely, $\neg v \approx \mathrm{EX}v$ and $\neg v \approx \mathrm{EX}^2 v$) are satisfied in the root of the top frame.



The construction of T_S is bottom-up, and we show three steps: (a), (b) and (c). Notice that in (b) each of T_{S_1} and T_{S_2} has its own partition (no intersection). In (c) we process the root of S_3 by tying together T_{S_1} and T_{S_2} with a common parent, who lives in a single class of the partition. Notice that the partitions of

T_{S_1} and T_{S_2} are properly joined (grey area), according to the information in the root of S_3. Finally in (d) we construct T_S. The root of S is a point of increment, so we match $\oplus$ with some $\ominus$ in T_{S_3}. In this case, we match it with the right-hand $\ominus$, and so we join them by putting them in the same partition (grey area). We have satisfied the future requirement $v \approx \mathrm{EF}v$ of φ.

On the one hand, any incrementing derivation S that is a solution to $\mathrm{CSReach}(\mathcal{A}^k_\varphi, q_0, \widehat{Q})$ for some $q_0 \in Q_0$, can be translated into a data tree T_S whose root satisfies φ. In fact, any semantic information contained in the labels of nodes in frames of S will be satisfied in the corresponding nodes of T_S. The difficult part is to show that $\ominus$-leaves will have the necessary conditions to be joined with the equivalence class of an $\oplus$-ancestor, making true the formula $v \approx \mathrm{EF}v$. This will be a consequence of the fact that the incrementing derivation satisfies CSReach.

On the other hand, if φ is satisfiable in some k-ranked data tree T then we can build an incrementing derivation tree S_T that is a solution to $\mathrm{CSReach}(\mathcal{A}^k_\varphi, q_0, \widehat{Q})$ for some $q_0 \in Q_0$. Following the ideas of the previous part, we proceed from the root toward the leaves using the structure and equivalence classes of T to determine in each step the corresponding states (including the semantic labels and the labels $\ominus$, $\oplus$) and rules of S_T. From the construction, and using the incrementing nature of the derivation, it will follow that S_T is a solution to the control-state reachability problem.

For the construction of $\mathcal{B}^k_\varphi$, the information given by the (d,k)-frames will be supplemented by the addition of sets of formulas containing information about the EU operator and the Boolean connectives. The way to do this is standard (see *e.g.*, [6]).

Complexity. Let $\mathrm{LRV}^{\mathrm{D}}_{n,d}$ be the fragment of $\mathrm{LRV}^{\mathrm{D}}_{n}$ where each formula has EX-length at most d. By inspecting the above reduction, we can bound the number of states of the constructed $n\mathrm{VASS}_k$ by $O(p(n)^{k^{d+1}} \cdot (k^{d+1})^{k^{d+1}} \cdot 2^{p(|\varphi|)})$ for some polynomial p, and we can bound the maximum value among the entries in unary rules by k^d. Furthermore, we can reduce our branching VASS to an equivalent (single-state) BVAS with an addition of a constant number of new dimensions. This transformation increases the binary size of the maximum entry of the unary rules at most logarithmically over the number of states of our original $n\mathrm{VASS}_k$. Now, using Proposition 1, Theorem 3, and the above complexity analysis, we obtain:

Proposition 5. *SAT_k-$\mathrm{LRV}^{\mathrm{D}}_{n,d}$ is in ExpTime for fixed k, n, d; it is in 2ExpTime for fixed k, n or fixed d, k; and it is in 3ExpTime for fixed k.*

6 Obtaining Equivalence with VASS_k

In the previous section we have seen a reduction into the control-state reachability problem for VASS_k. A natural question is whether there exists a reduction in the other direction: can $\mathrm{CSReach}(\mathrm{VASS}_k)$ be reduced into the k-satisfiability

for LRV^{D}? For the case $k = 1$, this has been shown to be the case [7]: there exists a polynomial-space reduction from $\text{CSReach}(\text{VASS}_1)$ to $\text{SAT}_1(\text{LRV})$.

The existence of a reduction would show, intuitively, that one can express in the logic that there is a tree that verifies all the conditions for being a derivation. Without the use of data tests, one can easily encode trees that verify all the conditions except perhaps (1) and (2) regarding the vectors. For this, let us assume without loss of generality that all unary rules contain a vector $\bar{e}_i$ or $-\bar{e}_i$. The data values are used to ensure the next two conditions:

- Along any branch, every node containing a rule of the form $q \xrightarrow{\bar{e}_i} q'$ has a unique data value. In other words, we cannot find two nodes encoding an increment of component i with the same data value so that one is the ancestor of the other.
- For every node with a unary rule $q \xrightarrow{\bar{e}_i} q'$ there exists a descendant with a rule $p \xrightarrow{-\bar{e}_i} p'$ and the same data value.

