Basic Model Theory of XPath on Data Trees*

Diego Figueira
University of Edinburgh
UK

Santiago Figueira
Universidad de Buenos Aires
and CONICET
Argentina

Carlos Areces
Universidad Nacional de Córdoba
and CONICET
Argentina

ABSTRACT

We investigate model theoretic properties of XPath with data (in)equality tests over the class of data trees, i.e., the class of trees where each node contains a label from a finite alphabet and a data value from an infinite domain.

We provide notions of (bi)simulations for XPath logics containing the `child`, `descendant`, `parent` and `ancestor` axes to navigate the tree. We show that these notions precisely characterize the equivalence relation associated with each logic. We study formula complexity measures consisting of the number of nested axes and nested subformulas in a formula; these notions are akin to the notion of quantifier rank in first-order logic. We show characterization results for fine grained notions of equivalence and (bi)simulation that take into account these complexity measures. We also prove that positive fragments of these logics correspond to the formulas preserved under (non-symmetric) simulations. We show that the logic including the `child` axis is equivalent to the fragment of first-order logic invariant under the corresponding notion of bisimulation. If upward navigation is allowed the characterization fails but a weaker result can still be established. These results hold over the class of possibly infinite data trees and over the class of finite data trees.

Besides their intrinsic theoretical value, we argue that bisimulations are useful tools to prove (non)expressivity results for the logics studied here, and we substantiate this claim with examples.

Categories and Subject Descriptors

F.4.1 [**Mathematical Logic**]: Model theory; H.2.3 [**Languages**]: Query Languages; I.7.2 [**Document Preparation**]: Markup Languages

*This work was partially supported by grant ANPCyT-PICT-2010-688, 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".

General Terms

Theory, Languages

1. INTRODUCTION

We study the expressive power and model theory of XPath—arguably the most widely used XML query language. Indeed, XPath is implemented in XSLT and XQuery and it is used as a constituent part of many specification and update languages. XPath is, fundamentally, a general purpose language for addressing, searching, and matching pieces of an XML document. It is an open standard and constitutes a World Wide Web Consortium (W3C) Recommendation [6].

Core-XPath (term coined in [13]) is the fragment of XPath 1.0 containing the navigational behavior of XPath. It can express properties of the underlying tree structure of the XML document, such as the label (tag name) of a node, but it cannot express conditions on the actual data contained in the attributes. In other words, it only allows to reason about trees over a finite alphabet. Core-XPath has been well studied and its satisfiability problem is known to be decidable even in the presence of DTDs [17, 1]. Moreover, it is known that it is equivalent to FO^2 (first-order logic with two variables over an appropriate signature on trees) in terms of expressive power [18], and that it is strictly less expressive than PDL with converse over trees [2]. From a database perspective, however, Core-XPath fails to include the single most important construct in a query language: the join. Without the ability to relate nodes based on the actual *data values* of the attributes, the logic's expressive power is inappropriate for many applications.

The extension of Core-XPath with (in)equality tests between attributes of elements in an XML document is named Core-Data-XPath in [4]. Here, we will call this logic $\text{XPath}_=$. Models of $\text{XPath}_=$ are *data trees* which can be seen as XML documents. A data tree is a tree whose nodes contains a *label* from a finite alphabet and a *data value* from an infinite domain (see Figure 1 for an example). We will relax the condition on finiteness and consider also infinite data trees, although all our results hold also on finite structures.

The main characteristic of $\text{XPath}_=$ is to allow formulas of the form $\langle\alpha = \beta\rangle$, where α, β are *path expressions*, that navigate the tree using *axes*: `descendant`, `child`, `ancestor`, `next-sibling`, etc. and can make tests in intermediate nodes. The formula is true at a node x of a data tree if there are nodes y, z that can be reached by the relations denoted by α, β, respectively, and such that the data value of y is equal to the data value of z.

Recent articles investigate several algorithmic problems

10.5441/002/icdt.2014.09

of logics evaluated over data trees. For example, satisfiability and evaluation are discussed in [8, 5]. In particular, all the logics studied in this article have a decidable satisfiability problem [10, 9]; but tools to investigate their *expressive power* are still lacking. There are good reasons for this: in the presence of joins and data values, classical notions such as Ehrenfeucht-Fraïssé games or structural bisimulations are difficult to handle. In this article we take the first steps towards understanding the expressive power and model theory of XPath$_=$ on data trees.

Contribution: XPath$_=$ can navigate the data tree by means of its axes: `child` (that we will note $\downarrow$), `descendant` ($\downarrow_*$), `parent` ($\uparrow$), `ancestor` ($\uparrow^*$), etc. XPath$_=$ can also navigate the data tree horizontally, by going to a next or previous `sibling` of the current node. However, we focus on the vertical axes that allow downward and upward exploration. In particular, we will discuss the following languages: XPath$^{\downarrow}_{=}$ (XPath$_=$ with $\downarrow$); XPath$^{\updownarrow}_{=}$ (XPath$_=$ with $\downarrow$ and $\uparrow$); XPath$^{\downarrow\downarrow_*}_{=}$ (XPath$_=$ with $\downarrow$ and $\downarrow_*$); XPath$^{\updownarrow\updownarrow^*}_{=}$ (XPath$_=$ with $\downarrow$, $\uparrow$, $\downarrow_*$ and $\uparrow^*$); and its positive fragments. Our main contributions can be summarized as follows:

- In §3 and §5 we introduce bisimulation notions for XPath$^{\downarrow}_{=}$, XPath$^{\downarrow\downarrow_*}_{=}$, XPath$^{\updownarrow}_{=}$, and XPath$^{\updownarrow\updownarrow^*}_{=}$ and show that they precisely characterize the logical equivalence relation of the respective logic. We also consider fine grained versions of these bisimulations that take into account the number of nested axes and subformulas. The notion of bisimulation for XPath$^{\updownarrow}_{=}$ relies on a strong normal form which we also introduce.
- In §4 we show that the simulations associated to the defined bisimulations characterize the positive fragments of the logics: a formula is equivalent to a positive formula if and only if it is invariant under simulations.
- In §6 we characterize XPath$^{\downarrow}_{=}$ as the fragment of first-order logic over data trees (over a signature that includes the `child` relation and an equivalence relation) that is invariant under bisimulations. If we consider XPath$^{\updownarrow}_{=}$ instead the characterization fails, but a weaker result can still be established.
- Using bisimulations we show (non)expressivity results about XPath$_=$ in §7. We characterize, for example, in which cases increasing the nesting depth increases the expressive power of XPath$^{\downarrow}_{=}$.
- All results are proved both over the class of arbitrary (possibly infinite) data trees, and over the class of finite data trees.

Related work: The notion of bisimulation was introduced independently by Van Benthem [26] in the context of modal correspondence theory, Milner [19] and Park [23] in concurrency theory, and Forti and Honsell [11] in non-wellfounded set theory (see [25] for a historical outlook). This classical work defines a *standard notion of bisimulation* but this notion has to be suitably adapted for a particular, given logic. The notion of bisimulation for a given logic $\mathcal{L}$ defines when two models are indistinguishable for $\mathcal{L}$, that is, when there is no formula of $\mathcal{L}$ that is true in one model but false in the other. Bisimulations can also be used to obtain model theoretic characterizations that identifies the expressive power of a logic $\mathcal{L}_1$ in terms of the bisimulation invariant fragment of a logic $\mathcal{L}_2$ which, hopefully, is better understood. The challenge, here, is to pinpoint both the appropriate notion

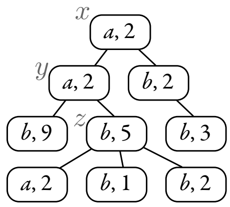

Figure 1: A data tree of *Trees*$(\mathbb{A}\times\mathbb{D})$ with $\mathbb{A} = \{a, b\}$ and $\mathbb{D} = \mathbb{N}$.

of bisimulation required and the adequate 'framework' logic $\mathcal{L}_2$. The classical example of a result of this kind is Van Benthem's characterization for the basic modal logic as the bisimulation (with the standard notion of bisimulation) invariant fragment of first-order logic [26]. Van Benthem's original result over arbitrary structures was proved to hold for finite structures by Rosen [24]. The proof was then simplified and unified by Otto [20, 22], and later expanded by Dawar and Otto [7] to other classes of structures.

Logics for semi-structured databases can often be seen as modal logics. In fact, structural characterizations for XPath without equality test were studied in [14], and XPath is known to be captured by PDL [15], whose bisimulation is well-understood [3]. It is then natural to look for an intuitive bisimulation definition for XPath$_=$.

2. PRELIMINARIES

2.1 Notation

Let $\mathbb{N} = \{1, 2, 3, \dots\}$ and let $[n] := \{1, \dots, n\}$ for $n \in \mathbb{N}$. We use the symbol $\mathbb{A}$ to denote a finite alphabet, and $\mathbb{D}$ to denote an infinite domain (e.g., $\mathbb{N}$) of **data values**. In our examples we will consider $\mathbb{D} = \mathbb{N}$. We write $X{\sim}Y$ to say that X is the result of replacing every data value $d \in \mathbb{D}$ from Y by $f(d)$ where $f : \mathbb{D} \to \mathbb{D}$ is some arbitrary bijection, for any objects X, Y. We write λ for the empty string.

2.2 Data trees

Let *Trees*(A) be the set of ordered and unranked 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**. Figure 1 shows an example of a (finite) data tree. 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, v, w as variables for nodes. Given a node $x \in T$ of $\mathcal{T}$, we write $\mathit{label}(x) \in \mathbb{A}$ to denote the node's label, and $\mathit{data}(x) \in \mathbb{D}$ 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. $(x\xrightarrow{n})$ denotes the set of all descendants of x at distance n, and $(\xrightarrow{n}y)$ denotes the sole ancestor of y at distance n (assuming it has one).

