Model Theory of XPath on Data Trees.
Part II: Binary Bisimulation and Definability

Sergio Abriola María Emilia Descotte Santiago Figueira

University of Buenos Aires

Abstract

We study the expressive power of the downward and vertical fragments of XPath equipped with (in)equality tests over data trees. We show definability theorems (which provide conditions under which a class of pointed data trees can be defined by a node expression or by a set of node expressions) and separation theorems (which provide conditions under which two disjoint classes of pointed data trees can be separated by a class definable by a node expression or a set of node expressions). To do so, we introduce a notion of saturation, and show that over saturated data trees, the already known notion of (unary) bisimulation coincides with logical equivalence.

Both for the downward and vertical fragments, we introduce new notions of binary bisimulations, which relate two *pairs* of nodes of data trees. We show that over finitely branching data trees, these notions correspond to the idea of 'indistinguishability by means of path expressions'. We show a characterization theorem, which states that a first-order formula with two free variables is expressible in the downward fragment of XPath with (in)equality tests if and only if it is binary-bisimulation-invariant and represents a 'forward property'. Using the new tool of binary bisimulations, together with suitable modifications of saturation, we show definability and separation theorems, this time with respect to classes of two-pointed data trees (with some restrictions) and in the context of path expressions as the language of description.

1 Introduction

The abstraction of an XML document is a data tree, i.e. a tree whose every node contains a tag or label (such as *LastName*) from a finite domain, and a data value (such as *Smith*) from an infinite domain. For instance, Figure 1 depicts a data tree whose labels are a or b and whose data values are integers.

XPath is the most widely used query language for XML documents; it is an open standard and constitutes a World Wide Web Consortium (W3C) Recommendation [5]. XPath has syntactic operators to navigate the tree using the 'child', 'parent', 'sibling', etc. accessibility relations, and can make tests on intermediate nodes. Core-XPath [14] is the fragment of XPath 1.0 containing only the navigational behavior of XPath. It can express properties on nodes with respect to the underlying tree structure of the XML document, such as

$$\textit{nodes with a child labeled } a \textit{ and a child labeled } b. \tag{1}$$

It can also express properties on paths along the tree such as

$$\begin{array}{c}\textit{downward paths of length two starting in a node}\\ \textit{with label } a \textit{ and ending in a node with label } b.\end{array} \tag{2}$$

The first types of formulas are evaluated on individual nodes and are called **node expressions**. In the data tree of Figure 1, condition (1) is true at node x and false at node u. Formulas of the second kind are evaluated on pairs of nodes and are called **path expressions**. In the same

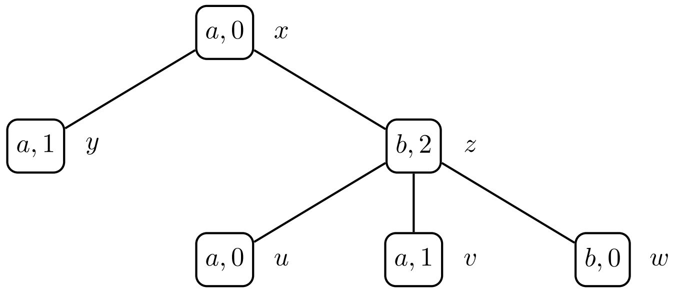

Figure 1: A data tree $\mathcal{T}$ with a and b as labels and integers as data values. Nodes x, y, z, u, v, w are names for nodes. They are not part of the data tree; they are mentioned here only for explanatory purposes.

example, condition (2) is true at (x, w) because there is a path with the condition expressed in (2) connecting x and w, but false at (x, u) because there is no way to connect x and u with a path satisfying the condition (2). However, Core-XPath [14] cannot express conditions on the actual data contained in the attributes, such as the node expression

$$\textit{nodes with two children with same label } a \textit{ but different data value} \tag{3}$$

(which in our example is true only at z), or the path expression

$$\textit{downward paths of length one whose starting and end point have the same data value} \tag{4}$$

(which is true nowhere). But Core-Data-XPath [3] —here called XPath$_=$— can. Indeed, XPath$_=$ is the extension of Core-XPath with (in)equality tests between attributes of elements in an XML document. Here we study two main fragments of XPath$_=$: the **downward** fragment (containing only the 'child' axis) and the **vertical** fragment (containing both the 'child' and the 'parent' axes). Whilst (2) and (4) above describe downward paths, we will see that the latter (4) cannot be formalized using only 'child', and needs the 'parent' axis. Of course, not all paths should be downward. An example of a path that is not necessarily downward could be:

$$\textit{paths of length one whose starting and end point have label } a, \tag{5}$$

which can be either downward or upward. In general, vertical XPath$_=$ allows us express paths of the form *go up and check some condition, then go down and check this other condition, etc.*

In a recent paper [10], the expressive power of XPath$_=$ was studied, from a logical and modal model theoretical point of view. Bisimulations are a classic tool of modal logics to determine equivalence between relational models. A node x of a data tree $\mathcal{T}$ and a node x' of a data tree $\mathcal{T}'$ are said to be *bisimilar* if they satisfy some special (depending of the studied fragment) back-and-forth conditions over the structure of the data tree. Of course, different logics have different definitions of bisimulations. For non-modal logics, such as first-order logic, the notions of bisimulation are typically stated in terms of Ehrenfeucht-Fraïssé games. In [10] suitable notions of bisimulations were given both for the downward and the vertical fragment of XPath$_=$. It is shown that if x and x' are bisimilar then they satisfy exactly the same node expressions, and that the converse is also true for trees whose every node has only finitely many children. Hence, bisimulation coincides with logical equivalence, i.e., with *indistinguishability by means of node expressions*. It is also shown a van Benthem-like characterization theorem for the downward fragment of XPath$_=$, which states that it coincides with the bisimulation-invariant fragment of first-order logic with one free variable (over the adequate signature). For the case of the vertical fragment of XPath$_=$ this characterization fails.

This article is a natural continuation of [10], as we develop new tools and delve in some aspects of the model theory of the downward and vertical fragments of XPath$_=$. In the first part we show a definability theorem with respect to node expressions, which answers the basic question of when a class of pointed data trees is definable by a set of node expressions, or by a single node expression, over the downward and the vertical fragments. Our main result in this part is the analog of the classic first-order definability theorem (see, e.g. [4, Cor. 6.1.16]), which can be stated as follows:

> *A class of models K is definable by means of a set of first-order formulas if and only if K is closed under ultraproducts and isomorphisms, and the complement of K is closed under ultrapowers. Also, K is definable by a single first-order formula if and only if both K and its complement are closed under ultraproducts and isomorphisms.*

As a corollary, we obtain the analog of the classic first-order separation theorem (see, e.g. [4, Cor. 6.1.17]), which says:

> *Let K_1 and K_2 be two disjoint classes of models such that K_1 is closed under isomorphisms and ultraproducts and K_2 is closed under isomorphisms and ultrapowers. Then there exists a class K that is definable by a set of first-order formulas, contains K_1, and is disjoint from K_2. Furthermore, if both K_1 and K_2 are closed under isomorphisms and ultraproducts, then such K is definable by a single first-order formula.*

Though we take as motivation the current relevance of XML documents (which of course are finite) and the logics for reasoning over them, we do not restrict ourselves to the finite case. Indeed, an infinite set of node expressions may force all its data tree models to be infinite. Hence, since we aim at working with arbitrary sets of node expressions, we must consider arbitrary (i.e. finite or infinite) data trees.

Our definability and separation theorems for XPath$_=$ themselves are shown using rather known techniques. The main contribution of this part, however, is to devise and calibrate the adequate notions to be used in the XPath$_=$ setting, and to study the subtle interaction between them:

- *Bisimulation*: already introduced in [10], it is the counterpart of *isomorphisms* in the classical theorem for first-order logic. It is known that if two (possibly infinite) data trees are bisimilar then they are logically equivalent (that is, they are not distinguishable by an XPath$_=$ node expression) but that the converse is not true in general.

- *Saturation*: we define and study the new notion of *XPath$_=$-saturation*. We show that for XPath$_=$-saturated data trees, being bisimilar is the same as being logically equivalent. It is also shown that a 2-saturated data tree (regarded as a first-order structure) is already XPath$_=$-saturated.

- *Ultraproducts*: contrary to other adaptations of the classical first-order definability theorem to modal logics, in our case we have to adjust also the notion of *ultraproduct*, and so we work with a variant of it called *quasi-ultraproduct*. The reason is that we must not abandon the universe of data trees, as these are the only allowed models of XPath$_=$.

In the second part of our work we start a model theoretical study of path expressions of XPath$_=$. We introduce a new kind of *binary* bisimulation for both the downward and the vertical fragment, which captures, over finitely branching trees, when two pairs of nodes (instead of single nodes, as in [10]) are indistinguishable by means of path expressions (instead of by node expressions). Our binary bisimulations subsume, in fact, the already known unary bisimulation, since over finitely branching trees, (x, x) is binary-bisimilar to (x', x') if and only if x is unary-bisimilar to x'. The definitions of binary bisimulations require more rules than the unary ones, but they all have the flavor of back-and-forth conditions. Furthermore, these rules are slightly simpler than those for the unary bisimulation.

We show a characterization theorem for the path expressions of the downward fragment of XPath$_=$: a first-order formula $\varphi(x, y)$ with two free variables is expressible by a path expression in the downward fragment of XPath$_=$ if and only if $\varphi(x, y)$ is binary-bisimulation-invariant and represents a forward property (that is, a property that holds only for pairs (u, v) where v is a descendant of u).

Finally we show restricted definability theorems with respect to path expressions, which answer the question of when a class of two-pointed data trees (satisfying some further conditions) is definable by a set of path expressions, or by a single path expression, over the downward and the vertical fragments of XPath$_=$. We show separation results over such classes of two-pointed data trees in the expected way. We adapt some of the tools developed for single-node pointed data trees, and we use the binary bisimulations as the main ingredient. The major obstacle in this case is to deal with the absence of complementation and intersection in the of language XPath$_=$ path expressions.

1.1 Related work

There are many works in the literature studying the expressive power of Core-XPath (see e.g. [15, 18, 25]). All these consider the navigational fragment of XPath. A first step towards the study of the expressive power of XPath when equipped with (in)equality test over data trees, is the recent paper [10], and its full version [9]. The present development is a natural continuation of that work.

The notion of bisimulation was introduced independently by van Benthem [26] in the context of modal correspondence theory, by Milner [19] and Park [22] in concurrency theory, and by Forti and Honsell [13] in non-wellfounded set theory (see [24] for a historical outlook). With respect to our notions of binary bisimulations, we can mention the recent work [12], where some notions of bisimulations are given for some fragments of Tarsky's calculus of binary relations, with the aim of understanding the expressive power of the calculus of relations as a database query language for binary relation structures.

The classical result of definability for first-order logic was adapted to the context of many modal logics, where the notion of *isomorphism* is replaced by the weaker concept of *bisimulation* (the one which turns to be adequate for the chosen modal logic). Thus, definability theorems were established for the basic modal logic [7], for temporal logics with *since* and *until* operators [16], for negation-free modal languages [17], etc. A global counterpart was studied in [8], and a general framework stating sufficient conditions for an arbitrary (modal) logic $\mathcal{L}$ to verify it was given in [1]. One of those requirements is that the models of $\mathcal{L}$ are closed under ultraproducts, which is true for the aforementioned logics, but not for XPath$_=$. Indeed, models of XPath$_=$ are data trees, which may not remain connected under ultraproducts. Hence one cannot expect to use that framework in this case. The Separation theorem for the basic modal logic was shown by de Rijke in [7], and it was studied for other specific modal logics such as the temporal logic [16]. For more general modal logics, Separation was studied in [1], but again, XPath$_=$ does not fit in here.

In [26] van Benthem characterizes the basic modal logic as the bisimulation invariant fragment of first-order logic. Van Benthem's original result over arbitrary structures was proved to hold for finite structures by Rosen [23]. The proof was then simplified and unified by Otto [20, 21], and later expanded by Dawar and Otto [6] to other classes of structures. We follow the ideas of [20] to show the characterization result for binary bisimulations in the downward fragment.

1.2 Outline

The paper is organized as follows, In §2 we introduce the formal syntax and semantics of the downward and vertical fragments of XPath$_=$, together with notions of (unary) bisimulations from [10, 9]. Suitable notions of saturation for both fragments are given in §3.1, where it is also shown that for saturated trees bisimilarity coincides with logical equivalence. In §3.2 we explain the connection between XPath$_=$ and first-order logic, and we introduce the idea of quasi-ultraproducts for the downward and vertical fragments. In §3.3 and §3.4 we state the theorems on definability and separation, respectively.

In §4 we start our study of path expressions, which is divided in the downward fragment (§4.1) and the vertical fragment (§4.2). For the downward fragment, we begin with some needed facts

(§4.1.1), and we then define the notions of logical equivalence to be used (§4.1.2). The definitions of binary bisimulations for the downward fragment are given in §4.1.3, where it is also shown their coincidence to the logical equivalence for path expressions. The characterization theorem is given in §4.1.4. For the vertical fragment, we first show some needed facts in §4.2.1, and then introduce the definition of binary bisimulation in §4.2.2, where, again, it is shown that it matches logical equivalence. In §5 we introduce the needed changes to the notions of saturation and quasi-ultraproducts for the case of two-pointed data trees, and we state the theorems of definability and separation for the scenario of path expressions, over a restricted class of two-pointed data trees. In this section some proofs are not given since they are analogous to those of §3.3 and §3.4. However, it is explained in detail how to express, in the language of path expression, the needed operations of complementation and intersection, provided the universe of two-pointed data trees is restricted. Finally, in §6 we show some simple applications of our definability theorems, and we close in §7 with some conclusions and future work.

The following table summarizes the organizational structure of the paper and points out the main results:

	Node expressions		Path expressions	
	Downward	Vertical	Downward	Vertical
Bisimulation notion	[9, §3.1.2]	[9, §3.2.4]	§4.1.3	§4.2.2
Characterization	[9, §4.1]	Fails [9, §4.2]	§4.1.4 Thm. 52	Fails
Def. saturation	§3.1	§3.1	§5.1	§5.1
Def. quasi-ultraprod.	§3.2	§3.2	§5.2	§5.2
Definability	§3.3 Thms. 20,21,22	§3.3 Thms. 25,26,27	§5.3 Thms. 67,68	§5.3
Separation	§3.4 Thms. 28,29	§3.4	§5.3 Thms. 69,70	§5.3

2 Preliminaries

2.1 Data trees

Let *Trees*(A) be the set of ordered and unranked (finite or infinite) trees over an arbitrary alphabet A. We say that $\mathcal{T}$ is a **data tree** if it is a tree from *Trees*$(\mathbb{A} \times \mathbb{D})$, where $\mathbb{A}$ is a finite set of **labels** and $\mathbb{D}$ is an infinite set of **data values**. For instance, the tree $\mathcal{T}$ of Figure 1 belongs to *Trees*$(\{a, b\} \times \mathbb{N})$. A data tree is **finitely branching** if every node has finitely many children. For any given data tree $\mathcal{T}$, we denote by T its set of nodes. We use letters x, y, z, u, v, w as variables for nodes. Given a node $x \in T$ of $\mathcal{T}$, we write $label(x) \in \mathbb{A}$ to denote the node's label, and $data(x) \in \mathbb{D}$ to denote the node's data value.

Given two nodes $x, y \in T$ we write $x \to y$ if y is a child of x, and $x \xrightarrow{n} y$ if y is a descendant of x at distance n. In particular, $\xrightarrow{1}$ is the same as $\to$, and $\xrightarrow{0}$ is the identity relation. We denote by $x \xrightarrow{\leq m} y$ the fact that $x \xrightarrow{n} y$ for some $n \leq m$. $(\xrightarrow{n} y)$ denotes the sole ancestor of y at distance n (assuming it has one).

2.2 Vertical and Downward XPath with data tests

We work with a simplification of XPath, stripped of its syntactic sugar. We consider fragments of XPath that correspond to the navigational part of XPath 1.0 with data equality and inequality. XPath$_=$ is a two-sorted language, with **path expressions** (that we write α, β, γ) expressing properties of paths, and **node expressions** (that we write φ, ψ, η), expressing properties of nodes. The **vertical XPath**, notated XPath$_=(\uparrow\downarrow)$ is defined by mutual recursion as follows:

$$\begin{array}{lll} \alpha, \beta ::= o \mid [\varphi] \mid \alpha\beta \mid \alpha \cup \beta & & o \in \{\varepsilon, \uparrow, \downarrow\} \\ \varphi, \psi ::= a \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \langle\alpha\rangle \mid \langle\alpha = \beta\rangle \mid \langle\alpha \neq \beta\rangle & & a \in \mathbb{A} \end{array}$$

We call **downward XPath**, notated XPath$_=(\downarrow)$, to the syntactic fragment which only uses the navigation axis $\downarrow$, but not $\uparrow$. An **XPath$_=(\uparrow\downarrow)$-formula** [resp. **XPath$_=(\downarrow)$-formula**] is either a node expression or a path expression of XPath$_=(\uparrow\downarrow)$ [resp. XPath$_=(\downarrow)$].

Semantics of XPath$_=(\uparrow\downarrow)$ in a given data tree $\mathcal{T}$ are defined as follows:

$$
\begin{aligned}
[\![\downarrow]\!]^{\mathcal{T}} &= \{(x,y) \mid x{\to}y\} \\
[\![\uparrow]\!]^{\mathcal{T}} &= \{(x,y) \mid y{\to}x\} \\
[\![\varepsilon]\!]^{\mathcal{T}} &= \{(x,x) \mid x \in T\} \\
[\![\alpha\beta]\!]^{\mathcal{T}} &= \{(x,z) \mid (\exists y \in T)\ (x,y) \in [\![\alpha]\!]^{\mathcal{T}}, (y,z) \in [\![\beta]\!]^{\mathcal{T}}\} \\
[\![\alpha \cup \beta]\!]^{\mathcal{T}} &= [\![\alpha]\!]^{\mathcal{T}} \cup [\![\beta]\!]^{\mathcal{T}} \\
[\![[\varphi]]\!]^{\mathcal{T}} &= \{(x,x) \mid x \in [\![\varphi]\!]^{\mathcal{T}}\} \\
[\![a]\!]^{\mathcal{T}} &= \{x \in T \mid \mathit{label}(x) = a\} \\
[\![\neg\varphi]\!]^{\mathcal{T}} &= T \setminus [\![\varphi]\!]^{\mathcal{T}} \\
[\![\varphi \wedge \psi]\!]^{\mathcal{T}} &= [\![\varphi]\!]^{\mathcal{T}} \cap [\![\psi]\!]^{\mathcal{T}} \\
[\![\langle\alpha\rangle]\!]^{\mathcal{T}} &= \{x \in T \mid (\exists y \in T)\ (x,y) \in [\![\alpha]\!]^{\mathcal{T}}\} \\
[\![\langle\alpha = \beta\rangle]\!]^{\mathcal{T}} &= \{x \in T \mid (\exists y{,}z \in T)(x,y) \in [\![\alpha]\!]^{\mathcal{T}}, (x,z) \in [\![\beta]\!]^{\mathcal{T}}, \mathit{data}(y) = \mathit{data}(z)\} \\
[\![\langle\alpha \neq \beta\rangle]\!]^{\mathcal{T}} &= \{x \in T \mid (\exists y{,}z \in T)(x,y) \in [\![\alpha]\!]^{\mathcal{T}}, (x,z) \in [\![\beta]\!]^{\mathcal{T}}, \mathit{data}(y) \neq \mathit{data}(z)\}
\end{aligned}
$$

Example 1. Some examples of node expressions over the data tree of Figure 1:

- $\varphi_1 = \langle\downarrow[a]\rangle \wedge \langle\downarrow[b]\rangle$ expresses property (1), i.e. "nodes with a child labeled a and a child labeled b", and $[\![\varphi_1]\!]^{\mathcal{T}} = \{x, z\}$.
- $\varphi_2 = \langle\downarrow[a] \neq \downarrow[a]\rangle$ expresses property (4), i.e. "*nodes with two children with same label a but different data value*", and $[\![\varphi_2]\!]^{\mathcal{T}} = \{z\}$
- $\varphi_3 = \langle\varepsilon \neq \uparrow[\langle\varepsilon \neq \uparrow\rangle]\rangle$ expresses "*nodes with a data value different from the one of his parent, who, in turn, has a data value different from his parent*", and $[\![\varphi_3]\!]^{\mathcal{T}} = \{u, v, w\}$.
- $\varphi_4 = \langle\varepsilon \neq \downarrow\downarrow[\varphi_3]\rangle$ expresses "*nodes with a downward path of length 2, with all distinct data values*", and $[\![\varphi_4]\!]^{\mathcal{T}} = \{x\}$.

Example 2. Some examples of path expressions over the data tree of Figure 1:

- $\alpha_1 = [a]\downarrow\downarrow[b]$ expresses property (2), i.e. "*downward paths of length two starting in a node with label a and ending in a node with label b*", and $[\![\alpha_1]\!]^{\mathcal{T}} = \{(x, w)\}$.
- $\alpha_2 = [a]\downarrow[a] \cup [a]\downarrow\downarrow[a]$ expresses "*downward paths of length one or two, starting and ending in a node with label a*", and $[\![\alpha_2]\!]^{\mathcal{T}} = \{(x, y), (x, u), (x, v)\}$.
- $\alpha_3 = \downarrow[\langle\epsilon = \uparrow\rangle]$ expresses property (4), i.e. "*downward paths of length one whose starting and end point have the same data value*", and $[\![\alpha_3]\!]^{\mathcal{T}} = \emptyset$. Notice that we needed to use the 'parent' relation. As we will see next, this is unavoidable.
- $\alpha_4 = [a]\downarrow[a] \cup [a]\uparrow[a]$ expresses property (5) "*paths of length one whose starting and end point have label a*", and $[\![\alpha_4]\!]^{\mathcal{T}} = \{(x, y), (y, x)\}$.

For a data tree $\mathcal{T}$ and $u, v \in T$, we say that $\mathcal{T}, u$ is a **pointed data tree**, and that $\mathcal{T}, u, v$ is a **two-pointed data tree**. For a node expression φ, we write $\mathcal{T}, u \models \varphi$ to denote $u \in [\![\varphi]\!]^{\mathcal{T}}$, and in that case we say that $\mathcal{T}, u$ satisfies φ or that φ is true at $\mathcal{T}, u$. In the same way, for a path expression α, we write $\mathcal{T}, u, v \models \alpha$ to denote $(u, v) \in [\![\alpha]\!]^{\mathcal{T}}$, and we say that $\mathcal{T}, u, v$ satisfies α or that α is true at $\mathcal{T}, u, v$. We say that the node expressions φ, ψ of XPath$_=$ are **equivalent** (notation: $\varphi \equiv \psi$) iff $[\![\varphi]\!]^{\mathcal{T}} = [\![\psi]\!]^{\mathcal{T}}$ for all data trees $\mathcal{T}$. Similarly, path expressions α, β of XPath$_=$ are **equivalent** (notation: $\alpha \equiv \beta$) iff $[\![\alpha]\!]^{\mathcal{T}} = [\![\beta]\!]^{\mathcal{T}}$ for all data trees $\mathcal{T}$.

Let $\mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, u)$ [resp. $\mathrm{Th}_{\downarrow}(\mathcal{T}, u)$] be the set of all $\mathrm{XPath}_=(\uparrow\downarrow)$-node expressions [resp. $\mathrm{XPath}_=(\downarrow)$-node expressions] true at $\mathcal{T}, u$. Similarly, let $\mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, u, v)$ [resp. $\mathrm{Th}_{\downarrow}(\mathcal{T}, u, v)$] be the set of all $\mathrm{XPath}_=(\uparrow\downarrow)$-path expressions [resp. $\mathrm{XPath}_=(\downarrow)$-path expressions] true at $\mathcal{T}, u, v$.

In terms of expressive power of node expressions, it is easy to see that $\cup$ is unessential (see [10, §2.2]): every $\mathrm{XPath}_=$ node expression φ has an equivalent φ' with no $\cup$ in its path expressions. It is enough to use the following equivalences to eliminate occurrences of $\cup$:

$$\begin{aligned}
\langle \alpha \star \beta \rangle &\equiv \langle \beta \star \alpha \rangle \\
\langle \beta(\alpha \cup \alpha')\beta' \rangle &\equiv \langle \beta\alpha\beta' \rangle \vee \langle \beta\alpha'\beta' \rangle \\
\langle \gamma \star \beta(\alpha \cup \alpha')\beta' \rangle &\equiv \langle \gamma \star \beta\alpha\beta' \rangle \vee \langle \gamma \star \beta\alpha'\beta' \rangle
\end{aligned}$$

where $\star \in \{=, \neq\}$. Observe also that $\langle \alpha \rangle \equiv \langle \alpha = \epsilon \rangle \vee \langle \alpha \neq \epsilon \rangle$, so we can restrict ourselves to the fragment without formulas of the form $\langle \alpha \rangle$.

For the case of path expressions, there is no such possible elimination of $\cup$. Indeed the path expression α_2 of Example 2 cannot be restated without using $\cup$. The reason of this is that there are no intersections or complementations of path expressions (in §4 and §5 we will study this issue).

2.3 Equivalences and syntactic measures

Let $\mathcal{T}$ and $\mathcal{T}'$ be data trees, and let $u \in T$, $u' \in T'$. We say that $\mathcal{T}, u$ and $\mathcal{T}', u'$ are **equivalent for $\mathbf{XPath}_=(\uparrow\downarrow)$** [resp. **equivalent for $\mathbf{XPath}_=(\downarrow)$**] (notation: $\mathcal{T}, u \equiv^{\uparrow\downarrow} \mathcal{T}', u'$ [resp. $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'$]) iff for all node expressions $\varphi \in \mathrm{XPath}_=(\uparrow\downarrow)$ [resp. $\varphi \in \mathrm{XPath}_=(\downarrow)$], we have $\mathcal{T}, u \models \varphi$ iff $\mathcal{T}', u' \models \varphi$.