These two conditions imply that after incrementing component i there must be *at least* one corresponding decrement of component i. Note that there could be *more* decrements than increments, which is not a problem since we work under the 'incrementing' semantics.

Interestingly, these two conditions can be expressed in LRV, but we do not know how to encode it in LRV^{D} (we conjecture that they are not expressible).

Adding the operator $\mathbf{AG}_{\approx v}(\varphi)$. We add a new operator $\text{AG}_{\approx v}(\varphi)$ to LRV^{D}, where $T, x \models \text{AG}_{\approx v}(\varphi)$ if every descendant of x with the same v-attribute verifies φ. The fragment of LRV^{D}_n extended with positive occurrences of $\text{AG}_{\approx v}(\varphi)$ (that is, where $\text{AG}_{\approx}$ occurs always under an even number of negations) is called $\text{LRV}^{\text{D}}_n(\text{AG}^{+}_{\approx})$.

Now, in $\text{LRV}^{\text{D}}_n(\text{AG}^{+}_{\approx})$ one can express: *for every node x containing a rule $q \xrightarrow{\bar{e}_i} q'$, we have that all descendants of x with the same v_i attribute contain a rule of the form $p \xrightarrow{-\bar{e}_i} p'$*. This, added to the property that every increment for component i must verify $v_i \approx \text{EF}v_i$, ensures that the tree indeed encodes a derivation tree.

Theorem 6. $\text{CSReach}(n\text{VASS}_k)$ *is PTime-reducible to* SAT_k-$\text{LRV}^{\text{D}}_1(AG^{+}_{\approx})$.

Proof (idea). We show the idea for $n = 1$, as this case generalizes to any n straightforwardly, and without changing the number of variables in the logic. For every 1VASS_k $\mathcal{C}^k = \langle Q, U, B\rangle$, $q_0 \in Q$ and $\widehat{Q} \subseteq Q$ we define $\varphi \in \text{LRV}^{\text{D}}_1(\text{AG}^{+}_{\approx})$, such that $\text{SAT}_k(\varphi)$ iff $\text{CSReach}(\mathcal{C}^k_\varphi, q_0, \widehat{Q})$. We want this φ to force various properties in all its models, so that every model corresponds to a derivation tree of $\text{CSReach}(\mathcal{C}^k_\varphi, q_0, \widehat{Q})$. In particular, we want:

- Each node is labeled with either a rule of $U \cup B$ or an extra label $*$ for dummy nodes that will be ignored (this is to force exact k-branching for all non-leaves). We can assume without loss of generality that all unary rules in U of the form $q \xrightarrow{c} q'$ have either $c = 1$ or $c = -1$. In particular, formulas φ_{inc} and φ_{dec} express that the label is an increment or a decrement rule, respectively.

- If a node is labeled with an empty rule $q \to \bar{\emptyset}$, then it is a leaf.
- The root is labeled with a rule of the form $(q_0, \dots) \in U \cup B$.
- Each node labeled with an increment rule has a descendant in the same equivalence class (*i.e.* with same value for the only attribute v), and all its descendants in the same equivalence class are labeled with a decrement rule: this can be expressed by $\varphi_{\text{inc}} \to \big(v \approx \text{EF}v \wedge \text{AG}_{\approx v}(\varphi_{\text{dec}})\big)$.

All the above properties, except the last one, can be expressed in LRV^{D}_1; for the last one we use (positively) $\text{AG}_{\approx}$. The final formula φ consists of a conjunction of all these properties, among others (so that the occurrence of $\text{AG}_{\approx}$ remains positive). Then one verifies that a solution to the control-state reachability problem of $\mathcal{C}^k, q_0, \widehat{Q}$ can be used to construct a model for φ; and that a data tree satisfying φ can be used to construct an incrementing derivation tree for $\text{CSReach}(\mathcal{C}^k, q_0, \widehat{Q})$. □

The satisfiability for this extension still has a reduction to the control-state reachability for VASS_k:

Theorem 7. *SAT_k-$\text{LRV}^{\text{D}}(\text{AG}^{+}_{\approx})$ is ExpSpace-reducible to* $\text{CSReach}(\text{VASSR}_k)$.

