On XPath with transitive axes and data tests
Diego Figueira

To cite this version:
Diego Figueira. On XPath with transitive axes and data tests. ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), Jun 2013, New York, NY, United States. 10.1145/2463664.2463675. hal-01795148

HAL Id: hal-01795148
https://hal.archives-ouvertes.fr/hal-01795148
Submitted on 18 May 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
On XPath with Transitive Axes and Data Tests

Diego Figueira
University of Edinburgh
Edinburgh, UK

ABSTRACT

We study the satisfiability problem for XPath with data equality tests. XPath is a node selecting language for XML documents whose satisfiability problem is known to be undecidable, even for very simple fragments. However, we show that the satisfiability for XPath with the rightward, leftward and downward reflexive-transitive axes (namely following-sibling-or-self, preceding-sibling-or-self, descendant-or-self) is decidable. Our algorithm yields a complexity of $3\text{ExpSpace}$, and we also identify an expressive-equivalent normal form for the logic for which the satisfiability problem is in $2\text{ExpSpace}$. These results are in contrast with the undecidability of the satisfiability problem as soon as we replace the reflexive-transitive axes with just transitive (non-reflexive) ones.

1. INTRODUCTION

The simplest way of abstracting an XML document is by seeing it as a tree over a finite alphabet of tags or labels. However, this abstraction ignores all actual data stored in the document attributes. This is why there has been an increasing interest in data trees: trees that also carry data from an infinite domain. Here, we consider an XML modeled as an unranked ordered finite tree whose every node contains a label, and a vector of data values, one for each attribute. Labels belong to some finite alphabet, and data values to some infinite domain. We call these models multi-attribute data trees (see Figure 1). We study logics on these models, that can express data properties, namely equality of attributes’ data values.

Categories and Subject Descriptors
I.7.2 [Document Preparation]: Markup Languages; H.2.3 [Database Management]: Languages; H.2.3 [Languages]: Query Languages

General Terms
Algorithms, Languages

Keywords
XML, XPath, unranked unordered tree, reflexive transitive axes, data-tree, infinite alphabet, data values

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.
on data trees that can capture \( FO^2(<_a, succ_a, <_b, succ_b, \sim) \), XPath, ATRA, and other models. Although its emptiness problem is undecidable, classes of data trees for which it is decidable are studied in [1]. Other formalisms include tree automata with set and linear constraints on cardinalities of sets of data values [6, 23].

XPath.

Here we concentrate on XPath, which is incomparable in terms of expressiveness with all the previously mentioned formalisms (except for Class Automata).

XPath is arguably the most widely used XML query language. It is implemented in XSLT and XQuery and it is used as a constituent part of several specification and update languages. XPath is fundamentally a general purpose language for addressing, searching, and matching pieces of an XML document. It is an open standard and constitutes a World Wide Web Consortium (W3C) Recommendation [5].

Query containment and query equivalence are important static analysis problems, which are useful to, for example, query optimization tasks. In logics closed under boolean operators—as the one treated here—, these problems reduce to checking for satisfiability: Is there a document on which a given query has a non-empty result? By answering this question we can decide at compile time whether the query contains a contradiction and thus the computation of the query (or subquery) on the document can be avoided. Or, by answering the query equivalence problem, one can test if a query can be safely replaced by another one which is more optimized in some sense (e.g., in the use of some resource). Moreover, the satisfiability problem is crucial for applications on security [2], type checking transformations [21], and consistency of XML specifications.

Core-XPath (term coined in [17]) is the fragment of XPath 1.0 that captures all the navigational behavior of XPath. It has been well studied and its satisfiability problem is known to be decidable even in the presence of DTDs. The extension of this language with the possibility to make equality and inequality tests between attributes of elements in the XML document is named Core-Data-XPath in [4].

In a nutshell, the important formulas of Core-Data-XPath (henceforth XPath) are of the form

\[ (\alpha \circ \theta_1 = \beta \circ \theta_2), \]

where \( \alpha, \beta \) are path expressions, that navigate the tree using axes: descendant, child, ancestor, next-sibling, etc. and can make tests in intermediary nodes. Such a formula is true at a node \( x \) of a multi-attribute data tree if there are two nodes \( y, z \) in the tree that can be reached with the relations denoted by \( \alpha, \beta \) respectively, so that the \( i \)th attribute of \( y \) carries the same value as the \( j \)th attribute of \( z \).

Unfortunately, the satisfiability problem for XPath is undecidable [16]. How can we regain decidability for satisfiability of XPath then? We can restrict the models, or restrict the logic. The first possibility is to restrict the classes of documents in which one is interested, which is the approach taken in [1]. Another, more studied, approach is to restrict the syntax, which is the one taken here. One way to regain decidability is to syntactically restrict the amount of nodes that the XPath properties can talk about. In this vein, there have been studies on fragments without negation or without transitive axes [2, 16]. These fragments enjoy a small model property and are decidable. However, they cannot state global properties, involving all the nodes in an XML document. Ideally, we seek fragments with the following desirable features:

- closed under boolean operators,
- having as much freedom as possible to navigate the tree in many directions: up, down, left, right,
- having the possibility to reach any node of the tree, with transitive axes, like descendant, following sibling (the transitive closure of the next sibling axis), etc.

However, decidability results are scarce, and most fragments with the characteristics just described are undecidable. There are, however, some exceptions [10].

- The downward fragment of XPath, containing the child and descendant axes, is decidable, \( \text{ExpTime-complete} \) [8, 13].
- The forward fragment of XPath, extending the downward fragment with the next sibling and the following sibling axes, is decidable with non-primitive recursive complexity [9] [12].
- The vertical fragment of XPath, extending the downward fragment with the parent and ancestor axes, is decidable with non-primitive recursive complexity [15].
- A last example is the present work: XPath with the reflexive transitive closure of the child, next-sibling and previous-sibling relations is decidable.

All the non-primitive recursive (NPR) upper bounds of the forward and vertical fragments are also matched with NPR lower bounds. That is, there is no primitive recursive function that bounds the time or space needed by any algorithm that computes the satisfiability for any of these two logics. Moreover, it is known that any fragment of XPath containing a transitive rightward, leftward or upward axis has a satisfiability problem which is either undecidable or decidable with a NPR lower bound [14]. Further, as soon as we have both the rightward and leftward transitive axes, the satisfiability becomes undecidable [14]. (Indeed, the downward fragment of XPath seemed to be the only one with elementary complexity up to now.)

The aforementioned hardness results make use of non-reflexive transitive relations. Surprisingly, the reductions do not seem to work when the relations are also reflexive. What is then the decidability status of the fragments of XPath with reflexive-transitive relations? This was a question raised in [14].

A partial answer to this question was given in [11]. There, it was shown that XPath restricted to data words is decidable even when we have both a reflexive-transitive future and past relations. (One can think of data words as XML documents of height 1, with only one attribute per node.) This result may seem surprising taking into account that if one of these relations is non-reflexive it is no longer decidable; and if we have only one non-reflexive transitive relation it is decidable with non-primitive recursive complexity. In [11] it was shown that the satisfiability problem is in \( 2\text{ExpSpace} \) (or ExpSPACE if we adopt a certain normal form of

\[1\text{These are the axes that are called } \text{preceding-sibling, following-sibling and ancestor in the XPath jargon.} \]
the formulas). This was a first step in our study of the computational behavior of XPath with reflexive-transitive axes. The present work corresponds to the second part, in which we study XPath on XML documents (i.e., trees) instead of words.

Contribution

We show decidability of the satisfiability for XPath with data equality tests between attributes, where navigation can be done going downwards, rightwards or leftwards in the XML document. The navigation can only be done by reflexive-transitive relations. These correspond to the XPath axes: preceding-sibling-or-self, following-sibling-or-self, and descendant-or-self axes. Here we denote these axes with "\(\preceq\), \(\rightarrow\)" and \(\downarrow\), respectively. As already mentioned, the fact that the relations are reflexive-transitive (as opposed to just transitive) is an essential feature to achieve decidability. Given the known complexity results on XPath, this fragment seems to be in balance between navigation and complexity. This work then argues in favor of studying XPath-like logics for trees with data with reflexive-transitive relations, since they may behave computationally much better than the non-reflexive counterpart, as evidenced here.

The extension of the prior work \cite{Contribution} on data words to work with trees with a descendant axis is highly non-trivial, and we prove some monotonicity properties, which are necessary more involved than those used in \cite{Contribution} to account for the interplay between horizontal and vertical navigation of the logic.

Our algorithm yields a \(3\text{ExpSpace}\) upper bound for the satisfiability problem of this XPath fragment. We also show that this can be lowered to \(2\text{ExpSpace}\) if we work with an expressive-equivalent normal form, called direct normal form. Since XPath with just one reflexive-transitive relation is already \(\text{ExpSpace}\)-hard (even when the formula is in direct normal form) by \cite{Contribution}, we cannot aim for much better complexities.

Due to space limitations, all the proofs are included in the Appendix, although the main intuitions, general strategy, and decomposition of the proof in simpler problems are presented in the body of the paper.

2. PRELIMINARIES

Basic notation.

Let \(\mathbb{N}_0 \defeq \{0, 1, 2, \ldots\}\), \(\mathbb{N} \defeq \{1, 2, 3, \ldots\}\), and let \([n] \defeq \{1, \ldots, n\}\) for any \(n \in \mathbb{N}\). We fix once and for all \(\mathbb{D}\) to be any infinite domain of data values; for simplicity in our examples we will consider \(\mathbb{D} = \mathbb{N}_0\). In general we use the symbols \(\mathbb{A}\), \(\mathbb{B}\) for finite alphabets, and the symbols \(\mathbb{E}\) and \(\mathbb{F}\) for any kind of alphabet. By \(\mathbb{E}^*\) we denote the set of finite sequences over \(\mathbb{E}\), by \(\mathbb{E}^+\) the set of finite sequences with at least one element over \(\mathbb{E}\). We write \(\epsilon\) for the empty sequence and \(\bar{\epsilon}\) as the concatenation operator between sequences. By \(|S|\) we denote the length of \(S\) (if \(S\) is a finite sequence), or its cardinality (if \(S\) is a set). We use \((a_i)_{i \in \{j, \ldots, j + n\}}\) as short for \(a_j a_{j+1} \cdots a_{j+n}\).

Unranked finite trees with data.

By \(\text{Trees}(\mathbb{E})\) we denote the set of finite ordered and unranked trees over an alphabet \(\mathbb{E}\). We view each position in a tree as an element of \(\mathbb{N}^*\). Formally, we define \(\text{POS} \subseteq 2^\mathbb{N}^*\) as the set of sets of finite tree positions, such that: \(X \in \text{POS}\) if (a) \(X \subseteq \mathbb{N}^*, |X| < \infty\); (b) \(X\) is prefix-closed; and (c) if \(n(i + 1) \in X\) for \(i \in \mathbb{N}\), then \(n \cdot i \in X\). A tree is then a mapping from a set of positions to labels of the alphabet \(\text{Trees}(\mathbb{E}) \defeq \{t : \mathbb{P} \rightarrow \mathbb{E} \mid P \in \text{POS}\}\). The root’s position is the empty string \(\epsilon\). The position of any other node in the tree is the concatenation of the position of its parent and the node’s index in the ordered list of siblings.

Given a tree \(t \in \text{Trees}(\mathbb{E})\), \(\text{pos}(t)\) denotes the domain of \(t\), which consists of the set of positions of the tree, and \(\text{alph}(t) = \mathbb{E}\) denotes the alphabet of the tree. From now on, we informally refer by ‘node’ to a position \(x\) together with the value \(t(x)\).

Given two trees \(t_1, t_2 \in \text{Trees}(\mathbb{E})\) such that \(\text{pos}(t_1) = \text{pos}(t_2) = P\), we define \(t_1 \odot t_2 : \mathbb{P} \rightarrow (\mathbb{E}^\times)^*\) as \((t_1 \odot t_2)(x) \defeq (t_1(x), t_2(x))\).

The set of multi-attribute data trees over a finite alphabet \(\mathbb{A}\) of labels, \(\mathbb{K}\) different attributes, and an infinite domain \(\mathbb{D}\) is defined as \(\text{Trees}(\mathbb{A} \times \mathbb{D}^k)\). Note that every tree \(t \in \text{Trees}(\mathbb{A} \times \mathbb{D}^k)\) can be decomposed into two trees \(a \in \text{Trees}(\mathbb{A})\) and \(d \in \text{Trees}(\mathbb{D}^k)\) such that \(t = a \odot d\). Figure 1 shows an example of a multi-attribute data tree. The notation for the set of data values used in a data tree is \(\text{data}(a \odot d) \defeq \{d(x)(i) \mid x \in \text{pos}(d), i \in \mathbb{K}\}\). With an abuse of notation we write \(\text{data}(X)\) to denote all the elements of \(\mathbb{D}\) contained in \(X\), for whatever object \(X\) may be.

A forest is a sequence of trees, and the set of multi-attribute data forests over \(\mathbb{A}\) and \(\mathbb{K}\) is \((\text{Trees}(\mathbb{A} \times \mathbb{D}^k))^\omega\). We will normally use the symbol \(t\) for a forest of multi-attribute data trees. That is, \(t \in (\text{Trees}(\mathbb{A} \times \mathbb{D}^k))^\omega\). (Note that in particular \(t\) can be an empty forest.) For any \((a, d) \in \mathbb{A} \times \mathbb{D}^k\), let us write \(\langle a, d \rangle \bar{t}\) for the multi-attribute data tree that results from adding \((a, d)\) as a root of \(t\). We call this operation rooting. We will usually write \(t\) (resp. \(\bar{t}\)) to denote multi-attribute data trees (resp. forests) and \(t\) (resp. \(\bar{t}\)) to denote trees (resp. forests) over a finite alphabet.

![Figure 1: A multi-attribute data tree, where \(\mathbb{A} = \{a, b, c\}\) and \(k = 2\).](image-url)
Next we define transitive XPath, the fragment of XPath where all axes are reflexive and transitive.

Transitive XPath is a two-sorted language, with path expressions (that we write $\varphi, \psi, \eta$) and node expressions (that we write $\varphi, \psi, \eta$). Path expressions denote binary relations, resulting from composing the descendant, ancestor, preceding sibling and following sibling relations (which are denoted respectively by $\downarrow, \uparrow, \prec, \succ$ respectively), and node expressions. Node expressions are boolean formulas that test a property of a node like, for example, that is has a certain label, or that it has a descendant labeled a with the same data value in attribute i as the attribute j of an ancestor labeled b, which is expressed by $\langle \downarrow[a][b] \rangle \psi$. We write XPath($\downarrow_+, \uparrow^+, \prec, \succ$) to denote this logic, and we write XPath($\downarrow_+, \uparrow^+, \prec, \succ$, $\varphi$) to denote the logic containing only the axes in $\mathcal{O}$. A formula of XPath($\downarrow_+, \uparrow^+, \prec, \succ$, $\varphi$) is either a node expression or a path expression of the logic. Its syntax and semantics are defined in Figure 2. As another example, we can select the nodes that have a sibling labeled a to the left whose first attribute is the same as the second attribute of some descendant of a right sibling by the formula $\varphi = \left( \downarrow^{\ast} [a][a] \right) \psi$. Given a tree $t$ as in Figure 2, we have $\downarrow^{\ast} [a][a] = \{ 2, 3, 4, 5, 311 \}$.

We write $\varphi$ as $\varphi$ if $t, x \models \varphi$ (resp. $t, (x, y) \models \varphi$) for $x \in pos(t)$ (resp. $x, y \in pos(t)$) as short for $x \in [\varphi^t]$ (resp. $x, y \in [\varphi^t]$). We write $t \models \varphi$ as short for $\varphi \in [\varphi^t]$.

In the case of XPath($\downarrow_+, \uparrow^+, \prec, \succ$, $\varphi$), we also extend the evaluation to multi-attribute data forests. Let $(a, \delta)$ be an arbitrary fix element of $A \times D^k$. Given a forest $t$ and $x, y \in pos(a, \delta)t$, $x \neq y$, we define the satisfaction relation $\models$, as $t, x \models \varphi$ (resp. $t, (x, y) \models \varphi$) if $(a, \delta)t, x \models \varphi$ (resp. $(a, \delta)t, (x, y) \models \varphi$). (Note that since XPath($\downarrow_+, \uparrow^+, \prec, \succ$, $\varphi$) has no ascending axes, whether $t, x \models \varphi$ or not does not depend on $(a, \delta)$, we use it as a simple way of defining its semantics.)

The satisfiability problem for XPath($\downarrow_+, \uparrow^+, \prec, \succ$, $\varphi$) (henceforth noted SAT-XPath($\downarrow_+, \uparrow^+, \prec, \succ$, $\varphi$)) is the problem of, given a formula $\varphi$ of XPath($\downarrow_+, \uparrow^+, \prec, \succ$, $\varphi$), whether there exists a multi-attribute data tree $t$ such that $t \models \varphi$.

3. PROOF SKETCH

The main contribution of this paper is the following.

**Theorem 3.1.** SAT-XPath($\downarrow_+, \uparrow^+, \prec, \succ$, $\varphi$) is decidable in $\exists$ExpSpace.

We reduce the problem of whether a formula $\varphi$ of our logic XPath($\downarrow_+, \uparrow^+, \prec, \succ$, $\varphi$) is satisfiable, to the problem of whether one can obtain an element with a certain property by repeated applications of operations in some algebra, starting from a basic set of elements. We call it the derivation problem. First we introduce the algebra (Section 4), we then solve the derivation problem (Section 5) and finally we show the reduction from the logic into the derivation problem (Section 6).

We first introduce forest profiles in Section 4, which constitute the algebra domain. A forest profile is an abstraction of a multi-attribute data forest inside a context, where the context consists of the two (possibly empty) forests that are to the left and to the right. Figure 3 depicts one such possible forest, together with the left and right context forests. A forest profile contains, for each data value $d$ and each path expression, the information of whether $d$ can be reached by the path expression, and where it can be reached, either

- inside the main forest, starting from the leftmost root (the node $2^0$ in Figure 3),
- inside the main forest but starting with the rightmost root (the node $3^0$ in Figure 3),
- in the left context forest (starting from the node $2^0$ in Figure 3), or
- in the right context forest (starting from the node $2^0$ in Figure 3).

In this setting, path expressions are called patterns and their navigation is greatly simplified. Patterns can go first to the left, and then down, or first to the right and then down, or only down. They correspond to path expressions like, for example, $\varphi$ or $\varphi$ as short for $\varphi$ or $\varphi$. Further, node expressions contained in patterns are simple boolean combinations of tests for labels. A forest profile also keeps track of a set of important data values called the rigid values. These are data values that play a determined function in the forest containing the abstracted forest (i.e., in the concatenation of the left, main and right forests). Intuitively, a data value is rigid in a forest if it can be pinpointed by a path expression, in the sense that it is the only data value that can be reached with some path expression $\alpha \otimes$.

We equip the set of forest profiles with two operations, one that corresponds to concatenating two of the forests being abstracted, and another operation that corresponds to adding a root to the forest, converting it into a tree. This algebra is introduced in Section 4.2. In particular, the root operation is restricted to work only with forest profiles that are from certain set of consistent profiles. Consistent profiles will play an important role in the reduction from the logic to the algebra. The idea is that they are those profiles that are not in contradiction with the formula $\varphi$ to test for satisfiability, that is, that could abstract subforests of a model of $\varphi$.

A root profile, is a profile that comes from the application of the root operation with a certain label of a certain alphabet $A_{\root}$ of root labels. An empty profile is the profile corresponding to the empty forest with an empty context. In Section 4 we define the derivation problem for forest profiles as the problem of whether there is a way of obtaining a root profile from the empty profile by repeated applications of the algebra operations.