For any binary relation R over elements of data trees, we say that a property P is **R-invariant** whenever the following condition holds: for every data tree $\mathcal{T}$ and $u \in T$, if $(\mathcal{T}, u)$ satisfies P and $(\mathcal{T}, u)$ is R-related to $(\mathcal{T}', u')$ then $(\mathcal{T}', u')$ satisfies P.

2.3 XPath

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

Table 1: Semantics of XPath$_=$ for a data tree $\mathcal{T}$.

We introduce the query language XPath adapted to data trees as abstractions of XML documents. 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) and **node expressions** (that we write φ, ψ, η). The fragment XPath$_=(\mathcal{O})$, with $\mathcal{O} \subseteq \{\downarrow, \downarrow_*, \uparrow, \uparrow^*\}$, is defined by mutual recursion as follows:

$$\begin{aligned} \alpha, \beta \ &::=\ o \mid [\varphi] \mid \alpha\beta \mid \alpha \cup \beta && o \in \mathcal{O} \cup \{\varepsilon\} \\ \varphi, \psi \ &::=\ a \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \langle\alpha\rangle \mid \\ &\qquad \langle\alpha = \beta\rangle \mid \langle\alpha \neq \beta\rangle && a \in \mathbb{A} \end{aligned}$$

A **formula** of XPath$_=(\mathcal{O})$ is either a node expression or a path expression. To save space, we use XPath$^{\downarrow}_{=}$ for XPath$_=(\downarrow)$; XPath$^{\updownarrow}_{=}$ for XPath$_=(\downarrow, \uparrow)$; XPath$^{\downarrow\downarrow_*}_{=}$ for XPath$_=(\downarrow, \downarrow_*)$; and XPath$^{\updownarrow\updownarrow^*}_{=}$ for XPath$_=(\downarrow, \uparrow, \downarrow_*, \uparrow^*)$.

We formally define the semantics of XPath$_=$ in Table 1. As an example, if $\mathcal{T}$ is the data tree shown in Figure 1, then $[\![\langle\downarrow_*[\, b \wedge \langle\downarrow[b] \neq \downarrow[b]\rangle\,]\rangle]\!]^{\mathcal{T}} = \{x, y, z\}$, where the formula reads: "*there is a descendant node labeled b, with two children labeled b with different data values.*" For a data tree $\mathcal{T}$ and $u \in T$, we write $\mathcal{T}, u \models \varphi$ to denote $u \in [\![\varphi]\!]^{\mathcal{T}}$, and we say that $\mathcal{T}, u$ **satisfies** φ. We say that the formulas 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 $\mathcal{T}$.

We call **downward XPath** to XPath$^{\downarrow}_{=}$ and **vertical XPath** to XPath$^{\updownarrow}_{=}$.

In terms of expressive power, it is easy to see that $\cup$ is unessential: every XPath$_=$ node expression φ has an equivalent φ' with no $\cup$ in its path expressions. φ' can be computed in exponential time without incrementing the number of nested axes or the number of nested subformulas. It is enough to use the following equivalences to eliminate occurrences of $\cup$

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

where $\odot \in \{=, \neq\}$. We will henceforth assume that formulas do not contain union of path expressions.

3. BISIMULATION

3.1 Downward XPath

We write $\mathrm{dd}(\varphi)$ to denote the **downward depth** of φ, defined in Table 2. Let ℓ-XPath$^{\downarrow}_{=}$ be the fragment of XPath$^{\downarrow}_{=}$ consisting of all formulas φ with $\mathrm{dd}(\varphi) \leq \ell$.

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

For every ℓ, there are finitely many different formulas φ of $\mathrm{dd}(\varphi) \leq \ell$ up to logical equivalence.

Proposition 3.1. *$\equiv^{\downarrow}_{\ell}$ has finite index.*

Corollary 3.2. *$\{\mathcal{T}', u' \mid \mathcal{T}, u \equiv^{\downarrow}_{\ell} \mathcal{T}', u'\}$ is definable by an ℓ-XPath$^{\downarrow}_{=}$-formula $\chi_{\ell,\mathcal{T},u}$.*

3.1.1 Bisimulation and ℓ-bisimulation

Let $\mathcal{T}$ and $\mathcal{T}'$ be two data-trees. We say that $u \in T$ and $u' \in T'$ are **bisimilar for XPath$^{\downarrow}_{=}$** (notation: $\mathcal{T}, u \underline{\leftrightarrow}^{\downarrow} \mathcal{T}', u'$) 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 $\mathit{label}(x) = \mathit{label}(x')$.
- **Zig (Figure 2):** 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. $\mathit{data}(v) = \mathit{data}(w) \Leftrightarrow \mathit{data}(v') = \mathit{data}(w')$,
  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.

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 bisimulation:

Proposition 3.3. *$\mathcal{T}, u \underline{\leftrightarrow}^{\downarrow} (\mathcal{T}|u), u$.*

Proposition 3.4. *If $\mathcal{T}$ is a subtree of $\mathcal{T}'$ and $u \in T$ then $\mathcal{T}, u \underline{\leftrightarrow}^{\downarrow} \mathcal{T}', u$.*

We say that $u \in T$ and $u' \in T'$ are **ℓ-bisimilar for XPath$^{\downarrow}_{=}$** (notation: $\mathcal{T}, u \underline{\leftrightarrow}^{\downarrow}_{\ell} \mathcal{T}', u'$) 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

$\mathrm{dd}(a) = 0$	$\mathrm{vd}(a) = (0, 0)$	$\mathrm{nd}(a) = 0$
$\mathrm{dd}(\varphi \wedge \psi) = \max\{\mathrm{dd}(\varphi), \mathrm{dd}(\psi)\}$	$\mathrm{vd}(\varphi \wedge \psi) = \max\{\mathrm{vd}(\varphi), \mathrm{vd}(\psi)\}$	$\mathrm{nd}(\varphi \wedge \psi) = \max\{\mathrm{nd}(\varphi), \mathrm{nd}(\psi)\}$
$\mathrm{dd}(\neg\varphi) = \mathrm{dd}(\varphi)$	$\mathrm{vd}(\neg\varphi) = \mathrm{vd}(\varphi)$	$\mathrm{nd}(\neg\varphi) = \mathrm{nd}(\varphi)$
$\mathrm{dd}(\langle\alpha\rangle) = \mathrm{dd}(\alpha)$	$\mathrm{vd}(\langle\alpha\rangle) = \mathrm{vd}(\alpha)$	$\mathrm{nd}(\langle\alpha\rangle) = \mathrm{nd}(\alpha)$
$\mathrm{dd}(\langle\alpha \odot \beta\rangle) = \max\{\mathrm{dd}(\alpha), \mathrm{dd}(\beta)\}$	$\mathrm{vd}(\langle\alpha \odot \beta\rangle) = \max\{\mathrm{vd}(\alpha), \mathrm{vd}(\beta)\}$	$\mathrm{nd}(\langle\alpha \odot \beta\rangle) = \max\{\mathrm{nd}(\alpha), \mathrm{nd}(\beta)\}$
$\mathrm{dd}(\lambda) = 0$	$\mathrm{vd}(\lambda) = (0, 0)$	$\mathrm{nd}(\alpha\beta) = \max\{\mathrm{nd}(\alpha), \mathrm{nd}(\beta)\}$
$\mathrm{dd}(\varepsilon\alpha) = \mathrm{dd}(\alpha)$	$\mathrm{vd}(\varepsilon\alpha) = \mathrm{vd}(\alpha)$	$\mathrm{nd}(\varepsilon) = 0$
$\mathrm{dd}([\varphi]\alpha) = \max\{\mathrm{dd}(\varphi), \mathrm{dd}(\alpha)\}$	$\mathrm{vd}([\varphi]\alpha) = \max\{\mathrm{vd}(\varphi), \mathrm{vd}(\alpha)\}$	$\mathrm{nd}([\varphi]) = 1 + \mathrm{nd}(\varphi)$
$\mathrm{dd}(\downarrow\alpha) = 1 + \mathrm{dd}(\alpha)$	$\mathrm{vd}(\downarrow\alpha) = \max\{(0,0), \mathrm{vd}(\alpha) + (1, -1)\}$	$\mathrm{nd}(\downarrow) = 0$
	$\mathrm{vd}(\uparrow\alpha) = \max\{(0,0), \mathrm{vd}(\alpha) + (-1, 1)\}$	$\mathrm{nd}(\uparrow) = 0$
Downward depth	**Vertical depth**	**Nesting depth**

Table 2: Definitions of downward depth, vertical depth and nesting depth. ($a \in \mathbb{A}$, $\odot \in \{=, \neq\}$, '+' and 'max' are performed component-wise, α is any path expression or the empty string λ.)

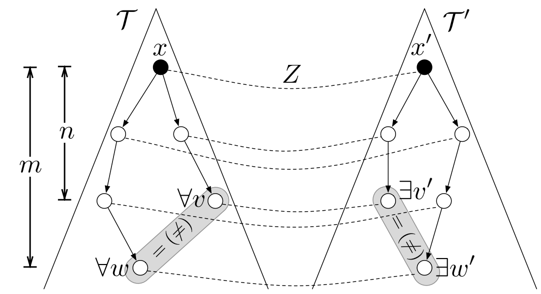

Figure 2: *Zig* clause of bisimulation for $\mathrm{XPath}^{\downarrow}_{=}$.

