2011 26th Annual IEEE Symposium on Logic in Computer Science

A decidable two-way logic on data words

Diego Figueira
U. of Warsaw & U. of Edinburgh

***Abstract*—We study the satisfiability problem for a logic on data words. A data word is a finite word where every position carries a label from a finite alphabet and a data value from an infinite domain. The logic we consider is two-way, contains *future* and *past* modalities, which are considered as reflexive and transitive relations, and data equality and inequality tests. This logic corresponds to the fragment of XPath with the 'following-sibling-or-self' and 'preceding-sibling-or-self' axes over data words. We show that this problem is decidable, EXPSPACE-complete. This is surprising considering that with the strict (non-reflexive) navigation relations the satisfiability problem is undecidable. To prove this, we first reduce the problem to a derivation problem for an infinite transition system, and then we show how to abstract this problem into a reachability problem of a finite transition system.**

I. INTRODUCTION

We study data words. A data word is a finite string where each position carries a label from a finite alphabet and a data value from some infinite domain. This kind of structure has been considered in the realm of semistructured data, timed automata, program verification, and generally in systems manipulating data values. Finding automata and logics with decidable satisfiability problems over data words is an important quest when studying systems manipulating data values.

Suppose that a data word models a database of 'actions' that were performed throughout the time, where the linear structure of the data word corresponds to the flow of time. Over these structures, one may want to model the behavior of a system, and may want to check that the specified behavior is satisfiable. In other words, that it does not contain any contradiction. The satisfiability problem is crucial to solve this and other static analysis problems.

In this paper we consider a formalism to express properties of data words, where the data values can only be tested for equality or inequality. Logics or automata that can test equality of data values and that can express meaningful properties are necessary. However, they are usually undecidable.

We study a two-way logic on data words and show that it has a decidable EXPSPACE satisfiability problem. We call it *Two-way Path Logic* (or PathLogic for short). PathLogic can be seen as an extended temporal logic [DL09] or as a fragment of XPath [CD99] on data words. It has *future* and *past* modalities to navigate the word and data equality test expressions. For example, it can test the following property of a node:

"there is a node x to the left and two nodes $y < z$ to the right with labels a, b, and c respectively, such that x and z have the same data value". $(\star)$

This property is expressed in PathLogic with the formula

$$\langle\ \overleftarrow{[a]} = \overrightarrow{[b]\cdot[c]}\ \rangle\ .$$

Although we can test the relative appearances of labels, the left and right relations are interpreted as transitive-reflexive relations. Thus, $\langle\ \overleftarrow{[a]} = \overrightarrow{[b]\cdot[b]}\ \rangle$ is equivalent to $\langle\ \overleftarrow{[a]} = \overrightarrow{[b]}\ \rangle$.

Our main result is the following.

Main Theorem. *The satisfiability problem for PathLogic is* EXPSPACE*-complete.*

This logic has links with other logics. Indeed, PathLogic is inspired by the logic for XML documents XPath with data equality tests and the reflexive-transitive axes 'following-sibling-or-self' and 'preceding-sibling-or-self'.[1] We note these axes $\rightarrow^*$ and ${}^*\!\leftarrow$ respectively for economy of space and XPath$(\rightarrow^*, {}^*\!\leftarrow, =)$ to the logic. PathLogic is in fact expressive-equivalent (through an EXPTIME translation) to XPath$(\rightarrow^*, {}^*\!\leftarrow, =)$ interpreted over data words. In fact, PathLogic is a simplification of the syntax of XPath$(\rightarrow^*, {}^*\!\leftarrow, =)$ that preserves only the essential. This has also the advantage of simplifying the decidability proofs. In terms of XPath our result is translated as follows.

Theorem 1. *The satisfiability problem for* XPath$(\rightarrow^*, {}^*\!\leftarrow, =)$ *is in* 2EXPSPACE*.*

This may be surprising given the known results in this area. In fact, [FS09] shows that XPath$(\rightarrow^+, =)$ is decidable with non-primitive recursive complexity. This is the fragment that contains the non-reflexive 'following-sibling' axis. What is more, XPath$(\rightarrow^+, {}^+\!\leftarrow, =)$ or even XPath$(\rightarrow^*, {}^+\!\leftarrow, =)$ are undecidable. Simply reducing the navigation by having reflexive-transitive relations instead of only transitive, we turn an undecidable problem into an elementarily decidable one. Thus, the $\rightarrow^*$ and ${}^*\!\leftarrow$ axes behave surprisingly much better than $\rightarrow^+$ and ${}^+\!\leftarrow$. This result opens some promising conjectures on the decidability of several fragments of XPath on XML documents.

Decidable two-way logics on data words are scarce. Very often two-way logics on data words have an undecidable satisfiability problem. One such prominent example is the case of $\mathrm{LTL}_1^{\downarrow}(\mathrm{F}, \mathrm{F}^{-1})$, a temporal logic with with future and past modalities and one register to store and compare data values. Whether we interpret the F and F^{-1} modalities as either reflexive or strict, its satisfiability problem remains undecidable. Further, if only one of these modalities is allowed, it becomes decidable, but with non-primitive recursive

Work supported by the FET-Open grant agreement FOX (FP7-ICT-233599).

[1]Strictly speaking, these axes do not exist in XPath [CD99]. They must be interpreted as the reflexive version of the 'following-sibling' and 'preceding-sibling' axes respectively.

complexity [DL09], [FS09]. PathLogic can also be seen as a fragment of $\text{LTL}_1^{\downarrow}(\text{F}, \text{F}^{-1})$, as we will discuss later on.

Related work

This paper addresses a topic related to the work of [FS09], about the complexity and decidability of logics for data words and trees with transitive relations. [FS09] showed lower bounds for several fragments of $\text{LTL}_1^{\downarrow}$ and XPath with transitive relations for navigation. In particular, it was shown that all fragments of XPath with data equality tests containing transitive and non-reflexive axes $\rightarrow^+$, ${}^+\!\leftarrow$ or $\uparrow^+$ have a satisfiability problem of non-primitive recursive complexity. To obtain such results it was necessary to study the behavior of XPath on non-branching structures, *i.e.*, on *data words*. The present paper also centers on data words, showing this time a positive result with the spirit of obtaining in the future decidability results for fragments of XPath over XML documents.

Bojańczyk et al. [BDM+10] study the satisfiability problem for first order logics with two variables on data words. In particular they study the fragment $\text{FO}^2(<, \sim)$. This is first-order logic restricted to two variables, a binary relation $<$ which corresponds to the order in the word, and a binary relation $\sim$ to test for data equality or inequality of nodes. They show that the satisfiability problem is decidable, NExpTime-complete. In fact, they show that the decidability is also preserved if we allow any predicate '$+k$', relating any two nodes at distance k in the data word.[2] This logic is in some sense two-way, since it can navigate both in the future and past directions. Since it also includes equality and inequality of nodes, the transitive $<$ relation and the reflexive-transitive relation $\leq$ are one definable in terms of the other. However close, $\text{FO}^2(<, \sim)$ is incomparable with PathLogic in terms of expressive power. This decidability result was extended to $\text{FO}^2(<, \precsim)$, where $x \precsim y$ tests that x carries a data value smaller or equal to that of y. The satisfiability of this logic is ExpSpace [SZ10].

Demri and Lazić [DL09] study the already mentioned logic $\text{LTL}_1^{\downarrow}$, which is a two-way temporal logic on data words. It is based on the linear temporal logic LTL with special constructs to store and test for equality of data values. This logic with Future, Next and Until modalities is shown to be decidable [DL09], but any fragment with Future and Past modalities becomes undecidable [FS09]. Nevertheless, [DL09] identifies a decidable fragment that is equivalent to $\text{FO}^2(<, \sim, +1, \ldots, +k)$, and hence decidable. [DDG07] and [KSZ10] also investigate decidable two-way logics but this time over *multi-attributes data words*, were every position carries a *vector* of data values. These logics are intimately related with FO^2 and they are hence also incomparable to PathLogic in expressive power.

Organization: In Section II we introduce our logic PathLogic and we briefly comment on the relation with XPath, $\text{LTL}_1^{\downarrow}$, and FO^2. Sections III, IV and V are dedicated to prove our main result, that the satisfiability problem of PathLogic is in ExpSpace. In Section VI we discuss about the lower bound. Section VII contains some corollaries of our result in terms of XPath fragments. Finally, in Section VIII we propose some conjectures and future lines of work.

[2]For this logic no primitive-recursive upper bound is known.