We show that the derivation problem is decidable in $2\exists$Space in Section 5. We first define a partial ordering on profiles in Section 5.2, this ordering will be of chief importance in our decidability result. We show a series of monotonicity properties that show that the set of derivable profiles is upward-closed. The purpose of the partial ordering is to reduce the derivation problem on the infinite set of forest profiles into a problem on a finite set of minimal profiles. The fact that the derivable profiles is upward-closed is indeed a key ingredient for this reduction to work.
$$\alpha, \beta ::= o \mid \alpha[\varphi] \mid [\varphi]\alpha \mid \alpha\beta$$

$$\varphi, \psi ::= a \mid \neg \varphi \mid \varphi \land \psi \mid \varphi \lor \psi \mid \langle \alpha \rangle \mid \langle \alpha \circ_i = \beta \circ_j \rangle \mid \langle \alpha \circ_i \neq \beta \circ_j \rangle \quad a \in \mathcal{A}, i, j \in [k] .$$

$$\llbracket \alpha \rrbracket = \{ (x, i) \mid x \in \text{pos}(t) \}^*$$

$$\llbracket \varphi \rrbracket = \{ (x, i, x(i+1)) \mid x(i+1) \in \text{pos}(t) \}^*$$

$$\llbracket \alpha \rrbracket = \{ (x, i) \mid x \in \text{pos}(t) \}$$

$$\llbracket \varphi \rrbracket = \{ (x, i) \mid x \in [\varphi]^t \}$$

$$\llbracket \alpha \rrbracket = \{ x \in \text{pos}(t) \mid \alpha(x) = a \}$$

$$\llbracket \neg \varphi \rrbracket = \text{pos}(t) \setminus \llbracket \varphi \rrbracket$$

$$\llbracket \alpha \circ_i = \beta \circ_j \rrbracket = \{ (x, i) \mid x \in \text{pos}(t) \}^*$$

$$\llbracket \varphi \land \psi \rrbracket = \{ (x, i) \mid x \in \text{pos}(t) \}^*$$

$$\llbracket \alpha \beta \rrbracket = \{ (x, z) \mid \text{there exists } y \text{ such that } (x, y) \in [\alpha]^t, (y, z) \in [\beta]^t \}$$

$$\llbracket \alpha \circ_i \neq \beta \circ_j \rrbracket = \{ (x, z) \mid \text{for some } y \in \llbracket \alpha \circ_i \rrbracket, (x, y) \in [\alpha]^t, (z, y) \notin [\beta]^t \}$$

$$\llbracket \alpha \circ_i = \beta \circ_j \rrbracket = \{ (x, z) \mid \text{for some } y \in \llbracket \alpha \circ_i \rrbracket, (x, y) \in [\alpha]^t, (z, y) \notin [\beta]^t \}$$

Figure 2: The syntax of transitive XPath; and its semantics for a multi-attribute data tree $t = a \odot d$.

Figure 3: A multi-attribute data forest, with its left and right forests.

However, one problem we need to face is that the ordering has infinite antichains: every two profiles with different sets of rigid values are incomparable. We tackle this in Section 5.4, where we show that we can bound the set of rigid values, obtaining an equivalent derivation problem on profiles with a small set of rigid values. Once we obtain this bound, the set of minimal profiles becomes finite, doubly exponential. Next, in Section 5.5 we show that, thanks to the monotonicity properties enjoyed by the algebra, we can work only with minimal elements. Finally, in Section 5.6 we give the concrete saturation-style algorithm that solves the derivation problem using doubly exponential space.

In Section 6 we show that the satisfiability problem for XPath($\alpha \rightarrow \downarrow_\varphi \rightarrow \gamma \rightarrow \epsilon = \) can be reduced to the derivation problem in EXPSPACE. In Section 6.1 we show a normal form, called direct unnested normal form, where direct unnested path expressions correspond, precisely, to the pattern expressions used in the forest profiles (basically all path expressions are of the form already described). We then show in Section 6.2 that one can reduce, in EXPSPACE, the satisfiability problem for formulas in this normal form into the derivation problem, obtaining a 3EXPSPACE decidability procedure for SAT-XPath($\alpha \rightarrow \downarrow_\varphi \rightarrow \gamma \rightarrow \epsilon = $), obtaining Theorem 6.1.

4. FOREST PROFILES

We define abstractions of forests of multi-attribute data trees. These are called forest profiles. They are the main construct in our solution. One must think of a forest profile as the description, for every data value $d \in D$, of all the possible ways of reaching the data value $d$ via path expressions of XPath($\alpha \rightarrow \downarrow_\varphi \rightarrow \gamma \rightarrow \epsilon = $). Some ways of reaching the data value may lie inside the forest being abstracted, and some outside the forest. Take for instance the forest in the middle of Figure 5. For every forest there we identify 4 nodes: the leftmost root, the rightmost root, the node to the left of the leftmost root (if any), and the node to the right of the rightmost root (if any). These are the nodes identified by $\overline{t}$, $\overline{t}^t$, $\overline{t}^e$, $\overline{t}^e$ respectively in the figure. The profile of this forest is represented by all the paths that can reach the data value $4$, all those that can reach $2$, etc. Take as an example the data value $5$; this data value can be reached by

(i) $\rightarrow^* \psi[1] \odot 1$ from $\overline{t}^e$,

(ii) $\psi[1] \downarrow_\varphi[1] \psi[1] \odot 1$ from $\overline{t}$,

(iii) $\rightarrow^* \psi[1] \odot 1$ from $\overline{t}$, etc.

Remember that expressions are evaluated in a forest and, for example, an expression starting with $\rightarrow^*$ denotes the possibility that there are moves forward in the sequence of tree roots of the forest. The idea is that we limit ourselves to those whenever there are paths departing from $\overline{t}$ or $\overline{t}^t$ they must be internal to the forest (i.e., internal to the gray forest in Figure 6), whereas there are paths from $\overline{t}^e$ or $\overline{t}^e$ they must be external to the forest (i.e., either in the forest depicted to the left or to the right of the gray forest in Figure 6).

Let $A$ be a finite alphabet of labels, let $\mathcal{A}_{\text{root}} \subseteq \mathcal{A}$ be the set of root labels, and let $D$ be an infinite domain of data values. The set $\mathcal{B}(A)$ is the boolean closure of tests for labels from $A$. For any $a \in \mathcal{A}$ and $\psi \in \mathcal{B}(A)$, we write $a \models \psi$ if the interpretation assigning true to $a$, and false to every other $b \in \mathcal{A}$, satisfies $\psi$. Let $k \in \mathbb{N}$ be a fixed natural number, corresponding to the number of attributes at each node. We say that $i \in [k]$ is an attribute index. We define the set of patterns, as any finite, subword-closed, subset of $(\mathcal{B}(A))^k$, and we denote it by $\mathcal{P}$. We generally use the symbols $\alpha, \beta, \gamma, \delta \in \mathcal{P}$ to denote patterns. For every label $a \in \mathcal{A}$ we define the following set of patterns $\sigma_a \subseteq \mathcal{P}$

$$\sigma_a \overset{\text{def}}{=} \{ \psi_1 \cdots \psi_k \in \mathcal{P} \mid a \models \psi_1 \land \cdots \land \psi_k \} .$$

Note that $\epsilon \in \sigma_\emptyset$. The set of composed patterns is

$$\Pi \overset{\text{def}}{=} (\mathcal{P} \setminus \{ \epsilon \}) \times \mathcal{P} \times [k] .$$
The intended meaning is that the first component operates on the siblings, the second on a downward path, and the third retrieves a data value from an attribute index. We will sometimes use the symbol \( \tilde{\alpha} \) to represent elements from \( \Pi \), or \( (\alpha, \beta, i) \) if we need to explicit the components of the composed pattern.

A forest profile \( f \) is a tuple 

\[
f = (\vec{x}, \vec{x}, \vec{x}, \vec{x}, R)
\]

where \( R \subseteq \mathbb{D} \), and we call it the set of rigid values of \( f \), and \( \vec{x}, \vec{x}, \vec{x}, \vec{x} \subseteq \mathbb{D} \times \Pi \), and we call them the set of left/right external/internal descriptions respectively. In the example before, one shall interpret \( f \) as \((5, a, e, 1) \in \vec{x}\), \( \vec{x} \) as \((5, b, c, 1) \in \vec{x}\) and \( \vec{x} \) as \((5, a-b, c, 1) \in \vec{x}\). We use \( \chi \) to denote a subset of \( \mathbb{D} \times \Pi \), and we write \( \vec{\chi} \) (resp. \( \vec{\chi} \)) to denote the 4-uple \((\vec{x}, \vec{x}, \vec{x}, \vec{x})\) (resp. \((\vec{x}, \vec{x}, \vec{x}, \vec{x})\)). Likewise, we use \( f \) (resp. \( f \)) to denote \((\vec{\chi}, \vec{\chi})\) (resp. \((\vec{\chi}, \vec{\chi}, \vec{\chi}, \vec{\chi})\)).

We define, for every \( \chi \subseteq \mathbb{D} \times \Pi \),

\[
\chi(d) = \{(\alpha, \beta, i) \in \Pi \mid (d, \alpha, \beta, i) \in \chi\},
\]

\[
\chi(\alpha, \beta, i) \text{ def } = \{d \in \mathbb{D} \mid (d, \alpha, \beta, i) \in \chi\},
\]

and \( \bar{\chi}(d) = \{(\alpha, \beta, i) \in \chi \mid (d, \alpha, \beta, i) \in \chi\} \).

We define data-values \( \text{def} \) of \( f \) \( \subseteq \mathbb{D} \times \mid \chi(d) \neq (0, 0, 0, 0) \}. \) We call data-values \( \text{def} \) \( \mathbb{D} \times \Pi \) the set of flexible values of \( f \). We use the symbol \( \pi \) to denote \((\vec{x}, \vec{x}, \vec{x}, \vec{x})\) where \( \vec{x}, \vec{x}, \vec{x}, \vec{x} \subseteq \mathbb{D} \times \Pi \). We further say that \( \pi \) is the description of \( d \subseteq \mathbb{D} \). We formalize this by defining an ordering on profiles corresponding to the operation just described, so that the forest profile abstracting \( f \) is bigger than the profile abstracting \( f \).

We equip \( \delta \) with two operations. The idea is that these operations correspond to the concatenation of two forests, and to the addition of a root to a forest (called rooting), turning it into a tree.

### Preliminaries

The set of root patterns of a forest profile \( f \), denoted by \( [f], \{f\} \subseteq \mathbb{P} \) is defined as follows

\[
[f] \text{ def } = \{\alpha \mid (d, \alpha, \beta, i) \in \chi \text{ for some } d, \beta, i\},
\]

\[
\{f\} \text{ def } = \{\alpha \mid (d, \alpha, \beta, i) \in \chi \text{ for some } d, \beta, i\}.
\]

Given \( P \subseteq \mathbb{P} \) and \( \chi \subseteq \mathbb{D} \times \Pi \), we define the extension of \( \chi \) by \( P \), denoted by \( P.\chi \), as the set

\[
P.\chi \text{ def } = \chi \cup \{(d, \alpha', \alpha, \beta, i) \in \mathbb{D} \times \Pi \mid (d, \alpha', \beta, i) \in \chi, \alpha' \in P\}.
\]

It is easy to see that the extension operation distributes over union (i.e., \( P.(\chi \cup \chi') = P.\chi \cup P.\chi' \) and \( (P \cup P').\chi = P.\chi \cup P'.\chi \)).

### Fingerprint

We now define the fingerprint of a forest profile. It contains a summary information, sufficient to decide whether the tree abstracted by the profile satisfies a formula of XPath—as we show in Section 6.2.

Let \( A = \{\vec{s}, \vec{\alpha}, \vec{\alpha}' \} \). Given a profile \( f \in \mathbb{G} \) and \( a \in A \), we define the set \( f.\chi_a \) as

\[
\chi_a \cup \{f \} \text{ if } a = \vec{s},
\]

\[
\chi_a \cup \{f \} \text{ if } a = \vec{\alpha},
\]

\[
\{(d, \alpha', \beta, i) \in \mathbb{D} \times \Pi \mid \exists \alpha.(d, \alpha, \beta, i) \in \chi \cup \chi' \}
\]

\[
\{d, \alpha, \beta, i) \in \chi \cup \chi' \mid \beta = \epsilon \}
\]

Note that \( f.\chi_a(\alpha, \beta, i) \) is independent of \( a \) when \( a = \vec{s} \), but it takes an element of \( A \) as argument for the sake of uniformity of notation. The fingerprint of a profile \( f \), noted \( \xi(f) \), is an element of

\[
\mathbb{F} \text{ def } = \mathbb{P} \times A \to \{0, 1, 2+, \}
\]

where for \( \vec{\alpha}, \vec{\alpha}' \in A, \vec{\alpha} \in A \), we define \( \xi(f)(\vec{\alpha}, a, \vec{\alpha}') \) as 0 or 1+ depending on whether \( \{f.\chi_a(\vec{\alpha}) \cap \chi_v(\vec{\alpha}') \} = 0 \) or not; and we define \( \xi(f)(\vec{\alpha}, a) \) as 0, 1, or 2+ depending on \( \{f.\chi_a(\vec{\alpha}) \} \) being 0, 1 or greater than 1 respectively.

We fix the set of consistent fingerprints, as a set of fingerprints \( \Gamma \subseteq \mathbb{F} \). The usefulness of this set will become apparent in the reduction from XPath to the derivation problem of forest profiles in Section 6.2, but we can anticipate
that this set will represent all the profiles abstracting multi-attribute data trees that do not contradict the formula we are trying to satisfy. For the moment, however, the reader may simply consider \( \Gamma \) as a given arbitrary set of fingerprints.

**Concatenation**

For every two \( f_1, f_2 \in \mathcal{F} \) so that

(a) \( R_1 = R_2 \),
(b) \( \overline{\chi}_1 = \overline{\chi}_2 = \overline{\chi}_3 \),
(c) \( \overline{\chi}_2 = \overline{\chi}_1 \cup \{ [f_1] \cdot \overline{\chi}_1 \} \),

we define the **concatenation** of \( f_1 \) and \( f_2 \), denoted as \( f_1 + f_2 \) as \( f_3 \), where

\[
R_3 = R_1 = R_2 \\
\overline{\chi}_3 = \overline{\chi}_2 \\
\overline{\varepsilon}_3 = \overline{\varepsilon}_1 \\
\overline{\chi}_3 = \overline{\chi}_1 \cup \{ [f_1] \cdot \overline{\chi}_1 \}.
\]

Notice that

- the concatenation is associative ((\( f_1 + f_2 \) + \( f_3 \)) = \( f_1 + (\{ f_2 \} + f_3) \)),
- the extension operation distributes over the concatenation operation \( ([f_1] \cdot ([f_2] \cdot \overline{\chi}_2)) \),
- if \( f_1 + f_2 = f_3 \) and \( f_1, f_2 \in \mathcal{F} \), then \( f_3 \in \mathcal{F} \).

**Rooting**

Given \( a \in \mathcal{A} \) and \( d \in \mathcal{D}^k \), we define \( (a, d)f_1 \subseteq \mathcal{F} \), where \( f_2 \in (a, d)f_1 \) if

(a) \( \xi(f_2) \in \Gamma \),
(b) \( \overline{\chi}_2 = \overline{\chi}_3 = \emptyset \),
(c) \( \overline{\chi}_2 = \overline{\chi}_3 = \emptyset \).

We say that \( f_2 \) is a **rooting** of \( (a, d) \) with \((a, d)\).

Notice that since the root pattern of any pair of profiles \( f_1, f_2 \in (a, d)f_3 \) is the same, it is idempotent and absorbing \((f_1 \cdot f_2) \cdot f = (f_1 \cdot f) \cdot (f_2 \cdot f)\) and \( f_3 \).

**4.3 The derivation problem**

We define the **empty profile** as \( f_0 \equiv \emptyset(\emptyset, \emptyset, \emptyset, \emptyset) \). Note that \( f_0 \in \mathcal{F} \). The set of profiles that can be obtained from empty profiles by applying the rooting and concatenation operations is called the set of **derivable profiles**, and noted \( \mathcal{D} \). We say that \( f \) is a **derivable root profile** if \( \overline{\chi}_3 = \overline{\chi}_4 = \emptyset \) and \( f \in (a, d)f' \) for some \( f' \in \mathcal{D} \, a \in \mathcal{A} \) and \( d \in \mathcal{D}^k \).

Let a **derivation tree** \( f \) be a tree whose every node is labeled by a forest profile and an element from \( \mathcal{A} \times \mathcal{D}^k \), except the leaves that are labeled only by the forest profile \( f_0 \) and

- the root is labeled with \( f \),
- every internal node \( x \) of \( t \) labeled with a forest profile \( f' \) and \((a, d)\) is so that \( f' \equiv (a, d)(f_1 + \cdots + f_n) \), where \( f_1, \ldots, f_n \) are the labels of the children of \( x \).

Similarly, a **derivation forest** \( \bar{f} \) for \( f \) is a forest of derivation trees \( \bar{f} = t_1 \cdots t_n \) for some profiles \( f_1, \ldots, f_n \) so that \( f = f_1 + \cdots + f_n \). Therefore, a profile \( f \) is derivable if, and only if, there is a derivation forest for \( f \).

We can now state the **derivation problem**, that is, whether there exists a derivable root profile, given \( A, A_{\text{root}}, \mathcal{P} \) and \( \Gamma \).

**Problem:** The derivation problem

**Input:** A finite alphabet \( A \), \( A_{\text{root}} \subseteq A \), a set of patterns \( \mathcal{P} \), a set of fingerprints \( \Gamma \subseteq \mathcal{F} \).

**Question:** Is there a derivable root profile?

In the next section we show that this problem is decidable. Later, in Section 6.1, we show that this problem is reducible from SAT-XPath(\( \langle \tau_\ast, \iota_\ast, \tau_\ast, \rangle = \).}

**5. COMPUTING DERIVABLE PROFILES**

In this section we solve the derivation problem, showing that it is decidable in \( 2\text{ExpSpace} \). To show this problem we work with some partial ordering on forest profiles (Section 5.2) that has some good monotonicity closure properties with our forest profile algebra (Section 5.3). This allows us to reduce the problem to a restricted derivation problem in which solutions can be found by only inspecting profiles with a bounded number of rigid values (Section 5.4), that is, minimal elements of the ordering (Section 5.5). These are bounded and computable, allowing us to produce an algorithm solving the problem (Section 5.6).

**5.1 Preliminaries**

Given \( f_1, f_2 \in \mathcal{F} \) we define that \( f_1 \) and \( f_2 \) are **equivalent**, and we note it \( f_1 \sim f_2 \), if there is some bijection \( g : \mathcal{D} \to \mathcal{D} \) so that \( f_2 \) is the result of replacing \( d \) by \( g(d) \) in \( f_1 \); in this case we write \( g(f_1) = f_2 \). For a set \( C \subseteq \mathcal{F} \), we write \( f \in C \) if there is \( f' \sim f \) so that \( f' \in C \). Given a forest profile \( f \) and two data values \( d \in \text{data}(f) \), \( d' \notin \text{data}(f) \), we define \( \{ d \to d' \} \) as the result of replacing \( d \) by \( d' \) in \( f \). Note that if \( \{ d \to d' \} \sim f \), given two data values \( d, d' \) we write \( \{ d \to d' \} \) to denote \( f \) where \( R' = R \), \( \chi'(d') = \chi(d) \), and \( \chi(e) = \chi(e) \) for every other \( e \neq d' \). Note that if \( d \in \text{data}(f) \) \( R \) and \( d' \notin \text{data}(f) \), we have that if \( f \in \mathcal{F} \) then \( \{ d \to d', d' \} \in \mathcal{F} \).

We say that a data value \( d \in \mathcal{D} \) is an **external data value** of \( f \) if \( \chi'(d) \cup \chi'(d') \notin \emptyset \). If further \( \chi'(d) \cup \chi'(d') \notin \emptyset \), we say that \( d \) is a **strict external data value** of \( f \). If \( d \notin \text{data}(f) \) is not a strict external data value, it is then an **internal data value**, and if it is not an external data value, it is then a **strict internal data value**.

**5.2 Ordering on profiles**

We define a partial order \( \preceq \) on forest profiles, that follows from our discussion of Section 4 on the role of flexible and rigid data values. It is the order in which we can make a profile bigger by adding a fresh data value to it, with the same description as that of a flexible data value already contained in it.

Given \( f_1, f_2 \in \mathcal{F} \), we define \( f_1 \preceq f_2 \) if either \( f_1 = f_2 \), or there is a flexible datum \( d \in \text{data}(f_1) \) so that \( f_1[d \to d, d'] \preceq f_2 \) for some \( d' \notin \text{data}(f_1) \). Note that \( \preceq \) is recursive, reflexive and transitive, and it is hence a partial order.

Note that if \( f_1 \preceq f_2 \) then \( \xi(f_1) \equiv \xi(f_2) \). We write \( f \preceq f' \) if \( f \preceq f' \) for some \( f' \sim f' \). We say that a set of forest profiles \( G \subseteq \mathcal{F} \) is **upward closed** (resp. downward...
Lemma 5.1 (Figure 4). Let \( f_1 \geq f_2 \) in \( G \) and \( f' \geq f \) (resp. \( f \geq f' \)), then \( f' \geq f \) (resp. \( f \geq f' \)).

Lemma 5.3 (Figure 5). If \( f_1 + f_2 = f_3 \) and \( f'_1 \geq f_1, f'_2 \geq f_2 \), then \( f'_1 + f'_2 \geq f_3 \), for some \( f'_1, f'_2 \in G \) so that \( f'_i \) is a bounded extension of \( f_i \) for all \( i \in \{1, 2\} \).

A similar lemma holds for the rooting operation.

Lemma 5.4 (Figure 6). For every \( f_1 \in (a, d)f_2 \) and \( f_2 \geq f_3 \), there is \( f_4 \geq f_3 \) so that \( |\text{data}(f_4)| \leq |\text{data}(f_3)| + |\Pi|^4 + |R_1| \) and \( f_4 \geq f_1, f_4 \geq f_2 \).