- **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.

Clearly if $\mathcal{T}, u \,\underline{\leftrightarrow}^{\downarrow}\, \mathcal{T}', u'$ then $\mathcal{T}, u \,\underline{\leftrightarrow}^{\downarrow}_{\ell}\, \mathcal{T}', u'$ for all ℓ. For a data tree $\mathcal{T}$ and $u \in T$, let $\mathcal{T}|_{\ell}u$ denote the subtree of $\mathcal{T}$ induced by $\{v \in T \mid (\exists n \leq \ell)\ u\xrightarrow{n}v\}$.

3.1.2 Equivalence and bisimulation

We now show that $\underline{\leftrightarrow}^{\downarrow}$ coincides with $\equiv^{\downarrow}$ on finitely branching data trees, and that $\underline{\leftrightarrow}^{\downarrow}_{\ell}$ coincides with $\equiv^{\downarrow}_{\ell}$.

THEOREM 3.5.

1. *$\mathcal{T}, u \,\underline{\leftrightarrow}^{\downarrow}\, \mathcal{T}', u'$ implies $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'$. The converse also holds when $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching.*
2. *$\mathcal{T}, u \,\underline{\leftrightarrow}^{\downarrow}_{\ell}\, \mathcal{T}', u'$ iff $\mathcal{T}, u \equiv^{\downarrow}_{\ell} \mathcal{T}', u'$.*

The Theorem above is a consequence of the next two propositions:

PROPOSITION 3.6. *$\mathcal{T}, u \,\underline{\leftrightarrow}^{\downarrow}_{\ell}\, \mathcal{T}', u'$ implies $\mathcal{T}, u \equiv^{\downarrow}_{\ell} \mathcal{T}', u'$.*

PROOF. We actually show that if $\mathcal{T}, u \,\underline{\leftrightarrow}^{\downarrow}_{\ell}\, \mathcal{T}', u'$ 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 xZ_jx' then $\mathcal{T}, x \models \varphi$ iff $\mathcal{T}', x' \models \varphi$;
2. If $x\xrightarrow{n}v$, $x'\xrightarrow{n}v'$ and $(\xrightarrow{i}v)\, Z_{(j-n)+i}\, (\xrightarrow{i}v')$ for all $0 \leq i \leq n$, then $(x, v) \in [\![\alpha]\!]^{\mathcal{T}}$ iff $(x', v') \in [\![\alpha]\!]^{\mathcal{T}'}$.

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, $label(x) = label(x')$ 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 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 $x\xrightarrow{n}v$, $x\xrightarrow{m}w$, $(x, v) \in [\![\alpha]\!]^{\mathcal{T}}$, $(x, w) \in [\![\beta]\!]^{\mathcal{T}}$ and $data(v) = data(w)$. By Zig, there are $v', w' \in T'$ such that $x'\xrightarrow{n}v'$, $x'\xrightarrow{m}w'$, $(\xrightarrow{i}v)\, Z_{j-n+i}\, (\xrightarrow{i}v')$ for all $0 \leq i \leq n$, $(\xrightarrow{i}w)\, Z_{j-m+i}\, (\xrightarrow{i}w')$ for all $0 \leq i \leq m$, and $data(v') = data(w')$. By inductive hypothesis 2 (twice), $(x', v') \in [\![\alpha]\!]^{\mathcal{T}'}$ and $(x', w') \in [\![\beta]\!]^{\mathcal{T}'}$. 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. The case $\varphi = \langle\alpha\rangle$ is similar (and simpler) to the previous case.

Let us now analyze item 2. We only show the 'only if' direction. The base case is when $\alpha \in \{\varepsilon, \downarrow\}$. If $\alpha = \varepsilon$ then $v = x$ and so $n = 0$. Since $v' = x'$, we conclude $(x', v') \in [\![\alpha]\!]^{\mathcal{T}'}$. If $\alpha = \downarrow$ then $x \to v$ in $\mathcal{T}$, and so $n = 1$. Since $x' \to v'$, we have $(x', v') \in [\![\alpha]\!]^{\mathcal{T}'}$.

For the inductive step, let

$$x_0, \ldots, x_n \in T \quad \text{and} \quad x'_0, \ldots, x'_n \in T'$$

be such that

$$\begin{aligned} x = x_0 \to x_1 \to x_2 \to \cdots \to x_n = v \qquad & \text{in } \mathcal{T}, \\ x' = x'_0 \to x'_1 \to x'_2 \to \cdots \to x'_n = v' \qquad & \text{in } \mathcal{T}', \end{aligned}$$

and $x_i Z_{j-i} x'_i$ for all $0 \leq i \leq n$. Assume, for contradiction, that $(x', v') \notin [\![\alpha]\!]^{\mathcal{T}'}$. Then, there is a subformula φ of α and $k \in \{0, \ldots, n\}$ such that $\mathcal{T}, x_k \models \varphi$ and $\mathcal{T}', x'_k \not\models \varphi$. This contradicts the inductive hypothesis 1. $\square$

PROPOSITION 3.7. *$\mathcal{T}, u \equiv^{\downarrow}_{\ell} \mathcal{T}', u'$ implies $\mathcal{T}, u \leftrightarrow^{\downarrow}_{\ell} \mathcal{T}', u'$.*

PROOF. Fix $u \in T$ and $u' \in T'$ such that $\mathcal{T}, u \equiv^{\downarrow}_{\ell} \mathcal{T}', u'$. Define $(Z_i)_{i \le \ell}$ by

$$xZ_ix' \quad \text{iff} \quad \mathcal{T}, x \equiv^{\downarrow}_{i} \mathcal{T}', x'.$$

We show that Z is an ℓ-bisimulation between $\mathcal{T}, u$ and $\mathcal{T}', u'$. By hypothesis, $uZ_\ell u'$. Fix $h \le \ell$, by construction, Z_h satisfies Harmony. Let us see that Z_h satisfies Zig (the case for Zag is analogous). Suppose xZ_hx',

$$x = v_0 \to v_1 \to \cdots \to v_n = v \qquad \text{in } \mathcal{T},$$
$$x = w_0 \to w_1 \to \cdots \to w_m = w \qquad \text{in } \mathcal{T},$$

and $data(v) = data(w)$ (the case $data(v) \ne data(w)$ is shown in a similar way), where $m, n \le h$. Let $P \subseteq T'^2$ be defined by

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

Since $\mathcal{T}, x \equiv^{\downarrow}_{h} \mathcal{T}', x'$, $\mathrm{dd}(\langle \downarrow^n = \downarrow^m \rangle) \le h$ and $\mathcal{T}, x \models \langle \downarrow^n = \downarrow^m \rangle$, we conclude that $P \ne \emptyset$. We next show that there exists $(v', w') \in P$ such that

i. $x' = v'_0 \to v'_1 \to \cdots \to v'_n = v'$ in $\mathcal{T}'$,

ii. $x' = w'_0 \to w'_1 \to \cdots \to w'_m = w'$ in $\mathcal{T}'$,

iii. $(\forall i \in \{0, \dots, n\})\ \mathcal{T}, v_i \equiv^{\downarrow}_{h-i} \mathcal{T}', v'_i$, and

iv. $(\forall j \in \{0, \dots, m\})\ \mathcal{T}, w_j \equiv^{\downarrow}_{h-j} \mathcal{T}', w'_j$,

and hence Zig is satisfied by Z_h. By way of contradiction, assume that for all $(v', w') \in P$ satisfying i and ii we have either

(a) $(\exists i \in \{0, \dots, n\})\ \mathcal{T}, v_i \not\equiv^{\downarrow}_{h-i} \mathcal{T}', v'_i$, or

(b) $(\exists j \in \{0, \dots, m\})\ \mathcal{T}, w_j \not\equiv^{\downarrow}_{h-j} \mathcal{T}', w'_j$.

Fix $\top$ as any tautology such that $\mathrm{dd}(\top) = 0$. For each $(v', w') \in P$ we define two families of formulas,

$$\varphi^0_{v',w'}, \dots, \varphi^n_{v',w'} \quad \text{and} \quad \psi^0_{v',w'}, \dots, \psi^m_{v',w'},$$

satisfying that $\mathrm{dd}(\varphi^i_{v',w'}) \le h - i$ for all $i \in \{0, \dots, n\}$ and $\mathrm{dd}(\psi^j_{v',w'}) \le h - j$ for all $j \in \{0, \dots, m\}$ as follows:

- Suppose that (a) holds and that i is the smallest number such that $\mathcal{T}, v_i \not\equiv^{\downarrow}_{h-i} \mathcal{T}', v'_i$. Let $\varphi^i_{v',w'}$ be such that $\mathrm{dd}(\varphi^i_{v',w'}) \le h-i$ and $\mathcal{T}, v_i \models \varphi^i_{v',w'}$ but $\mathcal{T}', v'_i \not\models \varphi^i_{v',w'}$. For $k \in \{0, \dots, n\} \setminus \{i\}$, let $\varphi^k_{v',w'} = \top$, and for $k \in \{0, \dots, m\}$, let $\psi^k_{v',w'} = \top$.
- Suppose that (a) does not hold. Then (b) holds. Let j be the smallest number such that $\mathcal{T}, w_j \not\equiv^{\downarrow}_{h-j} \mathcal{T}', w'_j$. Let $\psi^j_{v',w'}$ be such that $\mathrm{dd}(\psi^j_{v',w'}) \le h-j$ and $\mathcal{T}, w_j \models \psi^j_{v',w'}$ but $\mathcal{T}', w'_j \not\models \psi^j_{v',w'}$. For $k \in \{0, \dots, m\} \setminus \{j\}$, let $\psi^k_{v',w'} = \top$, and for $k \in \{0, \dots, n\}$, let $\varphi^k_{v',w'} = \top$.