We write $\mathrm{dd}(\varphi)$ to denote the **downward depth** of φ, which measures 'how deep' the formula can see, and it is defined as follows:

$$\begin{array}{rcl@{\qquad}rcl}
\mathrm{dd}(a) &=& 0 & \mathrm{dd}(\lambda) &=& 0 \\
\mathrm{dd}(\varphi \wedge \psi) &=& \max\{\mathrm{dd}(\varphi), \mathrm{dd}(\psi)\} & \mathrm{dd}(\varepsilon\alpha) &=& \mathrm{dd}(\alpha) \\
\mathrm{dd}(\neg\varphi) &=& \mathrm{dd}(\varphi) & \mathrm{dd}([\varphi]\alpha) &=& \max\{\mathrm{dd}(\varphi), \mathrm{dd}(\alpha)\} \\
\mathrm{dd}(\langle\alpha\rangle) &=& \mathrm{dd}(\alpha) & \mathrm{dd}(\downarrow\alpha) &=& 1 + \mathrm{dd}(\alpha) \\
\mathrm{dd}(\langle\alpha \odot \beta\rangle) &=& \max\{\mathrm{dd}(\alpha), \mathrm{dd}(\beta)\} & & &
\end{array}$$

where $a \in \mathbb{A}$, $\odot \in \{=, \neq\}$, and α is any path expression or the empty string λ. Let ℓ-$\mathrm{XPath}_=(\downarrow)$ be the fragment of $\mathrm{XPath}_=(\downarrow)$ consisting of all node expressions φ with $\mathrm{dd}(\varphi) \leq \ell$.

We say that $\mathcal{T}, u$ and $\mathcal{T}', u'$ are **ℓ-equivalent for $\mathbf{XPath}_=(\downarrow)$** (notation: $\mathcal{T}, u \equiv^{\downarrow}_{\ell} \mathcal{T}', u'$) iff for all node expression $\varphi \in \ell$-$\mathrm{XPath}_=(\downarrow)$, we have $\mathcal{T}, u \models \varphi$ iff $\mathcal{T}', u' \models \varphi$.

For the vertical fragment of $\mathrm{XPath}_=$, we need to define both the maximum distance r going downward and the maximum distance s going upward that the formula can reach. We call the pair (r, s) the **vertical depth** of a formula (notation: $\mathrm{vd}(\varphi)$). The **nesting depth** of a formula φ (notation: $\mathrm{nd}(\varphi)$) is the maximum number of nested [ ] appearing in φ.

$$\begin{array}{rcl@{\qquad}rcl}
\mathrm{vd}(a) &=& (0,0) & \mathrm{vd}(\lambda) &=& (0,0) \\
\mathrm{vd}(\varphi \wedge \psi) &=& \max\{\mathrm{vd}(\varphi), \mathrm{vd}(\psi)\} & \mathrm{vd}(\varepsilon\alpha) &=& \mathrm{vd}(\alpha) \\
\mathrm{vd}(\neg\varphi) &=& \mathrm{vd}(\varphi) & \mathrm{vd}([\varphi]\alpha) &=& \max\{\mathrm{vd}(\varphi), \mathrm{vd}(\alpha)\} \\
\mathrm{vd}(\langle\alpha\rangle) &=& \mathrm{vd}(\alpha) & \mathrm{vd}(\downarrow\alpha) &=& \max\{(0,0), \mathrm{vd}(\alpha) + (1,-1)\} \\
\mathrm{vd}(\langle\alpha \odot \beta\rangle) &=& \max\{\mathrm{vd}(\alpha), \mathrm{vd}(\beta)\} & \mathrm{vd}(\uparrow\alpha) &=& \max\{(0,0), \mathrm{vd}(\alpha) + (-1,1)\}
\end{array}$$

$$\begin{array}{rcl@{\qquad}rcl}
\mathrm{nd}(a) &=& 0 & \mathrm{nd}(\alpha\beta) &=& \max\{\mathrm{nd}(\alpha), \mathrm{nd}(\beta)\} \\
\mathrm{nd}(\varphi \wedge \psi) &=& \max\{\mathrm{nd}(\varphi), \mathrm{nd}(\psi)\} & \mathrm{nd}(\varepsilon) &=& 0 \\
\mathrm{nd}(\neg\varphi) &=& \mathrm{nd}(\varphi) & \mathrm{nd}([\varphi]) &=& 1 + \mathrm{nd}(\varphi) \\
\mathrm{nd}(\langle\alpha\rangle) &=& \mathrm{nd}(\alpha) & \mathrm{nd}(\downarrow) &=& 0 \\
\mathrm{nd}(\langle\alpha \odot \beta\rangle) &=& \max\{\mathrm{nd}(\alpha), \mathrm{nd}(\beta)\} & \mathrm{nd}(\uparrow) &=& 0
\end{array}$$

where, $a \in \mathbb{A}$, $\odot \in \{=, \neq\}$, the operations '+' and 'max' are performed component-wise, and α is any path expression or the empty string λ.

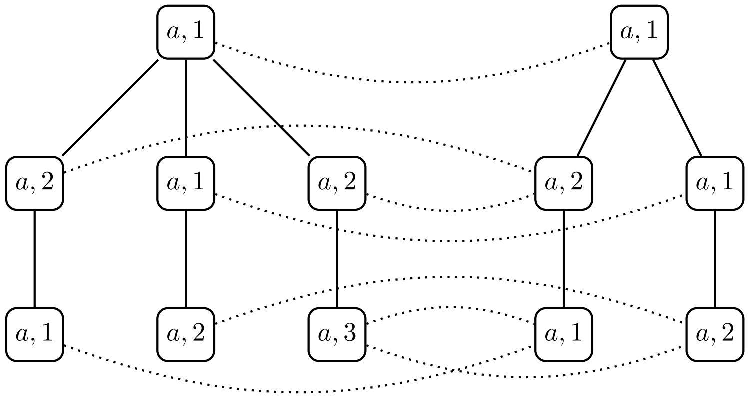

Figure 2: Two data trees and an XPath$_=(\downarrow)$-bisimulation between them (represented by dotted lines).

Let (r,s,k)-XPath$_=(\uparrow\downarrow)$ be the set of node expressions φ in XPath$_=(\uparrow\downarrow)$ with $\mathrm{vd}(\varphi) \leq (r,s)$ and $\mathrm{nd}(\varphi) \leq k$. Let $\mathcal{T}, u$ and $\mathcal{T}', u'$ be pointed data trees. We say that $\mathcal{T}, u$ and $\mathcal{T}', u'$ are (r,s,k)**-equivalent for XPath**$_=(\uparrow\downarrow)$ (notation: $\mathcal{T}, u \equiv^{\uparrow\downarrow}_{r,s,k} \mathcal{T}', u'$) if they satisfy the same node expressions of (r,s,k)-XPath$_=(\uparrow\downarrow)$.

2.4 Bisimulations

In [10] the notions of downward and vertical bisimulations are introduced. We reproduce them here, as they are key concepts for our results.

Let us start with the notions of bisimulation for the downward fragment of XPath$_=$. We say that $u \in T$ and $u' \in T'$ are **bisimilar for XPath**$_=(\downarrow)$ (or $\downarrow$**-bisimilar**; notation: $\mathcal{T}, u \leftrightarrow^{\downarrow} \mathcal{T}', u'$) iff there is a relation $Z \subseteq T \times T'$ such that uZu' and for all $x \in T$ and $x' \in T'$ we have

- **Harmony:** If xZx' then $label(x) = label(x')$.
- **Zig:** If xZx', $x \xrightarrow{n} v$ and $x \xrightarrow{m} w$ then there are $v', w' \in T'$ such that $x' \xrightarrow{n} v'$, $x' \xrightarrow{m} w'$ and
  1. $data(v) = data(w) \Leftrightarrow data(v') = data(w')$,
  2. $(\xrightarrow{i} v)\, Z\, (\xrightarrow{i} v')$ for all $0 \leq i < n$, and
  3. $(\xrightarrow{i} w)\, Z\, (\xrightarrow{i} w')$ for all $0 \leq i < m$.
- **Zag:** If xZx', $x' \xrightarrow{n} v'$ and $x' \xrightarrow{m} w'$ then there are $v, w \in T$ such that $x \xrightarrow{n} v$, $x \xrightarrow{m} w$ and items 1, 2 and 3 above are verified.

See Figure 2, taken from [10] for an example of a $\downarrow$-bisimulation. Notice that all pairs of leaves can be in a downward bisimulation relation as long as they have the same label.

There is also a notion of step-by-step bisimulation for the downward fragment. We say that $u \in T$ and $u' \in T'$ are ℓ**-bisimilar for XPath**$_=(\downarrow)$ (notation: $\mathcal{T}, u \leftrightarrow^{\downarrow}_{\ell} \mathcal{T}', u'$) if there is a family of relations $(Z_j)_{j \leq \ell}$ in $T \times T'$ such that $uZ_\ell u'$ and for all $j \leq \ell$, $x \in T$ and $x' \in T'$ we have

- **Harmony:** If xZ_jx' then $label(x) = label(x')$.
- **Zig:** If xZ_jx', $x \xrightarrow{n} v$ and $x \xrightarrow{m} w$ with $n, m \leq j$ then there are $v', w' \in T'$ such that $x' \xrightarrow{n} v'$, $x' \xrightarrow{m} w'$ and
  1. $data(v) = data(w) \Leftrightarrow data(v') = data(w')$,
  2. $(\xrightarrow{i} v)\, Z_{j-n+i}\, (\xrightarrow{i} v')$ for all $0 \leq i < n$, and

3. $(\xrightarrow{i} w)\, Z_{j-m+i}\, (\xrightarrow{i} w')$ for all $0 \leq i < m$.

- **Zag:** If xZ_jx', $x' \xrightarrow{n} v'$ and $x' \xrightarrow{m} w'$ with $n, m \leq j$ then there are $v, w \in T$ such that $x \xrightarrow{n} v$, $x \xrightarrow{m} w$ and items 1, 2 and 3 above are verified.

The following result of [10] establishes the connection between bisimulation and equivalence for the downward fragment:

Theorem 3. *1. $\mathcal{T}, u \leftrightarroweq^{\downarrow} \mathcal{T}', u'$ implies $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'$. The converse is not true in general, but it holds when $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching.*

2. $\mathcal{T}, u \leftrightarroweq^{\downarrow}_{\ell} \mathcal{T}', u'$ iff $\mathcal{T}, u \equiv^{\downarrow}_{\ell} \mathcal{T}', u'$.

Let us turn to bisimulation notions for the vertical fragment of XPath$_=$. We say that $u \in T$ and $u' \in T'$ are **bisimilar for XPath$_=(\uparrow\downarrow)$** (or $\uparrow\downarrow$**-bisimilar**; notation: $\mathcal{T}, u \leftrightarroweq^{\uparrow\downarrow} \mathcal{T}', u'$) iff there is a relation $Z \subseteq T \times T'$ such that uZu' and for all $x \in T$ and $x' \in T'$ we have

- **Harmony:** If xZx' then $label(x) = label(x')$,
- **Zig:** If xZx', $y \xrightarrow{n} x$ and $y \xrightarrow{m} z$ then there are $y', z' \in T'$ such that $y' \xrightarrow{n} x'$, $y' \xrightarrow{m} z'$, $data(z) = data(x) \Leftrightarrow data(z') = data(x')$, and zZz'.
- **Zag:** If xZx', $y' \xrightarrow{n} x'$ and $y' \xrightarrow{m} z'$ then there are $y, z \in T$ such that $y \xrightarrow{n} x$, $y \xrightarrow{m} z$, $data(z) = data(x) \Leftrightarrow data(z') = data(x')$, and zZz'.

The notion of step by step bisimulation for XPath$_=(\uparrow\downarrow)$ is as follows: We say that $u \in T$ and $u' \in T'$ are (r, s, k)**-bisimilar for XPath$_=(\uparrow\downarrow)$** (notation: $\mathcal{T}, u \leftrightarroweq^{\uparrow\downarrow}_{r,s,k} \mathcal{T}', u'$) if there is a family of relations $(Z^{\hat{k}}_{\hat{r},\hat{s}})_{\hat{r}+\hat{s} \leq r+s, \hat{k} \leq k}$ in $T \times T'$ such that $uZ^{k}_{r,s}u'$ and for all $\hat{r} + \hat{s} \leq r + s, \hat{k} \leq k$, $x \in T$ and $x' \in T'$ we have that the following conditions hold.

- **Harmony:** If $xZ^{\hat{k}}_{\hat{r},\hat{s}}x'$ then $label(x) = label(x')$.
- **Zig:** If $xZ^{\hat{k}}_{\hat{r},\hat{s}}x'$, $y \xrightarrow{n} x$ and $y \xrightarrow{m} z$ with $n \leq \hat{s}$ and $m \leq \hat{r} + n$ then there are $y', z' \in T'$ such that $y' \xrightarrow{n} x'$, $y' \xrightarrow{m} z'$, and the following hold
  1. $data(z) = data(x) \Leftrightarrow data(z') = data(x')$,
  2. if $\hat{k} > 0$, $zZ^{\hat{k}-1}_{\hat{r}',\hat{s}'}z'$ for $\hat{r}' = \hat{r} + n - m$, $\hat{s}' = \hat{s} - n + m$.
- **Zag:** If $xZ^{\hat{k}}_{\hat{r},\hat{s}}x'$, $y' \xrightarrow{n} x'$ and $y' \xrightarrow{m} z'$ with $n \leq \hat{s}$ and $m \leq \hat{r} + n$ then there are $y, z \in T$ such that $y \xrightarrow{n} x$, $y \xrightarrow{m} z$, and items (1) and (2) above are verified.

The following result of [10] establishes the connection between bisimulation and equivalence for the vertical fragment:

Theorem 4. *1. $\mathcal{T}, u \leftrightarroweq^{\uparrow\downarrow} \mathcal{T}', u'$ implies $\mathcal{T}, u \equiv^{\uparrow\downarrow} \mathcal{T}', u'$. The converse is not true in general, but it holds when $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching.*

2. $\mathcal{T}, u \leftrightarroweq^{\uparrow\downarrow}_{r,s,k\cdot(r+s+2)} \mathcal{T}', u'$ implies $\mathcal{T}, u \equiv^{\uparrow\downarrow}_{r,s,k} \mathcal{T}', u'$.

3. $\mathcal{T}, u \equiv^{\uparrow\downarrow}_{r,s,k} \mathcal{T}', u'$ implies $\mathcal{T}, u \leftrightarroweq^{\uparrow\downarrow}_{r,s,k} \mathcal{T}', u'$.

2.5 Connection to first order logic

We fix the signature σ with binary relations $\rightsquigarrow$ and $\sim$, and a unary predicate P_a for each $a \in \mathbb{A}$. Any data tree $\mathcal{T}$ can be seen as a first-order σ-structure, where

$$\begin{aligned}
\rightsquigarrow^{\mathcal{T}} &= \{(x,y) \in T^2 \mid x \to y \text{ in } \mathcal{T}\}; \\
\sim^{\mathcal{T}} &= \{(x,y) \in T^2 \mid \mathit{data}(x) = \mathit{data}(y)\}; \\
P_a^{\mathcal{T}} &= \{x \in T \mid \mathit{label}(x) = a\}.
\end{aligned}$$

If $\varphi(x)$ is a first-order formula with a free variable x, we use $\mathcal{T} \models \varphi[a]$, as usual, to denote that φ is true in $\mathcal{T}$ under the valuation which maps x to $a \in T$. In [10] is shown a truth preserving translation Tr_x mapping XPath$_=(\uparrow\downarrow)$-node expressions into first-order σ-formulas with one free variable x. The following translation is slightly more clear than the one described in [10], and it also considers translation of path expressions (resulting in first-order formulas with two variables):

$$\begin{aligned}
\mathrm{Tr}_x(a) &= P_a(x) && (a \in \mathbb{A}) \\
\mathrm{Tr}_x(\varphi \dagger \psi) &= \mathrm{Tr}_x(\varphi) \dagger \mathrm{Tr}_x(\psi) && (\dagger \in \{\wedge, \vee\}) \\
\mathrm{Tr}_x(\neg\varphi) &= \neg\mathrm{Tr}_x(\varphi) \\
\mathrm{Tr}_x(\langle\alpha\rangle) &= (\exists y)\mathrm{Tr}_{x,y}(\alpha) && (y \text{ a fresh variable}) \\
\mathrm{Tr}_x(\langle\alpha = \beta\rangle) &= (\exists y)(\exists z)\big(y \sim z \wedge \mathrm{Tr}_{x,y}(\alpha) \wedge \mathrm{Tr}_{x,z}(\beta)\big) && (y, z \text{ fresh variables}) \\
\mathrm{Tr}_x(\langle\alpha \neq \beta\rangle) &= (\exists y)(\exists z)\big(y \not\sim z \wedge \mathrm{Tr}_{x,y}(\alpha) \wedge \mathrm{Tr}_{x,z}(\beta)\big) && (y, z \text{ fresh variables}) \\
\mathrm{Tr}_{x,y}(\epsilon) &= (x = y) \\
\mathrm{Tr}_{x,y}(\downarrow) &= (x \rightsquigarrow y) \\
\mathrm{Tr}_{x,y}(\uparrow) &= (y \rightsquigarrow x) \\
\mathrm{Tr}_{x,y}(\alpha\beta) &= (\exists z)\big(\mathrm{Tr}_{x,z}(\alpha) \wedge \mathrm{Tr}_{z,y}(\beta)\big) && (z \text{ a fresh variable}) \\
\mathrm{Tr}_{x,y}(\alpha \cup \beta) &= \mathrm{Tr}_{x,y}(\alpha) \vee \mathrm{Tr}_{x,y}(\beta) \\
\mathrm{Tr}_{x,y}([\varphi]) &= \mathrm{Tr}_x(\varphi) \wedge (x = y).
\end{aligned}$$

It is easy to see that the above translation is truth preserving:

Proposition 5. *If φ is a node expression of XPath$_=(\uparrow\downarrow)$ then $\mathcal{T}, u \models \varphi$ iff $\mathcal{T} \models \mathrm{Tr}_x(\varphi)[u]$. If α is a path expression of XPath$_=(\uparrow\downarrow)$ then $\mathcal{T}, u, v \models \alpha$ iff $\mathcal{T} \models \mathrm{Tr}_{x,y}(\alpha)[u, v]$.*

3 Definability via node expressions

3.1 Saturation

In [10] it is shown that the reverse implication of Theorem 3 holds over finitely branching trees. However, it does not hold in general. In this section we introduce notions of saturation for the downward and vertical fragments of XPath, and show that the reverse implication of Theorem 3 is true over saturated data trees.

Saturation for the downward fragment. Let $\langle\Sigma_1, \dots, \Sigma_n\rangle$ and $\langle\Gamma_1, \dots, \Gamma_m\rangle$ be tuples of sets of XPath$_=(\downarrow)$-node expressions. Given a data tree $\mathcal{T}$ and $u \in T$, we say that $\langle\Sigma_1, \dots, \Sigma_n\rangle$ and $\langle\Gamma_1, \dots, \Gamma_m\rangle$ are **$=^{\downarrow}_{n,m}$-satisfiable** [resp. **$\neq^{\downarrow}_{n,m}$-satisfiable**] at $\mathcal{T}, u$ if there exist $v_0 \to v_1 \to \cdots \to v_n \in T$ and $w_0 \to w_1 \to \cdots \to w_m \in T$ such that $u = v_0 = w_0$ and

1. for all $i \in \{1, \dots, n\}$, $\mathcal{T}, v_i \models \Sigma_i$;

2. for all $j \in \{1, \dots, m\}$, $\mathcal{T}, w_j \models \Gamma_j$; and

3. $\mathit{data}(v_n) = \mathit{data}(w_m)$ [resp. $\mathit{data}(v_n) \neq \mathit{data}(w_m)$].

We say that $\langle \Sigma_1, \dots, \Sigma_n \rangle$ and $\langle \Gamma_1, \dots, \Gamma_m \rangle$ are $=^{\downarrow}_{n,m}$**-finitely satisfiable** [resp. $\neq^{\downarrow}_{n,m}$**-finitely satisfiable**] at $\mathcal{T}, u$ if for every finite $\Sigma'_i \subseteq \Sigma_i$ and finite $\Gamma'_j \subseteq \Gamma_j$, we have that $\langle \Sigma'_1, \dots, \Sigma'_n \rangle$ and $\langle \Gamma'_1, \dots, \Gamma'_m \rangle$ are $=^{\downarrow}_{n,m}$-satisfiable [resp. $\neq^{\downarrow}_{n,m}$-satisfiable] at $\mathcal{T}, u$.

Definition 6. We say that a data tree $\mathcal{T}$ is $\downarrow$**-saturated** if for every $n, m \in \mathbb{N}$, every pair of tuples $\langle \Sigma_1, \dots, \Sigma_n \rangle$ and $\langle \Gamma_1, \dots, \Gamma_m \rangle$ of sets of XPath$_=(\downarrow)$-node expressions, every $u \in T$, and $\star \in \{=, \neq\}$, the following is true:

> if $\langle \Sigma_1, \dots, \Sigma_n \rangle$ and $\langle \Gamma_1, \dots, \Gamma_m \rangle$ are $\star^{\downarrow}_{n,m}$-finitely satisfiable at $\mathcal{T}, u$ then $\langle \Sigma_1, \dots, \Sigma_n \rangle$ and $\langle \Gamma_1, \dots, \Gamma_m \rangle$ are $\star^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}, u$.

Proposition 7. *Any finitely branching data tree is $\downarrow$-saturated.*

Proof. Suppose by contradiction that there is $u \in T$ and tuples $\langle \Sigma_1, \dots, \Sigma_n \rangle$ and $\langle \Gamma_1, \dots, \Gamma_m \rangle$ of sets of XPath$_=(\downarrow)$-node expressions which are finitely $=^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}, u$ but not $=^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}, u$ (the case for $\mathcal{T}$ being $\neq^{\downarrow}_{n,m}$-satisfiable is analogous). Let

$$P = \{(v, w) \in T^2 \mid u \xrightarrow{n} v \wedge u \xrightarrow{m} w \wedge \mathit{data}(v) = \mathit{data}(w)\}.$$

Observe that P is finite because $\mathcal{T}$ is finitely branching. It is clear that if $(v, w) \in P$, so that $u = v_0 \to v_1 \to \dots \to v_n = v \in T$, and $u = w_0 \to w_1 \to \dots \to w_m = w \in T$ then either

1. there is $i \in \{1, \dots, n\}$ such that $\mathcal{T}, v_i \not\models \Sigma_i$, or
2. there is $j \in \{1, \dots, m\}$ such that $\mathcal{T}, w_j \not\models \Gamma_j$.

We will define sets $(\Sigma_{i,v,w})_{1 \le i \le n}$ and $(\Gamma_{j,v,w})_{1 \le j \le m}$, each one of them with at most one element, as follows: If case 1 holds, assume i_0 is the least such number and define $\Sigma_{i_0,v,w}$ as $\{\rho\}$ for some node expression $\rho \in \Sigma_{i_0}$ such that $\mathcal{T}, v_{i_0} \not\models \rho$, define $\Sigma_{i,v,w} = \emptyset$ for any $i \in \{1, \dots, n\} \setminus \{i_0\}$, and define $\Gamma_{j,v,w} = \emptyset$ for any $j \in \{1, \dots, m\}$. If case 1 does not hold then case 2 holds, so assume j_0 is the least such number and define $\Gamma_{j_0,v,w}$ as $\{\rho\}$ for some node expression $\rho \in \Gamma_{j_0}$ such that $\mathcal{T}, w_{j_0} \not\models \rho$, define $\Gamma_{j,v,w} = \emptyset$ for any $j \in \{1, \dots, m\} \setminus \{j_0\}$, and define $\Sigma_{i,v,w} = \emptyset$ for any $i \in \{1, \dots, n\}$. Finally, define the finite sets $\Sigma'_i = \bigcup_{(v,w) \in P} \Sigma_{i,v,w}$ and $\Gamma'_j = \bigcup_{(v,w) \in P} \Gamma_{j,v,w}$. By construction, we have $\Sigma'_i \subseteq \Sigma_i$, $\Gamma'_j \subseteq \Gamma_j$ and $\langle \Sigma'_1, \dots, \Sigma'_n \rangle$ and $\langle \Gamma'_1, \dots, \Gamma'_m \rangle$ are not $=^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}, u$ which is a contradiction. $\square$

Proposition 8. *Let $\mathcal{T}$ and $\mathcal{T}'$ be $\downarrow$-saturated data trees, and let $u \in T$ and $u' \in T'$. If $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'$, then $\mathcal{T}, u \leftrightarroweq^{\downarrow} \mathcal{T}', u'$.*

Proof. We show that Z, defined by xZx' iff $\mathcal{T}, x \equiv^{\downarrow} \mathcal{T}', x'$ is a $\downarrow$-bisimulation between $\mathcal{T}, u$ and $\mathcal{T}', u'$. Clearly uZu', and **Harmony** holds. We only need to show that **Zig** and **Zag** are satisfied. We see only **Zig**, as **Zag** is analogous.