II. A two-way logic

Notation: We consider a finite word over $\mathbb{E}$ as a function $\mathbf{w} : [n] \to \mathbb{E}$ for some $n > 0$, where $[n] = \{1, \ldots, n\}$. We define the set of words as $\mathit{Words}(\mathbb{E}) := \{\mathbf{w} : [n] \to \mathbb{E} \mid n > 0\}$. We write $\mathit{pos}(\mathbf{w}) = \{1, \ldots, n\}$ to denote the set of **positions** (that is, the domain of $\mathbf{w}$). Given $\mathbf{w} \in \mathit{Words}(\mathbb{E})$ and $\mathbf{w}' \in \mathit{Words}(\mathbb{F})$ with $\mathit{pos}(\mathbf{w}) = \mathit{pos}(\mathbf{w}') = P$, we write $\mathbf{w} \otimes \mathbf{w}' \in \mathit{Words}(\mathbb{E} \times \mathbb{F})$ for the word such that $\mathit{pos}(\mathbf{w} \otimes \mathbf{w}') = P$ and $(\mathbf{w} \otimes \mathbf{w}')(x) = (\mathbf{w}(x), \mathbf{w}'(x))$. A **data word** is a word $\mathbf{w} = \mathbf{a} \otimes \mathbf{d} \in \mathit{Words}(\mathbb{A} \times \mathbb{D})$, where $\mathbb{A}$ is a finite alphabet of letters and $\mathbb{D}$ is an infinite domain. We use $\mathbb{A}$ for a finite alphabet, $\mathbb{D}$ as an infinite domain, and we refer to its elements with the letters $a, b, c \in \mathbb{A}$ and $d, e \in \mathbb{D}$. We use $\mathbf{w}$, $\mathbf{a}$ and $\mathbf{d}$ for words of $\mathit{Words}(\mathbb{A} \times \mathbb{D})$, $\mathit{Words}(\mathbb{A})$ and $\mathit{Words}(\mathbb{D})$ respectively. We use loosely the term *node* to denote the pair of label and data value of a position. We write $\wp(S)$ and $\wp_{<\infty}(S)$ to denote the powerset and finite powerset of a set S respectively.

We work with a logic with a simple syntax and same expressive power as $\text{XPath}(\rightarrow^*, {}^*\!\leftarrow, =)$. We call this logic PathLogic. This is an equivalent formalism which is more convenient to work with for the proofs presented here.

PathLogic is a two-sorted logic with *path expressions* (that we note α, β) and *node expressions* (φ, ψ). A path expression is a sequence of node expressions, noted $\alpha = [\psi_1] \cdot \cdots \cdot [\psi_n]$ and we say that a position i of the data word reaches another position j through $\overrightarrow{\alpha}$ if there are witness positions $i \leq i_1 \leq \cdots \leq i_n = j$ where ψ_k holds true at position i_k, for every k. On the other hand, a node expression is a boolean combination of tests for labels and formulas of the form $\langle \overleftarrow{\alpha} = \overrightarrow{\beta} \rangle$. Such a formula demands the existence of one position to the left and one to the right reachable by $\overleftarrow{\alpha}$ and $\overrightarrow{\beta}$ respectively, that carry the *same* data value. Analogously, $\langle \overleftarrow{\alpha} \neq \overrightarrow{\beta} \rangle$ expresses the same property except that the positions must carry *different* data values. Expressions are formally defined as follows.

$$\begin{aligned} \varphi, \psi \; &::= \; \langle \overleftarrow{\alpha} = \overrightarrow{\beta} \rangle \mid \langle \overleftarrow{\alpha} \neq \overrightarrow{\beta} \rangle \mid a \mid \neg\varphi \mid \varphi \vee \psi \\ \alpha, \beta \; &::= \; \varepsilon \mid [\varphi] \cdot \alpha \end{aligned}$$

And semantics are defined according to the intuition we just gave. See Figure 1 for a precise definition.

In this context, we define $\mathbf{w}, i \models \varphi$ iff $i \in [\![\varphi]\!]^{\mathbf{w}}$, and $\mathbf{w} \models \varphi$ iff $\mathbf{w}, 1 \models \varphi$. We also define the test $\langle \overrightarrow{\alpha} \rangle$, which checks if there is a node reachable by α, as $\langle \overrightarrow{\alpha} = \overrightarrow{\alpha} \rangle$, and likewise for $\langle \overleftarrow{\alpha} \rangle$. The **satisfiability problem** is the problem of, given $\varphi \in$ PathLogic, whether there is a data word $\mathbf{w}$ such that $\mathbf{w} \models \varphi$. Our main result is that this problem is decidable.

$$
\begin{aligned}
[\![a]\!]^{\mathbf{w}} &\overset{def}{=} \{i \in pos(\mathbf{w}) \mid \mathbf{a}(i) = a\}\\
[\![\varphi \vee \psi]\!]^{\mathbf{w}} &\overset{def}{=} [\![\varphi]\!]^{\mathbf{w}} \cup [\![\psi]\!]^{\mathbf{w}}\\
[\![\neg\varphi]\!]^{\mathbf{w}} &\overset{def}{=} pos(\mathbf{w}) \setminus [\![\varphi]\!]^{\mathbf{w}}\\
[\![\langle \overleftarrow{\alpha} = \overrightarrow{\beta} \rangle]\!]^{\mathbf{w}} &\overset{def}{=} \{i \in pos(\mathbf{w}) \mid \exists j,k \in pos(\mathbf{w}),\\
&\qquad (i,j) \in [\![\alpha]\!]^{\mathbf{w}}_{\leftarrow}, (i,k) \in [\![\beta]\!]^{\mathbf{w}}_{\rightarrow}, \mathbf{d}(j) = \mathbf{d}(k)\}\\
[\![\langle \overleftarrow{\alpha} \neq \overrightarrow{\beta} \rangle]\!]^{\mathbf{w}} &\overset{def}{=} \{i \in pos(\mathbf{w}) \mid \exists j,k \in pos(\mathbf{w}),\\
&\qquad (i,j) \in [\![\alpha]\!]^{\mathbf{w}}_{\leftarrow}, (i,k) \in [\![\beta]\!]^{\mathbf{w}}_{\rightarrow}, \mathbf{d}(j) \neq \mathbf{d}(k)\}\\
[\![\varepsilon]\!]^{\mathbf{w}}_{\rightarrow} &\overset{def}{=} [\![\varepsilon]\!]^{\mathbf{w}}_{\leftarrow} = \{(i,i) \mid i \in pos(\mathbf{w})\}\\
[\![[\varphi]\cdot\alpha]\!]^{\mathbf{w}}_{\rightarrow} &\overset{def}{=} \{(i,j) \in (pos(\mathbf{w}))^2 \mid \exists k \geq i \text{ s.t.}\\
&\qquad k \in [\![\varphi]\!]^{\mathbf{w}}, (k,j) \in [\![\alpha]\!]^{\mathbf{w}}_{\rightarrow}\}\\
[\![[\varphi]\cdot\alpha]\!]^{\mathbf{w}}_{\leftarrow} &\overset{def}{=} \{(i,j) \in (pos(\mathbf{w}))^2 \mid \exists k \leq i \text{ s.t.}\\
&\qquad k \in [\![\varphi]\!]^{\mathbf{w}}, (k,j) \in [\![\alpha]\!]^{\mathbf{w}}_{\leftarrow}\}
\end{aligned}
$$

Fig. 1. Semantics of PathLogic for a data word $\mathbf{w} = \mathbf{a}\otimes\mathbf{d}$.

Comparison with other logics

a) XPath*:* PathLogic corresponds precisely to XPath($^*\!\leftarrow, \rightarrow^*, =$) in terms of expressiveness. The only difference is that the syntax is simplified. While XPath($^*\!\leftarrow, \rightarrow^*, =$) may change direction inside a path expression, PathLogic cannot. But in fact this does not change the expressive power: we can always factorize path expressions with mixed axes into formulas whose path expressions use only one axis. Thus, there is an EXPTIME translation from XPath($^*\!\leftarrow, \rightarrow^*, =$) into PathLogic.

b) $\mathrm{LTL}_1^{\downarrow}$*:* PathLogic also corresponds to the *simple* fragment of $\mathrm{LTL}_1^{\downarrow}$ as defined in [FS09][3], denoted by $\mathrm{sLTL}_1^{\downarrow}(\mathrm{F},\mathrm{F}^{-1})$.[4] Here, F and F^{-1} must be interpreted as reflexive-transitive relations. The logic $\mathrm{sLTL}_1^{\downarrow}(\mathrm{F},\mathrm{F}^{-1})$ is equivalent to XPath($^*\!\leftarrow, \rightarrow^*, =$) (and hence also to PathLogic) in terms of expressive power. As shown in [FS09], there is a PTIME translation in both directions.

c) FO^2*:* As already mentioned, $\mathrm{FO}^2(<,\sim)$ is incomparable with PathLogic. Indeed, $\mathrm{FO}^2(<,\sim)$ can express that the word contains at least two nodes, while we cannot express this property using only *reflexive*-transitive modalities. On the other hand, PathLogic can easily express a property like "all nodes labeled a verify the property ($\star$)", that cannot be expressed in $\mathrm{FO}^2(<,\sim)$.[5]

III. THE SATISFIABILITY PROBLEM

We solve the satisfiability problem through a series of two reductions. First, we reduce the problem into a derivation problem for an infinite transition system $\rightarrowtail$. And second, we abstract this transition system, reducing this problem into a similar problem for a *finite* transition system $\rightharpoonup_K$. In truth, $\rightharpoonup_K$ is not a finite transition system, but we will see that it can be easily restricted to a finite one.

Organization: In this section we present the general outline of the decidability proof, and we show our first reduction to a derivation problem for $\rightarrowtail$. Section IV contains the most difficult result, namely the reduction of this problem into the derivation problem for $\rightharpoonup_K$. Finally, in Section V we solve the derivation problem for $\rightharpoonup_K$ with an EXPSPACE procedure.