For each $i \in \{0, \dots, n\}$ and $j \in \{0, \dots, m\}$, let

$$\Phi^i = \bigwedge_{(v',w') \in P} \varphi^i_{v',w'} \quad \text{and} \quad \Psi^j = \bigwedge_{(v',w') \in P} \psi^j_{v',w'}. \qquad (1)$$

Since $\mathrm{dd}(\varphi^i_{v',w'}) \le h - i$, by Proposition 3.1, there are finitely many non-equivalent formulas $\varphi^i_{v',w'}$; the same applies to $\psi^j_{v',w'}$. Hence, both infinite conjunctions in (1) are equivalent to finite ones, and we may assume that Φ^i and Ψ^j are well-formed formulas. Finally, let

$$\alpha = [\Phi^0]\downarrow[\Phi^1]\downarrow \cdots \downarrow[\Phi^n] \text{ and } \beta = [\Psi^0]\downarrow[\Psi^1]\downarrow \cdots \downarrow[\Psi^m].$$

By construction, $\mathrm{dd}(\alpha), \mathrm{dd}(\beta) \le h$ and so $\mathrm{dd}(\langle \alpha = \beta \rangle) \le h$. Furthermore, $\mathcal{T}, x \models \langle \alpha = \beta \rangle$ and $\mathcal{T}', x' \not\models \langle \alpha = \beta \rangle$. This contradicts $\mathcal{T}, x \equiv^{\downarrow}_{h} \mathcal{T}', x'$. $\square$

3.2 Vertical XPath

We now study bisimulation for $\mathrm{XPath}^{\updownarrow}_{=}$. Interestingly, the notion we give is simpler than the one for $\mathrm{XPath}^{\downarrow}_{=}$ due to a normal form enjoyed by the logic.

In the downward fragment of $\mathrm{XPath}_{=}$ we used $\mathrm{dd}(\varphi)$ to measure the maximum depth from the current point of evaluation that the formula can access. 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. Formally, the **vertical depth** of a formula φ (notation: $\mathrm{vd}(\varphi)$) is the pair $\mathrm{vd}(\varphi) \in \mathbb{Z}^2_{\ge 0}$ defined in Table 2.

The **nesting depth** of a formula φ (notation: $\mathrm{nd}(\varphi)$) is the maximum number of nested [ ] appearing in φ. See Table 2 for the formal definition.

Let (r, s, k)-$\mathrm{XPath}^{\updownarrow}_{=}$ be the set of all formulas φ in $\mathrm{XPath}^{\updownarrow}_{=}$ with $\mathrm{vd}(\varphi) \le (r, s)$ and $\mathrm{nd}(\varphi) \le k$.

Let $\mathcal{T}$ and $\mathcal{T}'$ be data trees, let $u \in T$ and $u' \in T'$. We say that $\mathcal{T}, u$ and $\mathcal{T}', u'$ are **equivalent for $\mathrm{XPath}^{\updownarrow}_{=}$** (notation: $\mathcal{T}, u \equiv^{\updownarrow} \mathcal{T}', u'$) iff for all $\varphi \in \mathrm{XPath}^{\updownarrow}_{=}$, we have $\mathcal{T}, u \models \varphi$ iff $\mathcal{T}', u' \models \varphi$. $\mathcal{T}, x$ and $\mathcal{T}', x'$ are **(r, s)-equivalent** [resp. **(r, s, k)-equivalent**] **for $\mathrm{XPath}^{\updownarrow}_{=}$**, and we note it $\mathcal{T}, x \equiv^{\updownarrow}_{r,s} \mathcal{T}', x'$ [resp. $\mathcal{T}, x \equiv^{\updownarrow}_{r,s,k} \mathcal{T}', x'$] if they satisfy the same $\mathrm{XPath}^{\updownarrow}_{=}$ formulas φ so that $\mathrm{vd}(\varphi) \le (r, s)$ [resp. $\mathrm{vd}(\varphi) \le (r, s)$ and $\mathrm{nd}(\varphi) \le k$].

3.2.1 Normal form

We define a useful normal form for $\mathrm{XPath}^{\updownarrow}_{=}$ that will be implicitly used in the definition of bisimulation in the section. For $n \ge 0$, let $\downarrow^n$ denote the concatenation of n symbols $\downarrow$. I.e., $\downarrow^0$ is the empty string λ, $\downarrow^1 = \downarrow$, and $\downarrow^{n+1} = \downarrow\downarrow^n$ (similarly for $\uparrow^n$).

A path expression α of $\mathrm{XPath}^{\updownarrow}_{=}$ is **downward** [resp. **upward**] if it is of the form $\downarrow^n[\varphi]$ [resp. $[\varphi]\uparrow^n$] for some $n \ge 0$ with $\varphi \in \mathrm{XPath}^{\updownarrow}_{=}$. For example, $\downarrow[\langle\uparrow\rangle]$ is a downward expression whereas $\downarrow[\langle\downarrow\rangle]\downarrow$ is not. An **up-down** expression is any expression of the form ε, $\alpha^\uparrow$, $\alpha^\downarrow$ or $\alpha^\uparrow\alpha^\downarrow$ where $\alpha^\uparrow$ is upward and $\alpha^\downarrow$ is downward. Henceforth we will use $\alpha^\uparrow, \beta^\uparrow, \gamma^\uparrow$ to denote upward expressions and $\alpha^\downarrow, \beta^\downarrow, \gamma^\downarrow$ to denote downward expressions and $\alpha^{\uparrow\downarrow}, \beta^{\uparrow\downarrow}, \gamma^{\uparrow\downarrow}$ to denote up-down expressions. Note that in particular any downward or upward expression is an up-down expression. An $\mathrm{XPath}^{\updownarrow}_{=}$ formula or expression is in **up-down normal form** if every path expression contained in it is up-down and every data test is of the form $\langle \varepsilon \odot \alpha^{\uparrow\downarrow} \rangle$ with $\odot \in \{=, \ne\}$.

PROPOSITION 3.8. *Let $\varphi \in (r, s, k)$-$\mathrm{XPath}^{\updownarrow}_{=}$. There is $\varphi^{\uparrow\downarrow} \in \mathrm{XPath}^{\updownarrow}_{=}$ in up-down normal form such that*

1. *$\varphi^{\uparrow\downarrow} \equiv \varphi$;*
2. *$\mathrm{vd}(\varphi^{\uparrow\downarrow}) = (r, s)$; and*
3. *$\mathrm{nd}(\varphi^{\uparrow\downarrow}) \le k \cdot (r + s + 2)$.*

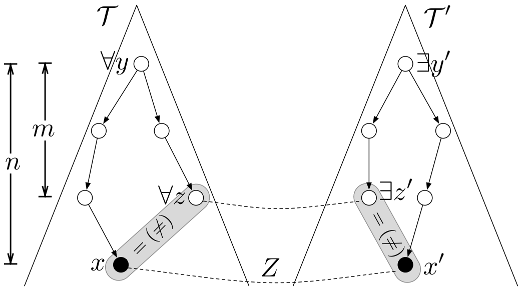

Figure 3: *Zig* clause of bisimulation for $\mathbf{XPath}^{\updownarrow}_{=}$

3.2.2 Finite index

Contrary to the case of $\mathrm{XPath}^{\downarrow}_{=}$ (*cf.*, Proposition 3.1), the logical equivalence relation restricted to $\mathrm{XPath}^{\updownarrow}_{=}$-formulas of bounded vertical depth has infinitely many equivalence classes.

PROPOSITION 3.9. *If $r + s \geq 2$ then $\equiv^{\updownarrow}_{r,s}$ has infinite index.*

In the proof of the above proposition we need to use formulas with unbounded nesting depth. In fact, when restricted to bounded nesting depth there are only finitely many formulas up to logical equivalence, as stated next.

PROPOSITION 3.10. *$\equiv^{\updownarrow}_{r,s,k}$ has finite index.*

COROLLARY 3.11. *$\{\mathcal{T}', u' \mid \mathcal{T}, u \equiv^{\updownarrow}_{r,s,k} \mathcal{T}', u'\}$ is definable by an (r, s, k)-$\mathrm{XPath}^{\updownarrow}_{=}$-formula.*

3.2.3 Bisimulation and (r, s, k)-bisimulation

The advantage of the normal form presented in Section 3.2.1, is that it makes it possible to use a very simple notion of bisimulation. The disadvantage is that, since it does not preserve nesting depth, $\underline{\leftrightarrow}^{\updownarrow}_{r,s,k}$ does not correspond precisely to $\equiv^{\updownarrow}_{r,s,k}$, although $\underline{\leftrightarrow}^{\updownarrow}$ corresponds precisely to $\equiv^{\updownarrow}$. Nonetheless, we obtain, for all r, s, k,

$$\underline{\leftrightarrow}_{r,s,k} \;\subseteq\; \equiv^{\updownarrow}_{r,s,k} \;\subseteq\; \underline{\leftrightarrow}^{\updownarrow}_{r,s,k\cdot(r+s+2)}.$$

Let $\mathcal{T}$ and $\mathcal{T}'$ be two data-trees. We say that $u \in T$ and $u' \in T'$ are **bisimilar for $\mathbf{XPath}^{\updownarrow}_{=}$** (notation: $\mathcal{T}, u \underline{\leftrightarrow}^{\updownarrow} \mathcal{T}', u'$) 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 (Figure 3):** 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'.