Suppose xZx', $x = v_0 \to v_1 \to \dots \to v_n$ and $x = w_0 \to w_1 \to \dots \to w_m$ are paths on $\mathcal{T}$, and $\mathit{data}(v_n) = \mathit{data}(w_m)$ (the case $\mathit{data}(v_n) \neq \mathit{data}(w_m)$ is shown analogously). For $i \in \{1, \dots, n\}$, let $\Sigma_i = \mathrm{Th}_{\downarrow}(\mathcal{T}, v_i)$, and for $j \in \{1, \dots, m\}$, let $\Gamma_j = \mathrm{Th}_{\downarrow}(\mathcal{T}, w_j)$. Furthermore, let Σ'_i be a finite subset of Σ_i, and let Γ'_j be a finite subset of Γ_j. Define

$$\varphi = \langle \downarrow [\wedge \Sigma'_1] \downarrow \dots \downarrow [\wedge \Sigma'_n] = \downarrow [\wedge \Gamma'_1] \downarrow \dots \downarrow [\wedge \Gamma'_m] \rangle.$$

It is clear that $\mathcal{T}, x \models \varphi$, and since by definition of Z we have $\mathcal{T}, x \equiv^{\downarrow} \mathcal{T}', x'$, we conclude that $\mathcal{T}', x' \models \varphi$. Hence $\langle \Sigma'_1, \dots, \Sigma'_n \rangle$ and $\langle \Gamma'_1, \dots, \Gamma'_m \rangle$ are $=^{\downarrow}_{n,m}$-satisfiable at x'. This holds for *any* finite sets $\Sigma'_i \subseteq \Sigma_i$ and $\Gamma'_j \subseteq \Gamma_j$, and so $\langle \Sigma_1, \dots, \Sigma_n \rangle$ and $\langle \Gamma_1, \dots, \Gamma_m \rangle$ are $=^{\downarrow}_{n,m}$-finitely satisfiable at x'. Since $\mathcal{T}'$ is $\downarrow$-saturated, $\langle \Sigma_1, \dots, \Sigma_n \rangle$ and $\langle \Gamma_1, \dots, \Gamma_m \rangle$ are $=^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}', x'$, so there are paths $x' = v'_0 \to v'_1 \to \dots \to v'_n$ and $x' = w'_0 \to w'_1 \to \dots \to w'_m$ on $\mathcal{T}'$ such that

i. $\mathit{data}(v'_n) = \mathit{data}(w'_m)$.

ii. For all $1 \le i \le n$, $\mathcal{T}', v'_i \models \mathrm{Th}_\downarrow(\mathcal{T}, v_i)$. This implies $\mathcal{T}, v_i \equiv^\downarrow \mathcal{T}', v'_i$: suppose by the way of contradiction that $\mathcal{T}', v'_i \models \varphi$ but $\mathcal{T}, v_i \not\models \varphi$. Then, $\mathcal{T}, v_i \models \neg\varphi$, and thus $\mathcal{T}', v'_i \models \neg\varphi$, a contradiction.

iii. For all $1 \le j \le m$, $\mathcal{T}', w'_j \models \mathrm{Th}_\downarrow(\mathcal{T}, w_j)$, i.e $\mathcal{T}, w_j \equiv^\downarrow \mathcal{T}', w'_j$.

By the definition of Z, conditions i, ii and iii above imply items 1, 2 and 3 of the **Zig** clause for $\downarrow$-bisimulation. $\square$

Saturation for the vertical fragment. Given a data tree $\mathcal{T}$ and $u \in T$, we say that the set of XPath$_=(\uparrow\downarrow)$-node expressions Γ is $=^{\uparrow\downarrow}_{n,m}$**-satisfiable** [resp. $\neq^{\uparrow\downarrow}_{n,m}$**-satisfiable**] at $\mathcal{T}, u$ if there exist $v, w \in T$ such that $v \xrightarrow{n} u$, $v \xrightarrow{m} w$, $w \models \Gamma$ and $data(u) = data(w)$ [resp. $data(u) \neq data(w)$]. We say that Γ is $=^{\uparrow\downarrow}_{n,m}$**-finitely satisfiable** [resp. $\neq^{\uparrow\downarrow}_{n,m}$**-finitely satisfiable**] at $\mathcal{T}, u$ if for every finite $\Gamma' \subseteq \Gamma$, we have that Γ' is $=^{\uparrow\downarrow}_{n,m}$-satisfiable [resp. $=^{\uparrow\downarrow}_{n,m}$-satisfiable] at $\mathcal{T}, u$.

Definition 9. We say that a data tree $\mathcal{T}$ is $\uparrow\downarrow$**-saturated** if for every set of XPath$_=(\uparrow\downarrow)$-node expressions Γ, every $u \in T$, every $n, m \in \mathbb{N}$, and $\star \in \{=, \neq\}$, the following is true:

if Γ is $\star^{\uparrow\downarrow}_{n,m}$-finitely satisfiable at $\mathcal{T}, u$ then Γ is $\star^{\uparrow\downarrow}_{n,m}$-satisfiable at $\mathcal{T}, u$.

Proposition 10. *Let $\mathcal{T}$ and $\mathcal{T}'$ be $\uparrow\downarrow$-saturated data trees, and let $u \in T$ and $u' \in T'$. If $\mathcal{T}, u \equiv^{\uparrow\downarrow} \mathcal{T}', u'$, then $\mathcal{T}, u \leftrightarrow^{\uparrow\downarrow} \mathcal{T}', u'$.*

Proof. We show that $Z \subseteq T \times T'$, defined by xZx' iff $\mathcal{T}, x \equiv^{\uparrow\downarrow} \mathcal{T}', x'$ is a $\uparrow\downarrow$-bisimulation between $\mathcal{T}, u$ and $\mathcal{T}', u'$. Clearly uZu', and **Harmony** also holds, so we only need to show that **Zig** and **Zag** are satisfied. We see only **Zig**, as **Zag** is analogous.

Suppose xZx', $y \xrightarrow{n} x$ and $y \xrightarrow{m} z$ are in $\mathcal{T}$, and $data(x) = data(z)$ (the case $data(x) \neq data(z)$ can be shown analogously). Let $\Gamma = \mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, z)$, and let Γ' be a finite subset of Γ. Define

$$\varphi = \langle \varepsilon = \uparrow^n \downarrow^m [\wedge\Gamma'] \rangle.$$

It is clear that $\mathcal{T}, x \models \varphi$, and since by definition of Z we have $\mathcal{T}, x \equiv^{\uparrow\downarrow} \mathcal{T}', x'$, we conclude that $\mathcal{T}', x' \models \varphi$. Hence Γ' is $=^{\uparrow\downarrow}_{n,m}$-satisfiable at x'. This holds for *any* finite set $\Gamma' \subseteq \Gamma$, and so Γ is $=^{\uparrow\downarrow}_{n,m}$-finitely satisfiable at x'. Since $\mathcal{T}'$ is $\uparrow\downarrow$-saturated, Γ is $=^{\uparrow\downarrow}_{n,m}$-satisfiable at x', and thus there are $y' \xrightarrow{n} x'$ and $y' \xrightarrow{m} z'$ on $\mathcal{T}'$ such that $data(x') = data(z')$ and $\mathcal{T}', z' \models \mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, z)$, i.e $\mathcal{T}, z \equiv^{\uparrow\downarrow} \mathcal{T}', z'$. By the definition of Z, we have zZz' and hence the **Zig** clause for $\uparrow\downarrow$-bisimulation is verified. $\square$

3.2 Weak Data Trees and Quasi-ultraproducts

For reasons that will become clearer later on, we will need to work with σ-structures which are slightly more general than data trees.

Definition 11. A σ-structure $\mathcal{T}$ is a **weak data tree** if $\sim$ is an equivalence relation; there is exactly one node r with no u such that $u \rightsquigarrow r$ (r is called *root* of $\mathcal{T}$); for all nodes $x \neq r$ there is exactly one y such that $y \rightsquigarrow x$; and for each $n \ge 0$ the relation $\rightsquigarrow$ has no cycles of length n.

Observe that a weak data tree need not be connected, and that the class of weak data trees is elementary, i.e. definable by a set of first-order σ-sentences (with equality). For a weak data tree $\mathcal{T}$ and $u \in T$, let $\mathcal{T}|u$ denote the substructure of $\mathcal{T}$ induced by $\{v \in T \mid u \rightsquigarrow^* v\}$. Observe that in this case $\mathcal{T}|u$ is a data tree.

The following proposition shows the 'local' aspect of XPath$_=(\downarrow)$ and XPath$_=(\uparrow\downarrow)$. It is stated in terms of first-order because models are weak data trees.

Proposition 12. *Let $\mathcal{T}$ be a weak data tree and let $r \rightsquigarrow^* u$ in $\mathcal{T}$.*

1. *If φ is a XPath$_=(\downarrow)$-node expression then $\mathcal{T} \models \mathrm{Tr}_x(\varphi)[u]$ iff $\mathcal{T}|r \models \mathrm{Tr}_x(\varphi)[u]$.*

2. *If r is the root of $\mathcal{T}$ and $\varphi \in XPath_=(\uparrow\downarrow)$ then $\mathcal{T} \models \mathrm{Tr}_x(\varphi)[u]$ iff $\mathcal{T}|r \models \mathrm{Tr}_x(\varphi)[u]$.*

Observe that the condition of r being the root in the second item is needed. Suppose for example we are on the data tree with only 2 nodes, the root r and its child u, with same data value. Consider now $\varphi = \langle \varepsilon = \uparrow \rangle$. Clearly $\mathcal{T} \models \mathrm{Tr}_x(\varphi)[u]$, but $\mathcal{T}|u \not\models \mathrm{Tr}_x(\varphi)[u]$.

If $\mathcal{M}$ is a first-order σ-structure and $A \subseteq M$, we denote by σ_A the language obtained by adding to σ constant symbols for each $a \in A$. $\mathcal{M}$ can be seen as a σ_A structure by interpreting the new symbols in the obvious way. Let $\mathrm{Th}_A(\mathcal{M})$ be the set of all σ_A-sentences true in $\mathcal{M}$. Let κ be a cardinal. We recall the definition of κ-saturated first-order structures:

Definition 13. We say that the σ-structure $\mathcal{M}$ is κ-saturated if for all $A \subseteq M$ and all n, if $|A| < \kappa$ and $\Gamma(x_1, \dots, x_n)$ is a set of σ_A-formulas with free variables among $x_1, \dots, x_n$ such that $\Gamma(x_1, \dots, x_n) \cup \mathrm{Th}_A(\mathcal{M})$ is satisfiable, then $\Gamma(x_1, \dots, x_n)$ is realized in $\mathcal{M}$.

We now show that 2-saturated data trees are already both downward and vertical saturated. For technical reasons we state these results in the more general setting of weak data trees.

Proposition 14. *Let $\mathcal{T}$ be a 2-saturated weak data tree and $r \in T$.*

1. *$\mathcal{T}|r$ is a $\downarrow$-saturated data tree.*
2. *If r is the root of T then $\mathcal{T}|r$ is a $\uparrow\downarrow$-saturated data tree.*

Proof. Let $\mathcal{T}' = \mathcal{T}|r$ and let $u \in T'$. For item 1, let $\langle \Sigma_1, \dots, \Sigma_n \rangle$ and $\langle \Gamma_1, \dots, \Gamma_m \rangle$ be tuples of sets of XPath$_=(\downarrow)$-node expressions. Suppose $\langle \Sigma_1, \dots, \Sigma_n \rangle$ and $\langle \Gamma_1, \dots, \Gamma_m \rangle$ are $=^{\downarrow}_{n,m}$-finitely satisfiable at $\mathcal{T}', u$ (the case for $\neq^{\downarrow}_{n,m}$-finitely satisfiable is analogous). We show that $\langle \Sigma_1, \dots, \Sigma_n \rangle$ and $\langle \Gamma_1, \dots, \Gamma_m \rangle$ are $=^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}', u$. Consider the following first-order $\sigma_{\{u\}}$-formula with free variables $\bar{x} = x_1, \dots, x_n$ and $\bar{y} = y_1, \dots, y_m$:

$$\varphi(\bar{x}, \bar{y}) = u \rightsquigarrow x_1 \wedge \bigwedge_{i=1}^{n-1} x_i \rightsquigarrow x_{i+1} \wedge u \rightsquigarrow y_1 \wedge \bigwedge_{j=1}^{m-1} y_j \rightsquigarrow y_{j+1} \wedge x_n \sim y_m.$$

Define the following set of first-order $\sigma_{\{u\}}$-formulas:

$$\Delta(\bar{x}, \bar{y}) = \{\varphi(\bar{x}, \bar{y})\} \cup \bigcup_{i=1}^{n} \mathrm{Tr}_{x_i}(\Sigma_i) \cup \bigcup_{j=1}^{m} \mathrm{Tr}_{y_j}(\Gamma_j).$$

Let $\Delta'(\bar{x}, \bar{y})$ be a finite subset of $\Delta(\bar{x}, \bar{y})$. Since $\langle \Sigma_1, \dots, \Sigma_n \rangle$ and $\langle \Gamma_1, \dots, \Gamma_m \rangle$ are $=^{\downarrow}_{n,m}$-finitely satisfiable at $\mathcal{T}', u$, then $\Delta'(\bar{x}, \bar{y})$ is satisfiable and, by item 1 of Proposition 12, consistent with $\mathrm{Th}_{\{u\}}(\mathcal{T})$. By compactness, $\Delta(\bar{x}, \bar{y})$ is satisfiable and consistent with $\mathrm{Th}_{\{u\}}(\mathcal{T})$. By 2-saturation, we conclude that $\Delta(\bar{x}, \bar{y})$ is realizable in $\mathcal{T}$, say at $\bar{v} = v_1, \dots, v_n$ and $\bar{w} = w_1, \dots, w_m$. Thus we have:

i. $u \rightsquigarrow v_1 \rightsquigarrow \cdots \rightsquigarrow v_n$ and $u \rightsquigarrow w_1 \rightsquigarrow \cdots \rightsquigarrow w_m$ in $\mathcal{T}$, and hence in $\mathcal{T}'$;

ii. for all $i \in \{1, \dots, n\}$, $\mathcal{T} \models \mathrm{Tr}_{x_i}(\Sigma_i)[v_i]$, and for all $j \in \{1, \dots, m\}$, $\mathcal{T} \models \mathrm{Tr}_{y_j}(\Gamma_j)[w_j]$; by item 1 of Proposition 12 this implies that $\mathcal{T}' \models \mathrm{Tr}_{x_i}(\Sigma_i)[v_i]$ and $\mathcal{T}' \models \mathrm{Tr}_{y_j}(\Gamma_j)[w_j]$;

iii. $v_n \sim w_m$ in $\mathcal{T}$, and hence in $\mathcal{T}'$.

Since Tr is truth preserving, we have that for all $i \in \{1, \dots, n\}$, $\mathcal{T}', v_i \models \Sigma_i$, and for all $j \in \{1, \dots, m\}$, $\mathcal{T}', w_i \models \Gamma_i$. Together with i and iii we conclude that $\langle \Sigma_1, \dots, \Sigma_n \rangle$ and $\langle \Gamma_1, \dots, \Gamma_m \rangle$ are $=^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}', u$.

For item 2, let Γ be a set of XPath$_=(\uparrow\downarrow)$-node expressions. Suppose Γ is $=^{\uparrow\downarrow}_{n,m}$-finitely satisfiable at $\mathcal{T}', u$ (the case for $\neq^{\uparrow\downarrow}_{n,m}$-finitely satisfiable is analogous). We show that Γ are $=^{\uparrow\downarrow}_{n,m}$-satisfiable at $\mathcal{T}', u$.

Consider the following first-order $\sigma_{\{u\}}$-formula with free variable y:

$$\begin{aligned}\varphi(y) \quad = \quad & (\exists x_0 \ldots \exists x_n)(\exists y_0 \ldots \exists y_m)[x_n = u \wedge y = y_m \wedge x_0 = y_0 \wedge \\ & \bigwedge_{i=0}^{n-1} x_i \rightsquigarrow x_{i+1} \wedge \bigwedge_{j=0}^{m-1} y_j \rightsquigarrow y_{j+1} \wedge x_n \sim y_m].\end{aligned}$$

Define the following set of first-order $\sigma_{\{u\}}$-formulas: $\Delta(y) = \{\varphi(y)\} \cup \mathrm{Tr}_y(\Gamma)$. Let $\Delta'(y)$ be a finite subset of $\Delta(y)$. Since Γ is $=^{\uparrow\downarrow}_{n,m}$-finitely satisfiable at $\mathcal{T}', u$, then $\Delta'(y)$ is satisfiable and, by item 2 of Proposition 12, consistent with $\mathrm{Th}_{\{u\}}(\mathcal{T})$. By compactness, $\Delta(y)$ is satisfiable and consistent with $\mathrm{Th}_{\{u\}}(\mathcal{T})$. By 2-saturation, we conclude that $\Delta(y)$ is realizable in $\mathcal{T}$, say at w. Thus we have:

iv. There is $v \in T$ such that $v \xrightarrow{n} u$ and $v \xrightarrow{m} w$ in $\mathcal{T}$ and hence in $\mathcal{T}'$.

v. $\mathcal{T} \models \mathrm{Tr}_y(\Gamma)[w]$; by item 2 of Proposition 12 this implies that $\mathcal{T}' \models \mathrm{Tr}_y(\Gamma)[w]$;

vi. $u \sim w$ in $\mathcal{T}$, and hence in $\mathcal{T}'$.

Since Tr is truth preserving, we have that $\mathcal{T}', w \models \Gamma$. Together with iv and vi we conclude that Γ is $=^{\uparrow\downarrow}_{n,m}$-satisfiable at $\mathcal{T}', u$. □

In what follows, we introduce the notion of quasi-ultraproduct, a variant of the usual notion of first-order model theory, which will be needed for the definability theorems.

Let $I \neq \emptyset$, let U be an ultrafilter over I and let $(\mathcal{T}_i)_{i\in I}$ be a family of data trees. As usual, we denote with $\prod_U \mathcal{T}_i$ the ultraproduct of $(\mathcal{T}_i)_{i\in I}$ modulo U. Observe that by the fundamental theorem of ultraproducts (see e.g. [4, Thm. 4.1.9]), $\prod_U \mathcal{T}_i$ is a weak data tree σ-structure —though it may not be a data tree because it may be disconnected, as it is shown next:

Example 15. For $i \in \mathbb{N}$, let $\mathcal{T}_i$ as any data tree of height at least i, and let u_i as any node of T_i at distance i from the root of $\mathcal{T}_i$. Let $\varphi_n(x)$ be the first-order property "x is at distance at least n from the root". It is clear that $\mathcal{T}_m \models \varphi_n[u_m]$ for every $m \geq n$. Let u^* be the ultralimit of $(u_i)_{i\in I}$ modulo U. Since $\{m \mid m \geq n\} \in U$ for any non-principal U, we conclude that $\prod_U \mathcal{T}_i \models \varphi_n[u^*]$ for every n, and so u^* is disconnected from the root of $\prod_U \mathcal{T}_i$.

Let $(\mathcal{T}_i, u_i)_{i\in I}$ be a family of pointed data trees. The ultraproduct of such *pointed* data trees is defined, as usual, by $(\prod_U \mathcal{T}_i, u^*)$, where u^* is the ultralimit of $(u_i)_{i\in I}$ modulo U.

Definition 16. Suppose $(\mathcal{T}_i, u_i)_{i\in I}$ is a family of pointed data trees, r_i is the root of $\mathcal{T}_i$, U is an ultrafilter over I, $\mathcal{T}^* = \prod_U \mathcal{T}_i$, and u^* and r^* are the ultralimits of $(u_i)_{i\in I}$ and $(r_i)_{i\in I}$ modulo U respectively.

1. The **$\downarrow$-quasi ultraproduct** of $(\mathcal{T}_i, u_i)_{i\in I}$ modulo U is the pointed data tree $(\mathcal{T}^*|u^*, u^*)$.
2. The **$\uparrow\downarrow$-quasi ultraproduct** of $(\mathcal{T}_i, u_i)_{i\in I}$ modulo U is the pair $(\mathcal{T}^*|r^*, u^*)$.

Observe that both $\mathcal{T}^*|u^*$ and $\mathcal{T}^*|r^*$ are data trees. However, while u^* is in the domain of the former, it may not be in the domain of the latter (cf. Example 15). Hence, in general, pointed data trees are not closed under $\uparrow\downarrow$-quasi ultraproduct. Let $k \geq 0$, let $\mathcal{T}$ be a data tree and let $u \in \mathcal{T}$. We say that $\mathcal{T}, u$ is a **k-bounded pointed data tree** if u is at distance at most k from the root of $\mathcal{T}$. In particular, if r is the root of $\mathcal{T}$ (as it is often the case) then $\mathcal{T}, r$ is a 0-bounded pointed data tree. The following proposition states that k-bounded data trees are closed under $\uparrow\downarrow$-quasi ultraproducts.

Proposition 17. *Let $(\mathcal{T}_i, u_i)_{i\in I}$ be a family of k-bounded pointed data trees. Then the $\uparrow\downarrow$-quasi ultraproduct of $(\mathcal{T}_i, u_i)_{i\in I}$ is a k-bounded pointed data tree.*

Proof. Let $(\mathcal{T}^{\uparrow\downarrow}, u^*)$ be the $\uparrow\downarrow$-quasi ultraproduct of $(\mathcal{T}_i, u_i)_{i\in I}$ modulo U. By definition it is clear that $\mathcal{T}^{\uparrow\downarrow}$ is a data tree. To see that $u^* \in T^{\uparrow\downarrow}$, let

$$\begin{aligned}\varphi(x) \quad &= \quad (\exists r)\ [\neg(\exists y) y \rightsquigarrow r \wedge [r = x \vee r \rightsquigarrow x \vee \\ &\qquad \bigvee_{1\le i<k} (\exists z_1 \ldots \exists z_i)[r \rightsquigarrow z_1 \wedge z_{i-1} \rightsquigarrow x \wedge \bigwedge_{1\le j<i-1} z_j \rightsquigarrow z_{j+1}]]],\end{aligned}$$

which is a first-order formula for "r is the root and x is at distance at most k from r". Since for every $i \in I$ we have $\mathcal{T}_i \models \varphi[u_i]$, we conclude that $\mathcal{T}^{\uparrow\downarrow} \models \varphi[u^*]$ and hence u^* is at distance at most k from the root of $\mathcal{T}^{\uparrow\downarrow}$. □

As a particular case one has the notion of **$\downarrow$-quasi ultrapower** and **$\uparrow\downarrow$-quasi ultrapower** of a family of pointed data trees. Observe that if $(\mathcal{T}^{\uparrow\downarrow}, u^*)$ is the $\uparrow\downarrow$-quasi ultrapower of $(\mathcal{T}, u)_{i\in I}$ then u^* belongs to the domain of $\mathcal{T}^{\uparrow\downarrow}$ and so $(\mathcal{T}^{\uparrow\downarrow}, u^*)$ is a pointed data tree.

3.3 Definability

In this section we state the main results. If K is a class of pointed data trees, we denote its complement by $\overline{K}$. We begin with the downward fragment.

Lemma 18. *Let $(\mathcal{T}, u)$ and $(\mathcal{T}', u')$ be two pointed data trees such that $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'$. Then there exist $\downarrow$-quasi ultrapowers $(\mathcal{T}^{\downarrow}, u^*)$ and $(\mathcal{T}'^{\downarrow}, u'^*)$ of $(\mathcal{T}, u)$ and $(\mathcal{T}', u')$ respectively such that $(\mathcal{T}^{\downarrow}, u^*) \underline{\leftrightarrow}^{\downarrow} (\mathcal{T}'^{\downarrow}, u'^*)$*

Proof. It is known that there is a suitable ultrafilter U such that $\prod_U \mathcal{T}$ and $\prod_U \mathcal{T}'$ are ω-saturated (see e.g. [2, Lemma 2.7.3]). By item 1 Proposition 14, $\mathcal{T}^{\downarrow} = (\prod_U \mathcal{T})|u^*$ and $\mathcal{T}^{\downarrow} = (\prod_U \mathcal{T}')|u'^*$ are $\downarrow$-saturated data trees. By hypothesis $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'$, and hence $\mathcal{T}^{\downarrow}, u^* \equiv^{\downarrow} \mathcal{T}'^{\downarrow}, u'^*$. Finally, by Proposition 8, $\mathcal{T}^{\downarrow}, u^* \underline{\leftrightarrow}^{\downarrow} \mathcal{T}'^{\downarrow}, u'^*$. □

Lemma 19. *Let K be a class of pointed data trees and let Σ be a set of XPath$_=(\downarrow)$-node expressions finitely satisfiable in K. Then Σ is satisfiable in some $\downarrow$-quasi ultraproduct of pointed data trees in K.*

Proof. Let $I = \{\Sigma_0 \subset \Sigma \mid \Sigma_0 \text{ is finite}\}$ and for each $\varphi \in \Sigma$, let $\hat{\varphi} = \{i \in I \mid \varphi \in i\}$. Then the set $E = \{\hat{\varphi} \mid \varphi \in \Sigma\}$ has the finite intersection property: $\{\varphi_1, \ldots, \varphi_n\} \in \hat{\varphi}_1 \cap \cdots \cap \hat{\varphi}_n$. By the Ultrafilter Theorem (see [4, Proposition 4.1.3]) E can be extended to an ultrafilter U over I.

Since Σ is finitely satisfiable in K, for each $i \in I$ there is $(\mathcal{T}_i, u_i) \in K$ such that $\mathcal{T}_i, u_i \models i$. Let $(\mathcal{T}^{\downarrow}, u^*)$ be the $\downarrow$-quasi ultraproduct of $(\mathcal{T}_i, u_i)_{i\in I}$ modulo U. We show that $\mathcal{T}^{\downarrow}, u^* \models \Sigma$: let $\varphi \in \Sigma$. Then $\hat{\varphi} \in E \subseteq U$ and $\hat{\varphi} \subset \{i \in I \mid \mathcal{T}_i, u_i \models \varphi\}$. Hence $\{i \in I \mid \mathcal{T}_i, u_i \models \varphi\} \in U$, which implies that $\prod_U \mathcal{T}_i \models \mathrm{Tr}_x(\varphi)[u^*]$, where u^* is the ultralimit of $(u_i)_{i\in I}$. Since $\mathcal{T}^{\downarrow} = (\prod_U \mathcal{T}_i)|u^*$, by item 1 of Proposition 12 we conclude that $\mathcal{T}^{\downarrow}, u^* \models \varphi$. □

Theorem 20. *Let K be a class of pointed data trees. Then K is definable by a set of XPath$_=(\downarrow)$-node expressions iff K is closed under $\downarrow$-bisimulations and $\downarrow$-quasi ultraproducts, and $\overline{K}$ is closed under $\downarrow$-quasi ultrapowers.*

Proof. For $(\Rightarrow)$, suppose that K is definable by a set of XPath$_=(\downarrow)$-node expressions. By Theorem 3 it is clear that K is closed under $\downarrow$-bisimulations. By the fundamental theorem of ultraproducts together with item 1 of Proposition 12 it is clear that K is closed under $\downarrow$-quasi ultraproducts. It is also clear that the fundamental theorem of ultraproducts and the fact that any XPath$_=(\downarrow)$-node expression is expressible in first-order imply that $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}^{\downarrow}, u^*$ for any $(\mathcal{T}^{\downarrow}, u^*)$ $\downarrow$-quasi ultrapower modulo U, and therefore that $\overline{K}$ is closed under $\downarrow$-quasi ultrapowers.