A. Outline of the proof

We first reduce the problem of testing whether some formula φ is satisfiable into a derivation problem for an infinite state transition system $\rightarrowtail$ that depends on φ. This problem consists in testing whether there exists a finite sequence $\mu_1 \rightarrowtail \cdots \rightarrowtail \mu_n$ such that μ_1 and μ_n satisfy certain local properties. The domain of this transition system is the set of *mosaics*: abstractions of positions of a data word, containing all the data values that can be found to the right and to the left, and further with which path expressions of the logic they can be reached. The transition system $\rightarrowtail$ relates any two mosaics that can be *matched*: that is, that could abstract two positions of a data word that are one next to another. We show that from a derivation $\mu_1 \rightarrowtail \cdots \rightarrowtail \mu_n$ with some good properties, one can build a data word of length n that satisfies the input formula φ. In turn, if there is no such derivation, φ is unsatisfiable. To obtain this reduction, we work with formulas of PathLogic in a certain normal form. This normal form enables to have a simple definition of the transition $\rightarrowtail$ specially convenient for our proofs.

Next, in Section IV, we solve the derivation problem for $\rightarrowtail$ by a reduction to a similar derivation problem for another transition system $\rightharpoonup_K$. This transition system is parametrized by a set K of data values. The definition of $\rightharpoonup_K$ introduces an important concept of *rigid* data values, which are data values that play a determined function in a sequence $\mu_1 \rightarrowtail \cdots \rightarrowtail \mu_n$. We denote by K such data values of important interest. Based on this concept we define a quasi-order $\leq_K$ over the set of mosaics, which is monotone with respect to $\rightarrowtail$. This monotonicity serves to finally show the reduction from derivation problem for $\rightarrowtail$ to the derivation problem for $\rightharpoonup_K$.

The derivation problem for $\rightharpoonup_K$ may seem in appearance more difficult than for $\rightarrowtail$, since we are dealing now with a *family* of transition systems $\rightharpoonup_K$, one for every K. Nevertheless, in Section V we show that the parameter K is harmless and we can bound and fix a value for it. Also, we will see that we only need to work with the minimal classes of equivalences of $\leq_K$ which are also finite and bounded. These observations give us indeed a *finite* transition system. Thus, we can use a standard reachability algorithm that solves the derivation problem for $\rightharpoonup_K$ and hence the satisfiability problem for PathLogic.

Next, we define the normal form for PathLogic and the transition system $\rightarrowtail$, together with some fundamental concepts and notation that we use throughout. We then state the target problem for the reduction.

[3]A formula of $\mathrm{LTL}_1^{\downarrow}$ in negated normal form is said to be *simple* if (i) there is at most one occurrence of $\uparrow$ within the scope of each occurrence of $\downarrow$ and, (ii) there is no negation between an occurrence of $\uparrow$ and its matching $\downarrow$, except maybe immediately before $\uparrow$. We denote by $\mathrm{sLTL}_1^{\downarrow}$ the fragment of $\mathrm{LTL}_1^{\downarrow}$ containing only simple formulas.

[4]This fragment has no connection with the *simple* fragment of $\mathrm{LTL}_1^{\downarrow}$ defined in [DL09].

[5]To express this property we would need 3 variables. (And $\mathrm{FO}^3(<,\sim)$ is undecidable since $\mathrm{FO}^3(<,+1,\sim)$ is undecidable [BDM+10, §9].)

B. A normal form

We will assume a certain normal form of the formula φ to test for satisfiability. This will simplify the necessary machinery to solve our problem. This normal form consists simply of having formulas without nesting of data tests. That is, we avoid treating formulas like

$$\langle \ \overleftarrow{[\ \langle\ \overleftarrow{[a]} = \overrightarrow{[b]}\ \rangle\ ]} \ = \ \overrightarrow{[c]\cdot[d]} \ \rangle \ .$$

Let us write $BC(labels)$ to denote all the formulas which are boolean combinations of tests for labels of $\mathbb{A}$. If a formula is such that all its path expressions α contain only tests for labels (i.e., $\alpha \in (BC(labels))^*$ abusing notation) we call it a **non-recursive** formula.

In the normal form we suppose that $\varphi = \varphi_1 \wedge \varphi_2$ where φ_1 is a non-recursive formula and φ_2 is a conjunction of tests of the form "a node satisfies ψ iff it has some of the labels $\{a_1, \dots, a_n\}$" for some non-recursive formula ψ and labels $a_1, \dots, a_n \in \mathbb{A}$. Formally, φ_2 contains a conjunction of tests of the form

$$\neg\langle \ \overrightarrow{[\xi \wedge \neg\psi]} \ \rangle \wedge \neg\langle \ \overrightarrow{[\neg\xi \wedge \psi]} \ \rangle$$

for ξ a disjunction of labels and ψ a non-recursive formula. Then, we obtain the following.

Lemma 1 (normal form)**.** *There is an exponential-time translation that for every formula $\eta \in$ PathLogic returns a formula φ in normal form such that η is satisfiable iff φ is satisfiable.*

Proof: Given a formula η we define the alphabet $\mathbb{A}_\varphi$ of the translation φ as all the locally consistent sets of subformulas of η. That is, the sets S such that for every subformula ψ of η: (1) if $\psi = \neg\psi'$ then $\{\psi', \neg\psi'\} \not\subseteq S$; (2) if $\psi = \psi' \wedge \psi''$ then $\psi \in S$ iff $\{\psi', \psi''\} \subseteq S$; and (3) if $\psi = \psi' \vee \psi''$ then $\psi \in S$ iff $\psi' \in S$ or $\psi'' \in S$.

Given a formula ψ, $tr(\psi)$ denotes the result of replacing every instance of a path expression $[\psi_1]\cdot\dots\cdot[\psi_n]$ in ψ (which does not appear nested inside another path expression) by $[\zeta_{\psi_1}]\cdot\dots\cdot[\zeta_{\psi_n}]$, and every test for label a (which does not appear inside a path expression) by ζ_a, where $\zeta_\psi = \bigvee_{S\in\mathbb{A}_\varphi, \psi\in S} S$.

To build the formula $\varphi = \varphi_1 \wedge \varphi_2$ in normal form, we define $\varphi_1 = tr(\eta)$, and we build φ_2 as a conjunction of formulas

$$\neg\langle\overrightarrow{[\zeta_\psi \wedge \neg tr(\psi)]}\rangle \quad \wedge \quad \neg\langle\overrightarrow{[\neg\zeta_\psi \wedge tr(\psi)]}\rangle$$

for all subformulas ψ of η. It is easy to see that this translation preserves satisfiability. ■

We fix once and for all $\mathbb{A}$ as the finite alphabet and $\mathbb{D}$ as the infinite domain we are going to work with. Given $\varphi = \varphi_1 \wedge \varphi_2$ in normal form, we write Ω_φ to denote all non-recursive subformulas of φ, and $\gamma_\varphi : \mathbb{A} \to \wp(\Omega_\varphi)$ to denote the function where $\gamma_\varphi(a)$ is the set of formulas ψ such that φ_2 contains $\neg\langle\overrightarrow{[\xi \wedge \neg tr(\psi)]}\rangle$ as a subformula, for some disjunctive formula ξ containing a. Finally, we define $\Omega^p_\varphi = \{\alpha \in \Omega_\varphi \mid \alpha \text{ is a path expression different from } \varepsilon\}$. We exclude ε from Ω^p_φ simply because we are interested in those path expressions that are 'moving', unlike ε.

In future sections, we will reduce the satisfiability for a formula in normal form into an ExpSpace problem. This, together with Lemma 1, immediately yields a 2ExpSpace decision procedure. However, the satisfiability for PathLogic is in ExpSpace, since the algorithm is doubly exponential only in the size of path subformulas. By the following observation, we will obtain an ExpSpace algorithm for arbitrary PathLogic formulas.

Corollary 1. *About the translation of Lemma 1:*

1. *The set of path subformulas Ω^p_φ resulting from the translation has cardinality polynomial in η.*
2. *Every path subformula of Ω^p_φ can be written using polynomial space.*

Proof of Corollary 1: The blowup in the exponential translation comes only from the formulas ζ_ψ. In fact, φ can be symbolically written in polynomial space just as we did, using a symbol ζ_ψ instead of a big exponential disjunction. Remark that testing whether a label $S \in \mathbb{A}_\varphi$ satisfies ζ_ψ reduces to testing $\psi \in S$. ■

C. The transition system

Given a formula φ in normal form, we show an abstraction of the important information that we need to keep track of in a model that satisfies φ. Each piece of information is called a **mosaic**. A mosaic maintains all necessary facts about some position i of a data word $\mathbf{w}$ such that $\mathbf{w} \models \varphi$, namely

- the label and datum of the current position,
- the data values that are to be found to the right of i, and with which paths of Ω^p_φ are reachable,
- and likewise for the data values to the left.

We say that a mosaic is the **φ-abstraction** of a position i of a data word $\mathbf{w}$ if it contains all the aforementioned details of the position. A mosaic that abstracts a position i of $\mathbf{w}$ contains enough information to test whether ψ is satisfied at position i (*i.e.*, $\mathbf{w}, i \models \psi$) for any subformula ψ of φ. In order to be a valid mosaic, the mosaic must always comply with $\gamma_\varphi(a)$ where $a \in \mathbb{A}$ is the label of the mosaic.

Given two mosaics μ_1, μ_2, we can check if they *match*, that one can be next to the other, that there are no incompatibilities. For example, suppose that μ_1 is the φ-abstraction of a position i. If μ_1 asserts that there is a data value d different from the current one that can be reached with $[a]$ navigating to the right, then the mosaic that abstracts $i+1$ must also contain the information that d can be reached with $[a]$. Otherwise these two mosaics would not match. We will write $\rightarrowtail$ for this matching relation, and $\rightarrowtail$ **sequence** for any sequence $\mu_1 \rightarrowtail \cdots \rightarrowtail \mu_n$. (In general, a $\to$ sequence denotes $\mu_1 \to \cdots \to \mu_n$ for any transition system $\to$.)