Observe that contrary to the definition of $\underline{\leftrightarrow}^{\downarrow}$, the conditions above do not require intermediate nodes to be related by Z. This is a direct consequence of the up-down normal form (Proposition 3.8).

We say that $u \in T$ and $u' \in T'$ are **(r, s, k)-bisimilar for $\mathbf{XPath}^{\updownarrow}_{=}$** (notation: $\mathcal{T}, u \underline{\leftrightarrow}^{\updownarrow}_{r,s,k} \mathcal{T}', u'$) 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.

OBSERVATION 3.12. *If $xZ^{\hat{k}}_{\hat{r},\hat{s}}x'$, $y \xrightarrow{n} x$ and $y' \xrightarrow{n} x'$ then it follows that $yZ^{\hat{k}-1}_{\hat{r}',\hat{s}'}y'$, for $\hat{r}' = \hat{r} + n$, $\hat{s}' = \hat{s} - n$. The same occurs with Z instead of $Z^{\hat{k}}_{\hat{r},\hat{s}}$ for the case of bisimilarity.*

For a data tree $\mathcal{T}$ and $u \in T$, let $\mathcal{T}|^{s}_{r}u$ denote the subtree of $\mathcal{T}$ induced by

$$\{v \in T \mid (\exists m \leq s)\ (\exists n \leq r + m)\ (\exists w \in T)\ w \xrightarrow{m} u \wedge\ w \xrightarrow{n} v\}.$$

3.2.4 Equivalence and bisimulation

The next result says that $\underline{\leftrightarrow}^{\updownarrow}$ coincides with $\equiv^{\updownarrow}$ on finitely branching data trees, and states precisely in what way $\underline{\leftrightarrow}^{\updownarrow}_{r,s,k}$ is related to $\equiv^{\updownarrow}_{r,s,k}$.

THEOREM 3.13.

1. *$\mathcal{T}, u \underline{\leftrightarrow}^{\updownarrow} \mathcal{T}', u'$ implies $\mathcal{T}, u \equiv^{\updownarrow} \mathcal{T}', u'$. The converse also holds when $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching.*
2. *$\mathcal{T}, u \underline{\leftrightarrow}^{\updownarrow}_{r,s,k\cdot(r+s+2)} \mathcal{T}', u'$ implies $\mathcal{T}, u \equiv^{\updownarrow}_{r,s,k} \mathcal{T}', u'$.*
3. *$\mathcal{T}, u \equiv^{\updownarrow}_{r,s,k} \mathcal{T}', u'$ implies $\mathcal{T}, u \underline{\leftrightarrow}^{\updownarrow}_{r,s,k} \mathcal{T}', u'$.*

COROLLARY 3.14. *$\underline{\leftrightarrow}^{\updownarrow}_{r,s,k}$ has finite index.*

4. SIMULATION

In this section we define notions of directed (non-symmetric) simulations for $\mathrm{XPath}^{\downarrow}_{=}$ and $\mathrm{XPath}^{\updownarrow}_{=}$, as it is done, e.g., in [16] for some modal logics. We obtain results similar to Theorems 3.5 and 3.13 but relating each simulation notion with the corresponding logical implication.

We say that an $\mathrm{XPath}_{=}$ formula is **positive** if it contains no negation $\neg$ and no inequality data tests $\langle \alpha \neq \beta \rangle$. For $\mathcal{L}$ one of $\mathrm{XPath}^{\downarrow}_{=}$, $\mathrm{XPath}^{\updownarrow}_{=}$, $\mathrm{XPath}^{\downarrow\downarrow^*}_{=}$, or $\mathrm{XPath}^{\updownarrow\updownarrow^*}_{=}$, we write $\mathcal{L}^{+}$ for the positive fragment of $\mathcal{L}$.

A **simulation for $\mathbf{XPath}^{\downarrow}_{=}$** [resp. for $\mathrm{XPath}^{\updownarrow}_{=}$] is simply a bisimulation from which the *Zag* clause and half of the first condition in the *Zig* clause have been omitted. Observe that simulations need not be symmetric.

Formally, we say that $u \in T$ is **similar to** $u' \in T'$ **for $\mathbf{XPath}^{\downarrow}_{=}$** (notation: $\mathcal{T}, u \underline{\rightarrow}^{\downarrow} \mathcal{T}', u'$) 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', $x\overset{n}{\to}v$ and $x\overset{m}{\to}w$ then there are $v', w' \in T'$ such that $x'\overset{n}{\to}v'$, $x'\overset{m}{\to}w'$ and
  1. $data(v) = data(w) \Rightarrow data(v') = data(w')$,
  2. $(\overset{i}{\to}v)\, Z\, (\overset{i}{\to}v')$ for all $0 \leq i < n$, and
  3. $(\overset{i}{\to}w)\, Z\, (\overset{i}{\to}w')$ for all $0 \leq i < m$.

$u \in T$ is **similar to** $u' \in T'$ **for** $\mathbf{XPath}^{\updownarrow}_{=}$ (notation: $\mathcal{T}, u \mathrel{\underline{\to}}^{\updownarrow} \mathcal{T}', u'$) 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', $y\overset{n}{\to}x$ and $y\overset{m}{\to}z$ then there are $y', z' \in T'$ such that $y'\overset{n}{\to}x'$, $y'\overset{m}{\to}z'$, zZz', and if $data(z) = data(x)$ then $data(z') = data(x')$.

Relations $\mathrel{\underline{\to}}^{\downarrow}_{\ell}$ and $\mathrel{\underline{\to}}^{\updownarrow}_{r,s,k}$ are defined accordingly. We define one-way (non-symmetric) logical implication between models as follows. We write $\mathcal{T}, u \Rrightarrow^{\downarrow} \mathcal{T}', u'$ for

$$(\forall \varphi \in \mathrm{XPath}^{\downarrow+}_{=})\ [\mathcal{T}, u \models \varphi \Rightarrow \mathcal{T}', u' \models \varphi].$$

Define $\Rrightarrow^{\downarrow}_{\ell}$, $\Rrightarrow^{\updownarrow}$, and $\Rrightarrow^{\updownarrow}_{r,s,k}$ in an analogous way for ℓ-$\mathrm{XPath}^{\downarrow+}_{=}$, $\mathrm{XPath}^{\updownarrow+}_{=}$, (r, s, k)-$\mathrm{XPath}^{\updownarrow+}_{=}$, respectively. As for bisimulation, we have that $\mathrel{\underline{\to}}$ coincides with $\Rrightarrow$.

Theorem 4.1.

1. *Let $\dagger \in \{\downarrow, \updownarrow\}$. $\mathcal{T}, u \mathrel{\underline{\to}}^{\dagger} \mathcal{T}', u'$ implies $\mathcal{T}, u \Rrightarrow^{\dagger} \mathcal{T}', u'$. The converse holds when $\mathcal{T}'$ is finitely branching.*
2. *$\mathcal{T}, u \mathrel{\underline{\to}}^{\downarrow}_{\ell} \mathcal{T}', u'$ iff $\mathcal{T}, u \Rrightarrow^{\downarrow}_{\ell} \mathcal{T}', u'$.*
3. *$\mathcal{T}, u \mathrel{\underline{\to}}^{\updownarrow}_{r,s,k\cdot(r+s+2)} \mathcal{T}', u'$ implies $\mathcal{T}, u \Rrightarrow^{\updownarrow}_{r,s,k} \mathcal{T}', u'$.*
4. *$\mathcal{T}, u \Rrightarrow^{\updownarrow}_{r,s,k} \mathcal{T}', u'$ implies $\mathcal{T}, u \mathrel{\underline{\to}}^{\updownarrow}_{r,s,k} \mathcal{T}', u'$.*

We say that $\mathcal{T}'$ is a **substructure** of $\mathcal{T}$ if $\mathcal{T}'$ is a data tree which results from removing some nodes of $\mathcal{T}$, i.e., $T' \subseteq T$ and for all $u, v \in T'$ we have: 1) $u{\to}v$ on $\mathcal{T}$ iff $u{\to}v$ on $\mathcal{T}'$; 2) $label(u)$ on $\mathcal{T}'$ equals $label(u)$ on $\mathcal{T}$; and 3) $data(u)$ on $\mathcal{T}'$ equals $data(u)$ on $\mathcal{T}$. Equivalently, seen as σ-structures, $\mathcal{T}'$ is the σ-substructure of $\mathcal{T}$ induced by $T' \subseteq T$. One can verify that the identity on T' is a simulation for $\mathrm{XPath}^{\updownarrow}_{=}$ from $\mathcal{T}'$ to $\mathcal{T}$.

Lemma 4.2. *If $\mathcal{T}'$ is a substructure of $\mathcal{T}$ and $u' \in T'$ then $\mathcal{T}', u' \mathrel{\underline{\to}}^{\updownarrow} \mathcal{T}, u'$.*

We obtain that the formulas of $\mathrm{XPath}_{=}$ invariant under simulations are, precisely, the positive ones.

Theorem 4.3.