For $(\Leftarrow)$, suppose K is closed under bisimulations and $\downarrow$-quasi ultraproducts, and $\overline{K}$ is closed under $\downarrow$-quasi ultrapowers. We show that $\Gamma = \bigcap_{(\mathcal{T},u)\in K} \mathrm{Th}_{\downarrow}(\mathcal{T}, u)$ defines K. It is clear that if $(\mathcal{T}, u) \in K$ then $\mathcal{T}, u \models \Gamma$.

Now suppose that $\mathcal{T}, u \models \Gamma$ and consider $\Sigma = \mathrm{Th}_{\downarrow}(\mathcal{T}, u)$. Let Δ be a finite subset of Σ, and assume that Δ is not satisfiable in K. Then $\neg \bigwedge \Delta$ is true in every pointed data tree of K, so

$\neg \wedge \Delta \in \Gamma$. Therefore $\mathcal{T}, u \models \neg \wedge \Delta$ which is a contradiction because $\Delta \subseteq \Sigma$. This shows that Σ is finitely satisfiable in K.

By Lemma 19, Σ is satisfiable in some $\downarrow$-quasi ultraproduct of pointed data trees in K, and since K is closed under $\downarrow$-quasi ultraproducts, Σ is satisfiable in K. Then there exists $(\mathcal{T}', u') \in K$ such that $\mathcal{T}', u' \models \Sigma$ and therefore $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'$. By Lemma 18, there exist $\downarrow$-quasi ultrapowers $(\mathcal{T}^{\downarrow}, u^*)$ and $(\mathcal{T}'^{\downarrow}, u'^*)$ of $(\mathcal{T}, u)$ and $(\mathcal{T}', u')$ respectively such that $(\mathcal{T}^{\downarrow}, u^*) \underline{\leftrightarrow}^{\downarrow} (\mathcal{T}'^{\downarrow}, u'^*)$. Since K is closed under $\downarrow$-bisimulations, $(\mathcal{T}^{\downarrow}, u^*) \in K$. Suppose $(\mathcal{T}, u) \in \overline{K}$. Since K is closed under $\downarrow$-quasi ultrapowers, $(\mathcal{T}^{\downarrow}, u^*) \in \overline{K}$, and this is a contradiction. Hence we conclude $(\mathcal{T}, u) \in K$. $\square$

Theorem 21. *Let K be a class of pointed data trees. Then K is definable by an XPath$_=(\downarrow)$-node expression iff both K and $\overline{K}$ are closed under $\downarrow$-bisimulations and $\downarrow$-quasi ultraproducts.*

Proof. For ($\Rightarrow$) suppose that K is definable by an XPath$_=(\downarrow)$-node expression. By Theorem 3 it is clear that K and $\overline{K}$ are closed under bisimulations. By the fundamental theorem of ultraproducts together with item 1 of Proposition 12 it is clear that K and $\overline{K}$ are closed under $\downarrow$-quasi ultraproducts.

For ($\Leftarrow$) suppose K and $\overline{K}$ are closed under bisimulations and $\downarrow$-quasi ultraproducts. Then, by Theorem 20, there exist sets Γ_1 and Γ_2 of XPath$_=(\downarrow)$-node expression defining K and $\overline{K}$ respectively. Consider the set of XPath$_=(\downarrow)$-node expressions $\Gamma_1 \cup \Gamma_2$. This set is clearly inconsistent and so, by compactness, there are finite sets Δ_1 and Δ_2 such that $\Delta_i \subseteq \Gamma_i$ ($i = 1, 2$) and

$$\mathcal{T}, u \models \wedge\Delta_1 \to \neg \wedge \Delta_2 \tag{6}$$

for any pointed data tree $(\mathcal{T}, u)$. We show that $\varphi = \wedge\Delta_1$ defines K. On the one hand, it is clear that if $(\mathcal{T}, u) \in K$ then $\mathcal{T}, u \models \varphi$. On the other hand, suppose that $\mathcal{T}, u \models \varphi$. From (6) we conclude $\mathcal{T}, u \models \neg \wedge \Delta_2$ and so $\mathcal{T}, u \not\models \Gamma_2$. Then $(\mathcal{T}, u) \notin \overline{K}$ as we wanted to prove.

$\square$

Like Theorem 21, the following result characterizes when a class of pointed data trees is definable by a single XPath$_=(\downarrow)$-node expression. However, instead of using the rather abstract notion of $\downarrow$-quasi ultraproducts, it uses the perhaps more natural notion of ℓ-bisimulation.

Theorem 22. *Let K be a class of pointed data trees. Then K is definable by a node expression of XPath$_=(\downarrow)$ iff K is closed by ℓ-bisimulations for XPath$_=(\downarrow)$ for some ℓ.*

Proof. ($\Rightarrow$) is a direct consequence of Theorem 3. Let us see ($\Leftarrow$). We know [10, Corollary 3.2] that $\{\mathcal{T}', u' \mid \mathcal{T}, u \equiv^{\downarrow}_{\ell} \mathcal{T}', u'\}$ is definable by an XPath$_=(\downarrow)$-node expression $\chi_{\ell,\mathcal{T},u}$ of downward depth $\leq \ell$. We show that

$$\varphi = \bigvee_{(\mathcal{T},u)\in K} \chi_{\ell,\mathcal{T},u}$$

defines K. In [10, Proposition 3.1] it is shown that $\equiv^{\downarrow}_{\ell}$ has finite index, and therefore the above disjunction is equivalent to a finite one. On the one hand, if $(\mathcal{T}', u') \in K$ then it is clear that $\mathcal{T}', u' \models \chi_{\ell,\mathcal{T}',u'}$ and so $\mathcal{T}', u' \models \varphi$. On the other hand, we have $\mathcal{T}', u' \models \varphi$ iff there is $(\mathcal{T}, u) \in K$ such that $\mathcal{T}', u' \models \chi_{\ell,\mathcal{T},u}$ iff there is $(\mathcal{T}, u) \in K$ such that $\mathcal{T}, u \underline{\leftrightarrow}^{\downarrow}_{\ell} \mathcal{T}', u'$. Hence since K is closed under $\underline{\leftrightarrow}^{\downarrow}_{\ell}$, if $\mathcal{T}', u' \models \varphi$ we have $(\mathcal{T}', u') \in K$. $\square$

We turn to the vertical fragment.

Lemma 23. *Let $(\mathcal{T}, u)$ and $(\mathcal{T}', u')$ be two pointed data trees such that $\mathcal{T}, u \equiv^{\uparrow\downarrow} \mathcal{T}', u'$. Then there exist $\uparrow\downarrow$-quasi ultrapowers $(\mathcal{T}^{\uparrow\downarrow}, u^*)$ and $(\mathcal{T}'^{\uparrow\downarrow}, u'^*)$ of $(\mathcal{T}, u)$ and $(\mathcal{T}', u')$ respectively such that $(\mathcal{T}^{\uparrow\downarrow}, u^*) \underline{\leftrightarrow}^{\uparrow\downarrow} (\mathcal{T}'^{\uparrow\downarrow}, u'^*)$*

Proof. The proof is analogous to the proof of Lemma 18 but using item 2 instead of item 1 of Proposition 14 and Proposition 10 instead of Proposition 8. $\square$

Lemma 24. *Let K be a class of k-bounded pointed data trees and let Σ be a set of XPath$_=(\uparrow\downarrow)$-node expressions finitely satisfiable in K. Then Σ is satisfiable in some $\uparrow\downarrow$-quasi ultraproduct of pointed data trees in K.*

Proof. The proof is analogous to the proof of Lemma 19 but taking $\uparrow\downarrow$-quasi ultraproducts instead of $\downarrow$-quasi ultraproducts and using item 2 instead of item 1 of Proposition 12. To apply this Proposition, one has to note that $u^* \in \mathcal{T}^{\uparrow\downarrow}$ since the $\mathcal{T}_i, u_i$ are k-bounded pointed. $\square$

In the next two theorems, the universe of pointed data trees is restricted to those which are k-bounded (for any fixed k). Therefore, the operations of closure and complement must be taken with respect to this universe.

Theorem 25. *Over k-bounded pointed data trees: K is definable by a set of XPath$_=(\uparrow\downarrow)$-node expressions iff K is closed under $\uparrow\downarrow$-bisimulations and $\uparrow\downarrow$-quasi ultraproducts, and $\overline{K}$ is closed under $\uparrow\downarrow$-quasi ultrapowers.*

Proof. The proof is analogous to the proof of Theorem 20 but replacing pointed data trees for k-bounded pointed data trees and every occurrence of $\downarrow$ for $\uparrow\downarrow$. Also, for ($\Rightarrow$), one has to use item 2 instead of item 1 of Proposition 12 and for ($\Leftarrow$), Lemmas 24 and 23 instead of Lemmas 19 and 18. $\square$

Theorem 26. *Over k-bounded pointed data trees: K is definable by an XPath$_=(\uparrow\downarrow)$-node expression iff both K and $\overline{K}$ are closed under $\uparrow\downarrow$-bisimulations and $\uparrow\downarrow$-quasi ultraproducts.*

As in Theorem 22, one can also restate Theorem 26 in terms of (r, s, k)-bisimulations for XPath$_=(\uparrow\downarrow)$.

Theorem 27. *Let K be a class of pointed data trees. Then K is definable by a node expression of XPath$_=(\uparrow\downarrow)$ iff K is closed by (r, s, k)-bisimulations for XPath$_=(\uparrow\downarrow)$ for some r, s, k.*

3.4 Separation

The theorem of Separation for first-order is closely related to Definability: it provides conditions to separate two disjoint classes of models K_1 and K_2 by means of a first-order formula, i.e. to find a class K, definable by a first-order formula or by a single formula, such that $K_1 \subseteq K$ and $K \cap K_2 = \emptyset$.

Theorem 28. *Let K_1 and K_2 be two disjoint classes of pointed data trees such that K_1 is closed under $\downarrow$-bisimulations and $\downarrow$-quasi ultraproducts and K_2 is closed under $\downarrow$-bisimulations and $\downarrow$-quasi ultrapowers. Then there exists a third class K which is definable by a set of XPath$_=(\downarrow)$-node expressions, contains K_1 and is disjoint from K_2.*

Proof. Let $K = \{(\mathcal{T}', u') \mid$ there is $(\mathcal{T}, u) \in K_1$ such that $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'\}$. Clearly, $K_1 \subseteq K$. We first show that $K \cap K_2 = \emptyset$. Suppose that there is a pointed model $(\mathcal{T}', u') \in K \cap K_2$. Then, there exists $(\mathcal{T}, u) \in K_1$ such that $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'$ and, by Lemma 18, there exist $\downarrow$-quasi ultrapowers $(\mathcal{T}^{\downarrow}, u^*)$ and $(\mathcal{T}'^{\downarrow}, u'^*)$ of $(\mathcal{T}, u)$ and $(\mathcal{T}', u')$ respectively such that $\mathcal{T}^{\downarrow}, u^* \leftrightarroweq^{\downarrow} \mathcal{T}'^{\downarrow}, u'^*$. Since K_1 is closed under $\downarrow$-quasi ultraproducts and $\downarrow$-bisimulations and K_2 is closed under $\downarrow$-quasi ultrapowers, $(\mathcal{T}'^{\downarrow}, u'^*) \in K_1 \cap K_2$ which is a contradiction.

To conclude the proof, we show that K is definable by a set of XPath$_=(\downarrow)$-node expressions. By Theorem 20, it is enough to check that K is closed under $\downarrow$-bisimulations and $\downarrow$-quasi ultraproducts and $\overline{K}$ is closed under $\downarrow$-quasi ultrapowers. Clearly, K is closed under $\downarrow$-bisimulations, as $\leftrightarroweq^{\downarrow}$ implies $\equiv^{\downarrow}$. Now, let $(\mathcal{T}'_i, u'_i)_{i \in I}$ be a family of pointed data trees contained in K. Then, for all $i \in I$, there is $(\mathcal{T}_i, u_i) \in K_1$ such that $\mathcal{T}_i, u_i \equiv^{\downarrow} \mathcal{T}'_i, u'_i$. By the fundamental theorem of ultraproducts, if U is an ultrafilter over I and $\mathcal{T}^*, u^*, \mathcal{T}'^*, u'^*$ are the ultraproducts of the families $(\mathcal{T}_i, u_i)_{i \in I}$ and $(\mathcal{T}'_i, u'_i)_{i \in I}$ respectively, then $(\mathcal{T}^*, u^*) \equiv^{\downarrow} (\mathcal{T}'^*, u'^*)$, and by Proposition 12 $(\mathcal{T}^{\downarrow}, u^*) \equiv^{\downarrow} (\mathcal{T}'^{\downarrow}, u'^*)$. Now, since K_1 is closed under $\downarrow$-quasi ultraproducts, $(\mathcal{T}'^{\downarrow}, u'^*) \in K$ which proves that K is closed under $\downarrow$-quasi ultraproducts. Finally, let $(\mathcal{T}', u') \in \overline{K}$. Suppose that $(\mathcal{T}'^{\downarrow}, u'^*)$, some $\downarrow$-quasi ultrapower of $(\mathcal{T}', u')$, belongs to K. By the fundamental theorem of ultraproducts, $(\mathcal{T}'^{\downarrow}, u'^*) \equiv^{\downarrow} (\mathcal{T}', u')$. So, since K is closed under $\equiv^{\downarrow}$, $(\mathcal{T}', u') \in K$, which is a contradiction. $\square$

Theorem 29. *Let K_1 and K_2 be two disjoint classes of pointed data trees closed under $\downarrow$-bisimulations and $\downarrow$-quasi ultraproducts. Then there exists a third class K which is definable by an XPath$_=(\downarrow)$-node expression, contains K_1 and is disjoint from K_2.*

Proof. By Theorem 28, there exists a class K' definable by a set of XPath$_=(\downarrow)$-node expressions Γ_1, containing K_1 and disjoint from K_2. Observe that as a consequence of Theorem 3, such K' is closed under $\downarrow$-bisimulations and $\downarrow$-quasi ultraproducts. Using Theorem 28 again for K_2 and K', we have another class K'' also definable by a set of XPath$_=(\downarrow)$-node expressions Γ_2, containing K_2 and disjoint from K'.

Now consider the set of XPath$_=(\downarrow)$-node expressions $\Gamma_1 \cup \Gamma_2$. This set is clearly inconsistent and so, by compactness, there are finite sets Δ_1 and Δ_2 such that $\Delta_i \subseteq \Gamma_i$ $(i = 1, 2)$ and $\bigwedge \Delta_1 \wedge \bigwedge \Delta_2$ is unsatisfiable. Now let $K = \{\mathcal{T}, u \mid \mathcal{T}, u \models \bigwedge \Delta_1\}$. This K satisfies the desired properties, as $K_1 \subset K' \subset K$ and $K_2 \cap K \subset K'' \cap K = \emptyset$. □

The same proofs apply for the case of XPath$_=(\uparrow\downarrow)$, using the corresponding notions of bisimulations and quasi ultraproducts and Lemmas 23 and 25 instead of Lemmas 18 and 20, with the proviso that the universe of data trees are restricted to those which are k-bounded (and so operations of closure and complement must be taken with respect to this universe).

4 Binary bisimulations

We introduce notions of binary bisimulations for the downward and vertical fragments. These notions are suitable in the sense that they capture the idea of *indistinguishability by path expressions.* For the case of the downward fragment, we show a van Benthem-like characterization theorem.

4.1 Downward

4.1.1 Some facts about path expressions over XPath$_=(\downarrow)$

The proofs of Theorem 3 or Theorem 4 of [10] assume that node expressions of XPath$_=(\downarrow)$ do not contain any $\cup$. Indeed, as explained at the end of §2.2, any $\cup$ of a path expression can be simulated with a $\vee$ within a suitable node expression. However, we have seen that it is not true that any XPath$_=(\downarrow)$-path expression is equivalent to a $\cup$-free one. Hence, in our context of studying a notion of binary bisimulation which captures the idea of *indistinguishability by path expressions*, we need to develop first some results that allow us to deal with the $\cup$ operator. Another difference with respect of the previous work is that there are no intersection nor complementation of path expressions. As we will see next, under certain contexts, we can define them within the language of XPath$_=(\downarrow)$.

Definition 30. If α is of the form $\alpha = [\varphi_0] \downarrow [\varphi_1] \downarrow \dots \downarrow [\varphi_n]$, we say that it is in **simple normal form**, and we say that the length of α (notation: $\mathrm{len}(\alpha)$) is n.

Fact 31. For each $\cup$-free XPath$_=(\downarrow)$ path expression α there is an XPath$_=(\downarrow)$ path expression β in simple normal form such that $\mathrm{dd}(\beta) = \mathrm{dd}(\alpha)$ and for all data tree $\mathcal{T}$, we have $[\![\alpha]\!]^{\mathcal{T}} = [\![\beta]\!]^{\mathcal{T}}$.

Fact 32. If α is a $\cup$-free XPath$_=(\downarrow)$ path expression then $\mathcal{T}, x, y \models \alpha$ implies $x \xrightarrow{n} y$ in $\mathcal{T}$, where $n = \mathrm{len}(\alpha)$.

The $\cup$ operator is unessential for distinguishing two pairs of nodes:

Lemma 33. *If $\mathcal{T}, x, y \models \alpha$ and $\mathcal{T}', x', y' \not\models \alpha$ then there is a $\cup$-free XPath$_=(\downarrow)$ path expression $\tilde{\alpha}$ with $\mathrm{dd}(\tilde{\alpha}) \leq \mathrm{dd}(\alpha)$ such that $\mathcal{T}, x, y \models \tilde{\alpha}$ and $\mathcal{T}', x', y' \not\models \tilde{\alpha}$.*

Proof. We show it by induction on α. The only interesting case is when $\alpha = \alpha_1 \cup \alpha_2$. Since $\mathcal{T}, x, y \models \alpha$ then there is $i \in \{1, 2\}$ such that $\mathcal{T}, x, y \models \alpha_i$. Since $\mathcal{T}', x', y' \not\models \alpha$ then $\mathcal{T}', x', y' \not\models \alpha_i$. By inductive hypothesis there is $\tilde{\alpha}_i$ which is $\cup$-free and such that $\mathcal{T}, x, y \models \tilde{\alpha}_i$ and $\mathcal{T}', x', y' \not\models \tilde{\alpha}_i$. □

The following lemma gives us a restricted form of negation for path expressions:

Lemma 34. *Let $x \xrightarrow{n} y$ in $\mathcal{T}$ and $x' \xrightarrow{n} y'$ in $\mathcal{T}'$. If α is an $\cup$-free XPath$_=(\downarrow)$ path expression such that $\mathcal{T}, x, y \models \alpha$ and $\mathcal{T}', x', y' \not\models \alpha$ then there is a $\cup$-free path expression $\overline{\alpha}$ such that* $\mathrm{dd}(\alpha) = \mathrm{dd}(\overline{\alpha})$ *and $\mathcal{T}, x, y \not\models \overline{\alpha}$ and $\mathcal{T}', x', y' \models \overline{\alpha}$.*

Proof. By Fact 31 we can assume that α is in simple normal form, say $\alpha = [\varphi_0] \downarrow [\varphi_1] \downarrow \dots \downarrow [\varphi_n]$. Let $x = x_0 \to x_1 \to \dots \to x_n = y$ and $x' = x'_0 \to x'_1 \to \dots \to x'_n = y'$. Since $\mathcal{T}, x, y \models \alpha$ and $\mathcal{T}', x', y' \not\models \alpha$ there is i such that $x_i \models \varphi_i$ and $x'_i \not\models \varphi_i$. One can check that $\overline{\alpha} = \downarrow^i [\neg\varphi_i] \downarrow^{n-i}$ is as we wanted. $\square$

The following lemma simplifies many of the proofs, and it will be used frequently and without mention.

Lemma 35. *If α is a XPath$_=(\downarrow)$ path expression, it is equivalent to a XPath$_=(\downarrow)$ path expression of the form $\beta_1 \cup \dots \cup \beta_n$, with the β_i in simple normal form.*

Definition 36. If $\alpha = [\varphi_0] \downarrow [\varphi_1] \downarrow \dots \downarrow [\varphi_i]$ and $\beta = [\psi_0] \downarrow [\psi_1] \downarrow \dots \downarrow [\psi_i]$ are XPath$_=(\downarrow)$ path expressions of the same length in simple normal form, we define the **intersection** of α and β as

$$\alpha \cap \beta := [\varphi_0 \wedge \psi_0] \downarrow [\varphi_1 \wedge \psi_1] \downarrow \dots \downarrow [\varphi_i \wedge \psi_i]. \tag{7}$$

Fact 37. If α and β are XPath$_=(\downarrow)$ path expressions in simple normal form of the same length, then $\mathrm{dd}(\alpha \cap \beta) = \max\{\mathrm{dd}(\alpha), \mathrm{dd}(\beta)\}$, and for every data tree $\mathcal{T}$, we have $[\![\alpha \cap \beta]\!]^{\mathcal{T}} = [\![\alpha]\!]^{\mathcal{T}} \cap [\![\beta]\!]^{\mathcal{T}}$.

4.1.2 Equivalence for XPath$_=(\downarrow)$ path expressions

For a data tree $\mathcal{T}$, let us define

$$D(\mathcal{T}) = \{(u, v) \in T^2 \mid u \xrightarrow{*} v\},$$

and for $\ell \geq 0$,

$$D_\ell(\mathcal{T}) = \{(u, v) \in T^2 \mid u \xrightarrow{\leq \ell} v\}.$$

We say that $(x, y) \in D(\mathcal{T})$ and $(x', y') \in D(\mathcal{T}')$ are **equivalent for XPath$_=(\downarrow)$ path expressions** (notation: $\mathcal{T}, x, y \equiv^{\downarrow} \mathcal{T}', x', y'$) if for all XPath$_=(\downarrow)$ path expressions α, we have $\mathcal{T}, x, y \models \alpha$ iff $\mathcal{T}', x', y' \models \alpha$. We say that $(x, y) \in D_\ell(\mathcal{T})$ and $(x', y') \in D_\ell(\mathcal{T}')$ are **ℓ-equivalent for XPath$_=(\downarrow)$ path expressions** (notation: $\mathcal{T}, x, y \equiv^{\downarrow}_\ell \mathcal{T}', x', y'$) if for all XPath$_=(\downarrow)$ path expressions α with $\mathrm{dd}(\alpha) \leq \ell$, we have $\mathcal{T}, x, y \models \alpha$ iff $\mathcal{T}', x', y' \models \alpha$.

Of course, one could have defined $\mathcal{T}, x, y \equiv^{\downarrow} \mathcal{T}', x', y'$ even for pairs $(x, y) \notin D(\mathcal{T})$ or for pairs $(x', y') \notin D(\mathcal{T}')$. For instance, if x is not an ancestor of y then $\mathcal{T}, x, y$ does not verify *any* path expression, and so one could say that $\mathcal{T}, x, y \equiv^{\downarrow} \mathcal{T}', x', y'$ only when x', y' does not verify any path expression (in other words, when x' is not an ancestor of y', i.e. $(x', y') \notin D(\mathcal{T})$). We restricted $\equiv$ to $D(\mathcal{T}) \times D(\mathcal{T}')$ for reasons of clarity when comparing logical equivalence with binary bisimulations, as we will see next.

Notice that if $\mathcal{T}, x, y \equiv^{\downarrow} \mathcal{T}', x', y'$ and $x \xrightarrow{n} y$ then $\mathcal{T}, x, y \models \downarrow^n$ and hence $\mathcal{T}', x', y' \models \downarrow^n$, which means $x' \xrightarrow{n} y'$ in $\mathcal{T}'$. The same holds in case $\mathcal{T}, x, y \equiv^{\downarrow}_\ell \mathcal{T}', x', y'$ when $n \leq \ell$.

Lemma 38. *Let $u \xrightarrow{m} v$ in $\mathcal{T}$ and $u' \xrightarrow{n} v'$ in $\mathcal{T}'$, and let $n, m \leq \ell$. If $\mathcal{T}, u, v \not\equiv^{\downarrow}_\ell \mathcal{T}', u', v'$ then there is a $\cup$-free XPath$_=(\downarrow)$ path expression α such that* $\mathrm{dd}(\alpha) \leq \ell$*, $\mathcal{T}, u, v \models \alpha$ and $\mathcal{T}', u', v' \not\models \alpha$.*

Proof. If $n \neq m$ then $\mathcal{T}, u, v \models \downarrow^m$ and $\mathcal{T}', u', v' \not\models \downarrow^m$. Suppose that $n = m$ and that there is an XPath$_=(\downarrow)$ path expression α, $\mathrm{dd}(\alpha) \leq \ell$, such that $\mathcal{T}, u, v \models \alpha$ and $\mathcal{T}', u', v' \not\models \alpha$. By Lemma 33, α can be taken $\cup$-free and we are done. The same argument applies in case $\mathcal{T}, u, v \not\models \alpha$ and $\mathcal{T}', u', v' \models \alpha$, via Lemma 34. $\square$

Proposition 39. *$\equiv^{\downarrow}_\ell$ has finite index in the context of path expressions, that is, there are finitely many non-equivalent path expressions of downward depth at most ℓ.*

Proof. Let qr be the quantifier rank of a fist order formula, i.e., the depth of nesting of its quantifiers. It can be easily shown by induction that for any path expression α of XPath$_=(\downarrow)$ with bounded downward depth and unnecessary uses of ε (recall that $\alpha\varepsilon\beta \equiv \alpha\beta$) we have that $\mathrm{qr}(\mathrm{Tr}_{x,y}(\alpha))$ is bounded. It is a well-known result of first order that there are finitely many nonequivalent formulas of bounded quantifier rank. Hence there are finitely many nonequivalent node expressions of bounded downward depth. □

Corollary 40. *Suppose $u \xrightarrow{n} v$, with $n \leq \ell$. Then $\{\mathcal{T}', u', v' \mid \mathcal{T}, u, v \equiv_\ell^\downarrow \mathcal{T}', u', v'\}$ is definable by an ℓ-XPath$_=(\downarrow)$ path expression $\gamma_{\ell,\mathcal{T},u,v}$.*

Proof. Let

$$A = \{\alpha \mid \mathcal{T}, u, v \models \alpha, \ \alpha \text{ is } \cup\text{-free and } \mathrm{dd}(\alpha) \leq \ell\}.$$

First, observe that by Fact 31, each $\alpha \in A$ can be written in simple normal form, and all of them have the same length. Hence it makes sense to take the intersection between finitely many elements of A. Second, notice that by Proposition 39 there are finitely many non-equivalent $\alpha \in A$, and hence the infinite intersection $\beta = \bigcap A$ is equivalent to a finite one.