Indeed, the φ-abstraction of the positions of a data word w and formula φ such that $w \models \varphi$ verify that they are a $\rightarrowtail$ sequence. Moreover, the leftmost mosaic must not contain information of data values to the left simply because there are no more elements to the left. For the same reasons, the rightmost mosaic must not contain information on the data values to the right. We call such mosaics **left-complete** and **right-complete** respectively. A $\rightarrowtail$ sequence is **complete** if the leftmost element is left-complete and the rightmost element is right-complete. We show that from any complete $\rightarrowtail$ sequence for φ whose first element verifies φ, we can build a data word satisfying φ. And in turn, the sequence of φ-abstractions of a word that satisfies φ is indeed a complete $\rightarrowtail$ sequence for φ. Now the satisfiability problem reduces to the problem of testing if there is a complete $\rightarrowtail$ sequence whose first element satisfies φ. We call this the **derivation problem**.

Definitions: Let us now turn to the basic definitions. An important object we will work with is that of a *profile*. A profile of a data value d is a subset of Ω^p_φ describing all the possible ways to reach d in a data word $\mathbf{w}$ from a position $i \in pos(\mathbf{w})$ by going in one direction (either to the right or to the left). We use $\pi, \rho \subseteq \Omega^p_\varphi$ as names for profiles.

We define some useful operations on profiles. There is a constant profile $\boldsymbol{\sigma}_a$ for every $a \in \mathbb{A}$, which consists in all the path expressions that can be satisfied locally at a certain node, assuming that the node's label is a. (Note that $\boldsymbol{\sigma}_a$ is a constant whereas π, ρ are names.)

$$\boldsymbol{\sigma}_a \stackrel{def}{=} \{\alpha \in \Omega^p_\varphi \mid \alpha = [\psi_1]\cdot \cdots \cdot[\psi_n], \forall i : a \models \psi_i\} \qquad a \in \mathbb{A}$$

where $a \models \psi_i$ means that ψ_i is verified by the label a—remember that $\psi_i \in BC(labels)$. Also, given a profile π and a label $a \in \mathbb{A}$, we write $a\pi$ for the concatenation of the profiles of $\boldsymbol{\sigma}_a$ and π.

$$a\pi \stackrel{def}{=} \pi \cup \{\alpha\cdot\beta \in \Omega^p_\varphi \mid \alpha \in \boldsymbol{\sigma}_a, \beta \in \pi\}$$

Finally, we write $\pi + a$ for the union of the profiles π and $\boldsymbol{\sigma}_a$.

$$\pi + a \stackrel{def}{=} \pi \cup \boldsymbol{\sigma}_a$$

Remark 1. *These are idempotent and commutative operations: $a(a\pi) = a\pi$; $(\pi + a) + a = \pi + a$; $a(\pi + a) = a\pi + a$.*

Definition 1 (mosaic)**.** *A **mosaic** over φ is a tuple $\mu = (a, d, \overleftarrow{\chi}, \overrightarrow{\chi})$ where $a \in \mathbb{A}$, $d \in \mathbb{D}$, and $\overleftarrow{\chi}, \overrightarrow{\chi} \in \wp_{<\infty}(\mathbb{D} \times \Omega^p_\varphi)$. The idea is that $\overleftarrow{\chi}$ is contains the profiles of the data values that can be found to the left and $\overrightarrow{\chi}$ of those to the right.*

We use letters μ, ν for mosaics and we fix the nomenclature $\mu_i = (a_i, d_i, \overleftarrow{\chi}_i, \overrightarrow{\chi}_i)$ for every i. We adopt the following notation for $\chi \in \{\overleftarrow{\chi}, \overrightarrow{\chi}\}$, $\alpha \in \Omega^p_\varphi$, and $e \in \mathbb{D}$.

$$\chi(e) \stackrel{def}{=} \{\alpha \in \Omega^p_\varphi \mid (e, \alpha) \in \chi\} \quad \mu(e) \stackrel{def}{=} (\overleftarrow{\chi}(e), \overrightarrow{\chi}(e))$$
$$\chi(\alpha) \stackrel{def}{=} \{e \in \mathbb{D} \mid (e, \alpha) \in \chi\}$$

We call $\mu(e)$ *the profile of* e *in* μ. Note that we use *profile* to denote either a subset of Ω^p_φ or a pair of subsets of Ω^p_φ. The intended meaning will always be clear from the context.

Given a formula $\varphi = \varphi_1 \wedge \varphi_2$ in normal form, a data word $\mathbf{w}$ and a position i, we say that a mosaic μ over φ is the φ**-abstraction** of $(\mathbf{w}, i)$ if it is defined in the expected way by taking into account the data values of $\mathbf{w}$ and the path expressions Ω^p_φ. That is, if $\mathbf{w} = \mathbf{a}\otimes\mathbf{d}$ and $\mu = (a, d, \overleftarrow{\chi}, \overrightarrow{\chi})$ we define $(a, d) = \mathbf{w}(i)$, $\overrightarrow{\chi} = \{(e, [\psi_1]\cdot \cdots \cdot[\psi_n]) \in \mathbb{D} \times \Omega^p_\varphi \mid$ there are $i \le j_1 \le \cdots \le j_n \le |\mathbf{w}|$ where $\mathbf{a}(j_k) \models \psi_k$ for all k and $\mathbf{d}(j_n) = e\}$. Likewise for $\overleftarrow{\chi}$.

For every atomic expression $\langle\alpha = \beta\rangle$, $\langle\alpha \neq \beta\rangle$ or a from Ω_φ one can decide if the formula holds at position i of $\mathbf{w}$ simply by inspecting the φ-abstraction of $(\mathbf{w}, i)$. Hence, one can define a satisfaction relation $\vdash$, where the following holds.

Lemma 2. *For every formula φ in normal form and subformula $\psi \in \Omega_\varphi$ and for every data word $\mathbf{w}$ and position i, $\mathbf{w}, i \models \psi$ iff $\mu \vdash \psi$, where μ is the φ-abstraction of $(\mathbf{w}, i)$.*

Not all mosaics are consistent with the φ-abstraction. For example, a mosaic over φ with a label $a \in \Omega_\varphi$ such that $\overrightarrow{\chi}([a]) = \emptyset$ cannot be the φ-abstraction of any $(\mathbf{w}, i)$: $\overrightarrow{\chi}([a])$ *must* contain the mosaic's datum.

Definition 2 (validity)**.** *A mosaic $(a, d, \overleftarrow{\chi}, \overrightarrow{\chi})$ is **valid** if it satisfies the following restrictions for all $\chi \in \{\overleftarrow{\chi}, \overrightarrow{\chi}\}$*

1. *for every $e \in \mathbb{D}$, $\chi(e)$ is suffix closed: if $\alpha\cdot\beta \in \chi(e)$ then $\beta \in \chi(d')$,*
2. *$\overrightarrow{\chi}(d) = a\pi + a$ and $\overleftarrow{\chi}(d) = a\rho + a$ for some $\pi, \rho \subseteq \Omega^p_\varphi$,*
3. *for every $e \in \mathbb{D}$, $\chi(e) = a\pi$ for some $\pi \subseteq \Omega^p_\varphi$, and*
4. *$\mu \vdash \bigwedge_{\psi \in \gamma_\varphi(a)} \psi$.*

Finally, $\mathcal{M}(\varphi)$ is the set of all valid mosaics over φ. Henceforth we write *mosaic* to denote a *valid* mosaic unless otherwise stated. Observe that the φ-abstraction of a data word $\mathbf{w}$ and position i results always in a valid mosaic.

Lemma 3. *The φ-abstraction of $(\mathbf{w}, i)$ is a valid mosaic of $\mathcal{M}(\varphi)$, assuming $\mathbf{w} \models \varphi$ and $i \in pos(\mathbf{w})$.*

The importance of the normal form presented earlier is that it allows us to work separately on each data value. The left profile of a data value at position $i + 1$ depends solely on the profile of that data value at position i, and the labels of these positions. In fact, all the interaction between the profiles of different data values is encapsulated in the restrictions imposed by the function γ_φ. This is due to the fact that all path expressions contained in Ω^p_φ depend only on the label of the mosaic. We can hence define a relation that tests whether it is possible that two profiles of a data value can abstract two consecutive positions.