1. *$\varphi \in \mathrm{XPath}^{\downarrow}_{=}$ is $\mathrel{\underline{\to}}^{\downarrow}$-invariant [resp. $\mathrel{\underline{\to}}^{\downarrow}_{\ell}$] iff it is equivalent to a formula of $\mathrm{XPath}^{\downarrow+}_{=}$ [resp. ℓ-$\mathrm{XPath}^{\downarrow+}_{=}$].*
2. *$\varphi \in \mathrm{XPath}^{\updownarrow}_{=}$ is $\mathrel{\underline{\to}}^{\updownarrow}$-invariant iff it is equivalent to a formula of $\mathrm{XPath}^{\updownarrow+}_{=}$.*
3. *If $\varphi \in \mathrm{XPath}^{\updownarrow}_{=}$ is $\mathrel{\underline{\to}}^{\updownarrow}_{r,s,k}$-invariant then it is equivalent to a formula of (r, s, k)-$\mathrm{XPath}^{\updownarrow+}_{=}$.*
4. *If $\varphi \in \mathrm{XPath}^{\updownarrow}_{=}$ is equivalent to a formula of (r, s, k)-$\mathrm{XPath}^{\updownarrow+}_{=}$ then φ is $\mathrel{\underline{\to}}^{\updownarrow}_{r,s,k'}$-invariant, for $k' = k\cdot(r+s+2)$.*

5. ADDING TRANSITIVITY

As it happens, for example, with the basic modal logic and propositional dynamic logic, the same notion of bisimulation [resp. simulation] of each logic captures the logical equivalence [resp. logical implication] for the corresponding fragments including the reflexive-transitive closure of the axes which are present. Intuitively, this occurs because $\downarrow_*$ is an infinite union of compositions of $\downarrow$, and similarly for $\uparrow$.

Let $\equiv^{\downarrow\downarrow_*}$ and $\equiv^{\updownarrow\updownarrow^*}$ be the logical equivalence relation for $\mathrm{XPath}^{\downarrow\downarrow_*}_{=}$ and $\mathrm{XPath}^{\updownarrow\updownarrow^*}_{=}$ respectively, and let $\Rrightarrow^{\downarrow\downarrow^*}$ and $\Rrightarrow^{\updownarrow\updownarrow^*}$ be the logical implication for $\mathrm{XPath}^{\downarrow\downarrow_*+}_{=}$ and $\mathrm{XPath}^{\updownarrow\updownarrow^*+}_{=}$ respectively.

Theorem 5.1. *Let $\dagger \in \{\downarrow\downarrow_*, \updownarrow\updownarrow^*\}$.*

1. *$\mathcal{T}, u \mathrel{\underline{\leftrightarrow}}^{\dagger} \mathcal{T}', u'$ implies $\mathcal{T}, u \equiv^{\dagger} \mathcal{T}', u'$. The converse also holds when $\mathcal{T}'$ is finitely branching.*
2. *$\mathcal{T}, u \mathrel{\underline{\to}}^{\dagger} \mathcal{T}', u'$ implies $\mathcal{T}, u \Rrightarrow^{\dagger} \mathcal{T}', u'$. The converse also holds when $\mathcal{T}'$ is finitely branching.*

6. CHARACTERIZATION

In §6.1 we show that there is a truth-preserving translation from $\mathrm{XPath}^{\updownarrow}_{=}$ to first-order logic over an appropriate signature. In §6.2 we characterize $\mathrm{XPath}^{\downarrow}_{=}$ as the fragment of first-order logic $\mathrel{\underline{\leftrightarrow}}^{\downarrow}$-invariant over data trees. In §6.3 we show that this result fails for $\mathrm{XPath}^{\updownarrow}_{=}$ in general, but a weaker result can still be proved.

6.1 Translating to first-order logic

We say that an $\mathrm{XPath}^{\updownarrow}_{=}$-path expression α is in **simple normal form** if it is of the form

$$[\varphi_0]o_1[\varphi_1]o_2 \cdots o_n[\varphi_n],$$

for $n \geq 0$, $\varphi_i \in \mathrm{XPath}^{\updownarrow}_{=}$, and $o_i \in \{\downarrow, \uparrow\}$. For any $\mathrm{XPath}^{\updownarrow}_{=}$- [resp. $\mathrm{XPath}^{\downarrow}_{=}$-] path expression α there is an equivalent $\mathrm{XPath}^{\updownarrow}_{=}$- [resp. $\mathrm{XPath}^{\downarrow}_{=}$-] path expression α' in simple normal form. Further, α' can be computed in polynomial time from α.[1] We say that an $\mathrm{XPath}^{\updownarrow}_{=}$-formula φ is in **simple normal form** if each path expression α occurring in φ is in simple normal form.

Fix the signature σ with binary relations $\rightsquigarrow$ and $\approx$, 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 such that

$$\begin{aligned}
\rightsquigarrow^{\mathcal{T}} &= \{(x, y) \in T^2 \mid y \text{ is a child of } x\};\\
\approx^{\mathcal{T}} &= \{(x, y) \in T^2 \mid data(x) = data(y)\};\\
P^{\mathcal{T}}_a &= \{x \in T \mid label(x) = a\}.
\end{aligned}$$

We define the following translation Tr mapping $\mathrm{XPath}^{\updownarrow}_{=}$ formulas in simple normal form to first-order σ-formulas:

$$\begin{aligned}
\mathrm{Tr}_x(a) &= P_a(x) && (a \in \mathbb{A})\\
\mathrm{Tr}_x(\varphi \dagger \psi) &= \mathrm{Tr}_x(\varphi) \dagger \mathrm{Tr}_x(\psi) && (\dagger \in \{\wedge, \vee\})\\
\mathrm{Tr}_x(\neg\varphi) &= \neg\mathrm{Tr}_x(\varphi)\\
\mathrm{Tr}_x(\langle\alpha\rangle) &= (\exists\bar{y})\big(x = y_0 \wedge \mathrm{Tr}_{\bar{y}}(\alpha)\big)\\
\mathrm{Tr}_x(\langle\alpha = \beta\rangle) &= (\exists\bar{y})(\exists\bar{z})\big(x = y_0 \wedge x = z_0 \wedge y_n \approx z_m \wedge\\
&\qquad \mathrm{Tr}_{\bar{y}}(\alpha) \wedge \mathrm{Tr}_{\bar{z}}(\beta)\big)\\
\mathrm{Tr}_x(\langle\alpha \neq \beta\rangle) &= (\exists\bar{y})(\exists\bar{z})\big(x = y_0 \wedge x = z_0 \wedge y_n \not\approx z_m \wedge
\end{aligned}$$

[1] Note that this proposition holds only for paths expressions without union.

$\mathrm{Tr}_{\bar{y}}(\alpha) \wedge \mathrm{Tr}_{\bar{z}}(\beta))$

$$\mathrm{Tr}_{\bar{y}}(\alpha) = \bigwedge_{i=0}^{n-1} o_{i+1}(y_i, y_{i+1}) \wedge \bigwedge_{i=0}^{n} \mathrm{Tr}_{y_i}(\varphi_i),$$

where $\bar{y} = y_0, \ldots, y_n$ and $\bar{z} = z_0, \ldots, z_m$, and are fresh when quantified in the fourth and fifth definition;

$$\alpha = [\varphi_0]o_1[\varphi_1]o_2[\varphi_2]o_3 \cdots o_n[\varphi_n];$$
$$\beta = [\psi_0]o'_1[\psi_1]o'_2[\psi_2]o'_3 \cdots o'_m[\psi_m];$$

$o_i, o'_i \in \{\downarrow, \uparrow\}$; $o_j(u,v)$ represents $u \rightsquigarrow v$ if $o_j = \downarrow$, and $v \rightsquigarrow u$ otherwise. For $\varphi \in \mathrm{XPath}^{\updownarrow}_{=}$ we have $\mathcal{T}, u \models \varphi$ iff $\mathcal{T} \models \mathrm{Tr}_x(\varphi)(u)$.

6.2 Downward XPath

Let $\mathrm{FO}(\sigma)$ be the set of first-order formulas over a given signature σ, and let $\mathcal{C}$ be a class of σ-models. An $\mathrm{FO}(\sigma)$-formula $\varphi(x)$ is **ℓ-local** if for all data trees $\mathcal{T}$ and $u \in T$, we have $\mathcal{T} \models \varphi(u) \Leftrightarrow \mathcal{T}|_\ell u \models \varphi(u)$. Finally, for $\varphi \in \mathrm{FO}(\sigma)$ let $\mathrm{qr}(\varphi)$ be its quantifier rank, i.e., the depth of nesting of its quantifiers.

Observe that the following result has two readings: one classical, and one restricted to finite models.

Theorem 6.1 (Characterization). *Let $\varphi(x) \in \mathrm{FO}(\sigma)$. The following are equivalent:*

(i) φ is $\underline{\leftrightarrow}^{\downarrow}$-invariant over [finite] data-trees;

(ii) φ is logically equivalent over [finite] data-trees to an ℓ-$\mathrm{XPath}^{\downarrow}_{=}$-formula, where $\ell = 2^{\mathrm{qr}(\varphi)} - 1$.