5.4 Bounding the rigid values

In this section we show that we can reduce the derivation problem into a similar problem where all the profiles have boundedly many rigid values. This will be combined with the result of the next sections, stating that the derivation problem restricted to profiles with boundedly many rigid values is decidable in 2ExpSpace, to solve the derivation problem.

Lemma 5.5. If there is a derivable root profile, then there is a derivation tree for a root profile so that all the profiles in the forest have no more than \( 2|\Pi| \) rigid values.

Let \( \mathcal{G}_0 \) be the set of all \( f \in \mathcal{G} \) that have no more than \( 2|\Pi| \) rigid values. Let \( \mathcal{D}_0 \) be the set of derivable profiles restricted to \( \mathcal{G}_0 \).

Remark 5.6. By Lemma 5.5 and in light of the definition of bounded extension, it follows that Lemma 5.4 when applied to profiles of \( \mathcal{G}_0 \), yields a profile \( f_4 \) that is a bounded extension of \( f_3 \).

By the Lemma just shown, we have the following

Lemma 5.7. There is a derivable root profile in \( \mathcal{D} \) if and only if there is a derivable root profile in \( \mathcal{D}_0 \).

We have then reduced the derivation problem into a simpler problem, the bounded derivation problem: testing whether there is a derivable root profile in \( \mathcal{D}_0 \).

Remark 5.8. We have that \( \mathcal{D}_0 \) is upward closed since \( \mathcal{D} \) is upward closed. That is, \( \mathcal{D}_0 = \uparrow \mathcal{D}_0 \).

Note that \( \mathcal{G}_0 \) has boundedly many \( \leq \)-minimal elements. In the next section we show how to restrict the problem to a problem that uses only these \( \leq \)-minimal profiles. We will show how this yields a 2ExpSpace algorithm in Section 5.9.

5.5 Restricting to minimal elements

Thanks to the result from the previous section stating that \( \mathcal{D}_0 \) is upward closed, we can now show that we can work only with the minimal elements of \( \mathcal{G}_0 \). The main necessary property concerns all those profiles \( f' \in \mathcal{G}_0 \) that are \( \leq \)-related to a profile \( f'' \in \mathcal{D}_0 \), in the sense that \( f' \leq f'' \in \mathcal{G}_0 \), for...
some $\mathcal{F}$. (Note that this set of profiles is precisely $\uparrow\downarrow \mathcal{D}_b$.) The property states that the forest profiles algebra preserves the $\sim$-relatedness.

Given $G \subseteq \mathcal{F}_b$, let

$$R_{a.(d)}^G \stackrel{\text{def}}{=} \{ f \in \mathcal{F}_b \mid f \sim (a, d) f', f' \in G \}$$

for $(a, d) \in A \times D^k$,

$$R_{a.}^G \stackrel{\text{def}}{=} \bigcup_{(a, d) \in A \times D^k} R_{a.(d)}^G,$$

$$R_+^G \stackrel{\text{def}}{=} \{ f \in \mathcal{F}_b \mid f = f_1 + f_2 \text{ where } f_1, f_2 \in G \},$$

$$R(G) \stackrel{\text{def}}{=} R_{a.}^G \cup R_+^G.$$

**Lemma 5.9.** $R(\uparrow \downarrow \mathcal{D}_b) \subseteq \downarrow \mathcal{D}_b$.

### 5.6 The algorithm

In this section we show how to compute, in 2ExpSpace, whether there exists a derivable root profile in $\mathcal{D}_b$, solving thus the derivation problem.

For $G \subseteq \mathcal{F}_b$, we define $G^{-} \stackrel{\text{def}}{=} \{ f \mid f \sim f' \text{ for some } f' \in G \}$. We define $G/\sim$ as the set containing one representative profile of $G$ for each $\sim$-equivalence class. We define $\text{MIN}(G)$ as the set of $\sim$-minimal elements of $G$,

$$\text{MIN}(G) \stackrel{\text{def}}{=} \{ f \in G \mid \text{ for all } f' \in G \text{ so that } f \sim f' \}. $$

For any $f \in \mathcal{F}_b$, we write $|f|$—the size of $f$—as the size needed to write $f$. Note that for all $f \in \text{MIN}(\mathcal{F}_b)$, $|f|$ is at most exponential in $|P|$. For any $G \subseteq \mathcal{F}_b$, we write $|G|$ to denote $\sum_{f \in G} |f|$.

Let us define $C_i$ for every $i \in \mathbb{N}_0$ as

$$C_0 \stackrel{\text{def}}{=} \{ \emptyset \},$$

$$C_{i+1} \stackrel{\text{def}}{=} C_i \cup \text{MIN}(\downarrow R(\uparrow \downarrow C_i)) / \sim.$$

Let $k_0 \in \mathbb{N}_0$ be the first index so that $C_{k_0} = C_{k_0+1}$.

**Remark 5.10.** For every $i \in \mathbb{N}_0$, $C_i \subseteq \text{MIN}(\mathcal{F}_b)$.

As a consequence of the property of the preceding section, we have that this algorithm computes $\text{MIN}(\downarrow \mathcal{D}_b)$.

**Lemma 5.11.** $C_{k_0} = \text{MIN}(\downarrow \mathcal{D}_b)$.

We further have that this computation is in 2ExpSpace since $|\text{MIN}(\mathcal{F}_b) / \sim|$ is doubly exponential in $|P|$, hence we obtain the following. (Details in the Appendix.)

**Proposition 5.12.** The derivation problem is decidable in 2ExpSpace.

### 6. FROM XPATH TO FOREST PROFILES

In this section we reduce the satisfiability problem for XPath($\preceq, \downarrow, \rightarrow, =)$ into the derivation problem for forest profiles.

In Section 6.1, we define a normal form for XPath($\preceq, \downarrow, \rightarrow, =)$, called direct unnested normal form, and in Section 6.2 we show the reduction from the satisfiability problem of direct unnested XPath($\preceq, \downarrow, \rightarrow, =)$ formulas into the derivation problem for forest profiles.

#### 6.1 Normal forms

We will assume a certain normal form of the formula $\varphi \in \text{XPath}(\preceq, \downarrow, \rightarrow, =)$ to test for satisfiability. This will simplify the reduction into the derivation problem for forest profiles.

The normal form has two main properties. Firstly, it contains only path expressions that are direct, in the sense that the navigation consists in going left and then down, or going right and then down. And secondly, path expressions do not contain data tests as node expressions, in other words the formula is unnested. Next, we explain in detail these properties.

**Preliminaries**

Let $\alpha = a_1 \cdots a_n$ with $n > 0$ be a XPath($\preceq, \downarrow, \rightarrow, =)$ path expression, where for every $i$, $a_i = [\psi]$ for some node expression $\psi$, or $a_i \in \{ \varepsilon, \downarrow, \rightarrow \}$. We say that $\alpha$ is in **alternating path normal form** if either $\alpha = \varepsilon$, or $n$ is even and for all $1 \leq i \leq n$

- if $i$ is even, $a_i = [\psi]$ for some node expression $\psi$,
- if $i$ is odd, $a_i \in \{ \downarrow, \rightarrow \}$.

In other words, the path alternates between axes and tests for node expressions. We say that a formula is in alternating path normal form if all its path expressions are in alternating path normal form. Note that one can turn any formula $\varphi \in \text{XPath}(\preceq, \downarrow, \rightarrow, =)$ into an equivalent formula $\varphi'$ in alternating path normal form in polynomial time, using the equivalences

$$\langle \psi \rangle \alpha \beta \equiv \psi \land (\alpha \psi \beta) \quad \text{for } \alpha \in \{ \varepsilon, \downarrow \},$$

$$\langle \psi \rangle \alpha \beta \equiv \psi \land (\alpha \psi \beta) \quad \text{for } \alpha \in \{ \varepsilon, \rightarrow \},$$

$$\alpha \psi \beta \equiv \alpha (\psi \beta) \beta \quad \text{and,}$$

$$\text{if } \alpha \neq \varepsilon, \alpha \beta \equiv \alpha [1] \beta \quad \text{and } \alpha \overline{\beta} \equiv \overline{\alpha} \beta.$$

For simplicity and without any loss of generality we can further assume that all our formulas do not contain formulas of the type $(\alpha)$, since it is equivalent to $(\alpha \varepsilon \beta)$.

We will henceforth assume that all the formulas we work with are in this form.

We say that a path expression in alternating path normal form is a **rightward path expression**, if it starts with $\rightarrow$ and all the axes in it are in $\rightarrow$ (similarly with leftward, downward and $\vdash$, $\downarrow$). Notice that, for example, a leftward expression may contain node tests using rightward or downward axes. For example, $\preceq [\{ \downarrow \} \varepsilon \downarrow \varepsilon]$ is a leftward expression while $\preceq [\{ \downarrow \} \varepsilon \downarrow \varepsilon]$ is not.

**Direct normal form**

The object of the direct normal form is to avoid having unnecessary mixed directions in path formulas, that use perhaps $\rightarrow$ and $\preceq$ in the same expression, or that contain a $\preceq$ (or $\rightarrow$) axis after a $\downarrow$ axis. That is, we avoid having formulas like

$$(\rightarrow \varepsilon \downarrow \varepsilon) \rightarrow \varepsilon \downarrow \varepsilon \rightarrow \varepsilon \downarrow \varepsilon \downarrow \varepsilon$$

in favor of equivalent formulas with a more direct navigation, like

$$(\rightarrow \varepsilon \downarrow [\rightarrow \varepsilon \downarrow \varepsilon \downarrow \varepsilon] \rightarrow \varepsilon \downarrow \varepsilon \downarrow \varepsilon \downarrow \varepsilon) \lor (\rightarrow \varepsilon \downarrow \varepsilon \downarrow \varepsilon \downarrow \varepsilon) \lor (\rightarrow \varepsilon \downarrow \varepsilon \downarrow \varepsilon \downarrow \varepsilon) \lor (\rightarrow \varepsilon \downarrow \varepsilon \downarrow \varepsilon \downarrow \varepsilon).$$

(A)
In the formula above we factor the loops that may be in the navigation of the path expression to obtain a simple navigation that goes in only one horizontal direction.

We say that a formula \( \varphi \in \text{XPath}(\tau \leftarrow \iota_\varsigma \rightarrow \iota_\rho, =) \) is in direct normal form, if every path expression is \( \varepsilon \), or of the form \( \alpha \cdot \beta \), where \( \alpha \cdot \beta \neq \varepsilon \) (i.e., it is not the empty string), \( \alpha \) is leftward, rightward or empty, and \( \beta \) is downward or empty. Note that, strictly speaking, the formula \( \Box \) is not in direct normal form since its second disjunct is not in alternating path normal form, but the equivalent alternating path expression—using \( \Box \)—is in direct normal form.

**Lemma 6.1 (Direct Normal Form).** There exists an exponential time translation that for every node expression \( \varphi \in \text{XPath}(\tau \leftarrow \iota_\varsigma \rightarrow \iota_\rho, =) \) returns an equivalent node expression \( \psi \) in direct normal form.

**Unnest normal form**

The second normal form consists in having formulas without nesting of data tests. That is, we avoid expressing formulas like, for example,

\[
(\downarrow_+(\downarrow_+\varphi_1 \circ \downarrow_+\varphi_2)) = \varphi_1 \circ \varphi_2
\]

If a formula is such that all its path expressions \( \alpha \) contain only (boolean combinations of) tests for labels we call it a non-recursive formula.

We say that \( \varphi \) is in unnested normal form if \( \varphi = \varphi_1 \wedge \varphi_2 \) where \( \varphi_1 \in B(\tilde{A}) \) and \( \varphi_2 \) is a conjunction of tests of the formula "if a node has some of the labels \( \{a_1, \ldots, a_n\} \) then it satisfies \( \psi \)" for some non-recursive formula \( \psi \) and labels \( a_1, \ldots, a_n \in \tilde{A} \). Formally, \( \varphi_2 \) contains a conjunction of tests of the form

\[
\neg(\downarrow_+[\tau \wedge \neg \psi])
\]

for \( \tau \) a disjunction of labels and \( \psi \) a non-recursive formula. Given \( \varphi = \varphi_1 \wedge \varphi_2 \) in unnested normal form, we write \( \gamma_\varphi(a) \) for \( a \in \tilde{A} \) to denote the function where \( \gamma_\varphi(a) \) is the conjunction of all the formulas \( \psi \) such that \( \varphi_2 \) contains \( \neg(\downarrow_+\varphi) \) as a subformula, for some disjunctive formula \( \tau \) containing the label \( a \).

Then, we obtain the following.

**Lemma 6.2 (Unnested Normal Form).** There exists an exponential time translation that for every formula \( \eta \in \text{XPath}(\tau \leftarrow \iota_\varsigma \rightarrow \iota_\rho, =) \) returns a formula \( \varphi \) in unnested normal form such that \( \eta \) is satisfiable iff \( \varphi \) is satisfiable. Further, the translation of a formula in direct normal form is in direct normal form.

**Corollary 6.3.** About the translation of Lemma 6.2

1. The set of path subformulas resulting from the translation has cardinality polynomial in \( \eta \).
2. Every path subformula resulting from the translation can be written using polynomial space.

### 6.2 Reduction to the derivation problem

In this section we show how we can reduce the satisfiability problem of direct unnested XPath formulas into the derivation problem for forest profiles.

Let us fix \( \varphi = \varphi_1 \wedge \varphi_2 \) in direct unnested normal form, where \( \tilde{A} \) as the finite alphabet, \( k \) as the number of attributes, \( \mathbb{D} \) as any infinite domain, and \( A_{\text{root}} \) is the set of all \( a \in \tilde{A} \) that make \( \varphi_1 \) true.

Given a pattern \( \alpha = \psi_1 \cdots \psi_k \in \mathcal{P} \), and an axis \( o \in \{\tau \leftarrow \iota_\varsigma \rightarrow \iota_{\rho}\} \), we can convert \( \alpha \) into a path expression as follows:

\[
P_o(\varepsilon) \overset{\text{def}}{=} \varepsilon \quad \text{if } k = 0,
\]

\[
P_o(\psi_1 \cdots \psi_k) \overset{\text{def}}{=} o[\psi_1] \circ \cdots \circ o[\psi_k] \quad \text{if } k > 0.
\]

Note that \( P_o \) is injective. Let us define \( P_\psi \) as the set of patterns consisting of

- the constant \( \top \) and the empty string \( \varepsilon \),
- \( \psi \), for every \( \psi \in B(\tilde{A}) \) that is a subformula of \( \phi \),
- every \( \psi \in (B(\tilde{A}))^* \) so that \( P_\alpha(\tau) \), \( P_{\varphi}(\alpha) \), or \( P_{\alpha}(\iota) \) is a substring of \( \phi \).

It follows that \( P_\psi \) is finite and subword-closed.

For any direct non-recursive formula \( \psi \) that is a boolean combination of subformulas of \( \phi \) and forest profile \( f \), we define \( f \vdash \psi \) as follows. If \( \psi \in \tilde{A} \), then \( f \vdash \psi \) if and only if there is some \( d \in \mathbb{D} \) and \( i \in [k] \) so that \( \psi(e, i) \in \chi^{d}(d) \).

For all the boolean cases \( = \) is homomorphic. Suppose now that \( \psi = (\alpha \cdot \beta) \neq \gamma \cdot \delta \), where \( \alpha \) is leftward, \( \varepsilon \) or empty, \( \gamma \) is rightward, \( \varepsilon \) or empty, and \( \beta, \delta \) are downward or empty. We define \( f \vdash \psi \) if there are some \( d, d' \in \mathbb{D} \) so that \( d \neq d' \) and

- if \( \alpha = \varepsilon \) or \( \alpha = \varepsilon, (\tau, P_{\alpha}^{\downarrow}((\iota)\beta), i) \in \chi^{d}(d) \),
- if \( \alpha \neq \varepsilon, (\alpha, \beta) \neq \gamma \cdot \delta \), \( (\tau, P_{\alpha}^{\downarrow}(\iota\beta), i) \in (\chi^{d}(d) \cup \chi^{d'}(d')) \),
- if \( \gamma = \varepsilon \) or \( \gamma = \varepsilon, (\tau, P_{\alpha}^{\downarrow}(\iota\beta), i) \in \chi^{d}(d) \),
- if \( \gamma \neq \varepsilon, \gamma \neq \varepsilon, (\tau, P_{\alpha}^{\downarrow}(\iota\beta), i) \in (\chi^{d}(d) \cup \chi^{d'}(d')) \).

Note that if \( \alpha = \varepsilon \) then \( \beta = \varepsilon \) (resp. with \( \gamma \) and \( \delta \)). If both \( \alpha \) and \( \gamma \) are rightwards or leftwards it is defined in an analogous way. The case for \( = \) is also analogous, where \( d = d' \). The idea is that \( f \vdash \psi \) makes only sense when the derivation forest for \( f \) is a tree, and the multi-attribute data tree \( t \) associated to the derivation tree is so that \( t \models \psi \).

For example, testing \( \psi \) is the same as testing if there is some pattern \( (\psi, \omega) \) in \( \chi^{d} \) or \( \chi^{d'} \). In a similar way, checking a formula like

\[
(\neg[\downarrow_+\beta] \circ \iota_\varsigma \circ [\downarrow_+\alpha_1 \circ \iota_\rho] = \downarrow_+\neg[\downarrow_+\alpha_2])
\]

reduces to checking if there is a data value \( d \in \mathbb{D} \) that can be reached with \( (\tau, c, 2) \) in the main forest (i.e., in \( \chi^{d} \) or \( \chi^{d'} \)), and either

- \( d \) can be reached by \( (a, b, 1) \) in the main forest, that is, \((a, b, 1) \in \chi^{d} \) (or equivalently \( \chi^{d'} \)),
- \( d \) can be reached in the right forest by \( (a, b, 1) \), where \( a \) could be tested in the main forest (i.e., \( a \in [\mathbb{P}] \)), that is, \((a, b, 1) \in [\mathbb{P}] \chi^{d} \).

Note that checking \( f \vdash \psi \) takes polynomial time in the size of \( f \) and \( \psi \). Also, whether \( f \vdash \bigwedge_{a \in \tilde{A}} (a \Rightarrow \gamma_\varphi(a)) \) holds or not depends only on \( \xi(f) \).

**Lemma 6.4.** Given a direct non-recursive formula \( \psi \) that is a boolean combination of subformulas of \( \phi \), and two forest profiles \( f, f' \in \mathbb{F} \) so that \( \xi(f) = \xi(f') \) then \( f \vdash \psi \) if, and only if, \( f' \vdash \psi \).
We can then write $\xi \models \psi$ for $\xi \in F$ instead of $f \models \psi$ for any $f$ so that $\xi(f) = \xi$. We define the set of consistent profiles $\Gamma_\phi$ as all $\xi \in F$ so that $\xi \vdash \bigwedge_{a \in A} (a \Rightarrow \gamma_\phi(a))$. The following lemma follows straight from the above definition of $\vdash$.

**Lemma 6.5.** $\xi \vdash \bigwedge_{a \in A} (a \Rightarrow \gamma_\phi(a))$ if, and only if, $\xi(f) \in \Gamma_\phi$.

**Abstractions.**

Given multi-attribute data forests $t_l, t_r$, we define

$$\text{abs}(t_l, t_r)$$

as the forest profile that abstracts the forest $t_l$ in the context of the forests $t_l$ to the left and $t_r$ to the right. We have already discussed the idea of this abstraction in Section 4. For example, for the forest of Figure 3 assuming $\mathcal{P} = \{\top, b, c, b, c, \epsilon\}$, we would obtain an abstraction where

$$\chi^e = \{(5, b, b, 1), (5, b, c, 1), (3, b, b, 2), (2, b, c, 1), \ldots\}.$$

We have that $\text{abs}$ is basically an algebra morphism between multi-attribute data forests with rooting and concatenation and forest profiles with profile rooting and profile concatenation. Further, the profile $\text{abs}(e, t, e)$ is a derivable root profile whenever $\xi \models \phi$ and every derivable root profile is the abstraction of some tree $t$ so that $t \models \phi$. (The formal definition of $\text{abs}$ and the properties are described in the Appendix.) As a corollary from these properties, we have the following.

**Corollary 6.6.** There is a derivable root forest profile if, and only if, $\phi$ is satisfiable.

By the above Corollary 6.6 and Proposition 5.12, we can check in $2\text{ExpSpace}$ if there is a derivable root profile. This is $2\text{ExpSpace}$ in the size of $\mathcal{P}_\phi$. Although bringing a formula $\varphi$ into direct unnested normal form may result in a doubly exponential formula, by Corollary 6.5 it can be stored in exponential space, and $\mathcal{P}_\phi$ is then singly exponential. Hence, the procedure is $3\text{ExpSpace}$ in the original formula $\varphi$. Thus, the decision procedure is in $3\text{ExpSpace}$ and Theorem 5.1 follows.