We define the relations $\overrightarrow{\mathbf{m}}$, $\overleftarrow{\mathbf{m}}$ that evidence this dependency of profiles for a given data value d. Intuitively, the first three components of the relation specify the situation for d on the left position: the label of the node, whether d is equal (1) or not (0) to the current data value, and its profile ($\overrightarrow{\chi}(d)$ or $\overleftarrow{\chi}(d)$ respectively for $\overrightarrow{\mathbf{m}}$, $\overleftarrow{\mathbf{m}}$). The other three components contain a similar description for the same data value on the right position. The idea is that $\overrightarrow{\mathbf{m}}$ and $\overleftarrow{\mathbf{m}}$ hold if it is possible to have this situation in a data word.

$$\overrightarrow{\mathbf{m}}(a, 1, \pi_1, b, j, \pi_2) \overset{def}{\Leftrightarrow} \pi_1 = a\pi_2 + a \tag{1}$$

$$\overrightarrow{\mathbf{m}}(a, 0, \pi_1, b, j, \pi_2) \overset{def}{\Leftrightarrow} \pi_1 = a\pi_2 \tag{2}$$

$$\overleftarrow{\mathbf{m}}(a, i, \rho_1, b, j, \rho_2) \overset{def}{\Leftrightarrow} \overrightarrow{\mathbf{m}}(b, j, \rho_2, a, i, \rho_1) \tag{3}$$

Using $\overrightarrow{\mathbf{m}}$ and $\overleftarrow{\mathbf{m}}$, we define $\mathbf{m}(a, i, (\rho_1, \pi_1), b, j, (\rho_2, \pi_2))$ iff $\overrightarrow{\mathbf{m}}(a, i, \pi_1, b, j, \pi_2)$ and $\overleftarrow{\mathbf{m}}(a, i, \rho_1, b, j, \rho_2)$. We use the expression $d \overset{?}{=} d'$ to denote 1 if $d = d'$ or 0 otherwise. Now we can define the transition system $\rightarrowtail$:

Definition 3 ($\rightarrowtail$)**.** *We say that μ_1 and μ_2 **match**, and we write it $\mu_1 \rightarrowtail \mu_2$, iff for all $d \in \mathbb{D}$*

$$\mathbf{m}(a_1,\, d_1 \overset{?}{=} d,\, \mu_1(d),\, a_2,\, d_2 \overset{?}{=} d,\, \mu_2(d)) \ .$$

We define the necessary properties that the abstractions of the borders (*i.e.*, of the first and last elements) should admit.

Definition 4. *We define the left and right demands of a mosaic $\mu = (a, d, \overleftarrow{\chi}, \overrightarrow{\chi})$. A left- (resp. right-) demand consists of the data values that necessarily have to be found to the left (resp. to the right).*

$$\overleftarrow{dem}(\mu) \overset{def}{=} \{(e, \rho) \in \overleftarrow{\chi} \mid (e \neq d \wedge \rho \neq \emptyset) \vee (e = d \wedge \rho \neq \boldsymbol{\sigma}_a)\}$$

$$\overrightarrow{dem}(\mu) \overset{def}{=} \{(e, \pi) \in \overrightarrow{\chi} \mid (e \neq d \wedge \pi \neq \emptyset) \vee (e = d \wedge \pi \neq \boldsymbol{\sigma}_a)\}$$

We say that a mosaic is left-complete (resp. right-complete) if it does not have left-demands (resp. right-demands).

Definition 5. *A $\rightarrowtail$ **sequence** is a sequence of mosaics $\mu_1, \ldots, \mu_n$ such that $\mu_i \rightarrowtail \mu_{i+1}$ for all $i < n$. We say that the sequence is **complete** if $\overleftarrow{dem}(\mu_1) = \overrightarrow{dem}(\mu_n) = \emptyset$.*

We state the derivation problem for a transition system $\rightarrow$.

	The derivation problem for $\rightarrow$
INPUT:	A formula $\varphi = \varphi_1 \wedge \varphi_2$ in normal form.
OUTPUT:	Is there a complete $\rightarrow$ sequence over $\mathcal{M}(\varphi)$ whose leftmost mosaic μ verifies $\mu \vdash \varphi_1$?

Note that any data word corresponding to a complete $\rightarrow$ sequence over $\mathcal{M}(\varphi)$ always satisfies φ_2. This is because $\mathcal{M}(\varphi)$ contains only valid mosaics, satisfying condition 4 of Definition 2.

D. From the satisfiability problem to the derivation problem

We can reduce the satisfiability problem for φ into a the derivation problem for $\rightarrowtail$ over $\mathcal{M}(\varphi)$.

Proposition 1. *A PathLogic formula $\varphi = \varphi_1 \wedge \varphi_2$ in normal form is satisfiable iff the derivation problem for $\rightarrowtail$ with input φ has a positive answer.*

IV. THE DERIVATION PROBLEM FOR $\rightarrowtail$

In this section we show how to solve the derivation problem for $\rightarrowtail$. For this, we introduce a quasi-order $\leq_K$ on the set of mosaics, parametrized by a set of data values K. The parameter of this relation relies on the concept of *rigid values*, which are data values with a specific role in a $\rightarrowtail$ sequence. We show that $\leq_K$ is (upward) monotone with respect to $\rightarrowtail$. This monotonicity allows us to define a transition system $\rightharpoonup_K$ that has an equivalent derivation problem. We postpone the solving of the derivation problem for $\rightharpoonup_K$ to Section V.

We fix once and for all a formula φ of PathLogic in normal form, and $\mathcal{M}(\varphi)$ as the domain of the transition system $\rightarrowtail$.

A. Rigid and flexible data values

Not all data values are the same in a data word satisfying a formula, different data values have different roles. We distinguish here two categories of data values: rigid and flexible. Rigid data values are important for the satisfaction of the formula and special care is needed to treat these, whereas flexible values are not crucial, and they can be sometimes removed from the word. Let us give some more precise intuition. We use the logic PathLogic to make this intuition clear, but we will then state the definitions in terms of our transition system.

Given a data word $\mathbf{w}$ such that $\mathbf{w} \models \varphi$, suppose there is a data value d such that: if d can be accessed through a path expression α starting at some position i, then d is the *only* value that can be accessed through α at position i. When there is such a d we call it a **rigid value** for i, since the logic can identify it and pinpoint it from the rest of the data values. If d is rigid for at least one position of $\mathbf{w}$ we say that d is rigid for $\mathbf{w}$. All the remaining data values (which are the **flexible values**) of $\mathbf{w}$ play the role of assuring that "there are at least two data values reachable through α from position i" for some α and i. As such, its importance is only relative. For example, if $\mathbf{w}$ is a data word satisfying φ and containing d as a flexible value, then consider $\mathbf{w}'$ as the result of replacing, for some fresh data value d', every appearance of (a, d) in $\mathbf{w}$ by $(a, d)\,(a, d')$ (*i.e.*, d' appears next to each occurrence of d, with the same label). $\mathbf{w}'$ will continue to satisfy φ. But this is not necessarily true if d is a rigid value. The same notions hold for our $\rightarrowtail$ sequences. We say that a mosaic μ is K**-compliant** for $K \subseteq \mathbb{D}$ if all its rigid values are inside K.

Definition 6 (K-compliant)**.** *Given a finite set $K \subseteq \mathbb{D}$ we say that a mosaic $\mu = (a, d, \overleftarrow{\chi}, \overrightarrow{\chi})$ is K-compliant iff for all $\alpha \in \Omega^p_\varphi$ and $\chi \in \{\overrightarrow{\chi}, \overleftarrow{\chi}\}$*

$$\chi(\alpha) = \{e\} \quad \textit{implies} \quad e \in K \ .$$

This notion of K-compliance will be central for our second reduction to the more abstract transition system. Observe that for every $\rightarrowtail$ sequence there is a trivial K such that all its mosaics are K-compliant: the set of all data values of the data word. Later on, we will see that we can effectively bound the size of K as a function of φ. This bound is independent of the length of the $\rightarrowtail$ sequence.

Based on this idea of rigid values, we define a relation $\leq_K$ specially tailored for mosaics that are K-compliant. In this definition, if two mosaics are in the relation, then all the profiles of data values from K must remain untouched. For any other flexible value, we are only interested in having at least as many data values with a certain profile in the bigger

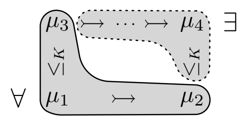
$\mu_3 \rightarrowtail \cdots \rightarrowtail \mu_4$
$\exists$
$\leq_K$
$\leq_K$
$\forall$
$\mu_1 \rightarrowtail \mu_2$

Fig. 2. The forward monotonicity Lemma.

profile. We also preserve the existence or non-existence of a flexible data value with a given profile.

Definition 7 ($\leq_K$). *$\mu_1 \leq_K \mu_2$ iff:*
1. they have the same label
2. $\forall d \in K$, $\mu_1(d) = \mu_2(d)$,
3. $\mu_1(d_1) = \mu_2(d_2)$ for d_i the data value of μ_i,
4. $\forall(\rho, \pi) \in \wp(\Omega_\varphi^p) \times \wp(\Omega_\varphi^p)$,

$$|\mu_1^{-1}(\rho,\pi) \setminus K| \quad \leq_1 \quad |\mu_2^{-1}(\rho,\pi) \setminus K|$$

where $n \leq_1 m$ iff $0 = n = m$ or $1 \leq n \leq m$.

Lemma 4 (about $\leq_K$)**.** *Given $\mu_1 \leq_K \mu_2$ we have*
1. if both mosaics are valid, then μ_1 is left-complete iff μ_2 is left-complete, and likewise for right-complete,
2. if μ_1 is K-compliant, then μ_2 is K-compliant,
3. if μ_1 is valid, then μ_2 is valid,
4. if both are K-compliant, for all $\psi \in \Omega_\varphi$, $\mu_1 \vdash \psi$ iff $\mu_2 \vdash \psi$.

B. A more abstract transition system

We define a transition system $\rightharpoonup_K$ over the set of K-compliant mosaics, which is also parametrized by K. We then reduce the derivation problem for $\rightarrowtail$ into the derivation problem for $\rightharpoonup_K$.