It is clear by Fact 37 that $\mathrm{dd}(\beta) \leq \ell$ and that $\mathcal{T}, u, v \models \beta$. Let us show that

$$\mathcal{T}', u', v' \models \beta \quad \text{iff} \quad \mathcal{T}, u, v \equiv_\ell^\downarrow \mathcal{T}', u', v'.$$

The right-to-left direction is straightforward. For the left-to-right direction, suppose by contradiction that $\mathcal{T}', u', v' \models \beta$ and $\mathcal{T}, u, v \not\equiv_\ell^\downarrow \mathcal{T}', u', v'$. By hypothesis, $\mathcal{T}, u, v \models \downarrow^n$ (where $n \leq \ell$), and thus, since $\mathcal{T}', u', v' \models \beta$, we have $\mathcal{T}', u', v' \models \downarrow^n$. By Lemma 38, there is a $\cup$-free XPath$_=(\downarrow)$ path expression γ such that $\mathrm{dd}(\gamma) \leq \ell$ and $\mathcal{T}, u, v \models \gamma$ and $\mathcal{T}', u', v' \not\models \gamma$. Since $\gamma \in A$ and $\mathcal{T}', u', v' \models \beta$ then $\mathcal{T}', u', v' \models \gamma$, which is a contradiction. □

4.1.3 Binary bisimulation for XPath$_=(\downarrow)$

We introduce a new notion of binary bisimulation between pairs of nodes (x, y) in one data-tree $\mathcal{T}$ and pairs of nodes (x', y') in another data tree $\mathcal{T}'$. For simplicity we only define binary bisimulation as a relation in $D(\mathcal{T}) \times D(\mathcal{T}')$. But it can be naturally extended to $T^2 \times T'^2$ if the definition of $\equiv$ is likewise extended.

We say that $(t, u) \in D(\mathcal{T})$ is **bisimilar** to $(t', u') \in D(\mathcal{T}')$ for XPath$_=(\downarrow)$ (notation: $\mathcal{T}, t, u \leftrightarrow^\downarrow \mathcal{T}', t', u'$) if there is a relation $Z \subseteq D(\mathcal{T}) \times D(\mathcal{T}')$ such that $(t, u)Z(t', u')$ and for all $x, y \in T$ and $x', y' \in T'$ we have:

- **Harmony:** if $(x, y)Z(x', y')$ then $label(x) = label(x')$.
- **Equidistance:** if $(x, y)Z(x', y')$ then there is k such that $x \xrightarrow{k} y$ and $x' \xrightarrow{k} y'$.
- **Split:** if $(x, y)Z(x', y')$, $x \xrightarrow{n} z \xrightarrow{m} y$ and $x' \xrightarrow{n} z' \xrightarrow{m} y'$ then $(x, z)Z(x', z')$ and $(z, y)Z(z', y')$.
- **Zig:** if $(x, y)Z(x', y')$, $x \xrightarrow{n} v$ and $x \xrightarrow{m} w$ then there are $v', w' \in T'$ such that: $x' \xrightarrow{n} v'$; $x' \xrightarrow{m} w'$; $(x, v)Z(x', v')$; $(x, w)Z(x', w')$; and $data(v) = data(w)$ iff $data(v') = data(w')$.
- **Zag:** if $(x, y)Z(x', y')$, $x' \xrightarrow{n} v'$ and $x' \xrightarrow{m} w'$ then there are $v, w \in T$ such that: $x \xrightarrow{n} v$; $x \xrightarrow{m} w$; $(x, v)Z(x', v')$; $(x, w)Z(x', w')$; and $data(v) = data(w)$ iff $data(v') = data(w')$.

See Figure 3 for an example of a binary-bisimulation for XPath$_=(\downarrow)$.

We say that $(t, u) \in D_\ell(\mathcal{T})$ is **ℓ-bisimilar** to $(t', u') \in D_\ell(\mathcal{T}')$ for XPath$_=(\downarrow)$ (notation: $\mathcal{T}, t, u \leftrightarrow_\ell^\downarrow \mathcal{T}', t', u'$) if there is a family of relations $(Z_j)_{j \leq \ell}$ in $D_j(\mathcal{T}) \times D_j(\mathcal{T}')$ such that $(t, u)Z_\ell(t', u')$ and for all $j \leq \ell$, $(x, y) \in D_j(\mathcal{T})$ and $(x', y') \in D_j(\mathcal{T}')$ we have:

- **Harmony:** if $(x, y)Z_j(x', y')$ then $label(x) = label(x')$.
- **Equidistance:** if $(x, y)Z_j(x', y')$ then there is $k \leq j$ such that $x \xrightarrow{k} y$ and $x' \xrightarrow{k} y'$.

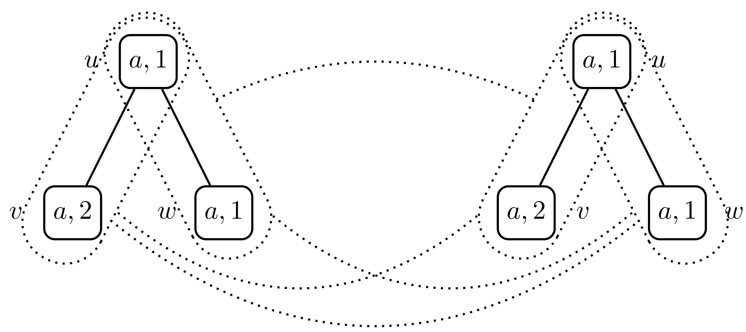

Figure 3: A XPath$_=(\downarrow)$ binary bisimulation Z between the same data tree $\mathcal{T}$. Pairs $(u,u)Z(u,u)$, $(v,v)Z(v,v)$, $(w,w)Z(w,w)$, $(w,w)Z(v,v)$ and $(v,v)Z(w,w)$ are not shown.

- **Split:** if $(x,y)Z_j(x',y')$, $x\xrightarrow{n}z\xrightarrow{m}y$ and $x'\xrightarrow{n}z'\xrightarrow{m}y'$ then $(x,z)Z_j(x',z')$ and $(z,y)Z_{j-n}(z',y')$.
- **Zig:** if $(x,y)Z_j(x',y')$, $x\xrightarrow{n}v$ and $x\xrightarrow{m}w$, with $n,m\leq j$, then there are $v',w'\in T'$ such that: $x'\xrightarrow{n}v'$, $x'\xrightarrow{m}w'$, $(x,v)Z_j(x',v')$, $(x,w)Z_j(x',w')$, and $\mathit{data}(v)=\mathit{data}(w)$ iff $\mathit{data}(v')=\mathit{data}(w')$.
- **Zag:** if $(x,y)Z_j(x',y')$, $x'\xrightarrow{n}v'$ and $x'\xrightarrow{m}w'$, with $n,m\leq j$, then there are $v,w\in T$ such that: $x\xrightarrow{n}v$, $x\xrightarrow{m}w$, $(x,v)Z_j(x',v')$, $(x,w)Z_j(x',w')$, and $\mathit{data}(v)=\mathit{data}(w)$ iff $\mathit{data}(v')=\mathit{data}(w')$.

Notice that, because of the **Split** condition, the rules **Zig** and **Zag** for binary bisimulations only require Z to relate (x,v) and (x',v') on one hand and (x,w) and (x',w') on the other, instead of relating *all* nodes along the path from x to v to the corresponding nodes in the path from x' to v', and the same for the paths from x to w and x' to w'.

For a data tree $\mathcal{T}$ and $u\in T$, let $\mathcal{T}|u$ denote the subtree of $\mathcal{T}$ induced by $\{v\in T\mid(\exists n)\ u\xrightarrow{n}v\}$. Observe that the root of $\mathcal{T}|u$ is u. The following results are straightforward consequences of the definition of binary bisimulation:

Proposition 41. *If $(u,v)\in D(\mathcal{T})$ then $\mathcal{T},u,v \mathrel{\underline{\leftrightarrow}}^{\downarrow} (\mathcal{T}|u),u,v$.*

Proposition 42. *If $\mathcal{T}$ is a subtree of $\mathcal{T}'$ and $(u,v)\in D(\mathcal{T})$ then $\mathcal{T},u,v \mathrel{\underline{\leftrightarrow}}^{\downarrow} \mathcal{T}',u,v$.*

For a data tree $\mathcal{T}$ and $u\in T$, let $\mathcal{T}|_\ell u$ denote the subtree of $\mathcal{T}$ induced by $\{v\in T\mid(\exists n\leq\ell)\ u\xrightarrow{n}v\}$.

Proposition 43. *If $(u,v)\in D_\ell(\mathcal{T})$ then $\mathcal{T},u,v \mathrel{\underline{\leftrightarrow}}^{\downarrow}_{\ell} (\mathcal{T}|_\ell u),u,v$.*

Proof. Define the family $(Z_j)_{j\leq\ell}$, $Z_j\subseteq D_j(\mathcal{T}|_\ell u)\times D_j(\mathcal{T})$ as following: given $j\leq\ell$, if $x\xrightarrow{\leq j}y$ and $u\xrightarrow{\ell-j}x$, then $(x,y)Z_j(x,y)$ (observe that $j=\ell-(\ell-j)$; intuitively, start with a Z_ℓ which matches all identical pairs of nodes in $D(\mathcal{T}|_\ell u)$, then consider $Z_{\ell-1}$ the subset where the first coordinate of the pairs must be at a downward distance of 1 from u, and so on). The reader can check that $\mathcal{T},u,v$ and $(\mathcal{T}|_\ell u),u,v$ are ℓ-bisimilar via this family of relations. $\Box$

Proposition 44. *Suppose $\mathcal{T}$ and $\mathcal{T}'$ have height at most ℓ, $(u,v)\in D_\ell(\mathcal{T})$, and $(u',v')\in D_\ell(\mathcal{T}')$. Then $\mathcal{T},u,v \mathrel{\underline{\leftrightarrow}}^{\downarrow}_{\ell} \mathcal{T}',u',v'$ iff $\mathcal{T},u,v \mathrel{\underline{\leftrightarrow}}^{\downarrow} \mathcal{T}',u',v'$.*

We now show that in the new context of path expressions of XPath$_=(\downarrow)$ we have an analog of Theorem 3 for binary bisimulations and path equivalence, i.e., $\mathrel{\underline{\leftrightarrow}}^{\downarrow}$ coincides with $\equiv^{\downarrow}$ on finitely branching data trees, and $\mathrel{\underline{\leftrightarrow}}^{\downarrow}_{\ell}$ always coincides with $\equiv^{\downarrow}_{\ell}$.

Theorem 45. *1. $\mathcal{T},u,v \mathrel{\underline{\leftrightarrow}}^{\downarrow} \mathcal{T}',u',v'$ implies $\mathcal{T},u,v\equiv^{\downarrow}\mathcal{T}',u',v'$. The converse also holds when $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching.*

2. $\mathcal{T}, u, v \leftrightarrow^{\downarrow}_{\ell} \mathcal{T}', u', v'$ *iff* $\mathcal{T}, u, v \equiv^{\downarrow}_{\ell} \mathcal{T}', u', v'$.

Item 2 of the above theorem is a consequence of the next two propositions. Item 1 can be shown analogously (the set P that will appear in the proof of Proposition 47 for showing **Zig** is finite when $\mathcal{T}'$ is finitely branching, and its version over $\mathcal{T}$ for showing **Zag** is finite when $\mathcal{T}$ is finitely branching).

Proposition 46. *If* $\mathcal{T}, t, u \leftrightarrow^{\downarrow}_{\ell} \mathcal{T}', t', u'$ *then* $\mathcal{T}, t, u \equiv^{\downarrow}_{\ell} \mathcal{T}, t', u'$.

Proof. We actually show that if $\mathcal{T}, t, u \leftrightarrow^{\downarrow}_{\ell} \mathcal{T}', t', u'$ via $(Z_i)_{i\leq\ell}$ then for all $0 \leq n \leq j \leq \ell$, for all φ with $\mathrm{dd}(\varphi) \leq j$, and for all α with $\mathrm{dd}(\alpha) \leq j$:

1. If $(x, x)Z_j(x', x')$ then $\mathcal{T}, x \models \varphi$ iff $\mathcal{T}', x' \models \varphi$,
2. If $(x, y)Z_j(x', y')$ then $\mathcal{T}, x, y \models \alpha$ iff $\mathcal{T}', x', y' \models \alpha$.

We show 1 and 2 by induction on $|\varphi| + |\alpha|$.

Let us see item 1. The base case is $\varphi = a$ for some $a \in \mathbb{A}$. By **Harmony**, $\mathit{label}(x) = \mathit{label}(x')$ and then $\mathcal{T}, x \models \varphi$ iff $\mathcal{T}', x' \models \varphi$. The Boolean cases for φ are straightforward.

Suppose $\varphi = \langle\alpha = \beta\rangle$. We will show $\mathcal{T}, x \models \varphi \Rightarrow \mathcal{T}', x' \models \varphi$, so assume $\mathcal{T}, x \models \varphi$. Suppose there are $v, w \in T$ and $n, m \leq j$ such that $x \xrightarrow{n} v$, $x \xrightarrow{m} w$, $\mathcal{T}, x, v \models \alpha$, $\mathcal{T}, x, w \models \beta$ and $\mathit{data}(v) = \mathit{data}(w)$. By **Zig**, there are $v', w' \in T'$ such that $x' \xrightarrow{n} v'$, $x' \xrightarrow{m} w'$, $(x, v)Z_j(x', v')$, $(x, w)Z_j(x', w')$ and $\mathit{data}(v') = \mathit{data}(w')$. By inductive hypothesis 2 (twice), $\mathcal{T}', x', v' \models \alpha$ and $\mathcal{T}', x', w' \models \beta$. Hence $\mathcal{T}', x' \models \varphi$. The implication $\mathcal{T}', x' \models \varphi \Rightarrow \mathcal{T}, x \models \varphi$ is analogous. The case $\varphi = \langle\alpha \neq \beta\rangle$ is shown similarly.

Let us now analyze item 2. We only show the 'only if' direction, as the 'if' is analogous. The base case is when $\alpha \in \{\varepsilon, \downarrow\}$. If $\alpha = \varepsilon$, we have:

$$\begin{aligned} \mathcal{T}, x, y \models \alpha \quad & \text{iff} \quad x \xrightarrow{0} y \\ & \text{iff} \quad x' \xrightarrow{0} y' && (\textbf{Equidistance}) \\ & \text{iff} \quad \mathcal{T}', x', y' \models \alpha \end{aligned}$$

If $\alpha = \downarrow$, we have the same argument but with $\xrightarrow{1}$ instead of $\xrightarrow{0}$. For the inductive step, suppose $\alpha = \beta\gamma$ and assume $\mathcal{T}, x, y \models \alpha$. Then there is $z \in T$ such that $x \xrightarrow{n} z \xrightarrow{m} y$, $\mathcal{T}, x, z \models \beta$ and $\mathcal{T}, z, y \models \gamma$. By **Split** we have $(x, z)Z_j(x', z')$ and $(z, y)Z_{j-n}(z', y')$ Observe that $\mathrm{dd}(\beta) \leq \mathrm{dd}(\alpha) \leq j$ and $\mathrm{dd}(\gamma) \leq \mathrm{dd}(\alpha) - n \leq j - n$. where z' is the only node such that $x' \xrightarrow{n} z' \xrightarrow{m} y'$ (observe that by **Equidistance**, $x' \xrightarrow{n+m} y'$). By inductive hypothesis 2 (again, twice), we conclude $\mathcal{T}', x', z' \models \beta$ and $\mathcal{T}', z', y' \models \gamma$, and hence $\mathcal{T}', x', y' \models \alpha$.

Suppose $\alpha = \alpha_1 \cup \alpha_2$ and assume $\mathcal{T}, x, y \models \alpha$. We have $\mathcal{T}, x, y \models \alpha_i$ for some $i \in \{1, 2\}$. By inductive hypothesis we have $\mathcal{T}', x', y' \models \alpha_i$, and so $\mathcal{T}', x', y' \models \alpha$

Finally, suppose $\alpha = [\varphi]$ and assume $\mathcal{T}, x, y \models \alpha$. By semantics we have $x = y$ and $\mathcal{T}, x \models \varphi$. By inductive hypothesis, $\mathcal{T}', x' \models \varphi$, and by **Equidistance** we have $x' = y'$. Hence we conclude $\mathcal{T}', x', y' \models \alpha$. □

Proposition 47. *If* $\mathcal{T}, t, u \equiv^{\downarrow}_{\ell} \mathcal{T}', t', u'$ *then* $\mathcal{T}, t, u \leftrightarrow^{\downarrow}_{\ell} \mathcal{T}', t', u'$.

Proof. Fix $(t, u) \in D_\ell(\mathcal{T})$ and $(t', u') \in D_\ell(\mathcal{T}')$ such that $\mathcal{T}, t, u \equiv^{\downarrow}_{\ell} \mathcal{T}', t', u'$. Define $(Z_j)_{j\leq\ell}$ by

$$(x, y)Z_j(x', y') \quad \text{iff} \quad \mathcal{T}, x, y \equiv^{\downarrow}_{j} \mathcal{T}', x', y'$$

for all $(x, y) \in D_\ell(\mathcal{T})$ and all $(x', y') \in D_\ell(\mathcal{T}')$. We show that $(Z_j)_{j\leq\ell}$ is an ℓ-bisimulation between $\mathcal{T}, u, v$ and $\mathcal{T}', u', v'$.

By hypothesis, $(t, u)Z_\ell(t', u')$. To check all the rules of ℓ-bisimulation for $\mathrm{XPath}_=(\downarrow)$, suppose $x \xrightarrow{k} y$ for some $k \leq j$, and assume $(x, y)Z_j(x', y')$. To see **Harmony**, let $a = \mathit{label}(x)$ and let

$\alpha = [a]{\downarrow}^k$, of downward depth $k \leq j$. It is clear that $\mathcal{T}, x, y \models \alpha$, and so $\mathcal{T}, x', y' \models \alpha$, which means that $label(x') = a$. The implication $label(x') = a \Rightarrow label(x) = a$ is seen analogously.

For **Equidistance**, since $\mathcal{T}, x, y \models {\downarrow}^k$, then $\mathcal{T}', x', y' \models {\downarrow}^k$, and so $x' \xrightarrow{k} y'$. The implication $x' \xrightarrow{k} y' \Rightarrow x \xrightarrow{k} y$ is seen analogously.

Let us see **Split**. Suppose $x \xrightarrow{n} z \xrightarrow{m} y$ and $x' \xrightarrow{n} z' \xrightarrow{m} y'$, where $k = m + n \leq j$. We prove that:

1. $\mathcal{T}, x, z \equiv^{\downarrow}_{j} \mathcal{T}', x', z'$ and
2. $\mathcal{T}, z, y \equiv^{\downarrow}_{j-n} \mathcal{T}', z', y'$.

To see 1, assume by contradiction that α is path expression with $\mathrm{dd}(\alpha) \leq j$ such that $\mathcal{T}, x, z \models \alpha$ and $\mathcal{T}', x', z' \not\models \alpha$ (the other case is analogous). Observe that $\mathrm{len}(\alpha) = n$. Now, $\mathcal{T}, x, y \models \alpha{\downarrow}^m$ and $\mathcal{T}', x', y' \not\models \alpha{\downarrow}^m$. But $\mathrm{dd}(\alpha{\downarrow}^m) = \max\{\mathrm{dd}(\alpha), m + \mathrm{len}(\alpha)\} \leq j$, so, since $\mathcal{T}, x, y \equiv^{\downarrow}_{j} \mathcal{T}', x', y'$, we have $\mathcal{T}', x', y' \models \alpha\, {\downarrow}^m$, a contradiction.

To see 2, assume by contradiction that α is a path expression with $\mathrm{dd}(\alpha) \leq j - n$ such that $\mathcal{T}, z, y \models \alpha$ and $\mathcal{T}', z', y' \not\models \alpha$ (the other case is analogous). Observe that $\mathrm{len}(\alpha) = m$. Now, $\mathcal{T}, x, y \models {\downarrow}^n\, \alpha$ and $\mathcal{T}', x', y' \not\models {\downarrow}^n \alpha$. But $\mathrm{dd}({\downarrow}^n \alpha) = n + \mathrm{dd}(\alpha) \leq n + j - n = j$, so, since $\mathcal{T}, x, y \equiv^{\downarrow}_{j} \mathcal{T}', x', y'$, we have $\mathcal{T}', x', y' \models {\downarrow}^n\, \alpha$, a contradiction.

Finally, let us show **Zig** (the case for **Zag** is analogous). Suppose $x \xrightarrow{n} v$, $x \xrightarrow{m} w$, where $n, m \leq j$, and $data(v) = data(w)$ (the case $\neq$ is analogous).

Let $P \subseteq T'^2$ be defined by:

$$P = \{(v', w') \mid x' \xrightarrow{n} v' \wedge x' \xrightarrow{m} w' \wedge data(v') = data(w')\}.$$

Observe that $\mathcal{T}, x \models \langle {\downarrow}^n = {\downarrow}^m \rangle$. Hence $\mathcal{T}, x, y \models [\langle {\downarrow}^n = {\downarrow}^m \rangle]{\downarrow}^k$ and so $\mathcal{T}', x', y' \models [\langle {\downarrow}^n = {\downarrow}^m \rangle]{\downarrow}^k$, which implies $\mathcal{T}', x' \models \langle {\downarrow}^n = {\downarrow}^m \rangle$. Therefore $P \neq \emptyset$.

We next show that there exists $(v', w') \in P$ such that $\mathcal{T}, x, v \equiv^{\downarrow}_{j} \mathcal{T}', x', v'$ and $\mathcal{T}, x, w \equiv^{\downarrow}_{j} \mathcal{T}', x', w'$, and hence **Zig** is satisfied by Z_j.

Suppose by way of contradiction that for all $(v', w') \in P$, either $\mathcal{T}, x, v \not\equiv^{\downarrow}_{j} \mathcal{T}', x', v'$ or $\mathcal{T}, x, w \not\equiv^{\downarrow}_{j} \mathcal{T}', x', w'$. Because of Lemma 38, for all (v', w') in P, either there exists a $\cup$-free path expression $\alpha_{v',w'}$ such that $\mathrm{dd}(\alpha_{v',w'}) \leq j$ and $\mathcal{T}, x, v \models \alpha_{v',w'}$ but $\mathcal{T}', x', v' \not\models \alpha_{v',w'}$, or there exists a path expression $\beta_{v',w'}$ such that $\mathrm{dd}(\beta_{v',w'}) \leq j$ and $\mathcal{T}, x, w \models \beta_{v',w'}$ but $\mathcal{T}', x', w' \not\models \beta_{v',w'}$. Call A the set of pairs of the first type, and B the set of pairs of the second type.

$$\alpha = \begin{cases} \bigcap_{(v',w') \in A} \alpha_{v',w'} & \text{if } A \neq \emptyset; \\ {\downarrow}^n & \text{otherwise.} \end{cases} \qquad \text{and} \qquad \beta = \begin{cases} \bigcap_{(v',w') \in B} \beta_{v',w'} & \text{if } B \neq \emptyset; \\ {\downarrow}^m & \text{otherwise.} \end{cases}$$

Now, by Proposition 39, there are only finitely many non-equivalent path expressions of downward depth at most ℓ, so the intersections that define α and β can be considered finite. Notice that by Fact 31 we may take all the $\alpha_{v',w'}$ involved in simple normal form, and they will all have the same length (namely, n, the distance from x to v). An analog argument holds for the $\beta_{v',w'}$ expressions. Therefore, it makes sense to take the operation $\cap$ among all the $\alpha_{v',w'}$ and among all the $\beta_{v',w'}$. Let $\psi = \langle \alpha = \beta \rangle$. By construction, $\mathcal{T}, x \models \psi$, and so $\mathcal{T}, x, y \models [\psi]{\downarrow}^k$. Furthermore, since A or B are nonempty, $\mathcal{T}', x' \not\models \psi$, and so $\mathcal{T}', x', y' \not\models [\psi]{\downarrow}^k$. Since $\mathrm{dd}(\psi) \leq j$ (by Fact 37) and $k \leq j$ we have $\mathrm{dd}([\psi]{\downarrow}^k) = \max\{\mathrm{dd}(\psi), k\} \leq j$. Hence $\mathcal{T}, x, y \not\equiv_j \mathcal{T}', x', y'$, which is a contradiction. This concludes the proof. $\square$

The following corollary shows that binary downward bisimulations subsume unary ones.

Corollary 48. *$\mathcal{T}, x \equiv^{\downarrow} \mathcal{T}', x'$ iff $\mathcal{T}, x, x \equiv^{\downarrow} \mathcal{T}', x', x'$. Thus, if $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching, then $\mathcal{T}, x \mathrel{\underline{\leftrightarrow}}^{\downarrow} \mathcal{T}', x'$ iff $\mathcal{T}, x, x \mathrel{\underline{\leftrightarrow}}^{\downarrow} \mathcal{T}', x', x'$.*

Proof. The second part follows from the first part, item 1 of Theorem 45 and the corresponding result for nodes [10].