Note that if the input formula is in direct normal form then we save one exponential in the reduction and we hence obtain a $2\text{ExpSpace}$ decision procedure.

**Theorem 6.7.** The satisfiability problem for formulas of XPath($\leftarrow, \downarrow, \rightarrow, =$) in direct normal form is decidable in $2\text{ExpSpace}$.

7. DISCUSSION

We have shown that XPath with downward, rightward and leftward reflexive-transitive axes is decidable. To show this, we devised an algebra with good monotonicity properties. This seems to be the right kind of approach to work with transitive relations, and it generalizes and simplifies, in some aspects, the work of [11].

**Upward axes**

One natural question that stems from the result presented here is whether it can be extended to work with an upward axis as well. However, we claim (without a proof) that already SAT-XPath($\uparrow, \rightarrow, =$) has a non-primitive recursive lower bound. Indeed, this can be proved by reusing the results on lower bounds of [11]. The cited work shows that XPath with one non-reflexive transitive axis is enough to prove non-primitive recursiveness provided that the axis is functional (i.e., the transitive closure of an axis like $\rightarrow$, $\leftarrow$, $\uparrow$ but unlike $\downarrow$). Here, however, we feature reflexive-transitive axes instead of only transitive. Therefore, in principle we cannot use this result. However, one can somehow code $\uparrow$ with $\rightarrow^a[t][¬a]$ for some label $a$. We leave the proof of this claim for the journal version of the present work.

By the previous claim, although it could be that full transitive XPath is decidable, it would have a non-primitive recursive lower bound. We can then answer negatively to the conjecture proposed in [11, Conjecture 2], stating that XPath($\leftarrow, \downarrow, \uparrow, \rightarrow, =$) is decidable in elementary time.

**Future work**

- The present work can be seen as a step forward in answering [11] Conjecture 1, suggesting that the extension of XPath($\leftarrow, \downarrow, \rightarrow, =$) with the child axis is decidable with elementary complexity. Our approach may perhaps be extended to handle the child relation.
- We suspect that XPath($\leftarrow, \downarrow, \uparrow, \rightarrow, =$) is in fact hard for $2\text{ExpSpace}$, even when the formulas are in direct normal form, and hence that SAT-directed-XPath($\leftarrow, \downarrow, \uparrow, \rightarrow, =$) is $2\text{ExpSpace}$-complete.
- We would also like to investigate further the approach taken in this paper to attempt to generalize it to work with the class of reflexive-transitive closures of regular languages.

8. REFERENCES


A. APPENDIX TO SECTION 4

LEMMA A.1 (Distributivity of (·, ∪)). For every \( P, P' \subseteq \mathcal{P} \) and \( \chi, \chi' \subseteq D \times \Pi \)

1. \( P(\chi \cup \chi') = P\chi \cup P\chi' \),
2. \( (P \cup P')\chi = P\chi \cup P'\chi \).

Proof. We have that

\[
P(\chi \cup \chi')
= \chi \cup \chi' \cup \{(d, \alpha' \cdot \alpha, \beta, i) \mid (d, \alpha, \beta, i) \in \chi \cup \chi', \alpha' \in P\}
= \chi \cup \chi' \cup \{(d, \alpha' \cdot \alpha, \beta, i) \mid (d, \alpha, \beta, i) \in \chi, \alpha' \in P\}
\]

(\text{by definition of } ')

\[
= \chi \cup \{(d, \alpha' \cdot \alpha, \beta, i) \mid (d, \alpha, \beta, i) \in \chi, \alpha' \in P\}
\]

\[
= \chi \cup \{(d, \alpha' \cdot \alpha, \beta, i) \mid (d, \alpha, \beta, i) \in \chi, \alpha' \in P\}
\]

which proves the first statement.

On the other hand,

\[
(P \cup P')\chi
= \chi \cup \{(d, \alpha' \cdot \alpha, \beta, i) \mid (d, \alpha, \beta, i) \in \chi, \alpha' \in P \cup P'\}
= \chi \cup \{(d, \alpha' \cdot \alpha, \beta, i) \mid (d, \alpha, \beta, i) \in \chi, \alpha' \in P \}
\]

\[
\cup \{(d, \alpha' \cdot \alpha, \beta, i) \mid (d, \alpha, \beta, i) \in \chi, \alpha' \in P'\}
\]

\[
= \chi \cup \{(d, \alpha' \cdot \alpha, \beta, i) \mid (d, \alpha, \beta, i) \in \chi, \alpha' \in P'\}
\]

(\text{by definition of } ')

which proves the second statement. \(\Box\)

LEMMA A.2. If \( f_1 + f_2 = f_3 \) and \( f_1, f_2 \in \mathcal{F} \), then \( f_3 \in \mathcal{F} \).

Proof. If \( \chi_3(\alpha, \beta, i) = \{d\} \), then \( \chi_1(\alpha, \beta, i) = \{d\} \), and hence \( d \in R_4 \) and therefore \( d \in R_4 = R_1 \).

A symmetrical argument applies for \( \chi_3 \) and \( \chi_2 \). \(\Box\)

LEMMA A.3 (Distributivity of (·, +)). For every \( f_1, f_2 \in \mathcal{F} \) and \( \chi \subseteq D \times \Pi \)

- \( [f_1 + f_2]\cdot \chi = [f_1]\cdot ([f_2]\cdot \chi) \),
- \( [f_1 + f_2]\cdot \chi = [f_2]\cdot ([f_1]\cdot \chi) \).

Proof. Let \( f_3 = f_1 + f_2 \).

\[
[f_1 + f_2]\cdot \chi
= \{(\alpha \mid (d, \alpha, \beta, i) \in \chi) \}
\]

(by definition of [ ]

\[
= \{(\alpha \mid (d, \alpha, \beta, i) \in \chi) \} \cup \{(d, \alpha, \beta, i) \mid (d, \alpha, \beta, i) \in \chi, \alpha' \in P\}
\]

(by definition of \( f_1 + f_2 \))

\[
= \{(\alpha \mid (d, \alpha, \beta, i) \in \chi) \} \cup \{(d, \alpha, \beta, i) \mid (d, \alpha, \beta, i) \in \chi, \alpha' \in P\}
\]

(\(\cup\)) distributivity

which proves the statement.
\[(d, \alpha' \cdot \alpha, \beta, i) \mid (d, \alpha, \beta, i) \in \chi, \alpha' \in [f_1]_\chi \cup [f_1]_\chi \]
\[= [f_1]_\chi \cup [f_1]_\chi \cup [f_1]_\chi \]
\[= [f_1]_\chi \cup [f_1]_\chi \cup [f_1]_\chi \]
\[= [f_1]_\chi \cup [f_1]_\chi \cup [f_1]_\chi \]
\[= [f_1]_\chi \cup [f_1]_\chi \cup [f_1]_\chi \]
\[= [f_1]_\chi \cup [f_1]_\chi \cup [f_1]_\chi \]
\[= [f_1]_\chi \cup [f_1]_\chi \cup [f_1]_\chi \]
\[= [f_1]_\chi \cup [f_1]_\chi \cup [f_1]_\chi \]
\[= [f_1]_\chi \cup [f_1]_\chi \cup [f_1]_\chi \]

Hence, we obtain that \(\chi^e_2 = \chi^e_2 \cup [f_2]_\chi \cdot \chi^e_2\), and thus condition \(4\) holds for \(f_2 + f_3\).

On the other hand, assuming that \(f_1 + (f_2 + f_3)\), we can show that the conditions to apply \((f_1 + f_2) + f_3\), we can show that the conditions to apply \((f_1 + f_2) + f_3\) hold, in the same way as before, since all the definitions are symmetrical, where \(\chi^e_1 = \chi^e_1 \cup [f_2]_\chi \cdot \chi^e_2\)

and \(\chi^e_2 = \chi^e_1 \cup [f_2]_\chi \cdot \chi^e_2\).

We now show that \((f_1 + f_2) + f_3 = f_1 + (f_2 + f_3)\). Let \(f_1 + (f_2 + f_3)\) and \((f_1 + f_2) + f_3\).

We have that \(\chi^e_{1(23)} = \chi^e_{23} = \chi^e_{12} = \chi^e_{123}\) by item condition \(5\) of +. In a similar way, by condition \(+3\) we have that \(\chi^e_{1(23)} = \chi^e_{123}\). On the other hand, we have

\[\chi^e_{1(23)} = \chi^e_{1} \cup [f_1]_\chi \cdot \chi^e_{23}\]
\[= \chi^e_{1} \cup [f_1]_\chi \cdot (\chi^e_{2} \cup [f_2]_\chi \cdot \chi^e_{3})\]
\[= \chi^e_{1} \cup [f_1]_\chi \cdot \chi^e_{2} \cup [f_2]_\chi \cdot \chi^e_{3}\]
\[= \chi^e_{1} \cup [f_1]_\chi \cdot \chi^e_{2} \cup [f_2]_\chi \cdot \chi^e_{3}\]

and by a similar reasoning using condition \(+5\) we obtain that \(\chi^e_{1(23)} = \chi^e_{123}\). Hence, \((f_1 + f_2) + f_3 = f_1 + (f_2 + f_3)\). \(\square\)

**Lemma A.5.** For every \(f_1, f_2 \in (a, d)fs\) and every \(\chi \subseteq \mathbb{D} \times \Pi,\)
\[\chi = [f_1]_\chi \cdot \chi = [f_2]_\chi \cdot \chi\]
\[\chi = [f_1]_\chi \cdot \chi = [f_2]_\chi \cdot \chi\]
\[\chi = [f_1]_\chi \cdot \chi = [f_2]_\chi \cdot \chi\]

**Proof.** By condition \(4\) of rooting and definition of \([\cdot, \cdot]\), we have that \([f_1]_\chi = [f_1]_\chi \cdot \chi = [f_2]_\chi = \alpha_\sigma\). By definition of \(\sigma_\alpha\), it is immediate that \(\alpha_\sigma \cdot \alpha_\sigma = \alpha_\sigma \cdot \chi\). Hence, the statement follows. \(\square\)

**Lemma A.6.** For every \(f_1 \in (a, d)fs,\)
\[\chi^e_{1} = [f_1]_\chi \cdot \chi^e_{1} = [f_1]_\chi \cdot \chi^e_{1}\]
\[\chi^e_{1} = [f_1]_\chi \cdot \chi^e_{1} = [f_1]_\chi \cdot \chi^e_{1}\]

**Proof.** By condition \(4\) of rooting, \([f_1]_\chi = \alpha_\sigma\) and since \(\chi^e_{1} = \{(d, \alpha, \beta, \gamma, i) \mid 3 \alpha', (d, \alpha', \gamma, i) \in \chi^e_{1} \cup \chi^e_{2}, \alpha, \beta \in \alpha\} \cup \bigcup_{i \in [k]} \{(d(i)) \times (\alpha \setminus \{e\}) \times \alpha \times \{i\}\}\) it follows that \(\alpha_\sigma \cdot \chi^e_{1} = \chi^e_{1}\), obtaining that \(\chi^e_{1} = [f_1]_\chi \cdot \chi^e_{1}\). Since \(\chi^e_{1} = \chi^e_{1}\) by condition \(4\), we also obtain that \(\chi^e_{1} = [f_1]_\chi \cdot \chi^e_{1}\). \(\square\)

**B. APPENDIX TO SECTION 5**

**Lemma B.1.** \(\preceq_{\chi} \text{ and } \succeq_{\chi}\) are partial orders over \(\mathcal{G}\).
Proof. If \( f \preceq f' \preceq f'' \), suppose \( f' = [d_1 \mapsto d_1, d'_1 \mapsto d'_1, \ldots, d_n \mapsto d_n, d'_n \mapsto d'_n] \) and \( f'' = [e_1 \mapsto e_1, e'_1 \mapsto e'_1, \ldots, e_m \mapsto e_m, e'_m \mapsto e'_m] \). Therefore, \( f'' = [d_1 \mapsto d_1, d'_1 \mapsto d'_1, \ldots, d_n \mapsto d_n, d'_n \mapsto d'_n, e_1 \mapsto e_1, e'_1 \mapsto e'_1, \ldots, e_m \mapsto e_m, e'_m \mapsto e'_m] \) and hence \( f \preceq f'' \).

On the other hand, it is plain that \( \sim \) preserves transitivity and reflexivity, and that \( f \preceq f \). □

**Lemma B.2.** If \( f_1 \preceq f_2 \) then \( \{f_1\} = \{f_2\} \) and \( \{f_1\} = \{f_2\} \).

Proof. This is immediate from the definition of \( \{ \cdot \} \). □

**Lemma B.3.** If \( f \preceq f' \) then \( \{f\} = \{f'\} \).

Proof. Suppose, without any loss of generality, that \( f' = [d \mapsto d, d'] \), where \( d \in \text{data}(f) \setminus R \) and \( d' \not\in \text{data}(f) \). Since \( d' \) behaves just as \( d \), we have that, for any \( \alpha, \alpha' \in \Pi, a, a' \in \{\varnothing, \exists, \sigma^i, \sigma^f, \sigma^o\} \), \( d' \in f'_i, \chi_a(\alpha) \cap \chi_{a'}(\alpha') \) iff \( d \in f_i, \chi_a(\alpha) \cap \chi_{a'}(\alpha') \). Hence, \( f_i, \chi_a(\alpha) \cap \chi_{a'}(\alpha') \neq \varnothing \) iff \( f_i, \chi_a(\alpha) \cap \chi_{a'}(\alpha') \neq \varnothing \).

Further, since \( d \not\in R \) we have that \( d \in \chi_a(\alpha) \) in \( f' \) (or in \( f \)) then \( |\chi_a(\alpha)| > 1 \). Hence \( \{f_i(\alpha, a) = \{f_i(\alpha, a) \) = \{f_i'(\alpha, a) \) = \{f_i'(\alpha, a) \) whenever \( d \in f_i \chi_a(\alpha) \). Further, since \( f \) preserves all the data values of \( f \) with their descriptions, \( \{f_i(\alpha, a) \) = \{f_i'(\alpha, a) \) whenever \( d \not\in f_i \chi_a(\alpha) \cap f_i \chi_{a'}(\alpha') \). □

**Lemma B.4.** If \( d \not\in \text{data}(f) \setminus R \) and \( d' \not\in \text{data}(f) \), we have that if \( f \not\in \) then \( \{f[d \mapsto d, d'] \not\in \) \.

Proof. This is immediate from the definition of validity. Since \( d' \not\in \text{data}(f) \), then \( f' = [d \mapsto f, d'] \) is the result of adding some fresh data value \( d' \not\in R \) with the same description as \( d' \). Since \( f' \not\in \), there is some \( \{d''\} = \chi^{\varnothing}(\bar{\alpha}) \) for \( \bar{\alpha} \in \Pi \) and \( d'' \not\in R \). By definition of \( f' \), this means that \( d'' \neq d \) and \( d'' \neq d' \), and hence that \( \{d''\} = \chi^{\varnothing}(\bar{\alpha}) \), and thus \( f \not\in \), which is a contradiction. A symmetric argument can also be applied to \( \chi^{\varnothing} \). □

**Appendix to Section 5.3**

**Lemma 5.1** (Figure 4). For every \( f_1 \preceq f_2 \in (a,d)\{f_3 \), there is \( n \in \mathbb{N} \), and \( f'_i \subseteq (a,d)\{f_3 \) for every \( i \in [n] \) so that

\[
\{f_1\} = \{f_1'\} + \cdots + \{f_n'\}.
\]

Proof of Lemma 5.1 Since \( f_1 \preceq f_2 \), suppose

\[
\{f_1\} = f_2[d_1 \mapsto d_1, d'_1 \mapsto d'_1, \ldots, d_t \mapsto d_t, d'_t] \tag{**}
\]

where \( \{d_1, \ldots, d_t\} \subseteq \text{data}(f_2) \setminus R_2; \{d'_1, \ldots, d'_t\} \cap \text{data}(f_2) = \{d'_1, \ldots, d'_t\} \) for all \( j \). We define \( n = t + 2 \).

The idea is that we will define \( f'_0, f'_1, \ldots, f'_{t+1} \) so that \( f'_i \) is, modulo some renaming of data values, the same as \( f_2 \) for all strictly internal values, and with the same descriptions. However, \( f'_i \) has more external data values than \( f_2 \) —namely \( d'_1, \ldots, d'_t \) —that have basically the same descriptions as \( d_1, \ldots, d_t \) in \( f_2 \). Although technically we could make use of less profiles, we prefer to define \( t+1 \) different profiles, to preserve some symmetries in the definitions, and thus to simplify proofs. We define \( f'_0, f'_1, \ldots, f'_{t+1} \) in Figure 7.

We shall now show that for every \( f'_i \) there is \( i \) so that \( f'_i \sim f''_i \) so that \( f'_i \sim f''_i \) in \( (a, d)\{f_3 \), the fact that they all have the same fingerprints is because:

- For every \( a \in A, \bar{a} \in \Pi \), and every data value \( e \in \text{data}(f_2) \), we have that \( e \in \chi_a(\bar{a}) \in f_2 \) iff \( e \in \chi_a(\bar{a}) \in f'_i \). This is because, since \( \chi_{a+1}(e) = \chi_0(e) = \chi_{\bar{a}'}(e), \chi_{a+1}(\bar{e}) = \chi_0(e), \chi_{a+1}(e) = \chi_{\bar{a}'}(e) \), we have that: for all \( a \in \{o, \sigma^i, \sigma^f\}, \chi_a(e) \) in \( f'_0 \) is equal to \( \chi_a(e) \) in \( f_2 \); and for all \( a \in \{o, \sigma^i, \sigma^f\}, \chi_a(e) \) in \( f'_{t+1} \) is equal to \( \chi_a(e) \) in \( f_2 \). And further,

- If \( a = \varnothing \)

\[
\chi_{\varnothing}(e) = \chi_0(e) \cup \chi_{\varnothing}'(e) = \chi_0'(e) \cup \chi_{\varnothing}'(e) \cup \chi_{\varnothing}'(e)
\]

(by Lemmas ?? and ??)

- If \( a = \exists \),

\[
\chi_{\exists}(e) = \chi_{\exists}'(e) \cup \{f_2[e_{t+1}] \chi_{\exists}'(e)
\]

- If \( a = \sigma^i \),

\[
\chi_{\sigma^i}(e) = \chi_{\sigma^i}'(e) \cup \{f_{t+1}[e_{t+1}] \chi_{\sigma^i}'(e)
\]

- If \( a = \sigma^f \),

\[
\chi_{\sigma^f}(e) = \chi_{\sigma^f}'(e) \cup \{f_{t+1}[e_{t+1}] \chi_{\sigma^f}'(e)
\]
We now check that Claim $i$ holds, we are going to show that conditions (b) and (c) of concatenation share the same set of internal data values, and that the internal data values $d_j$ and $d_j'$ share the same set of internal data values, and hence $d_j, d_j' \in f_2, \chi_2(a)$. Hence, $|f_2, \chi_2(a) \cap f_2, \chi_2(a')| \geq 1$ if $|f_2, \chi_2(a) \cap f_2, \chi_2(a')| \geq 1$.

As a result the fingerprint of $f_2$ and $f_0$ are equal, and hence $\xi(f_0, f_2) = \xi(f_2) \in \Gamma$. The same reasoning applies to $f_{t+1}$.

Thus, $f_0, f_{t+1} \in (a, d)\bar{f}_3$.

We now show that for every $i \in [t]$, there is $\xi'$ so that $f_i' \sim \xi'$ for every $i \in [t]$. By the properties of $f_i'$ and $f_2$, and the same set of internal data values, and the internal descriptions of these internal data values are the same in $f_0'$ and $f_2$. Also in this case we have that $\xi(f_i') = \xi(f_i) = \xi(f_2) \in \Gamma$. The same reasoning applies to $f_{t+1}$.

Thus, $f_0, f_{t+1} \in (a, d)\bar{f}_3$ since $f_2 \in (a, d)\bar{f}_3$.

We now check that $f_1 = f_0 + \cdots + f_{t+1}$.

CLAIM B.4.2. $f_1 = f_0 + \cdots + f_{t+1}$.

PROOF. For each data value $d$, we are going to show that conditions 1 and 2 of concatenation hold, and that $\chi_1(d)$ equals to the profile of $d$ in $f_1, + \cdots + f_{t+1}$.