7 From LRV to MVASS$_k$

The reduction from LRV^{D} to VASS_k from Sect. 5 cannot be extended to treat LRV. The main problem is that the branching nature of the counters in a $\text{CSReach}(\text{VASS}_k)$ will be insufficient to represent some classes of data trees (which can be needed to model some formulas). When we have tests of the form $u_1 \approx \text{EF}u_2$ with $u_1 \neq u_2$ distinct variables, we can no longer reason in terms of "one coordinate i for each variable u_i", where the i-th component in the configuration of the VASS_k codes, intuitively, how many distinct data values must be seen on variable u_i in the subtree as shown in Sect. 5. In fact, when working with LRV, a data value may appear in *several* variables, as a result of allowing formulas like $u_1 \approx \text{EF}u_2 \wedge u_1 \approx \text{EF}u_3$. This means that we need to reason in terms of *sets* of variables, where each component i is associated with a non-empty subset U_i of the variables appearing in the input formula φ; this time, component i counts how many data values must appear in the subtree under all the variables of U_i. This, in principle, poses no problem for the non-branching case: in fact, this kind of coding (indexing one coordinate of the configuration for each subset of variables) was used in [6] to show a reduction from LRV to VASS on data words. However, on data trees, this coding breaks with the semantics of the branching rules of VASS_k.

As an example, suppose we work with two variables u, v and we thus have dimension 3—the first component is associated with $\{u\}$, the second with $\{v\}$ and the third with $\{u, v\}$. Suppose that there are n ancestor nodes that have to satisfy both $u \approx \text{EF}u$ and $u \approx \text{EF}v$, which at the current configuration of the VASS_k is witnessed by the vector $(0, 0, n)$. Intuitively, this means that there are

n data values that must appear in the subtree under a variable u and also under v (though not necessarily at the same node) in the data tree the automaton is trying to find. Hence, as part of the "branching" instruction of this configurations into the configuration of the left and right children, one must contemplate the possibility of obtaining, for instance, $(n, 0, 0)\,(0, n, 0)$, saying that the left subtree contains n distinct data values for u, and the right child contains n data values for v. But it could be $(n - t, 0, t)\,(0, n - t, 0)$, or $(0, 0, n - t)\,(0, 0, t)$, etc. In other words, components need to be mixed in a more complex way that is not allowed in VASS_k branching rules. In particular, some sort of transfers between coordinates must be necessary. This is precisely the behavior that we can encode into MVASS.

Theorem 8. *SAT_k-LRV_n is reducible to* $\text{CSReach}(2^n\text{-MVASSR}_k)$.

Proof (idea). The crux of the reduction is in the use of the *merging* rules. We have one component associated to every non-empty subset. For every non-empty subset U of variables appearing in the input formula φ there is a component i associated to U, let us then define $\bar{e}_U$ as $\bar{e}_i$. Also, let $\bar{e}_\emptyset = \bar{0}$. The idea is that, on rank k, every merging instruction will contain the linear set consisting of every vector $(\bar{e}_U \bar{e}_{V_1} \cdots \bar{e}_{V_{k'}}) \in \mathbb{N}^{n\cdot(k'+1)}$ with $k' \leq k$, so that $U \neq \emptyset$ and $U = \bigcup_i V_i$. The partial order $\preceq$ will then be the subset ordering on the components: $i \preceq j$ if the set associated to i is contained in that associated to j.

Using the merging rules as described above, the reduction from LRV^{D} to VASS_k of Sect. 5 can be modified to obtain a reduction from LRV to MVASS_k. Frames and its notion of validity are extended to treat set of variables. In particular, now the points of increment and decrement are always relative to a set of variables. This follows, very roughly, the idea of coding from [7] in the setup built in Sect. 5, but now some special care must be considered because of the non-linearity of a tree. One must decide in advance to which leaf of the frame the satisfaction of data demands will be delegated. The resulting MVASS_k now has dimension *exponential* in the number of variables of the input formula. □

As a corollary, due to Theorem 2, we have that SAT_k-LRV is decidable. We remark that, similarly as done in [7], one can add formulas of the form $u \star \text{EF}[\varphi]v$ stating that there is a descendant witnessing $u \star \text{EF}v$ and verifying φ, while preserving this reduction.

8 Discussion

We have shown connections between counter systems and data logics on ranked data trees. In particular, this has yielded decision procedures for data logics and a new model of branching computation of VASS.

While in the present work the focus has been put on *ranked* data trees, we envisage working also on unranked trees in the future. In particular, we remark that these logics can be naturally extended to the unranked case, but that there

are no well-known models of branching counter systems with *unbounded* branching. This may lead to new natural models featuring some sort of unbounded parallel computations with good computational properties.

We are also interested in considering other modalities in our logics, with branching tests such as $\mathrm{EX}^i v \star \mathrm{EF} u$ and $\mathrm{EF} u \approx \mathrm{EF} v$, or tests including past such as $u \approx \mathrm{EF}^{-1} v$ and $\mathrm{EF}^{-1} u \approx \mathrm{EF} v$.

We were unable to show the precise complexity of $\mathrm{CSReach}(\mathrm{MVASS}_k)$, which lies between 2ExpTime and 3ExpTime. We leave this for future work. We believe that $\mathrm{SAT}_k\text{-LRV}(\mathrm{AG}^{+}_{\approx})$ is equivalent to the control-state reachability problem for MVASS_k, in the sense of existence of computable reductions from and to.