For the left-to-right implication, let $\mathcal{T}, x \equiv^{\downarrow} \mathcal{T}', x'$. Take $\alpha = [\varphi_0] \downarrow \dots \downarrow [\varphi_n]$ (we can assume α has this form from Lemma 33 and Fact 31). Suppose $\mathcal{T}, x, x \models \alpha$ and let us see that $\mathcal{T}, x', x' \models \alpha$ (the other implication is analogous). We have $n = 0$ and thus $\alpha = [\varphi_0]$, so $\mathcal{T}, x \models \varphi_0$. Then $\mathcal{T}', x' \models \varphi_0$, and $\mathcal{T}', x', x' \models [\varphi_0]$.

For the right-to-left implication, assume $\mathcal{T}, x, x \equiv^{\downarrow} \mathcal{T}', x', x'$. In particular, $\mathcal{T}, x, x \models [\varphi]$ iff $\mathcal{T}', x', x' \models [\varphi]$. Since $\mathcal{T}, x, x \models [\varphi]$ iff $\mathcal{T}, x \models \varphi$ and $\mathcal{T}', x', x' \models [\varphi]$ iff $\mathcal{T}', x' \models \varphi$, we arrive to $\mathcal{T}, x \models \varphi$ iff $\mathcal{T}', x' \models \varphi$, as we wanted. ☐

4.1.4 Characterization for XPath$_=(\downarrow)$ paths

In this section we show that for each formula $\varphi(x, y)$ of first order, over the appropriate signature and with two free variables x and y: there is a path expression α of XPath$_=(\downarrow)$ such that $\mathrm{Tr}_{x,y}(\alpha)$ is equivalent to $\varphi(x, y)$ if and only if φ is a 'forward property' (defined below), and it is bisimulation-invariant over data trees. We begin with some definitions.

We say that $\varphi(x, y) \in \mathrm{FO}(\sigma)$ is $\underline{\leftrightarrow}^{\downarrow}$**-invariant** (resp. $\underline{\leftrightarrow}^{\downarrow}_{\ell}$**-invariant**) if for all data trees $\mathcal{T}$ and $\mathcal{T}'$, $u \xrightarrow{*} v$ (resp. $u \xrightarrow{\leq \ell} v$) in $\mathcal{T}$, $u' \xrightarrow{*} v'$ (resp. $u' \xrightarrow{\leq \ell} v'$) in $\mathcal{T}'$, and $\mathcal{T}, u, v \,\underline{\leftrightarrow}^{\downarrow}\, \mathcal{T}', u', v'$ (resp. $\mathcal{T}, u, v \,\underline{\leftrightarrow}^{\downarrow}_{\ell}\, \mathcal{T}', u', v'$) we have $\mathcal{T} \models \varphi[u, v]$ iff $\mathcal{T}' \models \varphi[u', v']$.

A first-order σ-formula $\varphi(x, y)$ is said to be a **forward property** if for every σ-structure $\mathcal{A}$ and $u, v \in A$, we have that $\mathcal{A} \models \varphi(u, v)$ implies $u \rightsquigarrow^* v$ in $\mathcal{A}$. By Compactness, $\varphi(x, y)$ is a forward property iff there is k such that $\mathcal{A} \models \varphi(u, v)$ implies $u \rightsquigarrow^{\leq k} v$ in $\mathcal{A}$. In this case we say that $\varphi(x, y)$ is a k**-forward property**.

Recall that for a data tree $\mathcal{T}$ and $u \in T$, we denote by $\mathcal{T}|_{\ell} u$ the subtree of $\mathcal{T}$ induced by $\{v \in T \mid (\exists n \leq \ell)\ u \xrightarrow{n} v\}$. Let $k \leq \ell$. We say that a first-order formula $\varphi(x, y)$ with two free variables is (k, ℓ)**-local** whenever $\mathcal{T} \models \varphi[u, v]$ iff $\mathcal{T}|_{\ell} u \models \varphi[u, v]$ for all $(u, v) \in D_k(\mathcal{T})$.

We now state some lemmas that will be used for the proof.

Lemma 49. *Let $\varphi(x, y) \in \mathrm{FO}(\sigma)$ be $\underline{\leftrightarrow}^{\downarrow}$-invariant over [finite] data trees. Then for each k there is ℓ (large enough, depending on the quantifier rank of φ and k) such that φ is (k, ℓ)-local.*

Proof. A straightforward modification of the proof in [10, Prop 6.2], which, in turn, follows Otto's idea [20]. ☐

Lemma 50. *If $\varphi(x, y) \in \mathrm{FO}(\sigma)$ is a k-forward property, $\underline{\leftrightarrow}^{\downarrow}$-invariant over [finite] data trees and (k, ℓ)-local, then $\varphi(x, y)$ is $\underline{\leftrightarrow}^{\downarrow}_{\ell}$-invariant.*

Proof. Since $\varphi(x, y)$ is k-forward, it suffices to show that for $\mathcal{T}, u, v$ and $\mathcal{T}', u', v'$ such that $\mathcal{T}, u, v \,\underline{\leftrightarrow}^{\downarrow}_{\ell}\, \mathcal{T}', u', v'$ and $u \xrightarrow{\leq k} v$ (and so $u' \xrightarrow{\leq k} v'$) we have $\mathcal{T} \models \varphi[u, v]$ iff $\mathcal{T}' \models \varphi[u', v']$.

Now for such $\mathcal{T}, u, v$ and $\mathcal{T}', u', v'$ we have

$$\begin{aligned} \mathcal{T}, u, v \,\underline{\leftrightarrow}^{\downarrow}_{\ell}\, \mathcal{T}', u', v' &\text{ iff } (\mathcal{T}|_{\ell} u), u, v \,\underline{\leftrightarrow}^{\downarrow}_{\ell}\, (\mathcal{T}'|_{\ell} u'), u', v' && \text{(Prop. 43)} \\ &\text{ iff } (\mathcal{T}|_{\ell} u), u, v \,\underline{\leftrightarrow}^{\downarrow}\, (\mathcal{T}'|_{\ell} u'), u', v'. && \text{(Prop. 44)} \end{aligned}$$

By (k, ℓ)-locality, we have $\mathcal{T} \models \varphi[u, v]$ iff $\mathcal{T}|_{\ell} u \models \varphi[u, v]$. By $\underline{\leftrightarrow}^{\downarrow}$-invariance, $\mathcal{T}|_{\ell} u \models \varphi[u, v]$ iff $\mathcal{T}'|_{\ell} u' \models \varphi[u', v']$ and by (k, ℓ)-locality again, $\mathcal{T} \models \varphi[u, v]$ iff $\mathcal{T}' \models \varphi[u', v']$. ☐

Lemma 51. *If $\varphi(x, y) \in \mathrm{FO}(\sigma)$ is a k-forward property which is $\underline{\leftrightarrow}^{\downarrow}_{\ell}$-invariant over [finite] data trees, then there is an XPath$_=(\downarrow)$ path expression δ such that $\mathrm{dd}(\delta) \leq \ell$ and for all [finite] data trees $\mathcal{T}$ and $u, v \in T$ we have $\mathcal{T}, u, v \models \delta$ iff $\mathcal{T} \models \varphi[u, v]$.*

Proof. By Corollary 40, for every $\mathcal{T}, u, v$, with $u \xrightarrow{\leq \ell} v$, there is an ℓ-XPath$_=(\downarrow)$ path expression $\gamma_{\ell, \mathcal{T}, u, v}$ such that $\mathcal{T}, u, v \equiv^{\downarrow}_{\ell} \mathcal{T}', u', v'$ iff $\mathcal{T}', u', v' \models \gamma_{\ell, \mathcal{T}, u, v}$. Let

$$\delta = \bigcup_{\mathcal{T} \models \varphi[u, v]} \gamma_{\ell, \mathcal{T}, u, v}.$$

Since $\gamma_{\ell,\mathcal{T},u,v} \in \ell$-XPath$_=(\downarrow)$ and, by Proposition 39, $\equiv_\ell^\downarrow$ has finite index, it follows that δ is equivalent to a finite union.

We now show that $\varphi \equiv \mathrm{Tr}_{x,y}(\delta)$. Let us see that $\varphi \models \mathrm{Tr}_{x,y}(\delta)$. Suppose $\mathcal{T} \models \varphi[u,v]$. Since $\varphi(x,y)$ is a k-forward property, we have $u \xrightarrow{n} v$ for some $n \leq k \leq \ell$. Since $\mathcal{T}, u, v \models \gamma_{\ell,\mathcal{T},u,v}$, we have $\mathcal{T}, u, v \models \delta$ and so $\mathcal{T} \models \mathrm{Tr}_{x,y}(\delta)[u,v]$. Let us now see that $\mathrm{Tr}_{x,y}(\delta) \models \varphi$. Assume $\mathcal{T} \models \mathrm{Tr}_{x,y}(\delta)[u,v]$, and so $\mathcal{T}, u, v \models \delta$. Then there exists $\mathcal{T}', u', v'$ such that $\mathcal{T}' \models \varphi[u',v']$ and $\mathcal{T}, u, v \models \gamma_{\ell,\mathcal{T}',u',v'}$. By the property of $\gamma_{\ell,\mathcal{T}',u',v'}$, we have $\mathcal{T}, u, v \equiv_\ell^\downarrow \mathcal{T}', u', v'$ and since φ is $\leftrightarrow_\ell^\downarrow$-invariant (and hence $\equiv_\ell^\downarrow$-invariant by Theorem 45 we conclude $\mathcal{T} \models \varphi[u,v]$. □

The main result has two readings: one classical, and one restricted to finite models.

Theorem 52 (Characterization)**.** *Let $\varphi(x,y) \in \mathrm{FO}(\sigma)$. The following are equivalent:*

(i) φ is a forward property $\leftrightarrow^\downarrow$-invariant over [finite] data trees.

(ii) φ is expressible in XPath$_=(\downarrow)$.

Observe that the condition on φ to be a forward property is necessary. Indeed, if $\varphi(x,y)$ is universally valid then it is trivially $\leftrightarrow^\downarrow$-invariant over [finite] data trees, but it is clearly not XPath$_=(\downarrow)$-expressible, as its semantics include pairs of nodes with arbitrarily large distance between them, or even pairs (x,y) where y is not descendant of x.

Proof of Theorem 52. The implication (ii) $\Rightarrow$ (i) follows straightforwardly from Theorem 45. The proof of (i) $\Rightarrow$ (ii) goes as in the proof of [10, Th. 6.1], by Lemma 49, Lemma 50, and Lemma 51. □

4.2 Vertical

4.2.1 Some facts about path expressions over XPath$_=(\uparrow\downarrow)$

We use the following definitions introduced in [10]. We say that a path expression α of XPath$_=(\uparrow\downarrow)$ is **downward** [resp. **upward**] if it is of the form $\downarrow^n [\varphi]$ [resp. $[\varphi]\uparrow^n$] for some $n \geq 0$, with $\varphi \in$ XPath$_=(\uparrow\downarrow)$. We say that a path expression $\alpha^{\uparrow\downarrow}$ is in **up-down form** if either $\alpha^{\uparrow\downarrow} = \epsilon$, $\alpha^{\uparrow\downarrow} = \beta^\uparrow$, $\alpha^{\uparrow\downarrow} = \beta^\downarrow$, or $\alpha^{\uparrow\downarrow} = \beta^\uparrow\beta^\downarrow$, where $\beta^\uparrow$ is upward and $\beta^\downarrow$ is downward. We say that a node or path expression in XPath$_=(\uparrow\downarrow)$ is in **up-down normal form** if every path expression contained in it is up-down and every data test is of the form $\langle \epsilon \star \alpha^{\uparrow\downarrow} \rangle$, where $\alpha^{\uparrow\downarrow}$ is up-down and $\star \in \{=, \neq\}$.

Proposition 53. *[9, Prop. 12] Given a XPath$_=(\uparrow\downarrow)$ node expression φ, there is $\varphi^{\uparrow\downarrow}$ in up-down normal form such that $\varphi \equiv \varphi^{\uparrow\downarrow}$.*

Proposition 54. *[9, Lem. 13] Given a $\cup$-free XPath$_=(\uparrow\downarrow)$ path expression α, there is $\alpha^{\uparrow\downarrow}$ in up-down normal form such that $\alpha \equiv \alpha^{\uparrow\downarrow}$.*

We say that a path expression α is in $\cup$-NF (**union normal form**) if $\alpha = \beta_1 \cup \beta_2 \cup \cdots \cup \beta_n$ and the β_i are in up-down normal form (and thus $\cup$-free).

Proposition 55. *For all path expressions α in XPath$_=(\uparrow\downarrow)$, there is α' in $\cup$-NF such that $\alpha \equiv \alpha'$.*

Proof. We proceed by structural induction over α. If $\alpha = \epsilon$ or $\alpha = \downarrow$ or $\alpha = \uparrow$, the result holds trivially. If $\alpha = [\varphi]$, with φ a node expression, we can take, by Proposition 53, a node expression ψ in up-down normal form (and therefore $\cup$-free) with $\psi \equiv \varphi$. Finally, for the concatenation $\alpha = \beta\gamma$, we can assume by induction that $\beta \equiv \beta_1 \cup \cdots \cup \beta_m$, and $\gamma \equiv \gamma_1 \cup \cdots \cup \gamma_n$, with β_i, γ_i being in up-down normal form. The conclusion follows from the fact that

$$(\beta_1 \cup \cdots \cup \beta_m)(\gamma_1 \cup \cdots \cup \gamma_n) \equiv \beta_1\gamma_1 \cup \beta_1\gamma_2 \cup \cdots \cup \beta_1\gamma_n \cup \beta_2\gamma_1 \cup \cdots \cup \beta_m\gamma_n$$

and the application of Proposition 54 on the $\cup$-free path expressions $\beta_i\gamma_j$. □

From now on, we only consider the fragment of XPath$_=(\uparrow\downarrow)$ where all path expressions are in $\cup$-NF and all node expressions are in up-down normal form. Observe that, by Proposition 53 and Proposition 55, this fragment is semantically equivalent to full XPath$_=(\uparrow\downarrow)$.

Lemma 56. *Let $y \xrightarrow{n} x$, $y \xrightarrow{m} z$ in $\mathcal{T}$ and $y' \xrightarrow{n} x'$, $y' \xrightarrow{m} z'$ in $\mathcal{T}'$. If α is an XPath$_=(\uparrow\downarrow)$ path expression (in $\cup$-NF) such that $\mathcal{T}, x, z \models \alpha$ and $\mathcal{T}', x', z' \not\models \alpha$ then there is a path expression $\overline{\alpha}$ in up-down form such that $\mathcal{T}, x, z \not\models \overline{\alpha}$ and $\mathcal{T}', x', z' \models \overline{\alpha}$.*

Proof. Let $\alpha = \beta_1 \cup \beta_2 \cup \cdots \cup \beta_n$, with $\beta_i = [\varphi_i]\uparrow^{n_i}\downarrow^{m_i}[\psi_i]$. Let β_j be such that $\mathcal{T}, x, z \models \beta_j$. Since for all i we have $\mathcal{T}', x', z' \not\models \beta_i$, we have that either $\mathcal{T}', x' \not\models \langle\uparrow^{n_j}\downarrow^{m_j}\rangle$ (recall that both $y \xrightarrow{n} x$, $y \xrightarrow{m} z$, and $y' \xrightarrow{n} x'$, $y' \xrightarrow{m} z'$), or $\mathcal{T}', x' \not\models \varphi_j$, or $\mathcal{T}', z' \not\models \psi_j$. So either $\mathcal{T}', x' \models \neg\langle\uparrow^{n_j}\downarrow^{m_j}\rangle$ or $\mathcal{T}', x' \models \neg\varphi_j$ or $\mathcal{T}', z' \models \neg\psi_j$. In the first case, let $\overline{\alpha} = [\neg\langle\uparrow^{n_j}\downarrow^{m_j}\rangle]\uparrow^n\downarrow^m$, in the second case, let $\overline{\alpha} = [\neg\varphi_j]\uparrow^n\downarrow^m$, and in the third case, let $\overline{\alpha} = \uparrow^n\downarrow^m[\neg\psi_j]$. □

4.2.2 Binary bisimulation for XPath$_=(\uparrow\downarrow)$

Let $\mathcal{T}$ and $\mathcal{T}'$ be data trees. We say that $(u, v) \in T^2$ is **bisimilar** to $(u', v') \in T'^2$ for XPath$_=(\uparrow\downarrow)$ (notation: $\mathcal{T}, u, v \leftrightarrow^{\uparrow\downarrow} \mathcal{T}', u', v'$) if there is a relation $Z \subseteq T^2 \times T'^2$ such that $(u, v)Z(u', v')$ and for all $x, y \in T$ and $x', y' \in T'$ we have:

- **Harmony:** if $(x, y)Z(x', y')$ then $label(x) = label(x')$.
- **Reverse:** $(x, y)Z(x', y')$ iff $(y, x)Z(y', x')$.
- **Split-Zig:** If $(x, y)Z(x', y')$, then for all z such that $z \xrightarrow{m} x, z \xrightarrow{n} y$, there is z' such that $z' \xrightarrow{m} x', z' \xrightarrow{n} y'$, $(x, z)Z(x', z')$, and $(z, y)Z(z', y')$.
- **Split-Zag:** If $(x, y)Z(x', y')$, then for all z' such that $z' \xrightarrow{m} x', z' \xrightarrow{n} y'$, there is z such that $z \xrightarrow{m} x, z \xrightarrow{n} y$, $(x, z)Z(x', z')$, and $(z, y)Z(z', y')$.
- **Zig:** If $(x, y)Z(x', y')$, then for all z, w such that $z \xrightarrow{m} x, z \xrightarrow{n} w$, there are z', w' such that $z' \xrightarrow{m} x', z' \xrightarrow{n} w'$, $(z, w)Z(z', w')$, and $data(x) = data(w)$ iff $data(x') = data(w')$.
- **Zag:** If $(x, y)Z(x', y')$, then for all z', w' such that $z' \xrightarrow{m} x', z' \xrightarrow{n} w'$, there are z, w such that $z \xrightarrow{m} x, z \xrightarrow{n} w$, $(z, w)Z(z', w')$, and $data(x) = data(w)$ iff $data(x') = data(w')$.

Observe that any of **Split-Zig** or **Split-Zag** imply that $(x, y)Z(x', y') \Rightarrow (x, x)Z(x', x')$, and this property in conjunction with **Reverse** implies that $(x, y)Z(x', y') \Rightarrow (y, y)Z(y', y')$. We call these two implications **Endpoints**. See Figure 4 for an (incomplete) example of a XPath$_=(\uparrow\downarrow)$ bisimulation.

We say that $(x, y) \in T^2$ and $(x', y') \in T'^2$ are **equivalent for XPath$_=(\uparrow\downarrow)$ path expressions** (notation: $\mathcal{T}, x, y \equiv^{\uparrow\downarrow} \mathcal{T}', x', y'$) if for all XPath$_=(\uparrow\downarrow)$ path expressions α, we have $\mathcal{T}, x, y \models \alpha$ iff $\mathcal{T}', x', y' \models \alpha$.

Again, in the context of path expressions of XPath$_=(\uparrow\downarrow)$ we have an analog of Theorem 4 for binary bisimulations and path equivalence.

Theorem 57. *$\mathcal{T}, u, v \leftrightarrow^{\uparrow\downarrow} \mathcal{T}', u', v'$ implies $\mathcal{T}, u, v \equiv^{\uparrow\downarrow} \mathcal{T}', u', v'$. The converse also holds when $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching.*

Proof. We first show that if $\mathcal{T}, u, v \leftrightarrow^{\uparrow\downarrow} \mathcal{T}', u', v'$ then $\mathcal{T}, u, v \equiv^{\uparrow\downarrow} \mathcal{T}', u', v'$. We actually show that if $\mathcal{T}, u, v \leftrightarrow^{\uparrow\downarrow} \mathcal{T}', u', v'$ via Z, then:

1. If $(x, x)Z(x', x')$, then $\mathcal{T}, x, x \models [\varphi]$ iff $\mathcal{T}', x', x' \models [\varphi]$.
2. If $(x, y)Z(x', y')$, then $\mathcal{T}, x, y \models \alpha$ iff $\mathcal{T}', x', y' \models \alpha$.

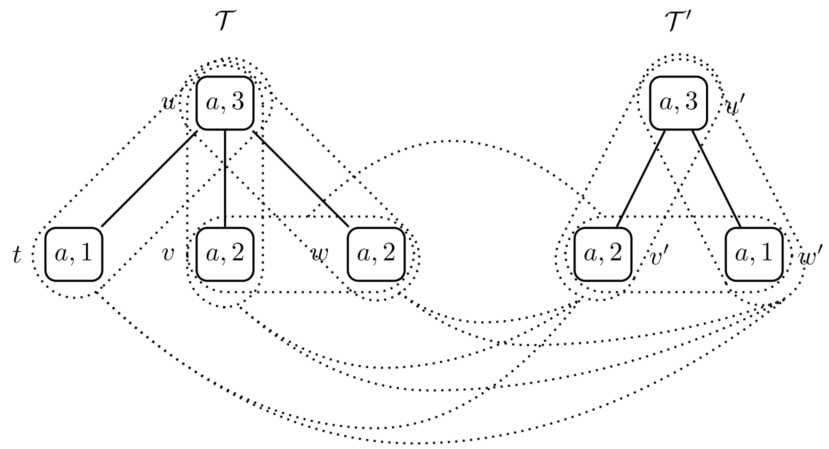

Figure 4: Part of a XPath$_=(\uparrow\downarrow)$ binary bisimulation Z between $\mathcal{T}$ and $\mathcal{T}'$. Not shown: bisimulation of the all the reverse and singleton paths, and of the other two pairs of leaves in $\mathcal{T}$.

We show 1 and 2 by structural induction on $|\varphi| + |\alpha|$. We start with the Item 1. The base case for item 1 is $\varphi = a$, for some label a. Suppose $\mathcal{T}, x, x \models [a]$. By **Harmony**, since $(x, x)Z(x', x')$, $label(x) = label(x')$, so $\mathcal{T}', x', x' \models [a]$. The case for $\mathcal{T}', x', x' \models [a]$ is identical. The boolean cases for φ are straightforward.

Now suppose $\varphi = \langle \epsilon = \alpha^{\uparrow\downarrow}\rangle$, and further assume that $\alpha^{\uparrow\downarrow} = [\psi_1]\uparrow^m\downarrow^n[\psi_2]$ (the cases with data inequality are analogous). Observe that by inductive hypothesis, it is enough to check $\mathcal{T}, x, x \models [\langle \epsilon = \uparrow^m\downarrow^n[\psi_2]\rangle]$ iff $\mathcal{T}', x', x' \models [\langle \epsilon = \uparrow^m\downarrow^n[\psi_2]\rangle]$. We show the left-to-right implication, as the reverse is analogous. So, suppose $\mathcal{T}, x, x \models \langle \epsilon = \uparrow^m\downarrow^n[\psi_2]\rangle$. There exist z, w such that $z \xrightarrow{m} x$, $z \xrightarrow{n} w$, $\mathcal{T}, x, w \models \uparrow^m\downarrow^n[\psi_2]$, and $data(x) = data(w)$. By **Zig**, there are z', w' such that $z' \xrightarrow{m} x'$, $z' \xrightarrow{n} w'$, $(x, z)Z(x', z')$, $(z, w)Z(z', w')$, and $data(x') = data(w')$. By inductive hypothesis, since $(z, w)Z(z', w')$ and $\mathcal{T}, z, w \models \downarrow^n[\psi_2]$ we have $\mathcal{T}', z', w' \models \downarrow^n[\psi_2]$. Since also $\mathcal{T}', x', z' \models \uparrow^m$, we conclude $\mathcal{T}', x', w' \models \uparrow^m\downarrow^n[\psi_2]$, and therefore (because $data(x') = data(w')$), $\mathcal{T}', x', x' \models \langle \epsilon = \uparrow^m\downarrow^n[\psi_2]\rangle$, as we wanted.

We now proceed to Item 2. We only show the left-to-right direction, as the reverse is analogous. The base case is when $\alpha \in \{\epsilon, \uparrow, \downarrow\}$. If $\alpha = \epsilon$ then $\mathcal{T}, x, y \models \alpha$ iff $x = y$. Thus, taking $z = x$ (and thus $m = n = 0$) in **Split-Zig**, it follows that $x' = y'$ and therefore $\mathcal{T}', x', y' \models \alpha$. If $\alpha = \uparrow$ then $\mathcal{T}, x, y \models \alpha$ implies $y \rightarrow x$. Now, $\mathcal{T}, y, x \models \downarrow$, and, from **Split-Zig** we deduce, $\mathcal{T}, y', x' \models \downarrow$. Therefore we conclude $\mathcal{T}, x', y' \models \alpha$, as we wanted. If $\alpha = \downarrow$, we proceed as before.

Finally, for the general case where $\alpha = \alpha^{\uparrow\downarrow}$. Suppose without loss of generality that $\mathcal{T}, x, y \models [\varphi]\uparrow^m\downarrow^n[\psi]$. Then, there exists z such that $\mathcal{T}, x, z \models [\varphi]\uparrow^m$ and $\mathcal{T}, z, y \models \downarrow^n[\psi]$. Since $z \xrightarrow{m} x$ and $z \xrightarrow{n} y$, by **Split-Zig**, we have a corresponding z' such that $z' \xrightarrow{m} x'$ and $z' \xrightarrow{n} y'$, $(x, z)Z(x', z')$, and $(z, y)Z(z', y')$. If $m = n = 0$, then $x = y$, and the problem consists of the already considered case $\mathcal{T}, x, x \models [\varphi]$. If $m \neq 0$ or $n \neq 0$, then $|[\varphi]\uparrow^m| < |\alpha|$ and $|\downarrow^n[\psi]| < |\alpha|$, and thus, since $(x, z)Z(x', z')$ and $(z, y)Z(z', y')$, we can use the inductive hypothesis to conclude that $\mathcal{T}', x', z' \models [\varphi]\uparrow^m$ and $\mathcal{T}', z', y' \models \downarrow^n[\psi]$, and therefore $\mathcal{T}', x', y' \models [\varphi]\uparrow^m\downarrow^n[\psi]$, as we wanted.