Definition 8 ($\rightharpoonup_K$)**.** *$\mu_1 \rightharpoonup_K \mu_2$ iff μ_1 and μ_2 are K-compliant and $\mu_1 \leq_K \mu_1' \rightarrowtail \mu_2' \geq_K \mu_2$ for some mosaics μ_1', μ_2'.*

Note that $\rightharpoonup_K$ is a looser relation than $\rightarrowtail$. This will enable us to easily bound the domain. But we postpone this issue to the next Section V. Our present goal is to show that there is indeed a reduction.

Proposition 2. *There exists a K-compliant complete $\rightharpoonup_K$ sequence over $\mathcal{M}(\varphi)$ whose first element μ_1 verifies $\mu_1 \vdash \varphi_1$ iff there exists a K-compliant complete $\rightarrowtail$ sequence over $\mathcal{M}(\varphi)$ whose first element μ_1' verifies $\mu_1' \vdash \varphi_1$.*

To prove the above proposition, first we need to show a monotonicity property of the transition system $\rightarrowtail$. The next Lemma asserts that $\rightarrowtail$ is upward compatible with respect to $\leq_K$, as in Figure 2. This is a crucial Lemma of our proof.

Lemma 5 (forward monotonicity)**.** *For every K-compliant mosaics $\mu_3 \geq_K \mu_1 \rightarrowtail \mu_2$ there is a K-compliant mosaic μ_4 such that $\mu_3 \rightarrowtail^+ \mu_4 \geq_K \mu_2$.*

Since all our definitions are symmetrical, by the proof of Lemma 5 we will also obtain a similar result for backward monotonicity.

Lemma 6 (backward monotonicity)**.** *For every K-compliant mosaics $\mu_1 \rightarrowtail \mu_2 \leq_K \mu_4$ there is a K-compliant mosaic μ_3 such that $\mu_1 \leq_K \mu_3 \rightarrowtail^+ \mu_4$.*

Before going into the details of the proof, we state some simple facts.

Lemma 7 (about $\mathbf{m}$)**.** *For any two valid mosaics $\mu_1, \mu_2 \in \mathcal{M}(\varphi)$ and $d \in \mathbb{D}$, suppose that $a_1 = a_2 = a$ for some $a \in \mathbb{A}$. Let $(\rho_1, \pi_1) = \mu_1(d)$, $(\rho_2, \pi_2) = \mu_2(d)$, $i = (d \overset{?}{=} d_1)$, $j = (d \overset{?}{=} d_2)$. Then $\overrightarrow{\mathbf{m}}(a, i, \pi_1, a, j, \pi_2)$ holds iff either*

$$i = 1, j = 0 \text{ and } \pi_1 = \pi_2 + a, \tag{4}$$
$$i = 0, j = 0 \text{ and } \pi_1 = \pi_2, \tag{5}$$
$$i = 1, j = 1 \text{ and } \pi_1 = \pi_2, \text{ or} \tag{6}$$
$$i = 0, j = 1 \text{ and } \pi_1 = \pi_2. \tag{7}$$

Analogously, $\overleftarrow{\mathbf{m}}(a, i, \pi_1, a, j, \pi_2)$ holds iff either

$$i = 1, j = 0 \text{ and } \rho_1 = \rho_2, \tag{8}$$
$$i = 0, j = 0 \text{ and } \rho_1 = \rho_2, \tag{9}$$
$$i = 1, j = 1 \text{ and } \rho_1 = \rho_2, \text{ or} \tag{10}$$
$$i = 0, j = 1 \text{ and } \rho_2 = \rho_1 + a. \tag{11}$$

Proof of Lemma 5 (forward monotonicity): Let $\mu_i = (a_i, d_i, \overleftarrow{\chi}_i, \overrightarrow{\chi}_i)$ for all $i \in \{1,2,3,4\}$. In this context we will write *flexible value* to denote a value $d \in \mathbb{D} \setminus K$ and *rigid value* to denote a value $d \in K$. Notice that $a_1 = a_3$ by definition of $\mu_1 \leq_K \mu_3$. By $\mu_1 \rightarrowtail \mu_2$ we can assume $\mu_1(d_1) = (\rho_1, a_1\pi_1 + a_1)$ and $\mu_2(d_1) = (a_2\rho_1, \pi_1)$ for some ρ_1 of the form $\rho_1 = a_1\rho' + a_1$. Similarly, we assume $\mu_2(d_2) = (a_2\rho_2 + a_2, \pi_2)$ and $\mu_1(d_2) = (\rho_2, a_1\pi_2)$ for some π_2 of the form $\pi_2 = a_2\pi' + a_2$. Hence, if $d_1 = d_2$ we have that $\rho_1 = \rho_2$ and $\pi_1 = \pi_2$.

We divide this proof into two parts. The first one groups all the easy cases, where the simulation can be done in one step: $\mu_3 \rightarrowtail \mu_4$. For the second part, we will need to use more than one step to arrive to such μ_4 mosaic.

Easy situation. The easy situation is when all data values in μ_3 can be 'simulated' by the behavior of data values of μ_1. We say that a flexible data value d of μ_3 can *simulate* another flexible data value d' of μ_1 if they have the same same profile and d is equal to d_3 iff d' is equal to d_1. If this holds, d can have the same profile as d' after one step of $\rightarrowtail$, by reading the same letter as μ_2. If all flexible data values of μ_3 simulate some data value of μ_1 and in turn every flexible data value from μ_1 is simulated by at least one data value from μ_3, we can arrive to a mosaic $\mu_4 \geq_K \mu_2$; but there is one caveat.

(A) There cannot be more than one data value that is chosen to simulate d_2, since only one of them can become the data value of μ_4. In this case the simulation would fail. If there is another flexible data value d' with the same profile as d_2 in μ_1, then we must choose only *one* data value to simulate d_2, and all the rest to simulate d'. We call this last restriction *the condition A*.

(B) By definition of $\leq_K$, every data value d such that $\mu_3(d) \neq \mu_3(d_3)$ ($= \mu_1(d_1)$) can simulate at least one data value in μ_1. Then, in order to assure that every data value of μ_3 can be simulated, it suffices to check that either d_1 is rigid (and hence also d_3), or if d_1 is flexible, that there are at least another flexible data value with the same profile as

d_1. We call this *the condition B*. If this would not hold, there could happen that there is only d_1 with the profile $\mu_1(d_1)$ in μ_1, and that there is a flexible data value $d \neq d_3$ with $\mu_3(d) = \mu_1(d_1)$. This data value d cannot simulate any value, and the simulation would then fail.

We formalize the properties we have discussed.

(A) If d_2 is a flexible value, then there are other flexible values with the same profile. That is, there exists another flexible value $d' \neq d_2$ with $\mu_1(d') = \mu_1(d_2)$.

(B) The same applies to d_1: if it is a flexible value, then there exists $d' \neq d_1$ with $\mu_1(d') = \mu_1(d_1)$.

If (A) and (B) hold, by definition of $\leq_K$ then there exists a surjective function $\xi : \mathbb{D} \to \mathbb{D}$ such that for every $d \in \mathbb{D}$,

1. $d = d_3$ iff $\xi(d) = d_1$,
2. $|\xi^{-1}(d_2)| = 1$,
3. $\mu_3(d) = \mu_1(\xi(d))$,
4. if $d \in K$ then $\xi(d) = d$.

In other words, ξ behaves as the identity on K, sends data values from μ_3 to data values with the same profile in μ_1, and sends only one data value to d_1 (namely d_3) and to d_2. We call ξ a *simulating function* between μ_3 and μ_1. If ξ is defined only for subsets $D_1, D_2 \subseteq \mathbb{D}$, we call $\xi : D_1 \to D_2$ a *partial* simulating function on D_1, D_2 between μ_3 and μ_1, and in this case conditions 1 and 2 are replaced by

$1'$. if $d_3 \in D_1$, $d = d_3$ iff $\xi(d) = d_1$,

$2'$. if $d_2 \in D_2$, $|\xi^{-1}(d_2)| = 1$.

We build $\mu_4 = (a_4, d_4, \overleftarrow{\chi}, \overrightarrow{\chi})$ where $a_4 = a_2$, $\{d_4\} = \xi^{-1}(d_2)$ and we define $\mu_4(d) = \mu_2(\xi(d))$ for all $d \in \mathbb{D}$. It is straightforward to check that for every $d \in \mathbb{D}$

$$\mathbf{m}(a_1, d \overset{?}{=} d_3, \mu_3(d), a_2, d \overset{?}{=} d_4, \mu_4(d)) \text{ iff}$$
$$\mathbf{m}(a_1, \xi(d) \overset{?}{=} d_1, \mu_1(\xi(d)), a_2, \xi(d) \overset{?}{=} d_2, \mu_2(\xi(d))) \ .$$

Complex situation 1. Suppose first that the condition B just seen does not hold, but that condition A holds. In other words, d_1 is the only flexible value with profile $\mu_1(d_1)$ in μ_1. Observe that $d_1 \neq d_2$. Otherwise we would have a contradiction, since (B) and (A) would express the same property, and we assume that one holds but the other does not.