Let us take any data value $d \in data(f_2) \setminus \{d_1, \ldots, d_t\}$. We first check that condition 1 holds for $d$, in other words, that for every $i \in \{0, \ldots, t\}$,

$\chi_i(d) \cup [f_i, \chi_i'](d) = \chi_i(d)$.
We have
\[
\overline{\chi_{i+1}}(d) \cup [f_{i+1}]^{\overline{\chi_{i+1}}}(d) = \overline{\chi_{i+1}}(d) \cup [f_i]^{\overline{\chi_{i+1}}}(d) = \overline{\chi_i}(d) \cup [f_2]^{\overline{\chi_i}}(d) \quad \text{(by Lemma ??)}
\]
\[
= \begin{cases} 
\overline{\chi_i}(d) \cup [f_2]^{\overline{\chi_i}}(d) & \text{if } i = t, \\
\overline{\chi_i}(d) \cup [f_2]^{\overline{\chi_i}}(d) \cup [f_2]^{\overline{\chi_i}}(d) & \text{otherwise.}
\end{cases}
\]

If \( i = t \), we further have that \( \overline{\chi_i}(d) \cup [f_2]^{\overline{\chi_i}}(d) = \overline{\chi_i}(d) \), verifying \([b]\). Otherwise, if \( i \neq t \),
\[
\overline{\chi_{i+1}}(d) \cup [f_{i+1}]^{\overline{\chi_{i+1}}}(d) = \cdots = 
\overline{\chi_2}(d) \cup [f_2]^{\overline{\chi_2}}(d) \cup [f_2]^{\overline{\chi_2}}(d) = \overline{\chi_i}(d) \cup [f_2]^{\overline{\chi_i}}(d) \cup [f_2]^{\overline{\chi_i}}(d) 
= \overline{\chi_i}(d) \cup [f_2]^{\overline{\chi_i}}(d) \cup [f_2]^{\overline{\chi_i}}(d) 
= [f_2]^{\overline{\chi_i}}(d) \cup [f_2]^{\overline{\chi_i}}(d) \cup [f_2]^{\overline{\chi_i}}(d) 
= [f_2]^{\overline{\chi_i}}(d) \cup [f_2]^{\overline{\chi_i}}(d) \cup [f_2]^{\overline{\chi_i}}(d) 
= \overline{\chi_i}(d). 
\]

In any case, condition \([b]\) holds between \( f_i \) and \( f_{i+1} \) for the data value \( d \). Note that since all definitions are symmetrical, it also follows that condition \([b]\) holds for \( d \).

We now check that, if we call \( f_+ \) to \( f_+ + \cdots + f_+ \), then \( \overline{\chi_+}(d) = \overline{\chi_1}(d) \). Here we only deal with \( \overline{\chi_+} \) and \( \overline{\chi_1} \) because the cases for \( \overline{\chi_+} \) and \( \overline{\chi_1} \) are symmetrical. By definition of \( + \), we have that
\[
\overline{\chi_+}(d) = \overline{\chi_2}(d) = \overline{\chi_1}(d), 
\]
\[
(\overline{\chi_2} \cup [f_2]^{\overline{\chi_2}}(d) \cup [f_2]^{\overline{\chi_2}}(d) \cup \cdots \cup [f_2]^{\overline{\chi_2}}(d)) \quad \text{(by \([b]\))}
\]
\[
= \overline{\chi_2}(d) = \overline{\chi_1}(d). 
\]

- Suppose now that we have \( d = d_j \) for some \( j \in [t] \). We check that condition \([b]\) holds for \( d \), in other words that for every \( i \in \{0, \ldots, t\} \),
\[
\overline{\chi_{i+1}}(d) \cup [f_{i+1}]^{\overline{\chi_{i+1}}}(d) = \overline{\chi_i}(d).
\]

We have
\[
\overline{\chi_{i+1}}(d) \cup [f_{i+1}]^{\overline{\chi_{i+1}}}(d) = \overline{\chi_{i+1}}(d) \cup [f_2]^{\overline{\chi_{i+1}}}(d) \quad \text{(by Lemma ??)}
\]
If \( i = t \),
\[
\overline{\chi_t}(d) \cup [f_2]^{\overline{\chi_t}}(d) = \overline{\chi_t}(d) \cup [f_2]^{\overline{\chi_t}}(d) = \overline{\chi_t}(d). 
\]
If \( i \neq t \) and \( j = i + 1 \),
\[
\overline{\chi_{i+1}}(d) \cup [f_2]^{\overline{\chi_{i+1}}}(d) = \emptyset \cup [f_2]^{\overline{\chi_{i+1}}}(d) = [f_2]^{\overline{\chi_{i+1}}}(d) \quad \text{(by definition)}
\]
\[
= [f_2]^{\overline{\chi_2}}(d) \cup [f_2]^{\overline{\chi_2}}(d) \quad \text{(by \([b]\))}
\]
\[
= [f_2]^{\overline{\chi_2}}(d) \cup [f_2]^{\overline{\chi_2}}(d) \quad \text{(by \([b]\))}
\]
\[
= \overline{\chi_2}(d) \cup [f_2]^{\overline{\chi_2}}(d) \quad \text{(by \([b]\))}
\]
\[
= \overline{\chi_2}(d). 
\]
If \( i \neq t \) and \( j \neq i + 1 \),
\[
\overline{\chi_{i+1}}(d) \cup [f_2]^{\overline{\chi_{i+1}}}(d) = \overline{\chi_i}(d) \cup [f_2]^{\overline{\chi_i}}(d) \quad \text{(by definition)}
\]
\[
= [f_2]^{\overline{\chi_i}}(d) \cup [f_2]^{\overline{\chi_i}}(d) \quad \text{(by \([b]\))}
\]
\[
= \overline{\chi_i}(d). 
\]
We check that $\bar{\chi}_i^+(d) = \bar{\chi}_i^+(d)$ and $\bar{\chi}_i^+(d) = \bar{\chi}_i^+(d)$. By definition of $+$, we have that

$$
\bar{\chi}_i^+(d) = \bar{\chi}_i^+(d),
$$

(by definition)

$$
\bar{\chi}_i^+(d) = \bar{\chi}_i^+(d),
$$

(by Lemma ??)

$$
\bar{\chi}_i^+(d) = \bar{\chi}_i^+(d),
$$

(by Lemma ??)

$$
\bar{\chi}_i^+(d) = \bar{\chi}_i^+(d).
$$

(by definition)

\[ \text{Condition [4] follows by symmetry.} \]

We check that $\bar{\chi}_i^+(d) = \bar{\chi}_i^+(d)$ and $\bar{\chi}_i^+(d) = \bar{\chi}_i^+(d)$. By definition of $+$, we have that

$$
\bar{\chi}_i^+(d) = \bar{\chi}_i^+(d),
$$

(by definition)

$$
\bar{\chi}_i^+(d) = \bar{\chi}_i^+(d),
$$

(by Lemma ??)

$$
\bar{\chi}_i^+(d) = \bar{\chi}_i^+(d),
$$

(by Lemma ??)

$$
\bar{\chi}_i^+(d).
$$

(by definition)

\[ \text{Finally, suppose } d = d'_j \text{ for some } j \in [t]. \text{ We check that condition [4] holds for } d, \text{ in other words that for every } i \in \{0, \ldots, t\}, \]

$$
\bar{\chi}_i^+(d'_j) \cup \bar{\chi}_i^+(d'_j) = \bar{\chi}_i^+(d'_j).
$$

By definition, we have

$$
\bar{\chi}_i^+(d'_j) \cup \bar{\chi}_i^+(d'_j) = \bar{\chi}_i^+(d'_j) \cup \bar{\chi}_i^+(d'_j),
$$

(by Lemma ??)

If $i = t$,

$$
\bar{\chi}_i^+(d'_j) \cup \bar{\chi}_i^+(d'_j) = \bar{\chi}_i^+(d'_j) \cup \bar{\chi}_i^+(d'_j),
$$

(by definition)

$$
\bar{\chi}_i^+(d'_j).
$$

(by definition)

If $i \neq t$ and $j = i + 1$,

$$
\bar{\chi}_i^+(d'_j) \cup \bar{\chi}_i^+(d'_j) = \bar{\chi}_i^+(d'_j) \cup \bar{\chi}_i^+(d'_j),
$$

(by definition)

$$
\bar{\chi}_i^+(d'_j) \cup \bar{\chi}_i^+(d'_j),
$$

(by Lemma ??)

$$
\bar{\chi}_i^+(d'_j).
$$

(by definition)

If $i \neq t$ and $j > i + 1$,

$$
\bar{\chi}_i^+(d'_j) \cup \bar{\chi}_i^+(d'_j) = \bar{\chi}_i^+(d'_j) \cup \bar{\chi}_i^+(d'_j),
$$

(by definition)

$$
\bar{\chi}_i^+(d'_j) \cup \bar{\chi}_i^+(d'_j),
$$

(by Lemma ??)

$$
\bar{\chi}_i^+(d'_j).
$$

(by definition)

If $i \neq t$ and $j < i + 1$,

$$
\bar{\chi}_i^+(d'_j) \cup \bar{\chi}_i^+(d'_j) = \bar{\chi}_i^+(d'_j) \cup \bar{\chi}_i^+(d'_j),
$$

(by definition)

$$
\bar{\chi}_i^+(d'_j). \quad \text{ (since } \bar{\chi}_1(d'_j) = \bar{\chi}_2(d'_j) \text{ by ??)}
$$


We check that $\bar{\chi}_i^+(d'_j) = \bar{\chi}_i^+(d'_j)$ and $\bar{\chi}_i^+(d'_j) = \bar{\chi}_i^+(d'_j)$. By definition of $+$, we have that

$$
\bar{\chi}_i^+(d'_j) = \bar{\chi}_i^+(d'_j),
$$

(by definition)
Therefore, we have that condition (a) as well as conditions (b) and (c) hold true for every data value, and that \( f_0 + \cdots + f'_i + 1 \) and \( f_i \) coincide. Thus, the Claim follows.

This concludes our proof.

As a corollary of the previous lemma, we obtain the following.

**Lemma B.5** (Figure 8). For every \( f \geq f_1 + \cdots + f_i \) with \( f_i \in (a_i, d_i) \) for every \( i \in [t] \), let \( \bar{f}_i \) be the forest of immediate subtrees of \( f_i \). Then, there must be a forest \( \bar{f} \) such that all the profiles in its derivation tree have no boundedly many rigid values.

**Proof of Lemma B.5.** This is a direct consequence of Lemma 5.11. Since \( f \geq f_1 + \cdots + f_i \), then

\[
\bar{f} = \bar{f}_1, \ldots, \bar{f}_i + 1, \ldots, \bar{f}_1, \ldots, \bar{f}_i,
\]

for some \( d_1, \ldots, d_i, d_i', \ldots, d_i'' \in \mathbb{D} \) so that \( \{d_1, \ldots, d_i\} \cap R = \emptyset \) and \( \{d_i', \ldots, d_i''\} \cap \text{data}(f_1 + \cdots + f_i) = \emptyset \). It follows that

\[
\bar{f} = (f_1 + \cdots + f_i)[d_1 \mapsto d_1, d_i'] \cdots [d_i \mapsto d_i, d_i'']
\]

Note that \( f_i[d_1 \mapsto d_1, d_i'] \cdots [d_i \mapsto d_i, d_i''] \leq f_i \in (a_i, d_i) \) for every \( i \in [t] \). We can hence apply Lemma 5.11 and we obtain, for some \( n_i \in \mathbb{N}, f_i, f_i', f_i'' \in (a_i, d_i) \) so that \( f_i + f_i' + f_i'' = f_i[d_1 \mapsto d_1, d_i'] \cdots [d_i \mapsto d_i, d_i''] \). Therefore,

\[
\bar{f} = f_i[d_1 \mapsto d_1, d_i'] \cdots [d_i \mapsto d_i, d_i''] + \cdots + f_i[d_1 \mapsto d_1, d_i'] \cdots [d_i \mapsto d_i, d_i'']
\]

which concludes the proof.

**Lemma 5.2** \( \mathcal{D} = \uparrow \mathcal{D} \).

**Proof of Lemma 5.2** This is a direct consequence of Lemma B.5. \( \square \)

**Lemma 5.3** (Figure 5). If \( f + f_2 = f_1 \) and \( f_1 \preceq f_1, f_2 \preceq f_2 \), then \( f_1 + f_2 \preceq f_3 \), for some \( f_1, f_2, f_3 \in \mathcal{D} \) so that \( f_3 \) is a bounded extension of \( f_i \) for all \( i \in \{1, 2\} \).

**Proof of Lemma 5.3** For every description \( \pi \) so that \( \tilde{\chi}(\pi) \setminus R_3 \neq \emptyset \), let \( d_b \in \mathcal{D} \) be a data value so that \( d_b \in \tilde{\chi}(\pi) \setminus R_3 \). Let \( D \) be the set of all these data values. Note that \( |D| \leq |\Pi|^4 \).

We define \( f'' = (R_3, f''_i) \), where

\[
\tilde{\chi}''_i(d) = \begin{cases} 
\tilde{\chi}_i(d) & \text{if } d \notin D \\
\tilde{\chi}_i(d) & \text{if } d \in D.
\end{cases}
\]
Note that $\text{data}(\tilde{f}_i') \leq \text{data}(f_i) + |D| \leq \text{data}(f_i) + |\Pi|^4$.

CLAIM B.5.1. $f_i' \preceq f_i'^i \preceq f_i$, for all $i \in \{1, 2\}$.

PROOF. By definition of $\preceq$, for every $d \in D$ there is some $d' \in \text{data}(f_i') \setminus R_i$ so that $\tilde{x}_i(d) = \tilde{x}_i'(d')$. Therefore, $f_i' = f_i'[d_1 \mapsto d_1', \ldots][d_n \mapsto d_n']$ where $\{d_1, \ldots, d_n\} = D \setminus \text{data}(f_i')$ and for every $j \in [n]$, $d_j' \not\in R_j$, $d_j \not\in \text{data}(f_i')$, and $\tilde{x}_i'(d_j') = \tilde{x}_i(d_j)$. In other words, we have that $f_i' \preceq f_i'^i \preceq f_i$, for all $i \in \{1, 2\}$. \qed

Therefore, $f_i'^i$ is a bounded extension of $f_i'$, for $i \in \{1, 2\}$.

CLAIM B.5.2. $f_i'^i + f_i'^2 \preceq f_3$.

PROOF. First, we have that $f_i'^i, f_i'^2 \in \mathcal{F}$ because otherwise $f_1$ or $f_2$ would not be in $\mathcal{F}$.

Since $f_i'^i \preceq f_i$, by Lemma ?? it follows that $f_3 = f_3' \equiv f_3''$, \quad (\dagger)

This means that, since $f_i'^i$ is the result of adding data values $d$ with profile $\tilde{x}_i(d)$, if conditions \dagger, \ddagger, \S hold for $f_1 + f_2$ they must also hold for $f_i'^i + f_i'^2$. Therefore, all the preconditions to apply $f_1 + f_2$ hold. Let $f_i'^i = f_i'^i + f_i'^2$.

Note that all the profiles in question share the same set of rigid values, $\mathcal{R} = \mathcal{R}' = \mathcal{R}'' = \mathcal{R} = \mathcal{R} \cup \mathcal{R} \cup \mathcal{R} \cup \mathcal{R} = \mathcal{R} \cup \mathcal{R} \cup \mathcal{R} \cup \mathcal{R} = \mathcal{R} \cup \mathcal{R} \cup \mathcal{R}$. Every data value $d \in \text{data}(f_i'^i)$ is so that $\tilde{x}_i(d) = \tilde{x}_i(d)$, and with (\dagger) this means that $\tilde{x}_i(d) = \tilde{x}_i(d)$. For every other data value $d \in \text{data}(f_i') \setminus \text{data}(f_i'^i)$, it must be that $d \not\in \mathcal{R}$ and there must be some $d \in D$ so that $\tilde{x}_i(d) = \tilde{x}_i(d)$ by definition of $D$. Then, if $\text{data}(f_3) \setminus \text{data}(f_i'^i) = \{d_1, \ldots, d_n\}$ we have that $f_3 = f_3'[d_1 \mapsto d_1', \ldots][d_n \mapsto d_n']$, and hence, $f_i'^i \preceq f_3$. \qed

By Claims ?? and ??, and since $\text{data}(f_i'^i) \leq \text{data}(f_i') + |\Pi|^4$, the lemma follows. \qed

LEMMA 5.4 (FIGURE 5). For every $f_1 \in (a, d)\bar{f}_2$ and $f_3 \geq f_3$, there is $f_1 \geq f_3$ and $f_1 \not\in (a, d)\bar{f}_3$ so that $|\text{data}(f_1)| \leq |\text{data}(f_3)| + |\Pi|^4 + |\Pi| + |\mathcal{R}|$ and $f_1 \preceq f_3$.

PROOF OF LEMMA 5.4. We are given $f_1, f_2, f_3$ and $(a, d)$. We show that there must be some $f_2$ and $f_3$ so that $f_1 \geq f_2 \preceq f_3$ with $f_1 \geq f_3$. Figure 5 contains a graphical representation of the profiles we work with in the proof.

We first define $f_4$. For every internal description $\pi$ so that $\mathcal{R}_i \cup \mathcal{R}_i \not\subseteq \emptyset$ and $\tilde{x}_i(\pi) \setminus \mathcal{R}_i \not\subseteq \emptyset$, let $d_\emptyset \in \emptyset$ be a data value of $\tilde{x}_i(\pi) \setminus \mathcal{R}_i$. Let $D$ be the set of all such data values. Note that $|D| \leq |\Pi|^4$.

$$f_4 : \begin{cases} R_4 = \mathcal{R} \setminus \mathcal{R}_i & \text{for every } d \in D \cup \mathcal{R}_i, \tilde{x}_4(d) = \tilde{x}_4(d) \\ R_4 = \emptyset & \text{for every } d \not\in D \cup \mathcal{R}_i, \tilde{x}_4(d) = \tilde{x}_4(d) \end{cases}$$

Note that $|\text{data}(f_4)| \leq |\text{data}(f_3)| + |\Pi|^4 + |\mathcal{R}|$. Since $f_3 \geq f_3$, we have that $f_3 \leq f_3 \leq f_3$.

CLAIM B.5.3. $f_3 \succeq f_4$. \quad (\dagger)

PROOF. First notice that $R_3 = R_2$ since $f_3 \succeq f_3$, and that $R_4 = R_3$ by definition of $R_4$. We then have that for every $d \in R_3 = R_4$, $\tilde{x}_4(d) = \tilde{x}_4(d) = \tilde{x}_4(d)$.

We show that for every $d \in \text{data}(f_4)$, $\tilde{x}_4(d) = \tilde{x}_4(d)$. If $d \in \text{data}(f_4) \setminus (D \cup \mathcal{R}_i)$, we have that $\tilde{x}_4(d) = \tilde{x}_4(d)$. If on the other hand $d \in D \cup \mathcal{R}_i$, we have $\tilde{x}_4(d) = \tilde{x}_4(d)$. Then $d \in \text{data}(f_4)$, and since $f_3 \succeq f_2$ we have that either $d \not\in \text{data}(f_3)$ or $\tilde{x}_3(d) = \tilde{x}_3(d)$. Hence, for every $d \in \text{data}(f_3)$, $\tilde{x}_4(d) = \tilde{x}_4(d)$. We finish the proof of $f_3 \succeq f_4$, by showing that for every $d \in \text{data}(f_4) \setminus \mathcal{R}_4$ there is some $d' \in \text{data}(f_3) \setminus \mathcal{R}_4$ so that $\tilde{x}_4(d) = \tilde{x}_4(d')$. If $d \not\in D \cup \mathcal{R}_i$, then of course we can take $d' = d$ and $\tilde{x}_4(d) = \tilde{x}_4(d')$. If $d \in D \cup \mathcal{R}_i$, we have that $\tilde{x}_4(d) = \tilde{x}_4(d)$ by definition. Since $f_3 \succeq f_3$, there must be some $d' \in \text{data}(f_3) \setminus \mathcal{R}_4$ so that $\tilde{x}_4(d) = \tilde{x}_4(d')$, and we are done. This finishes the proof that $f_3 \succeq f_4$.

We now show that $f_4 \preceq f_3$.

Again, notice that $R_4 = R_2 = R_5$, and since for every $d \in D$, $\tilde{x}_4(d)$ is either $\tilde{x}_4(d)$ or $\tilde{x}_5(d)$, it follows that for every $d \in R_4 = R_5$, $\tilde{x}_4(d) = \tilde{x}_4(d)$.

We show that for every $d \in \text{data}(f_4)$, $\tilde{x}_4(d) = \tilde{x}_4(d)$. If $d \in D \cup \mathcal{R}_i$, then $\tilde{x}_4(d) = \tilde{x}_4(d)$ by definition. If $d \in \mathcal{R}_i$, then $\tilde{x}_4(d) = \tilde{x}_4(d)$ as shown before. If $d \not\in D \cup \mathcal{R}_i$, then $\tilde{x}_4(d) = \tilde{x}_4(d)$. Notice that $\tilde{x}_4(d) \neq (\emptyset, \emptyset, \emptyset, \emptyset)$, since otherwise $d \not\in \text{data}(f_4)$. Then, $d \in \text{data}(f_3)$ and since $f_4 \succeq f_3$, $\tilde{x}_4(d) = \tilde{x}_4(d)$.