We now show that if $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching, then $\mathcal{T}, u, v \equiv^{\uparrow\downarrow} \mathcal{T}', u', v'$ implies $\mathcal{T}, u, v \leftrightarrow^{\uparrow\downarrow} \mathcal{T}', u', v'$. Let $\mathcal{T}, u, v \equiv^{\uparrow\downarrow} \mathcal{T}', u', v'$. Define the relation Z by:

$$(x, y)Z(x', y') \text{ iff } \mathcal{T}, x, y \equiv^{\uparrow\downarrow} \mathcal{T}', x', y'.$$

We show that Z is a bisimulation between $\mathcal{T}, u, v$ and $\mathcal{T}', u', v'$.

First of all, by construction, it holds that $(u, v)Z(u', v')$.

To prove **Harmony**, let $(x, y)Z(x', y')$. We will see that if $label(x) = a$ then $label(x') = a$ (the other implication is analogous). Note that, since $\mathcal{T}$ is a tree, there are m, n such that

$\mathcal{T}, x, y \models \uparrow^m\downarrow^n$. Also, if *label*$(x) = a$, $\mathcal{T}, x, y \models [a]\uparrow^m\downarrow^n$. Therefore, since $\mathcal{T}, x, y \equiv^{\uparrow\downarrow} \mathcal{T}', x', y'$, we have $\mathcal{T}', x', y' \models [a]\uparrow^m\downarrow^n$, and thus *label*$(x') = a =$ *label*(x).

Now we check **Reverse**. Let $(x, y)Z(x', y')$. Observe first that it is enough to check that $(x, y)Z(x', y') \Rightarrow (y, x)Z(y', x')$. Now, let β be a path expression, which we can assume to be in up-down normal form $\beta = [\psi] \uparrow^n \downarrow^m[\varphi]$, such that $\mathcal{T}, y, x \models \beta$. Then, $\mathcal{T}, x, y \models [\varphi] \uparrow^m \downarrow^n[\psi]$, and, since $(x, y)Z(x', y')$, this implies that $\mathcal{T}', x', y' \models [\varphi] \uparrow^m \downarrow^n[\psi]$. In turn, this implies that $\mathcal{T}', y', x' \models [\psi] \uparrow^n \downarrow^m[\varphi] = \beta$, as we wanted.

Now we check **Split-Zig** (**Split-Zag** is analogous). Let $(x, y)Z(x', y')$. We prove that if $z\xrightarrow{m}x$ and $z\xrightarrow{n}y$ then there is z' in T' such that $z'\xrightarrow{m}x'$, $z'\xrightarrow{n}y'$, $(x, z)Z(x', z')$, and $(z, y)Z(z', y')$. We have $\mathcal{T}, x, y \models \uparrow^m\downarrow^n$, and then so does $\mathcal{T}', x', y' \models \uparrow^m\downarrow^n$. In particular, there exists z' such that $z'\xrightarrow{m}x'$, $z'\xrightarrow{n}y'$. To verify $(x, z)Z(x', z')$, we see that if α is a path expression such that $\mathcal{T}, x, z \models \alpha$, then $\mathcal{T}', x', z' \models \alpha$ (the other implication is analogous). Observe that $\mathcal{T}, x, y \models \alpha\downarrow^n$, which implies that $\mathcal{T}', x', y' \models \alpha\downarrow^n$. As there is only one w such that $w\xrightarrow{n}y'$, namely z', we conclude that $\mathcal{T}', x', z' \models \alpha$, as we wanted. To verify $(z, y)Z(z', y')$, we see that if α is a path expression such that $\mathcal{T}, z, y \models \alpha$, then, $\mathcal{T}', z', y' \models \alpha$ (the other implication is analogous). Now, $\mathcal{T}, z, y \models \alpha$ implies $\mathcal{T}, x, y \models \uparrow^m\alpha$, and then $\mathcal{T}', x', y' \models \uparrow^m\alpha$. Since $\mathcal{T}'$ is a tree, this in turn implies that $\mathcal{T}', z', y' \models \alpha$, as we wanted.

For the last step, we check that **Zig** holds (**Zag** is analogous). Suppose $\mathcal{T}, x, y \equiv^{\uparrow\downarrow} \mathcal{T}', x', y'$ (that is, $(x, y)Z(x', y')$). Let z, w be such that $z\xrightarrow{m}x$, $z\xrightarrow{n}w$, and assume that *data*$(x) =$ *data*(w) (the case for $\neq$ is analogous). We want to see that there are z', w' in T' such that $z'\xrightarrow{m}x'$, $z'\xrightarrow{n}w'$, *data*$(x') =$ *data*(w'), and $(z, w)Z(z', w')$. By **Split-Zig**, let $z' \in T'$ such that $z'\xrightarrow{m}x'$ and $(x, z)Z(x', z')$. Let

$$P = \{w' \in T' \mid z'\xrightarrow{n}w' \text{ and } \mathit{data}(x') = \mathit{data}(w')\}.$$

Notice that P is finite since $\mathcal{T}'$ is **finitely branching**. We show that there is $w' \in P$ such that $(z, w)Z(z', w')$. By **Split-Zig** we had $\mathcal{T}, x, z \equiv^{\uparrow\downarrow} \mathcal{T}', x', z'$, and thus $\mathcal{T}, x, z \models [\langle\epsilon = \uparrow^m\downarrow^n\rangle]\uparrow^m$ implies $\mathcal{T}', x', z' \models [\langle\epsilon = \uparrow^m\downarrow^n\rangle]\uparrow^m$, and so there is w' such that $z'\xrightarrow{n}w'$ and *data*$(x') =$ *data*(w'). Hence $P \neq \emptyset$.

Now, suppose by the way of contradiction that for all $w' \in P$, we have $\mathcal{T}, z, w \not\equiv^{\uparrow\downarrow} \mathcal{T}', z', w'$. That is, for every $w' \in P$, there exists a path expression, which we can assume is in up-down form $\alpha_{w'} = [\varphi_{w'}]\uparrow^{a_{w'}}\downarrow^{b_{w'}}[\psi_{w'}]$, such that either

1. $\mathcal{T}, z, w \models \alpha_{w'}$ and $\mathcal{T}', z', w' \not\models \alpha_{w'}$, or
2. $\mathcal{T}, z, w \not\models \alpha_{w'}$ and $\mathcal{T}', z', w' \models \alpha_{w'}$.

First we are going to see that we can assume that $\alpha_{w'}$ is of the form $[\varphi_{w'}]\downarrow^n[\psi_{w'}]$. First of all, observe that since $\mathcal{T}, x, z \equiv^{\uparrow\downarrow} \mathcal{T}', x', z'$, by **Endpoints** we have that $(z, z)Z(z', z')$. Now suppose by the way of contradiction that $\uparrow^{a_{w'}}\downarrow^{b_{w'}}$ holds in $\mathcal{T}, z, w$ but not in $\mathcal{T}', z', w'$ (the other case is analogous). Since $z\xrightarrow{n}w$, it must be that $b_{w'} - a_{w'} = n$. Since also $z'\xrightarrow{n}w'$ but $\mathcal{T}', z', w' \not\models \uparrow^{a_{w'}}\downarrow^{b_{w'}}$, we have $\mathcal{T}', z', z' \not\models [\langle\uparrow^{a_{w'}}\downarrow^{b_{w'}}\rangle]$, or, equivalently, $\mathcal{T}', z', z' \models [\neg\langle\uparrow^{a_{w'}}\downarrow^{b_{w'}}\rangle]$. But then $\mathcal{T}, z, z \models [\neg\langle\uparrow^{a_{w'}}\downarrow^{b_{w'}}\rangle]$, and this implies that $\mathcal{T}, z, w \not\models \uparrow^{a_{w'}}\downarrow^{b_{w'}}$, a contradiction. So we can assume without loss of generality that always $\uparrow^{a_{w'}}\downarrow^{b_{w'}} = \downarrow^n$.

Now, by Lemma 56, we can always assume that case 1 applies. We take

$$\alpha = [\bigwedge_{w' \in P} \varphi_{w'}]\downarrow^n[\bigwedge_{w' \in P} \psi_{w'}]\uparrow^n$$

and observe that $\mathcal{T}, z, z \models \alpha$ but $\mathcal{T}', z', z' \not\models \alpha$, a contradiction. $\square$

Corollary 58. *$\mathcal{T}, x \equiv^{\uparrow\downarrow} \mathcal{T}', x'$ iff $\mathcal{T}, x, x \equiv^{\uparrow\downarrow} \mathcal{T}', x', x'$. Thus, if $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching, then $\mathcal{T}, x \leftrightarroweq^{\uparrow\downarrow} \mathcal{T}', x'$ iff $\mathcal{T}, x, x \leftrightarroweq^{\uparrow\downarrow} \mathcal{T}', x', x'$.*

Proof. The proof is similar to that of Corollary 48. For the second part we use that if $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching, then $\underline{\leftrightarrow}^{\uparrow\downarrow}$ and $\equiv^{\uparrow\downarrow}$ coincide (Theorem 57)

Assume first that $x \equiv^{\uparrow\downarrow} x'$. Suppose that $\mathcal{T}, x, x \models \alpha$ and let us prove that $\mathcal{T}', x', x' \models \alpha$ (the other implication is analogous). Without loss of generality we can assume that $\alpha = [\varphi]\uparrow^m\downarrow^n[\psi]$. So $n = m$, $\mathcal{T}, x \models \varphi$, and $\mathcal{T}, x \models \psi$. Since $\mathcal{T}, x \equiv^{\uparrow\downarrow} \mathcal{T}', x'$, we conclude that $\mathcal{T}', x', x' \models [\varphi]\uparrow^m\downarrow^n[\psi]$.

For the other implication, assume $x, x \equiv^{\uparrow\downarrow} x', x'$. In particular, $\mathcal{T}, x, x \models [\varphi]$ iff $\mathcal{T}', x', x' \models [\varphi]$. As $\mathcal{T}, x, x \models [\varphi]$ iff $\mathcal{T}, x \models \varphi$ and $\mathcal{T}', x', x' \models [\varphi]$ iff $\mathcal{T}', x' \models \varphi$, we get $\mathcal{T}, x \models \varphi$ iff $\mathcal{T}', x' \models \varphi$, as we wanted. ☐

5 Definability via path expressions

5.1 Saturation

Saturation for the downward fragment. Let Σ and Γ be sets of XPath$_=(\downarrow)$-path expressions. Given a data tree $\mathcal{T}$ and $u \in T$, we say that Σ and Γ are **$=^\downarrow$-satisfiable** [resp. **$\neq^\downarrow$-satisfiable**] at $\mathcal{T}, u$ if there exist $v, w \in T$ such that $\mathcal{T}, u, v \models \Sigma$, $\mathcal{T}, u, w \models \Gamma$, and $data(v) = data(w)$ [resp. $data(v) \neq data(w)$]. We say that Σ and Γ are **$=^\downarrow$-finitely satisfiable** [resp. **$\neq^\downarrow$-finitely satisfiable**] at $\mathcal{T}, u$ if for every finite $\Sigma' \subseteq \Sigma$ and finite $\Gamma' \subseteq \Gamma$, we have that Σ' and Γ' are $=^\downarrow$-satisfiable [resp. $\neq^\downarrow$-satisfiable] at $\mathcal{T}, u$.

Definition 59. We say that a data tree $\mathcal{T}$ is **binary $\downarrow$-saturated** if for every pair of sets Σ, Γ of XPath$_=(\downarrow)$-path expressions, every $u \in T$, and $\star \in \{=, \neq\}$, the following is true:

if Σ and Γ are $\star^\downarrow$-finitely satisfiable at $\mathcal{T}, u$ then Σ and Γ are $\star^\downarrow$-satisfiable at $\mathcal{T}, u$.

Proposition 60. *Let $\mathcal{T}$ and $\mathcal{T}'$ be $\downarrow$-saturated data trees, and let $u, v \in T$ and $u', v' \in T'$. If $\mathcal{T}, u, v \equiv^\downarrow \mathcal{T}', u', v'$, then $\mathcal{T}, u, v \underline{\leftrightarrow}^\downarrow \mathcal{T}', u', v'$.*

Proof. We show that Z, defined by $(x, y)Z(x', y')$ iff $\mathcal{T}, x, y \equiv^\downarrow \mathcal{T}', x', y'$ is a $\downarrow$-bisimulation between $\mathcal{T}, u, v$ and $\mathcal{T}', u', v'$. Clearly $(u, v)Z(u', v')$, and **Harmony** also holds. For **Equidistance**, if $(x, y)Z(x', y')$, assume $x \xrightarrow{k} y$. Then, since $(x, y) \equiv^\downarrow (x', y')$, $\mathcal{T}, x, y \models \downarrow^k$ iff $\mathcal{T}', x', y' \models \downarrow^k$. For **Split**, let $(x, y)Z(x', y')$, $x \xrightarrow{n} z \xrightarrow{m} y$, and $x' \xrightarrow{n} z' \xrightarrow{m} y'$. We check first that $(x, z)Z(x', z')$: $\mathcal{T}, x, z \models \alpha \Leftrightarrow \mathcal{T}, x, y \models \alpha\downarrow^m \Leftrightarrow \mathcal{T}', x', y' \models \alpha\downarrow^m \Leftrightarrow \mathcal{T}', x', z' \models \alpha$. The proof is similar for checking $(z, y)Z(z', y')$: $\mathcal{T}, z, y \models \alpha \Leftrightarrow \mathcal{T}, x, y \models \downarrow^n \alpha \Leftrightarrow \mathcal{T}', x', y' \models \downarrow^n \alpha \Leftrightarrow \mathcal{T}', z', y' \models \alpha$.

We now need to show that **Zig** and **Zag** are satisfied. We see only **Zig**, as **Zag** is analogous. Suppose $(x, y)Z(x', y')$, $x \xrightarrow{m} a$, $x \xrightarrow{n} b$ and $data(a) = data(b)$ (the case with $data(a) \neq data(b)$ is analogous). Let

$$\Sigma = \{\alpha \mid \mathcal{T}, x, a \models \alpha \text{ and } \alpha \text{ is } \cup\text{-free}\} \quad \text{and} \quad \Gamma = \{\alpha \mid \mathcal{T}, x, b \models \alpha \text{ and } \alpha \text{ is } \cup\text{-free}\}.$$

That is, Σ and Γ are the $\cup$-free theories of $\mathcal{T}, x, a$ and $\mathcal{T}, x, b$, respectively. Furthermore, let Σ' be a finite subset of Σ, and let Γ' be a finite subset of Γ. Observe that, being in XPath$_=(\downarrow)$, all path expressions in Γ and Σ are of the same length, and thus we have a notion of intersection as in Equation 7.

Now define $\varphi = \langle \cap\Sigma' = \cap\Gamma' \rangle$. Observe that, from **Split**, $(x, y) \equiv^\downarrow (x', y')$ implies $(x, x) \equiv^\downarrow (x', x')$ implies (Corollary 48) $x \equiv^\downarrow x'$. Now, it is clear that $\mathcal{T}, x \models \varphi$, and thus $\mathcal{T}', x' \models \varphi$. Therefore, there exist a', b' such that $\mathcal{T}', x', a' \models \Sigma'$ (in particular, $x' \xrightarrow{m} a'$), $\mathcal{T}', x', b' \models \Gamma'$ (in particular, $x' \xrightarrow{n} b'$), and $data(a') = data(b')$. Hence Σ' and Γ' are $=^\downarrow$-satisfiable at x', for *any* finite sets Σ', Γ' and thus Σ and Γ are $=^\downarrow$-finitely satisfiable at x'. Since $\mathcal{T}'$ is $\downarrow$-saturated, this implies that Σ and Γ are $=^\downarrow$-satisfiable at x', for some a' and b'.

Finally, we see that $\mathcal{T}', x', a' \models \Sigma$ implies that $\mathrm{Th}_\downarrow(\mathcal{T}, x, a) = \mathrm{Th}_\downarrow(\mathcal{T}', x', a')$ and thus $(x, a) \equiv^\downarrow (x', a')$ (the case for $(x, b) \equiv^\downarrow (x', b')$ is analogous). We are only going to prove that $\mathcal{T}', x', a' \models \alpha \Rightarrow \mathcal{T}, x, a \models \alpha$, as the other implication is clear. Suppose by way of contradiction that there is an α, which by Lemma 33 can be assumed to be $\cup$-free, such that $\mathcal{T}', x', a' \models \alpha$ but $\mathcal{T}, x, a \not\models \alpha$. Then, by Lemma 34, there is a $\cup$-free path expression $\bar{\alpha}$ such that $\mathcal{T}', x', a' \not\models \bar{\alpha}$ and $\mathcal{T}, x, a \models \bar{\alpha}$. Then, since $\mathcal{T}', x', a' \models \Sigma$, we have that $\mathcal{T}', x', a' \models \bar{\alpha}$, a contradiction. ☐

Saturation for the vertical fragment. Given a data tree $\mathcal{T}$ and $u \in T$, we say that the set of XPath$_=(\uparrow\downarrow)$-path expressions Γ is **$=^{\uparrow\downarrow}$-satisfiable** [resp. **$\neq^{\uparrow\downarrow}$-satisfiable**] at $\mathcal{T}, u$ if there exist $v, w \in T$ such that $v \xrightarrow{*} u$, $v \xrightarrow{*} w$, $\mathcal{T}, u, w \models \Gamma$ and $data(u) = data(w)$ [resp. $data(u) \neq data(w)$]. We say that Γ is **$=^{\uparrow\downarrow}$-finitely satisfiable** [resp. **$\neq^{\uparrow\downarrow}$ -finitely satisfiable**] at $\mathcal{T}, u$ if for every finite Γ', we have that Γ' is $=^{\uparrow\downarrow}$-satisfiable [resp. $=^{\uparrow\downarrow}$-satisfiable] at $\mathcal{T}, u$.

Definition 61. We say that a data tree $\mathcal{T}$ is **binary $\uparrow\downarrow$-saturated** if for every set of XPath$_=(\uparrow\downarrow)$-path expressions Γ, every $u \in T$ and $\star \in \{=, \neq\}$, the following is true:

if Γ is $\star^{\uparrow\downarrow}$-finitely satisfiable at $\mathcal{T}, u$ then Γ is $\star^{\uparrow\downarrow}$-satisfiable at $\mathcal{T}, u$.

Proposition 62. *Let $\mathcal{T}$ and $\mathcal{T}'$ be binary $\uparrow\downarrow$-saturated data trees, and let $u, v \in T$ and $u', v' \in T'$. If $\mathcal{T}, u, v \equiv^{\uparrow\downarrow} \mathcal{T}', u', v'$, then $\mathcal{T}, u, v \leftrightarrows^{\uparrow\downarrow} \mathcal{T}', u', v'$.*

Proof. We show that $Z \subseteq T^2 \times T'^2$, defined by

$$(x, y)Z(x', y') \text{ iff } \mathcal{T}, x, y \equiv^{\uparrow\downarrow} \mathcal{T}', x', y'$$

is a $\uparrow\downarrow$-bisimulation between $\mathcal{T}, u, v$ and $\mathcal{T}', u', v'$. Clearly $(u, v)Z(u', v')$. **Harmony**, **Reverse**, **Split-Zig**, and **Split-Zag** hold with the same proofs as in the second part of the proof of Theorem 57.

We now need to show that **Zig** and **Zag** are satisfied. We see only **Zig**, as **Zag** is analogous.

Suppose $(x, y)Z(x', y')$, $s \xrightarrow{a} x$, $s \xrightarrow{b} y$, $z \xrightarrow{m} x$, $z \xrightarrow{n} w$, and $data(x) = data(w)$ (the case $\neq$ is analogous). We want to see that there are $z', w' \in T'$ such that $z' \xrightarrow{m} x'$, $z' \xrightarrow{n} w'$, $(z, w)Z(z', w')$, and $data(x') = data(w')$. Let

$$\Gamma = \{\beta \mid \mathcal{T}, x, w \models \beta \text{ and } \beta \text{ is of the form } [\varphi] \uparrow^m\downarrow^n [\psi], \text{ for some } \varphi \text{ and } \psi\},$$

and let Γ' be a finite subset of Γ. If $\beta_1 = [\varphi_1] \uparrow^m\downarrow^n [\psi_1]$ and $\beta_2 = [\varphi_2] \uparrow^m\downarrow^n [\psi_2]$, we will define $\beta_1 \cap \beta_2 = [\varphi_1 \cap \varphi_2] \uparrow^m\downarrow^n [\psi_1 \cap \psi_2]$.

Now, define

$$\alpha = [\langle \varepsilon = \cap \Gamma' \rangle] \uparrow^a\downarrow^b .$$

It can be seen that $\mathcal{T}, x, y \models \alpha$, and thus, since by definition of Z we have $\mathcal{T}, x, y \equiv^{\uparrow\downarrow} \mathcal{T}', x', y'$, we conclude $\mathcal{T}', x', y' \models \alpha$. This implies that there are p', q' such that $p' \xrightarrow{m} x'$, $p' \xrightarrow{n} q'$, $data(x') = data(q')$, and $\mathcal{T}', x', q' \models \Gamma'$. Therefore, Γ is $=^{\uparrow\downarrow}$-finitely satisfiable at $\mathcal{T}', x'$. Since $\mathcal{T}'$ is binary $\uparrow\downarrow$-saturated, this implies that Γ is $=^{\uparrow\downarrow}$-satisfiable at $\mathcal{T}', x'$, and therefore there exist nodes $z', w' \in T'$ such that $t \xrightarrow{m} x'$, $t \xrightarrow{n} w'$, $data(x') = data(w')$, and $\mathcal{T}', x', w' \models \Gamma$.

It remains to prove that $\mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, x, w) = \mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}', x', w')$, as this property in conjunction with **Split-Zig** will imply that $(z, w)Z(z', w')$.

First we prove that $\mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, x, w) \subset \mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}', x', w')$. Let $\beta \in \mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, x, w)$. Without loss of generality, we can assume that β is $\cup$-free, and thus of the form $\beta = [\varphi] \uparrow^j\downarrow^k [\psi]$. Since $z' \xrightarrow{m} x'$ and $z' \xrightarrow{n} w'$, $\mathcal{T}', x', w' \models \beta$ iff $\mathcal{T}', x', w' \models \gamma$, with $\gamma = [\varphi \wedge \langle \uparrow^j\downarrow^k \rangle] \uparrow^m\downarrow^n [\psi]$. But $\gamma \in \Gamma$, and thus $\mathcal{T}', x', w' \models \gamma$.

We now see that $\mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}', x', w') \subset \mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, x, w)$. Suppose by way of contradiction that there is a β (which can be assumed to be $\cup$-free) such that $\beta = [\varphi] \uparrow^j\downarrow^k [\psi]$ and $\mathcal{T}', x', w' \models \beta$ but $\mathcal{T}, x, w \not\models \beta$. As $z \xrightarrow{m} x$ and $z \xrightarrow{n} w$, $\mathcal{T}, x, w \models \beta$ iff $\mathcal{T}, x, w \models [\varphi \wedge \langle \uparrow^j\downarrow^k \rangle] \uparrow^m\downarrow^n [\psi]$, and as $z' \xrightarrow{m} x'$ and $z' \xrightarrow{n} w'$, we also have $\mathcal{T}', x', w' \models \beta$ iff $\mathcal{T}', x', w' \models [\varphi \wedge \langle \uparrow^j\downarrow^k \rangle] \uparrow^m\downarrow^n [\psi]$. So from our supposition we have $\mathcal{T}, x, w \not\models [\varphi \wedge \langle \uparrow^j\downarrow^k \rangle] \uparrow^m\downarrow^n [\psi]$. Since $\mathcal{T}, x, w \models \uparrow^m\downarrow^n$, it must be that either $\mathcal{T}, x, w \models [\neg(\varphi \wedge \langle \uparrow^j\downarrow^k \rangle)] \uparrow^m\downarrow^n$ or $\mathcal{T}, x, w \models \uparrow^m\downarrow^n [\neg\psi]$. But since $\mathcal{T}', x', w' \models \Gamma$, this implies that either $\mathcal{T}', x', w' \models [\neg(\varphi \wedge \langle \uparrow^j\downarrow^k \rangle)] \uparrow^m\downarrow^n$ or $\mathcal{T}', x', w' \models \uparrow^m\downarrow^n [\neg\psi]$, which contradicts the fact that $\mathcal{T}', x', w' \models [\varphi \wedge \langle \uparrow^j\downarrow^k \rangle] \uparrow^m\downarrow^n [\psi]$.

So we have $(x, w)Z(x', w')$. As $z \xrightarrow{m} x$ and $z \xrightarrow{n} w$, we can use **Split-Zig** to finally obtain $(z, w)Z(z', w')$, as we wanted.

□

5.2 Weak Data Trees and Quasi-ultraproducts

The following proposition shows the 'local' aspect of XPath$_=(\downarrow)$ and XPath$_=(\uparrow\downarrow)$ for paths, whereas Proposition 12 showed it for nodes. It is stated in terms of first-order because models are weak data trees.

Proposition 63. *Let $\mathcal{T}$ be a weak data tree and let both $r \rightsquigarrow^* u$ and $r \rightsquigarrow^* v$ in $\mathcal{T}$.*

1. *If α is a XPath$_=(\downarrow)$-path expression then $\mathcal{T} \models \mathrm{Tr}_{x,y}(\alpha)[u,v]$ iff $\mathcal{T}|r \models \mathrm{Tr}_{x,y}(\alpha)[u,v]$.*
2. *If r is the root of $\mathcal{T}$ and $\alpha \in$ XPath$_=(\uparrow\downarrow)$ then $\mathcal{T} \models \mathrm{Tr}_{x,y}(\alpha)[u,v]$ iff $\mathcal{T}|r \models \mathrm{Tr}_{x,y}(\alpha)[u,v]$.*

We now show that 2-saturated data trees are already both binary $\downarrow$-saturated and binary $\uparrow\downarrow$-saturated. For technical reasons we state these results in the more general setting of weak data trees.