Note that in this case we cannot define a function ξ just as before, because there may be a flexible data value $d \neq d_3$ with $\mu_3(d) = \mu_1(d_1)$. (If there is no such data value, we proceed just as in the easy case.) We basically do not know what to do with this data value d, since it cannot simulate any other data value from μ_1. Let P be the set of these problematic data values,

$$P = \{d \in \mathbb{D} \setminus K \mid d \neq d_3, \mu_3(d) = \mu_1(d_1)\}$$

If $P = \{e_1, \dots, e_n\}$, we will create n intermediary mosaics $\mu'_1, \dots, \mu'_n$ with

$$\mu_3 \rightarrowtail \mu'_1 \rightarrowtail \cdots \rightarrowtail \mu'_n \rightarrowtail \mu_4 \geq_K \mu_2$$

where μ'_i has the same letter a_1 as μ_3 and e_i as data value. Further, all data values except $\{d_3, e_1, \dots, e_n\}$ will preserve the same profile they have in μ_3 all along $\mu'_1, \dots, \mu'_n$. This is just a consequence of (5). Each μ'_i is defined as follows:

- a_1 is the label, and e_i is the data value,
- $d_3, e_1, \dots, e_{i-1}$ have profile $(\rho_1, a_1\pi_1)$,
- $e_i, \dots, e_n$ have profile $(\rho_1, a_1\pi_1 + a_1)$, and
- any other data value $d \notin \{d_3, e_1, \dots, e_n\}$ has profile $\mu_3(d)$.

These are indeed legal mosaics.

Claim 1. *For all $i \in [n]$, μ'_i is valid and K-compliant.*

Claim 2. *$\mu_3 \rightarrowtail \mu'_1$, and for all $i \in [n-1]$, $\mu'_i \rightarrowtail \mu'_{i+1}$.*

Let $D_1 = \mathbb{D} \setminus \{d_3, e_1, \dots, e_n\}$, $D_2 = \mathbb{D} \setminus \{d_1\}$ and $\xi : D_1 \to D_2$ be a partial simulating function on D_1, D_2 between μ_3 and μ_1. It must exist since condition A holds.

Claim 3. *There exists a partial simulating function $\xi : D_1 \to D_2$ between μ_3 and μ_1.*

We then define μ_4 as $\mu_4 = (a_4, d_4, \overleftarrow{\chi}_4, \overrightarrow{\chi}_4)$ with $a_4 = a_2$, $\{d_4\} = \xi^{-1}(d_2)$, $\mu_4(d) = (a_2\rho_1, \pi_1)$ for all $d \in \{d_3, e_1, \dots, e_n\}$ and $\mu_4(d) = \mu_2(\xi(d))$ for all other $d \in \mathbb{D}$. It then follows that $\mu'_n \rightarrowtail \mu_4$ and that $\mu_4 \geq_K \mu_2$.

Claim 4. $\mu'_n \rightarrowtail \mu_4$

Claim 5. $\mu_4 \geq_K \mu_2$

There are two other complex situations: when neither conditions A nor B hold, and when condition B holds but A does not. These conditions are treated in an analogous way. If condition B holds but A does not, we use a symmetrical strategy, creating intermediate mosaics but this time with the same label as μ_2. If neither of the conditions hold, we combine both approaches. ■

We show that the reduction from the derivation problem for $\rightarrowtail$ into the derivation problem for $\rightharpoonup_K$ is sound and complete with the following two lemmas.

Lemma 8 (complete)**.** *For every K-compliant complete $\rightarrowtail$ sequence $\mu_1 \rightarrowtail \cdots \rightarrowtail \mu_n$ over $\mathcal{M}(\varphi)$ there exists a K-compliant complete sequence $\mu'_1 \rightharpoonup_K \cdots \rightharpoonup_K \mu'_n$ over $\mathcal{M}(\varphi)$.*

Proof: Trivial, since $\rightarrowtail$ over K-compliant mosaics is a particular case of $\rightharpoonup_K$ by reflexivity of $\leq_K$. ■

Lemma 9 (sound)**.** *For every K-compliant sequence $\mu_1 \rightharpoonup_K \cdots \rightharpoonup_K \mu_n$ over $\mathcal{M}(\varphi)$ there exists a K-compliant $\rightarrowtail$ sequence $\mu'_1 \rightarrowtail \cdots \rightarrowtail \mu'_m$ over $\mathcal{M}(\varphi)$ with $\mu_1 \leq_K \mu'_1$ and $\mu_n \leq_K \mu'_m$.*

Proof: We proceed by induction on n. The base case is immediate. For the inductive step, suppose we have a sequence $\mu_1 \rightharpoonup_K \cdots \rightharpoonup_K \mu_n \rightharpoonup_K \mu_{n+1}$. By inductive hypothesis we obtain $\nu_1 \rightarrowtail \cdots \rightarrowtail \nu_m$ with $\nu_1 \geq_K \mu_1$ and $\nu_m \geq_K \mu_n$, as shown in Figure 3. Since $\mu_n \rightharpoonup_K \mu_{n+1}$, then

$$\mu_n \leq_K \mu'_n \rightarrowtail \mu'_{n+1} \geq_K \mu_{n+1}$$

for some μ'_n and μ'_{n+1} by definition of $\rightharpoonup_K$. Let $\mu^\dagger$ be such that $\mu^\dagger \geq_K \nu_m$ and $\mu^\dagger \geq_K \mu'_n$. It always exists such $\mu^\dagger$.

$$\begin{tikzcd}
\text{backward monotonicity} & & & \text{claim} & \text{forward monotonicity} & \\
\nu_1^\dagger \arrow[r, tail] \arrow[d, dash] & \cdots \arrow[r, tail] & \nu_l^\dagger = \mu^\dagger \arrow[d, dash] \arrow[dr, dash] = \mu_1^\dagger \arrow[rr, tail] & & \cdots \arrow[r, tail] & \mu_k^\dagger \arrow[d, dash] \\
\nu_1 \arrow[r, tail, "\text{ind. hyp.}"] \arrow[d, dash] & \cdots \arrow[r, tail] & \nu_m \arrow[d, dash] & \mu'_n \arrow[rr, tail, "\text{definition}"] \arrow[dl, dash] & & \mu'_{n+1} \arrow[d, dash] \\
\mu_1 \arrow[r, harpoon, "K"'] & \cdots \arrow[r, harpoon, "K"'] & \mu_n \arrow[rrr, harpoon, "K"'] & & & \mu_{n+1}
\end{tikzcd}$$

Fig. 3. General argument of Lemma 9. Solid lines correspond to the relation $\leq_K$, where the upper element in the diagram is the bigger.

Claim 6. *For any three mosaics μ_1, μ_2, μ_3 with $\mu_1 \geq_K \mu_3$ and $\mu_2 \geq_K \mu_3$, there exists a mosaic ν such that $\mu_1 \leq_K \nu$ and $\mu_2 \leq_K \nu$.*

By applying the monotonicity lemma we obtain on the one hand a sequence

$$\nu_1^\dagger \rightarrowtail \cdots \rightarrowtail \nu_l^\dagger$$

where $\nu_l^\dagger = \mu^\dagger$ and $\nu_1 \leq_K \nu_1^\dagger$, and by the backwards monotonicity lemma that

$$\mu_1^\dagger \rightarrowtail \cdots \rightarrowtail \mu_k^\dagger$$

where $\mu_1^\dagger = \mu^\dagger$ and $\mu_k^\dagger \geq_K \mu'_{n+1}$. This is depicted in Figure 3. Further, these $\rightarrowtail$ sequences are K-compliant. Then,

$$\nu_1^\dagger, \ldots, \nu_{l-1}^\dagger, \mu^\dagger, \mu_2^\dagger, \cdots, \mu_k^\dagger$$

is the the desired $\rightarrowtail$ sequence. ■

By Lemmas 8 and 9 the Proposition 2 follows. The facts that the sequence is complete and that the first mosaic verifies φ_1 follow from Lemma 4.

V. The derivation problem for $\rightharpoonup_K$

In this section we solve the derivation problem for $\rightharpoonup_K$. Since the transition system $\rightharpoonup_K$ is parametrized by K, the derivation problem for $\rightharpoonup_K$ is in fact: "is there a finite set $K \subseteq D$ such that the derivation problem for $\rightharpoonup_K$ over $\mathcal{M}(\varphi)$ has a positive solution?".

Note that we are dealing with many sources of infinity here: the choice of K, the domain of the transition system (*i.e.*, the set of all mosaics), and the definition of $\rightharpoonup_K$, which involves making a potentially infinite number of tests. Nevertheless, they can all be tamed. This is the purpose of this section.

We now show that the object that we are dealing with can be effectively bounded, to end up with finite objects that can be treated by an algorithm.

A. A bound on the rigid values

For every complete $\rightarrowtail$ sequence over $\mathcal{M}(\varphi)$ there are only linearly many rigid values. This Lemma will allow us to bound the number of minimal $\leq_K$ classes of equivalence.

Lemma 10. *For every complete sequence $\mu_1 \rightarrowtail \cdots \rightarrowtail \mu_n$ over $\mathcal{M}(\varphi)$ there is a set $K \subseteq \mathbb{D}$ such that:*

1. $|K| \leq 2 \cdot |\Omega^p_\varphi|$
2. $\mu_1, \ldots, \mu_n$ are K-compliant.

Corollary 2. *If there is a finite set $K' \subseteq \mathbb{D}$ such that the derivation problem for $\rightharpoonup_{K'}$ has a solution, then the derivation problem for $\rightharpoonup_K$ has also a solution, for any subset $K \subseteq \mathbb{D}$ with $2 \cdot |\Omega^p_\varphi|$ elements.*

Therefore, henceforth we assume without any loss of generality that K is any fixed subset of $\mathbb{D}$ of $2 \cdot |\Omega^p_\varphi|$ elements.