We finish the proof of $f_4 \preceq f_3$ by showing that for every $d \in \text{data}(f_4) \setminus R_3$ there is some $d' \in \text{data}(f_4) \setminus R_3$ so that $\tilde{x}_4(d) = \tilde{x}_4(d')$. If $d \not\in D \cup \mathcal{R}_i$, then we can just simply take $d' = d$ and $\tilde{x}_4(d) = \tilde{x}_4(d')$. If $d \not\in D \cup \mathcal{R}_i$, then $\tilde{x}_4(d) = \tilde{x}_4(d)$, notice that $\tilde{x}_4(d) \neq (\emptyset, \emptyset, \emptyset, \emptyset)$ because otherwise $d \not\in \text{data}(f_3)$ (since $d \not\in \mathcal{R}_3$). Hence, $d \in \text{data}(f_3)$ and since $f_4 \preceq f_3$, we have that $\tilde{x}_4(d) = \tilde{x}_4(d)$, hence $\tilde{x}_4(d) = \tilde{x}_4(d)$. We just showed that $f_4 \preceq f_3$. \qed
We define $f_3$ as follows

$$
R_3 = R_1 \\
\text{for every } d \in R_1, \chi(d) = \chi_1(d) \\
\text{for every } d \notin data(f_1), \chi_3(d) = \chi_1(d) \\
\text{for every strict external value } d \text{ of } f_1, \chi_3(d) = \chi_1(d).
$$

**Claim B.5.4.** $f_3 \preceq f_1$.

**Proof.** By definition we have that $R_1 = R_3$, and that $\chi_3(d) = \chi_1(d)$ for all $d \in R_1$. Also by definition, for every $d \in data(f_3)$ we have $\chi_3(d) = \chi_1(d)$.

We must show that for every $d \in data(f_1) \setminus (R_1 \cup data(f_3))$ there is some $d' \in data(f_1) \setminus R_1$ so that $\chi_1(d) = \chi_1(d')$. Take any such $d$. Note that $d$ must be necessarily an internal data value, since $f_3$ contains any strict external data value of $f_1$. Hence, let $\pi = \chi_1(d)$, where $\pi_{\bar{1}} \cup \pi_{\bar{2}} \neq \emptyset$. By definition of $D$, there must be some $d_\pi \in D$ so that $\chi_1(d_\pi) = \pi$ and $d_\pi \notin R_1$. Since $\pi$ is internal and $f_1 \in (a, d) f_2$, $d_\pi \in d \cup data(f_2)$. If $d_\pi \notin d$, we have, by definition of $f_3$, that $\chi_3(d_\pi) = \pi = \chi_1(d)$. Hence, there is such $d' \in data(f_1) \setminus R_1$ so that $\chi_1(d) = \chi_1(d')$. Otherwise, suppose $d_\pi \in data(f_2)$, which means, since $\pi_{\bar{1}} \cup \pi_{\bar{2}} \neq \emptyset$, that $\chi_3(d_\pi) \neq \emptyset, \emptyset, \emptyset$. By definition of $f_4$, $\chi_4(d_\pi) = \chi_2(d_\pi) \neq \emptyset, \emptyset, \emptyset$, thus $d_\pi \notin data(f_2)$. Then, by definition of $f_3$ we have that $\chi_3(d_\pi) = \pi = \chi_1(d)$. Hence, in this case we also have that there is such $d' \in data(f_1) \setminus R_1$ so that $\chi_1(d) = \chi_1(d')$. □

**Claim B.5.5.** $f_3 \in (a, d) f_4$.

**Proof.** Since $f_4 \preceq f_2$ and $f_1 \in (a, d) f_2$, we have that

- $\xi(f_1) \in \Gamma$ since $\xi(f_2) \in \Gamma$, by Lemma ??, and
- $\chi_3 = \chi_1 = \emptyset$ since $\chi_3 = \emptyset$.

Therefore, conditions [a] and [b] hold for $f_1$. We must show condition [c], that is, $\chi_3 = \chi_1 = \emptyset$.

Take any data value of $d \in d \cup data(f_1)$. We have that $\chi_3(d) = \chi_1(d)$ by definition. We also have that $\chi_1(d) = \chi_1(d) = \chi_1(d) = \chi_1(d)$. So we also have that $\chi_1(d) = \chi_1(d) = \chi_1(d) = \chi_1(d)$. Since $f_2$ and $f_4$ coincide in $d$, we also have that the set above is equal to $\{\alpha, \beta, i \in \Pi \mid \exists d'.(d, \alpha', \gamma, i) \in \chi_3 \uplus \chi_4, \alpha, \beta \in \sigma_\pi \cup \bigcup_{\gamma \in [k]}\{(d(i)) \times (\sigma_\pi \setminus \{e\}) \times \sigma_\pi \times \{i\})\}$. This, together with the fact that all internal values of $f_3$ are in $data(f_4) \cup d$, implies that $\chi_3 = \chi_1 = \chi_1 = \chi_1$.

This concludes the proof. □

**Appendix to Section 5.4**

**Lemma 5.5.** If there is a derivable root profile, then there is a derivation tree for a root profile so that all the profiles in the forest have no more than $2|\Pi|$ rigid values.

**Proof of Lemma 5.5.** Suppose we have $f = f_1 + \cdots + f_n$. Let us first show that for every $(\alpha, \beta, i) \in \Pi$ there can be at most one data value $d \in D$ so that there is some $j \in [n]$ with $\chi_j^f(\alpha, \beta, i) = \{d\}$. By means of contradiction, if there were two distinct data values $d, d'$ then there would be two $f_j, f_k$ with $j \neq k$ so that $\chi_j^f(\alpha, \beta, i) = \{d\}$ and $\chi_k^f(\alpha, \beta, i) = \{d'\}$. Suppose without any loss of generality that $j < k$. Then, by definition of $+$, we must have that—since $d \in \chi_j^f(\alpha, \beta, i) = \{d\}$ and $f_j + \cdots + f_k$ is defined—$d \in \chi_k^f(\alpha, \beta, i) = \{d'\}$. This is in contradiction with the fact that $\chi_k^f(\alpha, \beta, i) = \{d'\}$. The same happens by symmetry with $\chi_k^f$.

Let us define $R_{\chi_1, \ldots, \chi_n} = \bigcup_{\alpha, \beta, i \in \Pi \text{ and } j \in [n]} R_{\chi_j} \uplus \chi_j^{d'} \uplus \chi_j^{d}$. By the discussion before, it follows that $|R_{\chi_1, \ldots, \chi_n}| \leq 2|\Pi|$. Consider the profiles $f', f_1', \ldots, f_n'$ to be as $f, f_1, \ldots, f_n$, but with $R_{\chi_1, \ldots, \chi_n}$ as the set of rigid values. By construction of $R_{\chi_1, \ldots, \chi_n}$ we have that $f' = f_1' + \cdots + f_n'$. In other words we have the following.

**Claim B.5.6.** For every $(R, \chi) = (R, \chi_1) + \cdots + (R, \chi_n)$ we have that $(R, \chi_1, \ldots, \chi_n, \chi) = (R, \chi_1, \ldots, \chi_n, \chi_1) + \cdots + (R, \chi_1, \ldots, \chi_n, \chi_n).$
Using the above statement we can now prove the lemma. Let \( t \) be a derivation tree for a root profile \((R, f)\). Let \( t' \) be the derivation tree that results from replacing in \( t \) the profile labels \((R, \bar{x}_1), \ldots, (R, \bar{x}_n)\) of any maximal sequence of siblings with the labels

\[
(R_{\bar{x}_1}, \ldots, \bar{x}_1), \ldots, (R_{\bar{x}_n}, \ldots, \bar{x}_n).
\]

By the claim above, \( t' \) is a derivation tree for \((R, f)\). (Note that this is true because in the rooting operation there is no restriction on which should be the set of rigid values of the parent profile.) It is immediate that \((R, f)\) is a root profile if and only if \((R, f)\) is a root profile. Thus, the lemma follows.

**Appendix to Section 5.5**

Lemma 5.5 is the consequence of the following properties.

**Lemma B.6** (Figure ??). For every \( f, f', f'', f''' \in \mathcal{F}_b \) and \( (a, \bar{d}) \in \mathcal{X} \times \mathcal{D}^k \) so that \( f \preceq f'' \preceq f' \) and \( f_1 \in (a, \bar{d})f' \), there are \( f''' \in \mathcal{F}_b \) so that \( f'' \preceq f''' \preceq f' \).

**Proof of Lemma ??**. Without any loss of generality, let us assume that \( f \preceq f' \) and \( f \preceq f'' \) so that \( data(f'') \cap data(f') = data(f) \). We define \( f''' \) so that \( R''' = R \) and

\[
\bar{x}''(d) = \begin{cases} \bar{x}'(d) & \text{if } d \in data(f'), \\ \bar{x}(d) & \text{else, if } d \in data(f''). \end{cases}
\]

The following statement follows straight from this definition.

**Claim B.6.1.** \( f'' \preceq f''' \preceq f' \).

**Proof.** By definition of \( f \preceq f'' \) and \( f \preceq f' \) for every data value \( d \in data(f) \cup data(f') \) there is a data value \( f(d) \in data(f') \) so that \( \bar{x}(d) = \bar{x}'(d) = \bar{x}''(d) \) and \( d \in R \) if \( f(d) \in R \). Then,

- \( f''' = f'[f(d_1) \rightarrow f(d_1), d_1] \cdots [f(d_n) \rightarrow f(d_n), d_n] \) for \( \{d_1, \ldots, d_n\} = data(f') \setminus data(f') \), and
- \( f'' = f'[f(d_1) \rightarrow f(d_1), d_1] \cdots [f(d_n) \rightarrow f(d_n), d_n] \) for \( \{d_1, \ldots, d_n\} = data(f') \setminus data(f') \).

Thus, \( f'' \preceq f''' \preceq f' \).

Remember that by definition of \( f \preceq f'' \), for every data value \( d \in data(f''') \) there must be some data value \( f(d) \in data(f') \) so that \( \bar{x}''(d) = \bar{x}'(d) \). We can further assume that for every \( d \in data(f''') \), \( d \in R'_1 \) if \( f(d) \in R_1 \) (i.e., we can simply define \( f(d) = d \) for all \( d \in R'_1 \cap data(f''') \)). We define \( f_1 \) as \( R_i'' = R_i \) and

\[
\bar{x}_i'''(d) = \begin{cases} \bar{x}_i'(d) & \text{if } d \in data(f_1), \\ \bar{x}_i(d) & \text{else, if } d \in data(f'''). \end{cases}
\]

**Claim B.6.2.** \( f_1'' \in (a, \bar{d})f'''' \).

**Proof.** For every data value of \( d \in data(f_1) \) we have that \( \bar{x}_1''(d) \) and \( \bar{x}_1'''(d) \) are equal to \( \bar{x}_1'(d) \) and \( \bar{x}_1(d) \), and it therefore verifies the conditions imposed by the rooting operation. Further, any other data value of \( data(f_1) \) (namely any data value from \( data(f''') \setminus data(f_1) \)) behaves as some data value \( f(d) \) in \( f_1 \). That is \( \bar{x}_1''(d) \) and \( \bar{x}_1'''(d) \) are equal to \( \bar{x}_1'(f(d)) \) and \( \bar{x}_1(f(d)) \). It then follows that the data value must verify the conditions imposed by the rooting operation. Finally, by Lemma ??, \( \xi(f_1') \in \Gamma \) and \( f_1' \preceq f_1'' \), we have that \( \xi(f_1') \in \Gamma \). Hence, \( f_1'' \in (a, \bar{d})f'''' \). By definition of \( f_1'' \), it also follows that \( f_1'' \preceq f_1''' \).

**Claim B.6.3.** \( f_1''' \preceq f_1'' \).

**Proof.** It is immediate from the definition that

\[
f_1''' = f_1'[f(\bar{d}_1) \rightarrow f(\bar{d}_1), \bar{d}_1, \ldots, f(\bar{d}_n) \rightarrow f(\bar{d}_n), \bar{d}_n],
\]

where \( \{\bar{d}_1, \ldots, \bar{d}_n\} = data(f''') \setminus data(f_1) \). By definition of \( f, f(\bar{d}_1), \ldots, f(\bar{d}_n) \in data(f') \setminus R'_1 \), and hence \( f_1''' \preceq f_1'' \).
This concludes the proof. □

**Lemma B.7.** $R_{up}(\uparrow\downarrow\mathcal{D}_b) \subseteq \downarrow\mathcal{D}_b$.

**Proof of Lemma B.6.** Let $f'_1 \in R_{sp}(\uparrow\downarrow\mathcal{D}_b)$. This means there is some $f'' \in \mathcal{D}_b$ and $f$ so that $f \preceq f''$, $f \succeq f'$, and $f'_1 \in (a, d)\ f''$ for some $(a, d) \in \mathcal{A} \times \mathcal{D}_a$. We can then apply Lemma B.6., obtaining that there is some $f'''$ and $f''_1$ so that $f' \preceq f'''$, $f' \succeq f''_1$, and $f''_1 \in (a, d)\ f'''$. Since $\mathcal{D}_b$ is upward-closed by Remark B.5., $f''' \in \mathcal{D}_b$, and therefore $f''_1 \in \mathcal{D}_b$ as well since $f''_1 \in (a, d)\ f'''$. Thus, as $f''_1 \succeq f'_1$, we obtain $f'_1 \in \downarrow\mathcal{D}_b$. □

**Lemma B.8.** (Figure B.7). For every $i \in \{1, 2\}$ and $f_i, f''_i, f'''_i \in \mathfrak{F}$ so that $f_i \preceq f''_i, f_i \succeq f'''_i$ and $f'_i + f''_i$ is defined, there are $f'''_1, f'''_2 \in \mathfrak{F}$ so that $f'_1 + f''_1 \preceq f'''_1 + f'''_2$ and $f'_2 \preceq f'''_1, f'_2 \preceq f'''_2$ for every $i \in \{1, 2\}$.

**Proof of Lemma B.8.** Without any loss of generality, assume that $f_i \preceq f''_i, f_i \succeq f'''_i$ and $\text{data}(f''_i) \cap \text{data}(f'''_i) = \text{data}(f_i)$ for every $i \in \{1, 2\}$. Remember that by definition of $f''_i$, for every data value $d \in \text{data}(f''_i)$ there must be some data value $f(d) \in \text{data}(f_i)$ so that $\chi_i(d) = \chi_i'(d) = \chi_i''(d)$. We then define $f''_i$ for every $i \in \{1, 2\}$ so that $R'_i = R_i$ and

$$
\chi_i''(d) = \begin{cases} 
\chi_i(d) & \text{if } d \in \text{data}(f'_i), \\
\chi_i'(d) & \text{else, if } d \in \text{data}(f''_i), \\
\chi_i''(d) & \text{else, if } d \in \text{data}(f'''_i), \\
(\emptyset, \emptyset, \emptyset, \emptyset) & \text{otherwise.}
\end{cases}
$$

In the definition above, notice that for every $d \in \text{data}(f''_{3-i})$ we have that $f(d) \in \text{data}(f_{3-i})$ and hence $f(d) \in \text{data}(f_{3-i})$ which, by definition of $f'_1 + f''_2$, means that $f(d) \in \text{data}(f_i)$.

The following claim follows straight from the definition just given.

**Claim B.8.1.** $f'_i \preceq f'''_i, f'_i \preceq f'''_i$ for all $i \in \{1, 2\}$.

**Proof of B.8.1.** Since $f_i \preceq f''_i$ and $f_i \preceq f'''_i$, there is some $\hat{f}$ so that for every $d \in \text{data}(f'_i), \hat{f}(d) \in \text{data}(f_i)$ so that $\chi_i''(d) = \chi_i(\hat{f}(d)) = \chi_i''(\hat{f}(d)) = \chi_i''(f(d))$ and $d \in R_i$ iff $\hat{f}(d) \in R_i$.

For every $i \in \{1, 2\}$, it is immediate from the definition of $f''_i$ that

$$f''_i = [f'(d_1) \Rightarrow f'(d_1), \ldots, f'(d_n) \Rightarrow f'(d_n)] \ [f'(d_1) \Rightarrow f'(d_1), \ldots, f'(d_m) \Rightarrow f'(d_m)]$$

where $\{d_1, \ldots, d_n\} = \text{data}(f''_{3-i})$ and $\{d_1, \ldots, d_n\} = \text{data}(f''_{3-i}) \setminus \text{data}(f'_i)$.

Since $f_i \preceq f''_i$ and $f_i \preceq f'''_i$, there is some $\hat{f}$ so that for every $d \in \text{data}(f_i), \hat{f}(d) \in \text{data}(f_i)$ so that $\chi_i''(d) = \chi_i(\hat{f}(d)) = \chi_i''(\hat{f}(d)) = \chi_i''(f(d))$ and $d \in R_i$ iff $\hat{f}(d) \in R_i$.

For every $i \in \{1, 2\}$, it is immediate from the definition of $f''_i$ that

$$f''_i = [f'(d_1) \Rightarrow f'(d_1), \ldots, f'(d_n) \Rightarrow f'(d_n)] \ [f'(d_1) \Rightarrow f'(d_1), \ldots, f'(d_m) \Rightarrow f'(d_m)]$$

where $\{d_1, \ldots, d_n\} = \text{data}(f''_{3-i})$ and $\{d_1, \ldots, d_n\} = \text{data}(f'_i)$. □

We are then left with the following easy claim.

**Claim B.8.2.** $f'_i + f''_i \preceq f'''_i + f'''_i$.

**Proof of B.8.2.** For every data value $d \in \text{data}(f'_i) \cup \text{data}(f''_i)$ it is easy to see that the conditions of $+$ apply for $f''_i$ and $f''_i$ since they have the same description for $d$. For any other data value $d \in \text{data}(f''_i) \cup \text{data}(f''_i)$ we have that $f'''_i$ and $f''_i$ behave just as $f'_i$ and $f''_i$ for the data value $f(d)$. Therefore, the conditions of $+$ hold, and $f''_i + f'''_i$ is well-defined. Moreover, $f'_i + f''_i \preceq f''_i + f''_i$ since, by definition of $f''_i, f'''_i$ we have

$$f''_i + f''_i = (f'_i + f''_i)[f'(d_1) \Rightarrow f'(d_1), \ldots, f'(d_n) \Rightarrow f'(d_n), f'(d_n), f'(d_m)]$$

for $\{d_1, \ldots, d_n\} = \text{data}(f''_i + f''_i) \setminus \text{data}(f''_i + f''_i)$. □

This concludes the proof of the lemma.

**Lemma B.9.** $R_i (\uparrow\downarrow\mathcal{D}_b) \subseteq \downarrow\mathcal{D}_b$. □
Therefore, 

\[ \text{Thanks to the property } R(\downarrow D_R) \subseteq \downarrow D_R \text{ one can show that } C_n = \min(\downarrow D_R) \text{ and that } C_{n+1} \text{ can be computed from } C_n \text{ in } 2\text{EXPSPACE in } |C_n| \text{ and } |P|. \text{ It is possible to test from the set } \min(\downarrow D_R) \text{ whether there is a root derivable profile in } D_R, \text{ hence obtaining that the derivation problem is decidable.} \]

**Lemma 5.11** \( C_n = \min(\downarrow D_R). \)

**Proof of Lemma 5.11** By Lemma 5.9 we know that \( R(\downarrow D_R) \subseteq \downarrow D_R. \) Therefore, since \( C_0 \subseteq \min(\downarrow D_R), \) we have that for every \( i, C_i \subseteq \min(\downarrow D_R). \) We then need to show that \( \min(\downarrow D_R) \subseteq C_{n+1}. \)

Suppose, by means of contradiction, that there is some \( D \in 2\text{ExpSpace} \) for some bounded extension \( \min \) such (doubly exponentially many) bounded extensions of elements of \( \mathcal{C} \) represent all the possible profiles of the rigid values. We must also remember which profiles have either \( f \) since \( \min \), hence obtaining that the derivation problem is decidable.

**Lemma 5.10.** \( k_0 \) is bounded by a doubly exponential function on \( |P|. \)

**Proof of Lemma 5.10**. Remember that \( \mathfrak{D}_k \) is the set of profiles that have less than \( 2|P| \) rigid values. Then,

\[ (2^n \times 2^n \times 2^n \times 2^n)^{2|P|} \]

represent all the possible profiles of the rigid values. We must also remember which profiles have either 0, or 1 or more flexible values, and this information can be represented with an element of \( 2^{2^n \times 2^n \times 2^n \times 2^n}. \)

Therefore, there are doubly exponentially many elements in \( \min(\mathfrak{D}_k)/\sim, \) and hence \( k_0 \) is bounded by a doubly exponential function in \( |P|. \) Since \( |P| \) is polynomial in \( |P|, \) the statement follows.