Proposition 64. *Let $\mathcal{T}$ be a 2-saturated weak data tree and $r \in T$.*

1. *$\mathcal{T}|r$ is a binary $\downarrow$-saturated data tree.*
2. *If r is the root of T then $\mathcal{T}|r$ is a binary $\uparrow\downarrow$-saturated data tree.*

Proof. The proof goes as the proof of Proposition 14. □

Let $(\mathcal{T}_i, u_i, v_i)_{i\in I}$ be a family of two-pointed data trees. The ultraproduct of such *two-pointed* data trees is defined, as usual, by $(\prod_U \mathcal{T}_i, u^*, v^*)$, where u^* and v^* are the ultralimits of $(u_i)_{i\in I}$ and $(u_i)_{i\in I}$ modulo U, respectively.

Example 65. For $i \in \mathbb{N}$, let $\mathcal{T}_i$ be any data tree of height at least i, and let u_i, v_i be any pair of nodes of T_i at distance i from each other. Let $\rho_n(x,y)$ be the first-order property "x is at distance at least n from y". It is clear that $\mathcal{T}_m \models \rho_n[u_m, v_m]$ for every $m \geq n$. Let u^* and v^* be the ultralimits of $(u_i)_{i\in I}$ and $(v_i)_{i\in I}$ modulo U. Since $\{m \mid m \geq n\} \in U$ for any non-principal U, we conclude that $\prod_U \mathcal{T}_i \models \rho_n[u^*, v^*]$ for every n, and so u^* is disconnected from v^* in $\prod_U \mathcal{T}_i$.

Hence, in general, two-pointed data trees are not closed under $\uparrow\downarrow$-quasi ultraproduct.

Let $k \geq 0$, let $\mathcal{T}$ be a data tree and let $u, v \in \mathcal{T}$. We say that $(\mathcal{T}, u, v)$ is a ***k*-bounded two-pointed data tree** if u, v are at distance at most k from the root of $\mathcal{T}$. In particular, if r is the root of $\mathcal{T}$ then $(\mathcal{T}, r, r)$ is a 0-bounded two-pointed data tree.

Let $n \geq 0$, let $\mathcal{T}$ be a data tree and let $u, v \in \mathcal{T}$. We say that a two-pointed data tree $\mathcal{T}, u, v$ is ***n*-two-pointed** if the minimum distance between u and v is at most n. That is, if w is the first common ancestor of u and v (i.e. the closest common ancestor), and $w \xrightarrow{c} u, w \xrightarrow{d} v$, then $c + d \leq n$.

Definition 66. Suppose $(\mathcal{T}_i, u_i, v_i)_{i\in I}$ is a family of n-two-pointed data trees, r_i is the root of $\mathcal{T}_i$, U is an ultrafilter over I, $\mathcal{T}^* = \prod_U \mathcal{T}_i$, and u^*, v^* and r^* are the ultralimits of $(u_i)_{i\in I}$, $(v_i)_{i\in I}$, and $(r_i)_{i\in I}$ modulo U respectively. Then

1. If $u_i \xrightarrow{*} v_i$, the **$\downarrow$-quasi ultraproduct** of $(\mathcal{T}_i, u_i, v_i)_{i\in I}$ modulo U is the n-two-pointed data tree $(\mathcal{T}^*|u^*, u^*, v^*)$.
2. If $(\mathcal{T}_i, u_i, v_i)_{i\in I}$ is also a family of k-bounded data trees, the **$\uparrow\downarrow$-quasi ultraproduct** of $(\mathcal{T}_i, u_i, v_i)_{i\in I}$ modulo U is the k-bounded n-two-pointed data tree $(\mathcal{T}^*|r^*, u^*, v^*)$.

Observe that in the definition of $\uparrow\downarrow$-quasi ultraproduct, u^* and v^* are effectively in $\mathcal{T}^*|r^*$ for similar reasons as those in Proposition 17.

5.3 Definability and separation

We begin with the downward fragment and we state our definability results of two-pointed data trees. Since the language of path expressions does not have complementation or negation, we will deal with a restricted class of data trees. We work with n-two-pointed data trees which are **forward**, that is, data trees of the form $\mathcal{T}, u, v$ where $u \xrightarrow{\leq n} v$. If α, β are XPath$_=(\downarrow)$-path expressions, we say that $\alpha \equiv^n \beta$ if for every forward n-two-pointed data tree $\mathcal{T}, u, v$ we have $\mathcal{T}, u, v \models \alpha$ iff $\mathcal{T}, u, v \models \beta$. For a path expression $\alpha = [\varphi_0]\downarrow \ldots \downarrow[\varphi_i]$ in simple normal form, we define the complement (over the class of forward n-two-pointed data trees) as

$$\sim_n \alpha = \begin{cases} \top & \text{if } i > n; \\ \bigcup_{0 \leq j \leq i} \downarrow^j [\neg\varphi_j] \downarrow^{i-j} \cup \bigcup_{0 \leq j \leq n, i \neq j} \downarrow^j & \text{otherwise.} \end{cases}$$

(We represent ϵ as $\downarrow^0$, and $\top$ as $\bigcup_{0 \leq j \leq n} \downarrow^j$.) $\sim_n \alpha$ is thus true for all downward paths of a length at most n and different to that of α, and for paths of the same length as α but that do not satisfy some intermediate node expression $[\varphi_j]$. So for every forward n-two-pointed data tree $\mathcal{T}, u, v$, we have $\mathcal{T}, u, v \models \alpha$ iff $\mathcal{T}, u, v \not\models \sim_n \alpha$, that is, $\sim_n \alpha$ works as a kind of path expression negation over this restricted class of data trees. Notice that it is not possible to negate path expressions without a restriction on the class of data trees.

Recall that for XPath$_=(\downarrow)$ expressions α, β in simple normal form and of the same length we have defined the intersection $\alpha \cap \beta$ in Definition 36. We extend this definition of intersection to path expressions in simple normal form, and of length at most n. Let α and β be path expressions in simple normal form, with $\text{len}(\alpha) = i$, $\text{len}(\beta) = j$. We define $\alpha \cap_n \beta$ as $\perp$ (we let $\perp$ to be $[\langle \epsilon \neq \epsilon \rangle]$) in case $i \neq j$ and as in (7) of Definition 36 otherwise. It is clear that for every forward n-two-pointed data tree $\mathcal{T}, u, v$, we have $\mathcal{T}, u, v \models \alpha \cap_n \beta$ iff $\mathcal{T}, u, v \models \alpha$ and $\mathcal{T}, u, v \models \beta$. These observations allow us to extend, *over the class of forward n-two-pointed data trees*, the operations of complement and intersection to any XPath$_=(\downarrow)$-path expression:

$$\begin{aligned} \alpha &\equiv^n \perp && (\alpha \text{ in simple normal form and } \text{len}(\alpha) > n) \\ \sim_n (\alpha \cup \beta) &\equiv^n (\sim_n \alpha) \cap_n (\sim_n \beta) && (\alpha, \beta \text{ in simple normal form and } \text{len}(\alpha), \text{len}(\beta) \leq n) \\ \sim_n (\alpha \cap_n \beta) &\equiv^n (\sim_n \alpha) \cup (\sim_n \beta) && \text{(idem)} \\ (\alpha \cup \beta) \cap_n \gamma &\equiv^n (\alpha \cap_n \gamma) \cup (\beta \cap_n \gamma) && \text{(idem)} \\ \alpha \cap_n \beta &\equiv^n \beta \cap_n \alpha && \text{(idem)} \end{aligned}$$

Therefore, when restricted to n-two pointed data trees, we can pretend to have complementation and intersection of path expressions with the standard meaning and within the language. This allows us to prove the results of definability for path expressions using the obvious modifications to the proofs of Theorems 20 and 21. It is important to remark that these results are true only when restricting the universe to forward n-two-pointed data trees. In what follows, the universe is restricted to such data trees, and the operations of closure and complement must be taken with respect to this universe.

Theorem 67. *Over n-two-pointed data trees: A class K is definable by a set of XPath$_=(\downarrow)$-path expressions iff K is closed under $\downarrow$-bisimulations and $\downarrow$-quasi ultraproducts, and $\overline{K}$ is closed under $\downarrow$-quasi ultrapowers.*

Theorem 68. *Over n-two-pointed data trees: A class K is definable by an XPath$_=(\downarrow)$-path expression iff both K and $\overline{K}$ are closed under $\downarrow$-bisimulations and $\downarrow$-quasi ultraproducts.*

Theorems 28 and 29 can also be straightforwardly adapted.

Theorem 69. *Over n-two-pointed data trees: Let K_1 and K_2 be two disjoint classes such that K_1 is closed under $\downarrow$-bisimulations and $\downarrow$-quasi ultraproducts and K_2 is closed under $\downarrow$-bisimulations and $\downarrow$-quasi ultrapowers. Then there exists a third class K which is definable by a set of XPath$_=(\downarrow)$-path expressions, contains K_1, and is disjoint from K_2.*

Theorem 70. *Over n-two-pointed data trees: Let K_1 and K_2 be two disjoint classes closed under $\downarrow$-bisimulations and $\downarrow$-quasi ultraproducts. Then there exists a third class K which is definable by an XPath$_=(\downarrow)$-path expression, contains K_1, and is disjoint from K_2.*

Let us move to the vertical fragment. Not having complementation or intersection is more cumbersome in this case. We will state definability theorems restricted to classes of special two-pointed data trees which we denote n, m, k**-two-pointed** data trees. These are two-pointed data trees $\mathcal{T}, u, v$ such that if w is the first common ancestor of u and v (i.e. the closest common ancestor) then $w \xrightarrow{n} u$, $w \xrightarrow{m} v$, and w is at distance k from the root of $\mathcal{T}$. Observe that any n, m, k-two-pointed data tree is $k + \max\{n, m\}$-bounded.

If α, β are XPath$_=(\uparrow\downarrow)$-path expressions, we say that $\alpha \equiv^{n,m,k} \beta$ if for every n, m, k-two-pointed data tree $\mathcal{T}, u, v$ we have $\mathcal{T}, u, v \models \alpha$ iff $\mathcal{T}, u, v \models \beta$. The following equivalences, which are straightforward to verify, allows us to express in XPath$_=(\uparrow\downarrow)$ the complementation $\sim_{n,m,k}$ and intersection $\cap_{n,m,k}$ over the class of n, m, k-two-pointed data trees.

$$
\begin{aligned}
[\varphi]\uparrow^{n'}\downarrow^{m'}[\psi] &\equiv^{n,m,k} \begin{cases} [\varphi]\uparrow^{n}\downarrow^{m}[\psi] & \text{if } n - n' = m - m' \text{ and } n \leq n' \leq n + k \\ \bot & \text{otherwise (where } \bot := [\langle \epsilon \neq \epsilon \rangle]\uparrow^{n}\downarrow^{m}) \end{cases} \\
\sim_{n,m,k} ([\varphi]\uparrow^{n}\downarrow^{m}[\psi]) &\equiv^{n,m,k} ([\neg\varphi]\uparrow^{n}\downarrow^{m}) \cup (\uparrow^{n}\downarrow^{m}[\neg\psi]) \\
([\varphi]\uparrow^{n}\downarrow^{m}[\psi]) \cap_{n,m,k} ([\varphi']\uparrow^{n}\downarrow^{m}[\psi']) &\equiv^{n,m,k} [\varphi \wedge \varphi']\uparrow^{n}\downarrow^{m}[\psi \wedge \psi'] \\
\sim_{n,m,k} (\alpha \cap_{n,m,k} \beta) &\equiv^{n,m,k} (\sim_{n,m,k} \alpha) \cup (\sim_{n,m,k} \beta) \\
(\alpha \cup \beta) \cap_{n,m,k} \gamma &\equiv^{n,m,k} (\alpha \cap_{n,m,k} \gamma) \cup (\beta \cap_{n,m,k} \gamma) \\
\alpha \cap_{n,m,k} \beta &\equiv^{n,m,k} \beta \cap_{n,m,k} \alpha
\end{aligned}
$$

In the last three equivalencies, α is of the form $[\varphi]\uparrow^{n}\downarrow^{m}[\psi]$ for some φ and ψ, and β is of the form $[\varphi']\uparrow^{n}\downarrow^{m}[\psi']$ for some φ' and ψ'.

In the first equivalence, the condition $n - n' = m - m'$ means that the navigation via $\uparrow^{n'}\downarrow^{m'}$ could actually connect the same nodes as $\uparrow^{n}\downarrow^{m}$, assuming the tree extends sufficiently upwards and a common ancestor is reached. The condition $n \leq n'$ assures that the upward portion of the navigation reaches at least the first common ancestor of the nodes, and the condition $n' \leq n + k$ means that $\uparrow^{n'}$ reaches at most up to the root of the tree, and not higher. Notice that if any of these conditions do not hold, the path expression $\uparrow^{n'}\downarrow^{m'}$ is always false in the context of n, m, k-two-pointed data trees. If the three conditions hold simultaneously, then $\mathcal{T}, u, v \models \uparrow^{n'}\downarrow^{m'}$ for any n, m, k-two-pointed data tree $\mathcal{T}, u, v$.

The reader can check that, as expected, we arrive to results of definability and separation for XPath$_=(\uparrow\downarrow)$ path expressions, as in Theorems 67, 68, 69, and 70, but over the class of n, m, k-two-pointed data trees and using the notions of $\uparrow\downarrow$-bisimulation and $\uparrow\downarrow$-quasi ultraproducts.

6 Applications

We list some simple applications of our theorems of definability:

A class of pointed data trees definable in first-order (over data trees) but not definable by a set of XPath$_=(\uparrow\downarrow)$-node expressions. Let K be the class of pointed data trees $(\mathcal{T}, u)$ where u is the root of $\mathcal{T}$ and $\mathcal{T}$ has some node labeled a. On the one hand, K is definable by a first-order σ-formula over the class of data trees. On the other, K is closed under XPath$_=(\uparrow\downarrow)$-bisimulations but not closed under $\uparrow\downarrow$-quasi ultraproducts: for $i \in \mathbb{N}$ define $\mathcal{T}_i$ as any tree of height i whose only node labeled a is at distance i from the root, and define u_i as the root of $\mathcal{T}_i$. By an argument similar to the one used in Example 15 one can show that if $(\mathcal{T}^{\uparrow\downarrow}, u^*)$ is any $\uparrow\downarrow$-quasi ultraproduct of $(\mathcal{T}_i, u_i)_{i \in \mathbb{N}}$ then no node of $\mathcal{T}^{\uparrow\downarrow}$ has label a. By Theorem 25, K is not definable by a set of XPath$_=(\uparrow\downarrow)$-node expression.

A class of pointed data trees definable by a single XPath$_=(\uparrow\downarrow)$-node expression but not definable by set of XPath$_=(\downarrow)$-node expressions. Let $dist_3(x)$ be the property stating that there are nodes y, z so that $x \rightarrow y \rightarrow z$ and x, y, z have pairwise distinct data values. It can be checked that the XPath$_=(\uparrow\downarrow)$-node expression $\varphi_4 = \langle \varepsilon \neq \downarrow\downarrow[\langle \varepsilon \neq \uparrow[\langle \varepsilon \neq \uparrow \rangle]\rangle]\rangle$ from Example 1 expresses $dist_3(x)$. Figure 2 shows that K is not closed under $\downarrow$-bisimulations and hence, by Theorem 20, K is not definable by a set of XPath$_=(\downarrow)$-node expressions.

A class of pointed data trees definable by set of XPath$_=(\uparrow\downarrow)$-node expressions but not definable by single XPath$_=(\uparrow\downarrow)$-node expression. Let K be the class of pointed data trees $(\mathcal{T}, u)$, where u is the root of $\mathcal{T}$, and for all $v \in T$ we have $dist_3(v)$. On the one hand, K is definable by the set of XPath$_=(\uparrow\downarrow)$-node expressions $\{\neg\langle\downarrow^n [\neg\varphi_4]\rangle \mid n \geq 0\}$. On the other, for $i \in \mathbb{N}$, let $(\mathcal{T}_i, u_i)$ be any pointed data tree not in K, of height at least $i+1$, where u_i is the root of $\mathcal{T}_i$, and such that for all $v \in T_i$ at distance at most i from u_i we have $dist_3(v)$. Let $(\mathcal{T}^{\uparrow\downarrow}, u^*)$ be any $\uparrow\downarrow$-quasi ultraproduct of $(\mathcal{T}_i, u_i)_{i\in\mathbb{N}}$. One can see that all nodes of $v \in T^{\uparrow\downarrow}$ satisfy $dist_3(v)$, and so $(\mathcal{T}^{\uparrow\downarrow}, u^*) \in K$. Therefore $\overline{K}$ is not closed under $\uparrow\downarrow$-quasi ultraproducts and by Theorem 26, K is not definable by an XPath$_=(\uparrow\downarrow)$-node expression.

A class of two-pointed data trees definable by a single XPath$_=(\uparrow\downarrow)$-path expression but not definable by set of XPath$_=(\downarrow)$-path expressions. Let K be the class of two-pointed data trees $\mathcal{T}, u, v$ such that v is a child of u and they have the same data value. On the one hand, this class is definable by the path expression $\alpha_3 = \downarrow[\langle \epsilon = \uparrow \rangle]$ of Example 2. On the other hand, the auto bisimulation on $\mathcal{T}$ shown in Figure 3 shows that K is not closed under binary bisimulations for XPath$_=(\downarrow)$', since $\mathcal{T}, u, w$ is bisimilar to $\mathcal{T}, u, v$ but $data(u) \neq data(v)$. Thus, by Theorem 68, K is not definable by a set of XPath$_=(\downarrow)$ path expressions.

A class of two-pointed data trees definable in first-order (over data trees) but not definable by a set of XPath$_=(\uparrow\downarrow)$-path expressions. Let K be the class of two-pointed data trees $\mathcal{T}, u, v$ such that u and v are both children of the root of $\mathcal{T}$, and they have the same data value. It is straightforward that this property is definable by a σ-first-order formula over the class of data trees. On the other, the auto bisimulation between $\mathcal{T}$ and $\mathcal{T}'$ shown in Figure 4 shows that K is not closed under binary bisimulations for XPath$_=(\uparrow\downarrow)$, as $\mathcal{T}, v, w$ is bisimilar to $\mathcal{T}', v', w'$ but $data(v') \neq data(w')$. Thus, by the corresponding theorem of definability, K is not definable by a set of XPath$_=(\uparrow\downarrow)$ path expressions.

7 Conclusions

In this work we introduced new tools for showing definability and separation results for the downward and vertical fragments of XPath with (in)equality tests over data trees, here called XPath$_=$. The general road to prove these theorems themselves was somewhat similar to the one used for the basic modal logic BML (namely, a detour to first-order), but the new concepts (and their interactions) needed to be used in the context of XPath$_=$ are more sophisticated. The notions of $\downarrow$-saturation and $\uparrow\downarrow$-saturation are more refined than the usual notions of BML, as they need to take care of the (in)equality tests over the data. Another difference with respect to the models of BML, namely Kripke models, is that models of XPath$_=$ are trees (in particular, connected) and so they are not closed under ultraproducts. Thus arose the notions of $\downarrow$-quasi and $\uparrow\downarrow$-quasi ultraproducts. These are variants of the classical first-order ultraproducts, and are, of course, absent in the BML framework.

We also introduced new notions of binary bisimulation for both the downward and vertical XPath$_=$, which consist of a relation Z linking pairs (x, y) in some data tree with pairs (x', y') in some other. In order to maintain the structure of the pairs (x, y) and (x', y') in the bisimulation Z, more rules were needed than in the case of unary bisimulations. Binary bisimulations were shown to match, within finitely branching data trees, to logical equivalences in terms of path expressions, i.e., a bisimulation links (x, y) with (x', y') if and only if (x, y) and (x', y') satisfy the same path expressions. They supersede unary bisimulations, in that they keep all the information of the latter (but they carry more). Furthermore, binary bisimulations are robust in the sense that they allow us to show the essential theorems of definability and separation using the language of path expressions, and evaluating in pairs of nodes. Finally, a characterization theorem à la van Benthem was shown for the case of downward $\mathrm{XPath}_=$—for the case of vertical it is known to be false for unary bisimulations, and so as well for binary ones.

An interesting question is what can be said about other fragments of $\mathrm{XPath}_=$ such as $\mathrm{XPath}_=(\downarrow\downarrow_*)$ ('child' and 'descendant' axes) or $\mathrm{XPath}_=(\downarrow\uparrow\downarrow_*\uparrow^*)$ ('child', 'parent', 'descendant' and 'ancestor' axes). As it is mentioned in [10, §5], the bisimulation notions of these two fragments correspond to those for $\mathrm{XPath}_=(\downarrow)$ and $\mathrm{XPath}_=(\uparrow\downarrow)$ respectively. However, in the case of $\mathrm{XPath}_=(\downarrow\downarrow_*)$ and $\mathrm{XPath}_=(\downarrow\uparrow\downarrow_*\uparrow^*)$, the connection to first-order logic is not clear, as we cannot express *transitive closure*.

We can also try to relax the restrictions in the classes of two-pointed data trees used to show definability and separation for the language of path expressions. Though it seems difficult to work with a broader class in the case of downward, it would still be possible to obtain stronger results for the case of vertical.

Another question is to investigate the size of shortest node or path expressions distinguishing two nodes or two pairs of nodes, respectively, in a data tree, following the ideas in [11], where the notion of bisimulation plays a central role.

Finally, one can devise $\mathrm{XPath}_=$ as a querying language over graph-structured databases (graph data is everywhere these days: from social media like Facebook and Twitter, to biological databases and the Semantic Web). What can we say when we consider general data graphs instead of data trees? It is easy to see that the notion of unary and even binary downward bisimulation could be suitable in this case. However, the notions of bisimulations for the vertical fragment rely heavily in the existence of the up-down normal form, which does not hold in data graphs (not even in DAGs). An interesting study would be to develop notions of bisimulations and understand the model theory of the vertical $\mathrm{XPath}_=$ in the context of data graphs.

Acknowledgements

This work was partially supported by grant ANPCyT-PICT-2013-2011, ANPCyT-PICT-2011-0365, UBACyT 20020110100025 and the FP7-PEOPLE-2011-IRSES Project "Mobility between Europe and Argentina applying Logics to Systems" (MEALS) and the Laboratoire International Associé "INFINIS".

References

[1] C. Areces, F. Carreiro, and S. Figueira. Characterization, definability and separation via saturated models. *Theoretical Computer Science*, 537:72–86, 2014.

[2] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[3] M. Bojańczyk, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. *Journal of the ACM*, 56(3):1–48, 2009.

[4] C.C. Chang and H.J. Keisler. *Model theory*. Studies in logic and the foundations of mathematics. North-Holland, 1990.

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

[6] A. Dawar and M. Otto. Modal characterisation theorems over special classes of frames. *Annals of Pure and Applied Logic*, 161(1):1–42, 2009.

[7] M. de Rijke. Modal model theory. Technical Report CS-R9517, CWI, Amsterdam, 1995.

[8] M. de Rijke and H. Sturm. Global definability in basic modal logic. *Essays on non-classical logic*, 1:111, 2001.

[9] D. Figueira, S. Figueira, and C. Areces. Model theory of XPath on data trees. Part I: Bisimulation and characterization. Submitted, `http://www.glyc.dc.uba.ar/santiago/papers/xpath-part1.pdf`.

[10] D. Figueira, S. Figueira, and C. Areces. Basic model theory of XPath on data trees. In *International Conference on Database Theory*, pages 50–60, 2014.

[11] S. Figueira and D. Gorín. On the size of shortest modal descriptions. In *Advances in Modal Logic*, volume 8, pages 114–132, 2010.

[12] G.H.L. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, and S. Vansummeren. Similarity and bisimilarity notions appropriate for characterizing indistinguishability in fragments of the calculus of relations. *Journal of Logic and Computation*, 2014. Published online.

[13] M. Forti and F. Honsell. Set theory with free construction principles. *Annali Scuola Normale Superiore, Pisa*, X(3):493–522, 1983.

[14] G. Gottlob, C. Koch, and R. Pichler. Efficient algorithms for processing XPath queries. *ACM Transactions on Database Systems*, 30(2):444–491, 2005.

[15] M. Gyssens, J. Paredaens, D. Van Gucht, and G.H.L. Fletcher. Structural characterizations of the semantics of XPath as navigation tool on a document. In *PODS*, pages 318–327. ACM, 2006.

[16] N. Kurtonina and M. de Rijke. Bisimulations for temporal logic. *Journal of Logic, Language and Information*, 6:403–425, 1997.

[17] N. Kurtonina and M. de Rijke. Simulating without negation. *Journal of Logic and Computation*, 7:503–524, 1997.

[18] M. Marx and M. de Rijke. Semantic characterizations of navigational XPath. *SIGMOD Record*, 34(2):41–46, 2005.

[19] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[20] M. Otto. Elementary proof of the van Benthem-Rosen characterisation theorem. Technical Report 2342, Fachbereich Mathematik, Technische Universität Darmstadt, 2004.

[21] M. Otto. Bisimulation invariance and finite models. In *Logic Colloquium'02*, volume 27 of *Lecture Notes in Logic*, pages 276–298, 2006.

[22] D. Park. Concurrency and automata on infinite sequences. In *Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.

[23] E. Rosen. Modal logic over finite structures. *Journal of Logic, Language and Information*, 6(4):427–439, 1997.

[24] D. Sangiorgi. On the origins of bisimulation and coinduction. *ACM Transactions on Programming Languages and Systems*, 31(4), 2009.

[25] B. ten Cate. The expressivity of XPath with transitive closure. In Stijn Vansummeren, editor, *PODS*, pages 328–337. ACM, 2006.

[26] J. van Benthem. *Modal Correspondence Theory*. PhD thesis, Universiteit van Amsterdam, 1976.