Proof. The implication (ii) $\Rightarrow$ (i) follows straightforwardly from Theorem 3.5. The proof of (i) $\Rightarrow$ (ii) goes as follows: First, we show that any $\underline{\leftrightarrow}^{\downarrow}$-invariant $\varphi(x) \in \mathrm{FO}(\sigma)$ is ℓ-local for $\ell = 2^{\mathrm{qr}(\varphi)} - 1$ (Proposition 6.2). Then, we prove that any $\underline{\leftrightarrow}^{\downarrow}$-invariant $\varphi(x) \in \mathrm{FO}(\sigma)$ that is ℓ-local is $\underline{\leftrightarrow}^{\downarrow}_{\ell}$-invariant. Finally, we show that any $\mathrm{FO}(\sigma)$-definable property which is $\underline{\leftrightarrow}^{\downarrow}_{\ell}$-invariant is definable in ℓ-$\mathrm{XPath}^{\downarrow}_{=}$. $\square$

Proposition 6.2. *Any $\underline{\leftrightarrow}^{\downarrow}$-invariant $\varphi(x) \in \mathrm{FO}(\sigma)$ over [finite] data-trees is ℓ-local for $\ell = 2^{\mathrm{qr}(\varphi)} - 1$.*

Proof. We follow Otto's proof [20]. Assume that $\varphi(x) \in \mathrm{FO}(\sigma)$ is $\underline{\leftrightarrow}^{\downarrow}$-invariant, let $q = \mathrm{qr}(\varphi)$, and put $\ell = 2^q - 1$. Given a data tree $\mathcal{T}$ and $u \in T$ it suffices to show the existence of data trees $\mathcal{T}'$ and $\mathcal{T}''$, with corresponding elements $u' \in T'$ and $u'' \in T''$ such that

(a) $\mathcal{T}', u' \underline{\leftrightarrow}^{\downarrow} \mathcal{T}, u$,

(b) $\mathcal{T}'', u'' \underline{\leftrightarrow}^{\downarrow} (\mathcal{T}|_\ell u), u$, and

(c) $\mathcal{T}', u' \equiv_q \mathcal{T}'', u''$.

Indeed, from the above conditions it follows that

$$\begin{aligned} \mathcal{T} \models \varphi(u) &\text{ iff } \mathcal{T}' \models \varphi(u') && \text{((a) and } \underline{\leftrightarrow}^{\downarrow}\text{-inv. of } \varphi) \\ &\text{ iff } \mathcal{T}'' \models \varphi(u'') && \text{(c)} \\ &\text{ iff } (\mathcal{T}|_\ell u) \models \varphi(u), && \text{((b) and } \underline{\leftrightarrow}^{\downarrow}\text{-inv. of } \varphi) \end{aligned}$$

and hence φ is ℓ-local. By Proposition 3.3 one may assume that $u \in T$ is the root of $\mathcal{T}$.

We define $\mathcal{T}'$ and $\mathcal{T}''$, as structures that are disjoint copies of sufficiently many isomorphic copies of $\mathcal{T}$ and $\mathcal{T}|_\ell u$, respectively, all tied together by some common root. Both structures have q isomorphic copies of both $\mathcal{T}$ and $\mathcal{T}|_\ell u$, and only distinguish themselves by the nature of the one extra subtree, in which u' and u'' live, respectively: u' is the root of one of the copies of $\mathcal{T}$ and u'' is the root of one of the copies of $\mathcal{T}|_\ell u$. We indicate the two structures in the diagram of Figure 4, with distinguished elements u' and u'' marked by $\bullet$; the open cones stand for copies of $\mathcal{T}$, the closed cones for copies of $\mathcal{T}|_\ell u$. The new isomorphic copies have the same data values as the original one. The new root has an arbitrary, fixed, data value and label.

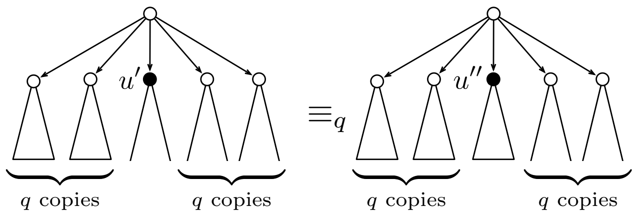

Figure 4: Definition of $\mathcal{T}', u'$ and $\mathcal{T}'', u''$.

By Proposition 3.4, it is straightforward that conditions (a) and (b) are satisfied. Condition (c) is true because one can exhibit a strategy for player **II** in the q-round Ehrenfeucht-Fraïssé game on structures $\mathcal{T}'$ and $\mathcal{T}''$. The strategy is exactly the same used in [20]. $\square$

6.3 Vertical XPath

The analog of Theorem 6.1 fails for $\mathrm{XPath}^{\updownarrow}_{=}$:

Lemma 6.3. *The $\mathrm{FO}(\sigma)$-formula*

$$(\exists x)\ P_a(x)$$

is $\underline{\leftrightarrow}^{\updownarrow}$-invariant though not logically equivalent over [finite] data-trees to any $\mathrm{XPath}^{\updownarrow}_{=}$-formula.

Hence $\mathrm{XPath}^{\updownarrow}_{=}$ is not the fragment of $\mathrm{FO}(\sigma)$ which is $\underline{\leftrightarrow}^{\updownarrow}$-invariant over [finite] data-trees. However, the following proposition still holds for the case of $\mathrm{XPath}^{\updownarrow}_{=}$:

Proposition 6.4. *Let $k' = k\cdot(r+s+2)$. If $\varphi(x) \in \mathrm{FO}(\sigma)$ is $\underline{\leftrightarrow}^{\updownarrow}_{r,s,k'}$-invariant over [finite] data-trees, then there is $\psi \in (r,s,k)$-$\mathrm{XPath}^{\updownarrow}_{=}$ such that $\mathrm{Tr}_x(\psi)$ is logically equivalent to φ over [finite] data-trees.*

Notice that the counterexample in Lemma 6.3 is an unrestricted, existential formula. One may wonder if it might be possible to extend the expressive power of $\mathrm{XPath}^{\updownarrow}_{=}$ to accout for unrestricted quantification. The natural candidate would be the modal operator E (usually known as the existential modality) which, intuitively, let us express that there is some node in the model where a formula holds. But even with the additional expressive power provided by E the analog of Theorem 6.1 fails. Formally, consider the logic $\mathrm{XPath}^{\updownarrow E}_{=}$, which results from adding the operator E to $\mathrm{XPath}^{\updownarrow}_{=}$ with the following semantics: $[\![E\varphi]\!]^{\mathcal{T}} = T$ if $[\![\varphi]\!]^{\mathcal{T}} \neq \emptyset$, and $[\![E\varphi]\!]^{\mathcal{T}} = \emptyset$ otherwise.

The following lemma shows a counterexample to the analog of Theorem 6.1, showing that $\mathrm{XPath}^{\updownarrow E}_{=}$ is not the fragment of $\mathrm{FO}(\sigma)$ $\underline{\leftrightarrow}^{\updownarrow}$-invariant over [finite] data-trees.

LEMMA 6.5. *The* $\mathrm{FO}(\sigma)$*-formula*

$$(\exists y, z)\ [y \approx z \wedge P_a(y) \wedge P_b(z)]$$

is $\underline{\leftrightarrow}^{\updownarrow}$*-invariant though not logically equivalent over [finite] data-trees to any* $XPath_{=}^{\updownarrow E}$*-formula.*

7. APPLICATIONS

We devote this section to exemplify how the model theoretic tools we developed can be used to show expressiveness results for $\mathrm{XPath}_{=}$. We do not intend to be comprehensive; rather we will exhibit a number of different results that show possible uses of the notions of bisimulation we introduced.

7.1 Expressiveness hierarchies

Define $\equiv^{\downarrow}_{\ell,k}$ as the equivalence $\equiv^{\downarrow}_{\ell}$ restricted to formulas of nesting depth at most k, that is, $\mathcal{T}, u \equiv^{\downarrow}_{\ell,k} \mathcal{T}', u'$ iff for all $\varphi \in \mathrm{XPath}^{\downarrow}_{=}$ such that $\mathrm{dd}(\varphi) \leq \ell$ and $\mathrm{nd}(\varphi) \leq k$ we have $\mathcal{T}, u \models \varphi$ iff $\mathcal{T}', u' \models \varphi$. Define a more fine-grained notion of bisimulation in a similar way. We say that $u \in T$ and $u' \in T'$ are (ℓ, k)**-bisimilar for** $\mathbf{XPath}^{\downarrow}_{=}$ (notation: $\mathcal{T}, u \underline{\leftrightarrow}^{\downarrow}_{\ell,k} \mathcal{T}', u'$) if there is a family of relations $(Z_{j,t})_{j\leq\ell, t\leq k}$ in $T \times T'$ such that $uZ_{\ell,k}u'$ and for all $j \leq \ell$, $t \leq k$, $x \in T$ and $x' \in T'$ we have

- **Harmony:** If $xZ_{j,t}x'$ then $\mathit{label}(x) = \mathit{label}(x')$.
- **Zig:** If $xZ_{j,t}x'$, $x\xrightarrow{n}v$ 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. $\mathit{data}(v) = \mathit{data}(w) \Leftrightarrow \mathit{data}(v') = \mathit{data}(w')$,
  2. if $t > 0$, $(\xrightarrow{i}v)\, Z_{j-n+i,t-1}\, (\xrightarrow{i}v')$ for all $0 \leq i < n$, and
  3. if $t > 0$, $(\xrightarrow{i}w)\, Z_{j-m+i,t-1}\, (\xrightarrow{i}w')$ for all $0 \leq i < m$.
- **Zag:** If $xZ_{j,t}x'$, $x'\xrightarrow{n}v'$ 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.

Following the same ideas used in Propositions 3.6 and 3.7, it is easy to show that (ℓ, k)-bisimulations characterize (ℓ, k)-equivalence.

PROPOSITION 7.1. $\mathcal{T}, u \underline{\leftrightarrow}^{\downarrow}_{\ell,k} \mathcal{T}', u'$ *iff* $\mathcal{T}, u \equiv^{\downarrow}_{\ell,k} \mathcal{T}', u'$.

The following theorem characterizes when an increase in nesting depth results in an increase in expressive power (see Figure 5). We conjecture that a similar hierarchy holds in the absence of data values, but this is not a direct consequence of our result.