**Lemma 5.11.** Given \( G \subseteq \mathfrak{D}_k \) and \( f \in \mathfrak{D}_k, \) testing \( f \in \min(\downarrow R(\downarrow G)) \) is computable in \( 2\text{ExpSpace} \) in \( |G| \) and \( |P|. \)

**Proof of Lemma 5.11**. First we check that \( f \in \min(\downarrow \mathfrak{D}_k). \)

To check \( f \in \min(\downarrow R_+(\downarrow G)) \), we verify if \( f \mid \preceq f_1 + f_2 \) in \( 2\text{ExpSpace}, \) where \( f_1, f_2 \) is a pair of the (doubly exponentially many) bounded extensions of elements of \( \min(\downarrow G). \) By Lemma 5.3 there are such \( f_1, f_2 \) if, and only if, \( f \in \min(\downarrow R_+(\downarrow G)). \)

Finally, to check \( f \in \min(\downarrow R_+(\downarrow G)) \), we verify that \( f \mid \preceq f_1 \in \min(\downarrow (a, d)f_2) \) in \( 2\text{ExpSpace}, \) where \( f_1 \) is a bounded extension of a profile of \( \min(\downarrow G). \) If the condition holds of course \( f \in \min(\downarrow R_+(\downarrow G)). \) On the other hand, by Lemma 5.4 sum Remark 5.6 if \( f \in (a, d)f_2 \) with \( f_2 \preceq f_1, \) then \( f \preceq f_3 \in \min(\downarrow (a, d)f_2) \) for some bounded extension \( f_3 \) of \( f_2, \) hence the condition holds.

By the lemma above, we immediately obtain the following.

**Lemma 5.12.** \( C_{n+1} \) can be computed from \( C_n \) in \( 2\text{ExpSpace} \) in \( |C_n| \) and \( |P|. \)

**Proof of Lemma 5.12**. For each \( f \in \min(\mathfrak{D}_k)/\sim \) we can check, in \( 2\text{ExpSpace}, \) whether \( f \in \min(\downarrow R(\downarrow C_i)) \) thanks to Lemma 5.7. We can therefore compute \( \min(\downarrow R(\downarrow C_i))/\sim \) in \( 2\text{ExpSpace}, \) and thus we can also compute \( C_{n+1} \) in \( 2\text{ExpSpace}. \)

**Proposition 5.12** The derivation problem is decidable in \( 2\text{ExpSpace}. \)

**Proof of Proposition 5.12**. We can compute all \( C_0, \ldots, C_{k_0} \) in \( 2\text{ExpSpace} \) by Lemmas 5.9 and 5.11. Since \( C_{k_0} = \min(\downarrow D_R) \) by Lemma 5.11 it follows that \( \min(\downarrow D_R) \) is hence computable in \( 2\text{ExpSpace}. \)

In order to test if there is a derivable profile in \( D_R \) we choose some \( f \in \min(\{f \in \mathfrak{D}_k \mid \xi(f) \in \Gamma \}) \) with no external values (i.e., so that \( \chi^1 = \chi^0 = 0, \) \( f \in \min(\downarrow D_R) \) and \( (a, d) \in \mathcal{A}_\text{root} \times \mathbb{D}^k, \) and test, in \( 2\text{ExpSpace} \) (Lemma 5.12), whether \( f \in \min(\downarrow R_+(\downarrow \{f \})) \).

**Claim 5.12.1.** The following statements are equivalent.
1. There is some \( f \in \text{MIN}(\{ f \in \mathcal{F} \mid \xi(f) \in \Gamma \}) \) with no external values so that
   - \( f \in \text{MIN}(\downarrow \mathcal{R}^{(a,d)}_\mathcal{A}(\downarrow \{f\})) \),
   - \( (a,d) \in \mathcal{A}_\mathcal{R} \times \mathcal{D}^k \) and
   - \( f' \in \text{MIN}(\downarrow \mathcal{D}_\mathcal{D}) \).

2. There is a derivable root profile in \( \mathcal{D}_\mathcal{D} \).

Proof. (\( \Rightarrow \)) Assume first that condition \( \Rightarrow \) holds. We change the names for the profiles to make the explanation clear. Suppose that there is some \( f_1 \in \text{MIN}(\downarrow \mathcal{R}^{(a,d)}_\mathcal{A}(\downarrow \{f\})) \) with no external data values, for some \( (a,d) \in \mathcal{A}_\mathcal{R} \times \mathcal{D}^k \), \( f_1 \in \text{MIN}(\{ f \in \mathcal{F}_b \mid \xi(f) \in \Gamma \}) \), and \( f \in \text{MIN}(\downarrow \mathcal{D}_\mathcal{D}) \). This means that there must be some \( f'' \in \mathcal{D}_\mathcal{D} \) so that \( f \preceq f'' \) and some \( f' \) so that \( f \preceq f' \) and \( f' \in (a,d)f' \). We can then apply Lemma \( \Rightarrow \) obtaining that there is some \( f''' \) and \( f''_1 \) so that \( f \preceq f''' \), \( f' \preceq f''' \), \( f_1 \preceq f''' \), and \( f_1 \in (a,d)f''' \). Since \( \mathcal{D}_\mathcal{D} \) is upward closed (Remark \( [5.8] \)), \( f'' \in \mathcal{D}_\mathcal{D} \) and hence \( f'' \in \mathcal{D}_\mathcal{D} \). By definition of \( \preceq \), it follows that, since \( f_1 \) has no external data values, \( f_1 \) has no external data values either. Thus, there is a derivable root profile, namely \( f_1'' \).

(\( \Leftarrow \))\( \Rightarrow \) Suppose, on the other hand, that condition \( \Rightarrow \) holds. If there is a derivable root profile in \( \mathcal{D}_\mathcal{D} \), then there are \( f, f' \in \mathcal{D}_\mathcal{D} \) and \( (a,d) \in \mathcal{A}_\mathcal{R} \times \mathcal{D}^k \) so that \( \chi = \chi = \emptyset \) and \( f \in (a,d)f' \). Let \( f_1 \in \text{MIN}(\{ f \})/\sim \), that is, \( f_1 \) is a minimal element corresponding to \( f \). Also, let \( f_1 \in \text{MIN}(\{ f \})/\sim \), that is, \( f_1 \) is a minimal element corresponding to \( f \). Of course, it follows that \( f_1 \in \text{MIN}(\downarrow \mathcal{R}^{(a,d)}_\mathcal{A}(\downarrow \{f\})) \).

Lemma \( [5.7] \) holds. Thus, there is a derivable root profile in \( \mathcal{D}_\mathcal{D} \) if and only if there is a derivable root profile in \( \mathcal{D} \) by Lemma \( [5.7] \) the proposition follows.

C. APPENDIX TO SECTION 6

Appendix to Section 6.1

Lemma 6.1 (Direct Normal Form). There exists an exponential time translation that for every node expression \( \varphi \in \text{XPath}^{(\text{eq}, \text{eq}, \rightarrow, =)} \) returns an equivalent node expression \( \psi \) in direct normal form.

Proof of Lemma 6.1. The idea is that every data test expression \( (\alpha \& \beta \& \delta) \) is translated into a big disjunction of expressions in direct normal form, where loops in the tree navigation of \( \alpha, \beta \) are factored as node expressions, as done in \( \mathcal{F} \).

For any finite alphabet \( \mathcal{A} \), we define \( \text{tree order morphisms} \) between forests over a powerset alphabet \( \mathcal{P} \). Given two forests \( t, t' \) over \( \mathcal{P} \), a tree order morphism from \( t \) to \( t' \) is a function \( f \) from the nodes of \( t \) to the nodes of \( t' \) so that
   - for every node \( x \) of \( t \), the label of \( x \) in \( t' \) is a subset of the label of \( f(x) \) in \( t' \), and
   - for every two nodes \( x, y \) of \( t \), if \( (x, y) \) is in the reflexive-transitive closure of the next-sibling relation (resp. of the child relation) in \( t \), then \( f(x), f(y) \) is also in the reflexive-transitive closure of the next-sibling relation (resp. of the child relation) in \( t' \).

We say that \( t', (x', y') \) is a contraction of \( t, (x, y) \) if there is a tree order morphism \( f \) from \( t \) to \( t' \) so that \( f(x) = x' \) and \( f(y) = y' \). We also say that \( t, (x, y) \) is an expansion of \( t', (x', y') \). Notice that the function need not be surjective, and hence the fact that one forest is a contraction of another does not have any implication on the sizes of the forests: a forest may have less or more nodes than its contraction. For example, both the tree of Figure 7-2 and the forest of Figure 7-3 are contractions of the forest of Figure 7-3.

Let \( \alpha \) be a path expression, and let \( ne(\alpha) \) be the set of all node expressions of \( \alpha \). Consider \( t^{\alpha} \) the tree over the alphabet \( \mathcal{P}^{|ne(\alpha)|} \), where every node is labeled by those \( \psi \in ne(\alpha) \) that are true at the node. For example, for the multi-attribute data tree \( t \) defined in Figure 7-4 and \( \alpha \) as defined in Figure 7-4, the tree \( t^{\alpha} \) is the one of Figure 7-5.

Let \( t \) be any forest over \( \mathcal{P}^{|ne(\alpha)|} \), and let \( x, y \) be two nodes of \( t \), so that \( x \) is a root. For such a forest and nodes, one can build a direct path expression \( \mu_{t, x, y}^{\alpha} \) that tests if there is a path between \( x \) and \( y \) satisfying the relative order of nodes satisfying the labels as in the tree. For example, for the forest \( t \) of Figure 7-3 and the nodes \( x, y \), depicted, the corresponding expression \( \mu_{t, x, y}^{\alpha} \) would be the one appearing in Figure 7-3.

Claim C.0.1. A direct path expression \( \mu_{t, x, y}^{\alpha} \) can be built from \( t \) in polynomial time, such that for every multi-attribute data tree \( t \) we have that \( t, (x', y') \models \mu_{t, x, y}^{\alpha} \) if and only if \( t^{\alpha}, (x', y') \) is a contraction of \( t, (x, y) \).

Proof. Given a forest \( t \) and nodes \( x, y \), so that \( x \) is a root, one can build \( \mu_{t, x, y}^{\alpha} \) iteratively. We first build a path expression that starts in \( x \) and ends in \( y \) and checks all the labels between \( x \) and \( y \) in the unique path between \( x \) and \( y \) that corresponds to a direct navigation. For example if \( t, x, y \) are as in Figure 7-7, we build a path as in Figure 7-7 that tests all the node expressions of the nodes
the path. We then build a more complex path expression nesting expressions that test the existence of paths as depicted in Figure ??-c. We iterate until we have covered the whole tree $\tilde{t}$, at each iteration, we add tests for paths that are at a deeper nesting degree in the resulting direct path expression. For example, the paths of Figure ??-c are at depth 1, the paths of Figure ??-d at depth 2, and the path of Figure ??-e at depth 3. Note that if we apply this construction to the forest of Figure ??-c and the path $\alpha$ of Figure ??-e, we obtain the expression $\mu_{t,x,y}^\alpha$ of Figure ??-e. \hfill \Box

Given a multi-attribute data tree $t$ so that $t, (x, y) \models \alpha$, we call a witness forest of $\alpha$ for $t, (x, y)$ to a forest $\tilde{t}$ over the alphabet $2^{\text{ax} \alpha}$ together with some nodes $x, y$, so that $\tilde{t}$ contains only the nodes involved in the satisfaction of $\alpha$, and they are labeled with the node expressions that they must verify. For example, given the multi-attribute data tree $t$ of Figure ??-a, and $\alpha$ as defined in Figure ??-e, a possible witness forest of $\alpha$ for $t, (x, y)$ is depicted in Figure ??-c. Of course, there may be several witness forests of $\alpha$ for a given $t, (x, y)$. Let $|\alpha|$ be the size of $\alpha$, computed as the number of axes in $\alpha$, irrespective of the size of node expressions. For example the path expression $\alpha$ defined in Figure ??-e is so that $|\alpha| = 7$. Notice that the number of nodes of any witness forest of $\alpha$ for $t, (x, y)$ is bounded by $p(|\alpha|)$ for some polynomial $p( )$.

Given a path expression $\alpha$, let $\tilde{\alpha}$ be the path expression over the alphabet $2^{\text{ax} \alpha}$, where every node expressions $\psi$ is replaced by $\bigvee_{\delta \subseteq \text{ax} \alpha, \psi \in S} S$, that is, it is treated as a disjunction of tests for labels from the alphabet. Notice that any witness forest $\tilde{t}, (x', y')$ of $\alpha$ for $t, (x, y)$ is an expansion of $\tilde{t}^\alpha, (x, y)$, and we have $\tilde{t}, (x', y') \models \tilde{\alpha}$. We then have the following.

---

Figure 11: Example for the proof of Lemma 6.1.

Figure 12: Idea of construction of $\mu_{t,x,y}^\alpha$. 
CLAIM C.0.2. \( t, (x, y) \models \alpha \) if, and only if, \( \ell, (x', y') \models \tilde{\alpha} \) for some expansion \( \ell, (x', y') \) of \( \tilde{t}\sigma, (x, y) \).

For every contraction \( \ell, (x, y) \) of \( \tilde{t}\sigma, (x', y') \), if \( \ell, (x', y') \models \tilde{\alpha} \) then \( \tilde{p}, (x', y') \models \tilde{\alpha} \). This observation, together with the previous claim and the fact that every witness forest \( \ell, (x', y') \) of \( \alpha \) for \( t, (x, y) \) is bounded by \( p(||\alpha||) \) yields the following.

CLAIM C.0.3. There is an expansion \( \ell, (x', y') \) of \( \tilde{t}\sigma, (x, y) \) of size at most \( p(||\alpha||) \) so that \( \tilde{\alpha} \) if, and only if, \( t, (x, y) \models \alpha \).

For every forest \( \tilde{t} \) over \( 2^{ne(\alpha)} \) of size \( \leq p(||\alpha||) \) and nodes \( x, y \) so that \( \tilde{t}, (x, y) \models \tilde{\alpha} \), we say that \( \mu_{\tilde{t}, x, y}^\alpha \) is a direct normal form linearization of \( \alpha \). It follows that \( t, (x, y) \models \mu_{\tilde{t}, x, y}^\alpha \) if, and only if, a contraction of \( \tilde{t}, (x', y') \) is a witness forest of \( \alpha \) for \( t, (x, y) \). Thus, the following claim follows.

CLAIM C.0.4. \( t, (x, y) \models \alpha \) if, and only if, \( t, (x, y) \models \mu_{\tilde{t}, x, y}^\alpha \) for some direct normal form linearization \( \mu_{\tilde{t}, x, y}^\alpha \) of \( \alpha \).

**Proof.** If \( t, (x, y) \models \alpha \), then by Claim ?? there is a forest \( \tilde{t}, (x', y') \) over \( 2^{ne(\alpha)} \) (the witness forest) of size \( \leq p(||\alpha||) \) so that \( \tilde{t}, (x', y') \models \tilde{\alpha} \). Hence, \( \mu_{\tilde{t}, x, y}^\alpha \) is a direct normal form linearization of \( \alpha \). Since \( \tilde{t}, (x', y') \) is an expansion of \( \tilde{t}\sigma, (x, y) \) and \( \tilde{t}, (x', y') \models \tilde{\alpha} \), then \( t, (x, y) \models \mu_{\tilde{t}, x, y}^\alpha \).

Suppose now that \( t, (x, y) \models \mu_{\tilde{t}, x, y}^\alpha \) for some \( \tilde{t}, (x', y') \) over \( 2^{ne(\alpha)} \) of size \( \leq p(||\alpha||) \) so that \( \tilde{t}, (x', y') \models \tilde{\alpha} \). Then, \( \tilde{t}, (x', y') \) must be an expansion of \( \tilde{t}\sigma, (x, y) \) by Claim ?? Since \( \tilde{t}, (x', y') \models \tilde{\alpha} \), we then have that \( t, (x, y) \models \alpha \) by Claim ??.

Note that there are exponentially many trees in \( ||\alpha|| \) over \( ne(\alpha) \) of size \( \leq p(||\alpha||) \). Hence, all the direct normal form linearizations of a given path expression \( \alpha \) are computable in exponential time \( ||\alpha|| \). Hence, given \( \varphi \in X\text{Path}(\{\epsilon, \downarrow, \rightarrow, \ast\}) \), consider \( \varphi' \) as the result of replacing every appearance of \( \langle \alpha\sigma_1 = \beta\sigma_2 \rangle \) (resp. \( \neq \)) by a disjunction of \( \mu_{\tilde{t}, x, y}^{\alpha_1, \sigma_1} \) and \( \mu_{\tilde{t}, x, y}^{\beta_1, \sigma_2} \) (resp. \( \neq \)) for every direct normal form linearization \( \mu_{\tilde{t}, x, y}^{\alpha_i, \sigma_i} \) of \( \alpha \) and \( \mu_{\tilde{t}, x, y}^{\beta_i, \sigma_i} \). Since these are all the possible linearizations, the translated formula is satisfied in the same multi-attribute data trees and nodes as the original formula. Further, \( \varphi' \) is in direct normal form. Since every replacement is exponential in \( ||\alpha|| \), and since \( ||\alpha|| \) does not depend on the size of sub-node expressions, it follows that \( ||\varphi'|| \) in time exponential in \( ||\varphi|| \). Thus, the lemma follows.

**Lemma 6.2 (Unnested Normal Form).** There exists an exponential time translation that for every formula \( \eta \in X\text{Path}(\{\epsilon, \downarrow, \rightarrow, \ast\}) \) returns a formula \( \varphi \) in unnested normal form such that \( \eta \) is satisfiable iff \( \varphi \) is satisfiable. Further, the translation of a formula in direct normal form is in direct normal form.

**Proof of Lemma 6.2:** Given a formula \( \eta \) we define the alphabet \( A_\varphi \) of the translation \( \varphi \) as all the locally consistent sets of subformulas of \( \eta \). That is, the sets \( S \) such that for every subformula \( \psi \) of \( \eta \): (1) if \( \psi = \neg \psi' \) then \( \{\psi', \neg \psi'\} \notin S \); (2) if \( \psi = \psi' \land \psi'' \) then \( \psi \in S \) iff \( \{\psi', \psi''\} \subseteq S \); and (3) if \( \psi = \psi' \lor \psi'' \) then \( \psi \in S \) iff \( \psi' \in S \lor \psi'' \). Given a formula \( \psi \), \( tr(\psi) \) denotes the result of replacing

- every instance of a path expression \( o_1[\psi_1]o_2\ldots o_n[\psi_n] \) in \( \psi \) (where \( o_1, \ldots, o_n \in \{\epsilon, \downarrow, \rightarrow, \ast\} \) which does not appear nested inside another path expression, with the expression
  \[ o_1[\zeta_{\psi_1}]o_2\ldots o_n[\zeta_{\psi_n}] \]
  and

- every test for label \( a \) which does not appear inside a path expression by \( \zeta_a \), where \( \zeta_a \equiv \bigvee_{S \in A_\varphi, \psi \in S} S \).

To build the formula \( \varphi = \varphi_1 \land \varphi_2 \) in normal form, we define \( \varphi_1 = \zeta_\psi \), and we build \( \varphi_2 \) as a conjunction of formulas

\[ \neg \langle \downarrow_s[\zeta_{\psi} \land \neg tr(\psi)] \rangle \]

for all subformulas \( \psi \) of \( \eta \). It is easy to see that this translation preserves satisfiability.

**Corollary 6.3** About the translation of Lemma 6.2.

1. The set of path subformulas resulting from the translation has cardinality polynomial in \( \eta \).
2. Every path subformula resulting from the translation can be written using polynomial space.

**Proof of Corollary 6.3:** The blowup in the exponential translation comes only from the formulas \( \zeta_\varphi \). In fact, \( \varphi \) can be symbolically written in polynomial space just as we did, using a symbol \( \zeta_\varphi \) instead of a big exponential disjunction. Remark that testing whether a label \( S \in A_\varphi \) satisfies \( \zeta_\psi \) reduces to testing \( \psi \in S \).

**Appendix to Section 6.2**

**Lemma 6.4** Given a direct non-recursive formula \( \psi \) that is a boolean combination of subformulas of \( \phi \), and two forest profiles \( \hat{f}, \hat{f}' \subset \hat{F} \) so that \( \xi(\hat{f}) = \xi(\hat{f}') \) then \( \hat{f} \models \psi \) if, and only if, \( \hat{f}' \models \psi \).