References

1. Baelde, D., Lunel, S., Schmitz, S.: A sequent calculus for a modal logic on finite data trees. In: 25th EACSL Annual Conference on Computer Science Logic, CSL 29, 1 September 2016, Marseille, France, pp. 32:1–32:16, August 2016
2. Bojańczyk, M., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on data trees and XML reasoning. JACM **56**(3), 13 (2009)
3. Bojańczyk, M., David, C., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on data words. ACM Trans. Comput. Log. 12(4) 2010
4. Bollig, B., Cyriac, A., Gastin, P., Narayan Kumar, K.: Model checking languages of data words. In: Birkedal, L. (ed.) FoSSaCS 2012. LNCS, vol. 7213, pp. 391–405. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28729-9_26
5. Demri, S., D'Souza, D., Gascon, R.: A decidable temporal logic of repeating values. In: Artemov, S.N., Nerode, A. (eds.) LFCS 2007. LNCS, vol. 4514, pp. 180–194. Springer, Heidelberg (2007). doi:10.1007/978-3-540-72734-7_13
6. Demri, S., D'Souza, D., Gascon, R.: Temporal logics of repeating values. J. Log. Comput. **22**(5), 1059–1096 (2012)
7. Demri, S., Figueira, D., Praveen, M.: Reasoning about data repetitions with counter systems. In: LICS, pp. 33–42. IEEE Press (2013)
8. Demri, S., Jurdziński, M., Lachish, O., Lazić, R.: The covering and boundedness problems for branching vector addition systems. J. Comput. Syst. Sci. **79**(1), 23–38 (2013)
9. Demri, S., Lazić, R.: LTL with the freeze quantifier and register automata. ACM Trans. Comput. Log. 10(3) (2009)
10. Emerson, E.A., Halpern, J.Y.: "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. JACM **33**(1), 151–178 (1986)
11. Figueira, D.: Forward-XPath and extended register automata on data-trees. In: ICDT. ACM (2010)
12. Figueira, D.: Alternating register automata on finite data words and trees. Log. Methods Comput. Sci. 8(1) (2012)
13. Figueira, D.: Decidability of downward XPath. ACM Trans. Comput. Log. 13(4) (2012)
14. Figueira, D.: On XPath with transitive axes and data tests. In: PODS, pp. 249–260. ACM (2013)
15. Figueira, D., Figueira, S., Areces, C.: Basic model theory of XPath on data trees. In: ICDT, pp. 50–60. ACM (2014)

16. Figueira, D., Libkin, L.: Pattern logics and auxiliary relations. In: CSL-LICS, pp. 40:1–40:10 (2014)
17. Figueira, D., Segoufin, L.: Future-looking logics on data words and trees. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 331–343. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03816-7_29
18. Figueira, D., Segoufin, L.: Bottom-up automata on data trees and vertical XPath. In: STACS, vol. 9 of LIPIcs, pp. 93–104. LZI (2011)
19. Jacquemard, F., Segoufin, L., Dimino, J.: FO2($<+1, \sim$) on data trees, data tree automata and branching vector addition systems. Log. Methods Comput. Sci. 12(2) (2016)
20. Jurdziński, M., Lazić, R.: Alternating automata on data trees and XPath satisfiability. ACM Trans. Comput. Log. **12**(3), 19 (2011)
21. Kara, A., Schwentick, T., Zeume, T.: Temporal logics on words with multiple data values. In: FST & TCS (2010)
22. Kupferman, O., Vardi, M.: Memoryful branching-time logic. In: LICS, pp. 265–274. IEEE Press (2006)
23. Lazić, R., Sylvain, S.: Nonelementary complexities for branching VASS, MELL, and extensions. ACM Trans. Comput. Log. **16**(3), 20 (2015)
24. Lisitsa, A., Potapov, I.: Temporal logic with predicate λ-abstraction. In: TIME, pp. 147–155. IEEE Press (2005)
25. Neven, F., Schwentick, T., Vianu, V.: Finite state machines for strings over infinite alphabets. ACM Trans. Comput. Log. **5**(3), 403–435 (2004)
26. Pnueli, A.: The temporal logic of programs. In: FOCS, pp. 46–57. IEEE Press (1977)
27. Rackoff, C.: The covering and boundedness problems for vector addition systems. Theoret. Comput. Sci. **6**(2), 223–231 (1978)
28. Segoufin, L.: Automata and logics for words and trees over an infinite alphabet. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 41–57. Springer, Heidelberg (2006). doi:10.1007/11874683_3
29. Verma, K.N., Goubault-Larrecq, J.: Karp-Miller trees for a branching extension of VASS. Discrete Math. Theor. Comput. Sci. **7**(1), 217–230 (2005)