B. A bound on the domain

Let us write $\mathrm{MIN}_{\mathcal{M}(\varphi)}$ for the set of mosaics of $\mathcal{M}(\varphi)$ that are minimal with respect to $\leq_K$. This set contains only one element (arbitrarily chosen) for every minimal class of equivalence of $\leq_K$.

Remark 2. *Notice that if $\mu_1 \rightharpoonup_K \mu_2$ then $\mu'_1 \rightharpoonup_K \mu'_2$ for any $\mu'_1 \leq_K \mu_1$ and $\mu'_2 \leq_K \mu_2$. Hence, we can restrict the domain of $\rightharpoonup_K$ to $\mathrm{MIN}_{\mathcal{M}(\varphi)}$. That is, to mosaics that are minimal with respect to $\leq_K$.*

Lemma 11. *The number of elements of $\mathrm{MIN}_{\mathcal{M}(\varphi)}$ is double exponential in $|\varphi|$.*

From the previous statements we obtain the following.

Lemma 12. *If there is a complete K-compliant $\rightharpoonup_K$ sequence $\mu_1 \rightharpoonup_K \cdots \rightharpoonup_K \mu_n$ over $\mathcal{M}(\varphi)$, then there is a complete K-compliant sequence $\mu'_1 \rightharpoonup_K \cdots \rightharpoonup_K \mu'_m$ where*

- *$\mu'_1 \leq_K \mu_1$, $\mu'_m \leq_K \mu_n$,*
- *$m \leq |\mathrm{MIN}_{\mathcal{M}(\varphi)}|$, and*
- *$\mu'_i \in \mathrm{MIN}_{\mathcal{M}(\varphi)}$ for all $1 \leq i \leq m$.*

C. A bound on the relation

The lemmas above show that we have that the problem now reduces to find a complete small $\rightharpoonup_K$ sequence for a small K. In fact, we can test the relation $\rightharpoonup_K$ between any two elements in polynomial space.

We define the *space* of a mosaic μ, that we denote by $|\mu|$, as the space needed to write the multiset of all non-empty profiles of the mosaic. It is not hard to see that $\rightharpoonup_K$ can be in fact checked in polynomial space.

Lemma 13. *Given two K-compliant mosaics μ_1, μ_2, we can test $\mu_1 \rightharpoonup_K \mu_2$ in space polynomial in $|\mu_1|$ and $|\mu_2|$.*

D. The algorithm

With all these ingredients we can now decide the derivation problem using a standard reachability algorithm.

Proposition 3. *The derivation problem for $\rightarrowtail$ is in* ExpSpace.

Hence, we obtain that PathLogic has a decidable, ExpSpace satisfiability problem.

Proposition 4. *The satisfiability problem for PathLogic is in* ExpSpace.

Proof: As remarked in Section III-B, this is not entirely straightforward, since we have an exponential translation from satisfiability of PathLogic into the derivation problem for $\rightarrowtail$.

Let $\eta \in$ PathLogic. By item (1) of Corollary 1, the translation of η into its normal form φ preserves a number of path subformulas polynomial. This, combined with Lemma 10 and Corollary 2, provides a bound on the size of the set K to consider that is polynomial in $|\eta|$.

Hence, the number of minimal mosaics $\text{MIN}_{\mathcal{M}(\varphi)}$ is still doubly exponential (Lemma 11) in $|\varphi|$ and in $|\eta|$. (Having an exponential number of labels does not change this fact.)

By item (2) of Corollary 1, any path subformula of φ can be written in polynomial space in $|\eta|$. Then, each minimal mosaic can be written in exponential space. Hence, the algorithm of Proposition 3 remains ExpSpace in $|\eta|$. ■

VI. Lower bound

There is also a lower bound of ExpSpace for the satisfiability problem of PathLogic.

Proposition 5. *The satisfiability problem for PathLogic is* ExpSpace*-hard.*

This can be shown by coding a solution for an instance of the 2^n corridor tiling problem. The coding, although it is not quite straightforward, utilizes several coding techniques developed in [FS09].

VII. Corollaries on XPath and XML documents

Let us call *plain*-XPath$({}^*\!\leftarrow, \rightarrow^*, =)$ to the fragment of XPath such that that

- all path expressions do not mix ${}^*\!\leftarrow$ and $\rightarrow^*$, *i.e.*, each path expressions uses only ${}^*\!\leftarrow$ or only $\rightarrow^*$, and
- all test node expressions $\langle \alpha = \beta \rangle$ and $\langle \alpha \neq \beta \rangle$ are such that α uses only ${}^*\!\leftarrow$ and β only $\rightarrow^*$.

It is immediate that this basically corresponds to PathLogic.

Lemma 14. *There are* PTime *translations such that for every node expression of plain*-XPath$({}^*\!\leftarrow, \rightarrow^*, =)$ *(resp. of PathLogic, one-way PathLogic) return an equivalent node expression PathLogic (resp. plain*-XPath$({}^*\!\leftarrow, \rightarrow^*, =)$, XPath$(\rightarrow^*, =)$*).*

Corollary 3. *The satisfiability problem for plain*-XPath$(\rightarrow^*, {}^*\!\leftarrow, =)$ *on data words is* ExpSpace*-complete.*

The coding to show Proposition 5 uses only one-way expressions (of the form $\langle \varepsilon = \overrightarrow{\alpha} \rangle$). Hence, this bound also translates into a ExpSpace-hardness for XPath$(\rightarrow^*, =)$.

Corollary 4. *The satisfiability problem for* XPath$(\rightarrow^*, =)$ *on data words is* ExpSpace*-complete.*

In fact, plain-XPath$(\rightarrow^*, {}^*\!\leftarrow, =)$ (and hence PathLogic) is expressive-equivalent to XPath$(\rightarrow^*, {}^*\!\leftarrow, =)$.

Lemma 15. *There exists a computable* ExpTime *translation that for every node expression φ of* XPath$({}^*\!\leftarrow, \rightarrow^*, =)$ *returns an equivalent node expression ψ of plain*-XPath$({}^*\!\leftarrow, \rightarrow^*, =)$.

VIII. Discussion

We believe that the proof we give here may allow to extend existing decidability results of XPath fragments with reflexive-transitive axes. However, one has to combine the techniques of those results with the ones contained here. We propose the following candidates of (non-trivial) corollaries of the work contained in this paper and known results on XPath.

Conjecture 1. *Satisfiability of* XPath$(\downarrow, \downarrow_*, \rightarrow^*, {}^*\!\leftarrow, =)$ *on* XML *documents is decidable, with elementary complexity.*

This would be an extension of the result of [Fig09] stating that XPath$(\downarrow, \downarrow_*, =)$ is ExpTime-complete, and in contrast to the fact that XPath$(\downarrow, \downarrow_*, \rightarrow^*, {}^+\!\leftarrow, =)$ is undecidable and XPath$(\downarrow, \downarrow_*, \rightarrow^+, =)$ has non-primitive recursive complexity [FS09]. In fact, we believe that this can be pushed further:

Conjecture 2. *Satisfiability of* XPath$(\downarrow, \downarrow_*, \uparrow^*, \rightarrow^*, {}^*\!\leftarrow, =)$ *on* XML *documents is decidable, with elementary complexity.*

Furthermore, it is plausible that the following fragment of XPath is decidable.

Conjecture 3. *Satisfiability of* XPath$(\downarrow, \downarrow_*, \uparrow, \uparrow^*, \rightarrow^*, {}^*\!\leftarrow, =)$ *on* XML *documents is decidable.*

This would be an extension of a recent result stating that XPath$(\downarrow, \downarrow_*, \uparrow, \uparrow^*, =)$ is decidable [FS11]. However, even XPath$(\downarrow, \downarrow_*, \uparrow, \uparrow^*, \rightarrow^+, =)$ or XPath$(\downarrow, \downarrow_*, \uparrow, \uparrow^*, \rightarrow, =)$ are undecidable [FS09].

Future work

We conclude with some open questions.

1. *Decidability of* XPath *fragments.* Is any of the aforementioned conjectures true?
2. *Infinite words.* Is PathLogic decidable on infinite data words? (It does not enjoy the finite model property.)
3. *Ordered data.* As a possible extension, one may add comparison between data values with respect to a linear or partial order.

References

[BDM+10] M. Bojańczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Log.*, 2010. To appear.

[CD99] J. Clark and S. DeRose. XML path language (XPath). Website, 1999. W3C Recommendation. http://www.w3.org/TR/xpath.

[DDG07] S. Demri, D. D'Souza, and R. Gascon. Decidable temporal logic with repeating values. In *LFCS*, volume 4514 of *LNCS*, pages 180–194. Springer, 2007.

[DL09] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3), 2009.

[Fig09] D. Figueira. Satisfiability of downward XPath with data equality tests. In *PODS*, pages 197–206. ACM Press, 2009.

[FS09] D. Figueira and L. Segoufin. Future-looking logics on data words and trees. In *MFCS*, volume 5734 of *LNCS*, pages 331–343. Springer, 2009.

[FS11] D. Figueira and L. Segoufin. Bottom-up automata on data trees and vertical XPath. In *STACS*. Leibniz-Zentrum für Informatik, 2011.

[KSZ10] A. Kara, T. Schwentick, and T. Zeume. Temporal logics on words with multiple data values. In *FSTTCS*, 2010.

[SZ10] T. Schwentick and T. Zeume. Two-variable logic with two order relations. In *CSL*, 2010.