**Proof.** If \( \psi \in S(\hat{A}) \), then it is immediate that \( \hat{f} \models \psi \) iff \( \xi(\hat{f})((\psi, e, i, o) \neq 0 \forall i \in [k]) \). Suppose now that \( \psi = \langle \alpha \beta \gamma \delta \rangle \neq 0 \) where \( \alpha \) is leftward, \( e \) or empty and \( \gamma \) is rightward, \( e \) or empty, and \( \beta, \delta \) are downward or empty. We show that whether \( \hat{f} \models \psi \) depends only on \( \xi(\hat{f}) \). We show
one of the possible cases in the definition of $\vdash$, while the remaining ones are analogous or simpler. Suppose, for example, that $\alpha \neq \epsilon$, $\beta \neq \epsilon$, $\gamma = \epsilon$. We then have that $\Gamma \vdash \psi$ if:

- there is at least one data value reachable with $\alpha \beta \ominus$, that is, $\xi(f)(\bar{\alpha}, \bar{\gamma}) \neq 0$, with $\bar{\alpha} = (P_{\epsilon}^{-1}(\alpha), P_{\epsilon}^{-1}(\beta), i)$, and
- there is at least one data value reachable with $\gamma \delta \ominus$, that is, $\xi(f)(\bar{\beta}, \bar{\gamma}) \neq 0$, with $\bar{\beta} = (\top, P_{\epsilon}^{-1}(\delta), j)$, and either
  - there are two data values reachable with $\alpha \beta \ominus$, that is, $\xi(f)(\bar{\alpha}, \bar{\gamma}) = 2+$, or
  - there are two data values reachable with $\gamma \delta \ominus$, that is, $\xi(f)(\bar{\beta}, \bar{\gamma}) = 2+$, or
  - there is one data value reachable with $\alpha \beta \ominus$, and only one data value reachable with $\gamma \delta \ominus$, and they are different, that is, $\xi(f)(\bar{\alpha}, \bar{\gamma}) = 1$, $\xi(f)(\bar{\beta}, \bar{\gamma}) = 1$, and $\xi(f)(\bar{\alpha}, \bar{\beta}, \bar{\gamma}) = 0$.

These are indeed the necessary and sufficient conditions for the existence of two data values $d \neq d'$ so that $d \in \chi_\alpha(\bar{a})$ and $d' \in \chi_\beta(\bar{\beta})$, where $a = \bar{\alpha}$, $d' = \bar{\alpha}$'. Hence, these are the necessary and sufficient conditions to verify $\Gamma \vdash \psi$.

The remaining cases are analogous or easier.

**Lemma 6.5** $\Gamma \vdash \bigwedge_{a \in A} (a \Rightarrow \gamma_\phi(a))$ if, and only if, $\xi(f) \in \Gamma_\phi$.

**Proof of Lemma 6.5.** This is a direct consequence of Lemma 6.4 and the definition of $\Gamma_\phi$. 

**Abstractions.**

We now define $\text{abs}(\bar{t}, \bar{t}, \bar{r})$ for any forests $\bar{t}, \bar{t}, \bar{r}$. Let $t = a_r \otimes d_r, t = a_d \otimes d, t = a_d \otimes d_r$ be the multi-attributes data trees $(a, d)\bar{t}$, $(a, d)\bar{r}$, and $(a, d)\bar{r}$, respectively, for some fixed arbitrary $(a, d)$. We then define $\text{abs}(\bar{t}, \bar{t}, \bar{r}) = \bar{\chi}$ where:

\[
\bar{\chi} = \{(d(y)(i), \alpha, \beta, i) | (1, y) \in \text{data}(\bar{t}) \},
\]

for $(\alpha, \beta, i) \in \Pi$ and $y \in \text{pos}(\bar{t})$.

\[
\bar{\chi} = \{(d(y)(i), \alpha, \beta, i) | (2, y) \in \text{data}(\bar{t}) \},
\]

for $n = \bar{t}$, $(\alpha, \beta, i) \in \Pi$ and $y \in \text{pos}(\bar{t})$.

\[
\bar{\chi} = \{(d(y)(i), \alpha, \beta, i) | (3, y) \in \text{data}(\bar{t}) \},
\]

for $n = \bar{t}$, $(\alpha, \beta, i) \in \Pi$ and $y \in \text{pos}(\bar{t})$.

For any $R \subseteq \mathbb{D}$ we define $\text{abs}(\bar{t}, \bar{t}, \bar{r}) = \bar{\chi}$.

**Remark C.1.** Note that $\text{abs}(\bar{t}, \bar{t}, \bar{r}) = \bar{\chi}$.

We show that $\text{abs}$ is basically an algebra morphism between multi-attribute data forests with rooting and concatenation and forest profiles with profile rooting and profile concatenation. This is necessary to show our reduction from SAT-XPath($\epsilon, \downarrow$, $\rightarrow$, $\Rightarrow$) into the derivation problem for forest profiles.

**Lemma C.2.** Given $R \subseteq \mathbb{D}$, and multi-attribute data forests $\bar{t}, \bar{t}, \bar{t}, \bar{t}, \bar{t}, \bar{t}$, we have:

1. Provided that $\text{abs}(\bar{t}, \bar{t}, \bar{t}, \bar{t}, \bar{t}, \bar{t}) = \bar{\chi}$.

2. Given $(a, d) \in \mathbb{A} \times \mathbb{D}$, $R \subseteq \mathbb{D}$ so that $\text{abs}(\bar{t}, \bar{t}, \bar{t}, \bar{t}, \bar{t}, \bar{t}) = \bar{\chi}$, we have:

\[
\text{abs}(\bar{t}, \bar{t}, \bar{r}) = (a, d) \text{ abs}(\bar{t}, \bar{t}, \bar{r}).
\]

**Proof of Lemma C.2.**

1. Follows from the semantics of XPath and the definition of $\text{abs}$. By definition of $\text{abs}$, the set $\chi_\alpha$ where $\chi_\alpha = \text{abs}(\bar{t}, \bar{t}, \bar{t}, \bar{t}, \bar{t}, \bar{t})$ is composed by all the paths that can reach data values in $\bar{t}, \bar{t}$, which is precisely $\chi_\alpha \cup \chi_\alpha$. In the same way, $\chi_\beta$ is composed by all the paths that can reach data values in $\bar{t}, \bar{t}$, which is precisely $\chi_\beta \cup \chi_\beta$. Finally, conditions $\text{abs}(\bar{t}, \bar{t}, \bar{t}, \bar{t}, \bar{t}, \bar{t})$ follow from the definition of $\text{abs}$.

2. Since $\xi(\text{abs}(\bar{t}, (a, d)\bar{t}, \bar{t})) = \bar{\chi}$, we only need to show that $\text{abs}(\bar{t}, (a, d)\bar{t}, \bar{t})$ is so that the rooting conditions $\text{abs}(\bar{t}, (a, d)\bar{t}, \bar{t})$ and $\text{abs}(\bar{t}, (a, d)\bar{t}, \bar{t})$ hold. Condition $\text{abs}(\bar{t}, (a, d)\bar{t}, \bar{t})$ is immediate since by definition of $\text{abs}$ we have that $\text{abs}(\bar{t}, (a, d)\bar{t}, \bar{t})$ has empty external descriptions. Condition $\text{abs}(\bar{t}, (a, d)\bar{t}, \bar{t})$ holds by definition of $\text{abs}$ and the semantics of XPath.

For any derivation forest $\bar{t}$ (as defined in Section 4.3) we associate a multi-attribute data forest $\bar{t}$ which is the result of removing all leaf nodes from $\bar{t}$ and projecting the tree onto $\mathbb{A} \times \mathbb{D}$.

**Lemma C.3.** For every boolean combination $\psi$ of non-recursive subformulas of $\phi$, we have that $\text{abs}(\bar{t}, \bar{t}, \bar{r}) \vdash \psi$ if, and only if, $\bar{t}, \bar{t}, \bar{r}, \bar{t}, \bar{t}, \bar{t}, \bar{t} \vdash \psi$, for $i_0 = |\bar{t}| + 1$ and $R = \text{data}(\bar{t}, \bar{t}, \bar{r}, \bar{t}, \bar{t}, \bar{t}, \bar{t})$. 

Proof of Lemma 77. Let $f = abs_R(f_t, t, t_i)$.

- Suppose first that $\psi = b$ with $b \in A$. Since $b \in P_\alpha$, we have that $f \vdash b$ iff $(b, e, i) \in \overline{\chi}(d)$ for some $d$, $i$ iff $(1, y) \in \overline{\Gamma}^f_y[\alpha, \delta]t$ for some $y$ by definition of $abs$ iff $(1, 1) \in \overline{\Gamma}^f_y[\alpha, \delta]t$ iff $t \vdash b$ iff $t \cdot t_{-i}, i \vdash b$, since $b \in A$.
- If $\psi = \psi_1 \lor \psi_2$ or $\psi = \neg \psi'$, we proceed by induction.
- Now suppose $\psi = (\alpha, \beta \equiv \gamma \delta \delta \delta)$, where $\alpha$ is leftward or empty, $\gamma$ is rightward or empty, and $\beta, \delta$ are downward or empty. Let $t' = a' \cdot d' = (a', d)(t \cdot t_{-i}).$

$\Rightarrow$ If $f \vdash \psi$ there must be some data value $d \in D$ that witnesses this fact, as required by the definition of $f$. Suppose first that $\alpha \neq \epsilon$, $\alpha \neq \epsilon$. Hence, there must be some $d \in D$ so that $(\overline{\Gamma}^f_y[\alpha, \delta]t, \epsilon, \beta, \delta) = (\alpha, \beta \equiv \gamma \delta \delta \delta)$.

1. If $(\overline{\Gamma}^f_y[\alpha, \delta]t, \epsilon, \beta, \delta) = (\alpha, \beta \equiv \gamma \delta \delta \delta)$, then there must be some $\psi'$ so that $(i_{0}, y)^{\prime} \in (\alpha, \beta \equiv \gamma \delta \delta \delta)(t, i_{-i})$. Let $d((\overline{\Gamma}^f_y[\alpha, \delta]t, \epsilon, \beta, \delta)) = \overline{\chi}(d)$.

2. If on the other hand $(\overline{\Gamma}^f_y[\alpha, \delta]t, \epsilon, \beta, \delta) = (\alpha, \beta \equiv \gamma \delta \delta \delta)$, then there must be some $\psi'$ so that $(i_{0}, y)^{\prime} \in (\alpha, \beta \equiv \gamma \delta \delta \delta)(t, i_{-i})$. Let $d((\overline{\Gamma}^f_y[\alpha, \delta]t, \epsilon, \beta, \delta)) = \overline{\chi}(d)$.

The case of $\psi = (\alpha, \beta \equiv \gamma \delta \delta \delta)$ is inside the subtree $t'$. Hence, there is some $d$ so that $(\overline{\Gamma}^f_y[\alpha, \delta]t, \epsilon, \beta, \delta) = (\alpha, \beta \equiv \gamma \delta \delta \delta)$.

The proof of Lemma 77 is now complete.

Lemma C.4. Let $f_{\text{root}} \in \mathcal{D}$ be a root profile, and let $t$ be its derivation tree. Let $\overline{\Gamma}$ be a maximal subforest of $t$, where $\overline{\Gamma}$ is the multi-attribute data forest associated with $t$, and $f_1, \ldots, f_n$ are the profile labels of the roots of $\overline{\Gamma}$. Let $R$ be the rigid set for $f_1, \ldots, f_n$. Then:

- $abs_R(\epsilon, t, \epsilon) = f_1 + \cdots + f_n$, and
- $f_i \vdash \phi_2$ for every $1 \leq i \leq \epsilon$.

Proof of Lemma 77. We will make use of the following easy fact.

Claim C.4.1. Given $f_1, \ldots, f_n \in \mathcal{S}$, so that $f_i = f_1 + \cdots + f_n$ where $\overline{\chi}_i = \overline{\chi}_i = \emptyset$, then the sets of $\{i\}$ determine the sets of $\overline{\chi}_i = \overline{\chi}_i$.

Proof. This is by conditions [1] and [2] of profile concatenation.

- $\overline{\chi}_i$ and $\overline{\chi}_i$ determine $\overline{\chi}_i$. Hence, $\overline{\chi}_i = \overline{\chi}_i$.
- $\overline{\chi}_i$ and $\overline{\chi}_i$ determine $\overline{\chi}_i$. Hence, $\overline{\chi}_i = \overline{\chi}_i$.

We now prove the main statement by generalized induction on the height of $\overline{\Gamma}$. Suppose that $f$ has height $m \geq 1$. Then there must be forests $f_1, \ldots, f_n$ so that

$\overline{\Gamma} = (a_i, d_i) in f_i$ and each $f_i$ has height less than $m$.

Let

$\hat{f}_i = abs_R((a_i, d_i) f_i, (a_i, d_i) f_i, ((a_i, d_i) f_i, (a_i, d_i) f_i))$

for every $i$. We then have the following.
CLAIM C.4.2. For every $i \in [n]$, $f_i = \hat{f}_i$.  

Proof. For every $i \neq \epsilon$, we have that $f_1^i + \cdots + f_n^i = \text{absR}(\epsilon, \hat{f}_i, \epsilon)$ by inductive hypothesis, where $f_1^i, \ldots, f_n^i$ are the profile labels of the children of the root of the $i$th tree of $\hat{f}$. Then, since $\hat{f}_i \in (a, \check{d}_i)(f_1^i + \cdots + f_n^i)$ by definition of derivation tree, we have that $f_i \in (a, \check{d}_i)\text{absR}(\epsilon, \hat{f}_i, \epsilon)$. If $\hat{f}_i = \epsilon$, then, since $\text{absR}(\epsilon, \epsilon, \epsilon) = \hat{f}_0$ and $f_i \in (a, \check{d}_i)\hat{f}_0$, we also have that $f_i \in (a, \check{d}_i)\text{absR}(\epsilon, \hat{f}_i, \epsilon)$. Hence, for all $i \in [n]$, $f_i \in (a, \check{d}_i)\text{absR}(\epsilon, \hat{f}_i, \epsilon)$.

Now, since

- all the internal descriptions of any forest profile inside $(a, \check{d}_i)\text{absR}(\epsilon, \hat{f}_i, \epsilon)$ is completely determined by $(a, \check{d}_i)$ and $\text{absR}(\epsilon, \hat{f}_i, \epsilon)$, and hence $\hat{f}_i$ and $\check{f}_i$ have the same internal descriptions,
- the external descriptions of $\hat{f}_1 + \cdots + \hat{f}_n$ are $\emptyset$,
- $\text{absR}(\epsilon, (a, \check{d}_i)\check{f}_1 \cdots (a, \check{d}_n)\check{f}_n, \epsilon) = \hat{f}_1 + \cdots + \hat{f}_n$ by Lemma ??-??, and the external descriptions of $\hat{f}_1 + \cdots + \hat{f}_n$ are $\emptyset$,

we can apply Claim ??, concluding that $f_i = \hat{f}_i$ for every $i$. $\square$

Therefore, since $\text{absR}(\epsilon, (a, \check{d}_i)\check{f}_1 \cdots (a, \check{d}_n)\check{f}_n, \epsilon) = \hat{f}_1 + \cdots + \hat{f}_n$ by Lemma ??-??, we have that

$$\text{absR}(\epsilon, (a, \check{d}_i)\check{f}_1 \cdots (a, \check{d}_n)\check{f}_n, \epsilon) = \hat{f}_1 + \cdots + \hat{f}_n.$$  

On the other hand, since $\xi(f_1), \ldots, \xi(f_n) \in \Gamma_\phi$ by condition $[a]$ of rooting, we have $\check{f}_i = f_i \vdash \bigwedge_{a \in A} (a \Rightarrow \gamma_\phi(a))$ for every $i$. Hence, by Lemma ??, $(a, \check{d}_i)\check{f}_1 \cdots (a, \check{d}_n)\check{f}_n, i \vdash \bigwedge_{a \in A} (a \Rightarrow \gamma_\phi(a))$ for every $i$. Also, by inductive hypothesis, for all $i \neq \epsilon$ we have that $\check{t}_i, j \models \phi_2$ for all $1 \leq j \leq |t_i|$. Then, by definition of $\phi_2$, it follows that

$$(a, \check{d}_i)\check{f}_1 \cdots (a, \check{d}_n)\check{f}_n, i \models \phi_2$$

for every $i$. $\square$

We now show that if $\phi$ is satisfiable, then there must be a derivable root profile.

Lemma C.5. Let $t_\phi$ be a tree so that $t_\phi \models \phi$. For every maximal subforest $t_1 \cdot t_2 \cdots t_r$ of $t_\phi$ so that $t_i \neq \epsilon$, there is a derivable forest profile $\check{f} \in D$ so that $\check{f} = \text{absR}(\hat{t}_1, \hat{t}_2, \cdots, \hat{t}_r)$, where $R = \text{data}(t_1 \cdot t_2 \cdots t_r)$. Further, if $t$ is a tree, $\check{f}$ is so that $\check{f} \in (a, d)\check{f}'$, with $\check{f}' \in \hat{D}$.

Proof of Lemma ???. We proceed by induction on the size of $t$, defined as its number of nodes.

Suppose first that $t$ consists in one tree with just one node $(a, d)$. We show that $\text{absR}(t_1, (a, d), t_r) \in (a, d)\hat{f}_0$.

Since $\check{t}_1 \cdot (a, d)\check{t}_r, j \models \phi_2$, for $j = |t_i| + 1$, we have that $\text{absR}(\check{t}_1, t, \check{t}_r) = \gamma_\phi(a)$ by Lemma ???. In other words, we have $\xi(\text{absR}(\check{t}_1, \check{t}_r, \epsilon)) \in \Gamma_\phi$. Then, by Lemma ??-?? we have $\text{absR}(\check{t}_1, (a, d), t_r) \in (a, d)\text{absR}(\epsilon, \epsilon, \epsilon)$. Since $\check{f}_0 = \text{absR}(\epsilon, \epsilon, \epsilon)$ (Remark ??), we have that $(a, d)\text{absR}(\epsilon, \epsilon, \epsilon) = (a, d)\hat{f}_0$. Therefore, $\text{absR}(\check{t}_1, (a, d), t_r) \in \hat{D}$.

Suppose now that $t$ consists in one tree $(a, d)\check{t}'$ where $\check{t}' \neq \epsilon$. By inductive hypothesis there is some $\check{f}' \in \hat{D}$ so that $\check{f}' = \text{absR}(\epsilon, \check{t}', \epsilon)$ for some $\check{D}'$. By Lemma ?? since $\check{t}_1 \cdot (a, d)\check{t}', j \models \phi_2$ for $j = |t_i| + 1$, we have that $\xi(\text{absR}(\check{t}_1, (a, d)\check{t}', \epsilon)) \in \Gamma_\phi$. We then have that

$$\text{absR}(\check{t}_1, (a, d)\check{t}', \epsilon) \in (a, d)\check{f}' \in (a, d)\text{absR}(\epsilon, \check{t}', \epsilon)$$

again by Lemma ??-??. Therefore, $\text{absR}(\check{t}_1, \hat{t}, \epsilon) \in \hat{D}$.

Suppose now that $t$ consists in more than one tree, let $t_1$ and $t_2$ be non-empty forests so that $\check{t} = \check{t}_1 \cdot \check{t}_2$. By applying twice the inductive hypothesis there must be $f_1, f_2 \in \hat{D}$ so that $\check{f}_1 = \text{absR}(\check{t}_1, \check{t}_1, \check{t}_2, \epsilon)$ and $f_2 = \text{absR}(\check{t}_2, \check{t}_1, \check{t}_2, \epsilon)$. As $\text{absR}(\check{t}_1, \check{t}_1, \check{t}_2, \epsilon) = \text{absR}(\check{t}_1, \check{t}_1, \check{t}_2, \epsilon) + \text{absR}(\check{t}_1, \check{t}_2, \epsilon)$ by Lemma ??-??, it follows that $\text{absR}(\check{t}_1, \hat{t}, \epsilon) = \check{f}_1 + \check{f}_2 \in \hat{D}$ since $\check{f}_1, \check{f}_2 \in \hat{D}$. $\square$

Corollary C.6. There is a derivable root forest profile if, and only if, $\phi$ is satisfiable.

Proof of Corollary C.6. $\Leftarrow$. Let $\check{f}$ be a derivable root forest profile. Then, there must be a derivation tree $t$ for $\check{f} = (R, \check{x})$. Let $t$ be the multi-attribute data tree associated to $t$. By Lemma ??, we have that $\text{absR}(\epsilon, t, \epsilon) = \check{f}$ and that $t \models \phi_2$. Since $\check{f}$ is a root profile, the root of $t$ must have a label $(a, d)$ with $a \in R_{\text{root}}$, and hence $t \models \phi_1$ as well. Therefore, $\phi$ is satisfiable.

$\Rightarrow$. Let $t$ be so that $t \models \phi$. Then by Lemma ?? we have that $\check{f} = \text{absR}(\epsilon, t, \epsilon) \in \hat{D}$. Further, since $t$ is a tree, we have that in fact $\hat{f} = (a, d)\check{f}'$ for some $\check{f}' \in \hat{D}$. Further, since $t \models \phi_1$, $a \in R_{\text{root}}$. Hence, $\check{f}$ is a derivable root profile. $\square$