THEOREM 7.2. *For all* $\ell, k \geq 0$, $i \geq 1$,

$$\equiv^{\downarrow}_{\ell,0} \supsetneq \equiv^{\downarrow}_{\ell,1} \supsetneq \cdots \supsetneq \equiv^{\downarrow}_{\ell,\ell} = \equiv^{\downarrow}_{\ell,\ell+i}, \text{ and}$$
$$\equiv^{\downarrow}_{\ell,k} \supsetneq \equiv^{\downarrow}_{\ell+i,k}.$$

7.2 Safe operations on models

Bisimulations can also be used to show that certain operations on models preserve truth. Such operations are usually called *safe* for a given logic, as they can be applied to a model without changing the truth values of any formula in the language. Proposition 3.3, for example, is already an example of this kind of results showing that the class of models of a formula is closed under sub-model generation. We will now show a more elaborate example.

$$\begin{array}{ccccccccccc}
\equiv^{\downarrow}_{0,0} & = & \equiv^{\downarrow}_{0,1} & = & \equiv^{\downarrow}_{0,2} & = & \equiv^{\downarrow}_{0,3} & = & \equiv^{\downarrow}_{0,4} & \cdots \\
\not\cup & & \not\cup & & \not\cup & & \not\cup & & \not\cup & \\
\equiv^{\downarrow}_{1,0} & \supsetneq & \equiv^{\downarrow}_{1,1} & = & \equiv^{\downarrow}_{1,2} & = & \equiv^{\downarrow}_{1,3} & = & \equiv^{\downarrow}_{1,4} & \cdots \\
\not\cup & & \not\cup & & \not\cup & & \not\cup & & \not\cup & \\
\equiv^{\downarrow}_{2,0} & \supsetneq & \equiv^{\downarrow}_{2,1} & \supsetneq & \equiv^{\downarrow}_{2,2} & = & \equiv^{\downarrow}_{2,3} & = & \equiv^{\downarrow}_{2,4} & \cdots \\
\not\cup & & \not\cup & & \not\cup & & \not\cup & & \not\cup & \\
\equiv^{\downarrow}_{3,0} & \supsetneq & \equiv^{\downarrow}_{3,1} & \supsetneq & \equiv^{\downarrow}_{3,2} & \supsetneq & \equiv^{\downarrow}_{3,3} & = & \equiv^{\downarrow}_{3,4} & \cdots \\
\vdots & & \vdots & & \vdots & & \vdots & & \vdots & \ddots
\end{array}$$

Figure 5: Hierarchy of $\mathbf{XPath}^{\downarrow}_{=}$**.**

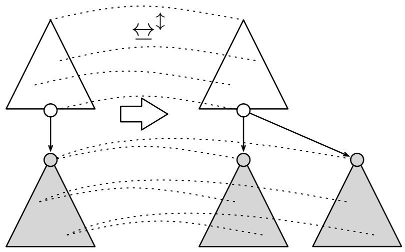

Figure 6: Closure under subtree replication.

We say that $\mathcal{T}'$ is a **subtree replication** of $\mathcal{T}$, if $\mathcal{T}'$ is the result of inserting $\mathcal{T}|x$ into $\mathcal{T}$ as a sibling of x, where x is any node of $\mathcal{T}$ different from the root. Figure 6 gives a schematic representation of this operation.

PROPOSITION 7.3. $XPath^{\updownarrow\updownarrow^*}_{=}$ *is closed under subtree replication, i.e. if* $\mathcal{T}'$ *is a subtree replication of* $\mathcal{T}$*, and* $u \in T$ *then* $\mathcal{T}', u \equiv^{\updownarrow\updownarrow^*} \mathcal{T}, u$.

PROOF. Suppose that $x \in T$ is not the root of $\mathcal{T}$, and that $\mathcal{T}'$ is the result of inserting $\mathcal{T}|x$ into $\mathcal{T}$ as a sibling of x. Let us call $\mathcal{T}_x$ to the new copy of $\mathcal{T}|x$ inserted into $\mathcal{T}'$, and let X be the set of nodes of $\mathcal{T}|x$. Furthermore, if $v \in X$ then v_x is the corresponding node of $\mathcal{T}_x$. Nodes v and v_x have the same label and data value, and the position of v in $\mathcal{T}|x$ coincides with the position of v_x in $\mathcal{T}_x$.

By Theorem 5.1, it suffices to verify that $\mathcal{T}, u \underline{\leftrightarrow}^{\updownarrow} \mathcal{T}', u$ via $Z \subseteq T \times T'$ defined by:

$$Z = \{(y, y) \mid y \in T\} \cup \{(v, v_x) \mid v \in X\}$$

(Z is depicted as dotted lines in Figure 6). $\square$

7.3 Non-expressivity results

Finally, we will use bisimulation to show the expressivity limits of different fragments of XPath. Let $\mathit{key}(a)$ be the property stating that every node with label a has a different data value. Let $\mathit{fk}(a, b)$ (for *foreign key*) be the property $(\forall x)[P_a(x) \Rightarrow (\exists y)[P_b(y) \wedge x \sim y]]$.

PROPOSITION 7.4.

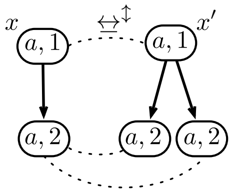

Figure 7: *key*(a) not in XPath$_=^{\updownarrow^*}$.

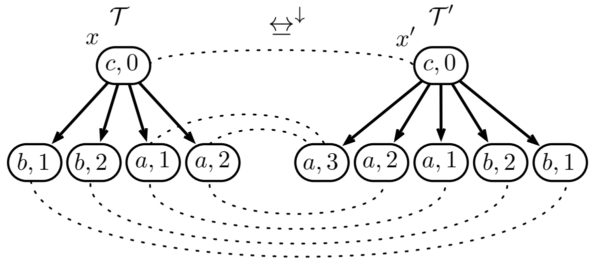

Figure 8: *fk*(a,b) not in XPath$_=^{\downarrow\downarrow_*}$.

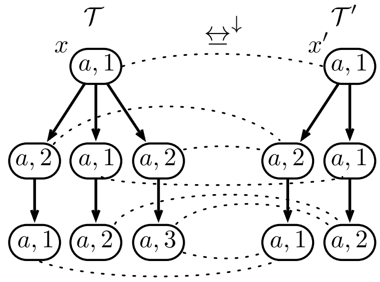

Figure 9: *dist*$_3$ not in XPath$_=^{\downarrow\downarrow_*}$.

1. *key*(a) *is not expressible in XPath*$_=^{\updownarrow^*}$.
2. *fk*(a,b) *is expressible in XPath*$_=^{\updownarrow^*}$ *but it is not expressible in XPath*$_=^{\downarrow\downarrow_*}$ *or XPath*$_=^{\updownarrow^*+}$.

Proof. The first item follows from Proposition 7.3. Since the logic is closed under subtree replication, the trees of Figure 7 are equivalent. As *key*(a) holds in one and not in the other, the statement follows.

For the second item, it is easy to see that *fk*(a,b) is expressible with the formula $\neg\langle\uparrow^*\downarrow_*[a \wedge \neg\langle\varepsilon = \uparrow^*\downarrow_*[b]\rangle]\rangle$. However, this property cannot be expressed in XPath$_=^{\downarrow\downarrow_*}$ because the models $\mathcal{T}$ and $\mathcal{T}'$ in Figure 8 are bisimilar for XPath$_=^{\downarrow}$ via Z, depicted as dotted lines. Since $\mathcal{T}, x$ satisfies *fk*(a,b) but $\mathcal{T}', x'$ does not, from Theorem 5.1 it follows that *fk*(a,b) is not expressible in XPath$_=^{\downarrow\downarrow_*}$.

Finally, suppose there exists $\psi \in$ XPath$_=^{\updownarrow^*+}$ expressing *fk*(a,b). Since $\mathcal{T}$ is a substructure of $\mathcal{T}'$ we have $\mathcal{T}, x \underline{\to}^{\updownarrow} \mathcal{T}', x$ by Lemma 4.2. By Theorem 5.1(2) and the fact that $\mathcal{T}, x \models \psi$, we have $\mathcal{T}', x \models \psi$, which is a contradiction. $\square$

Let *dist*$_3(x)$ be the property stating that there are nodes y, z so that $x \to y \to z$ and x, y, z have pairwise distinct data values.

Proposition 7.5.

1. *dist*$_3$ *is expressible in XPath*$_=^{\updownarrow}$;
2. *dist*$_3$ *is not expressible in XPath*$_=^{\downarrow\downarrow_*}$;
3. *neither dist*$_3$ *nor its complement can be expressed in XPath*$_=^{\updownarrow^*+}$.

Proof. For 1, one can check that $\mathcal{T}, x \models \varphi$ iff $\mathcal{T}, x$ satisfies *dist*$_3$, for $\varphi = \langle\varepsilon \neq \downarrow\downarrow[\langle\varepsilon \neq \uparrow\rangle]\rangle$.

Let us see 2. Consider the data trees $\mathcal{T}, x$ and $\mathcal{T}', x'$ depicted in Figure 9. It is straightforward that $\mathcal{T}, x$ satisfies *dist*$_3$ and $\mathcal{T}', x'$ does not.

Let v_1' and v_2' be the leaves of $\mathcal{T}'$ and let v be the only node of $\mathcal{T}$ with data value 3. One can check that $\mathcal{T}, x \underline{\leftrightarrow}^{\downarrow} \mathcal{T}', x'$ via $Z \subseteq T \times T'$ defined by

$$Z = \{\langle u, u'\rangle \mid h(u) = h(u') \wedge \mathit{data}(u) = \mathit{data}(u')\} \cup \{\langle v, v_1'\rangle, \langle v, v_2'\rangle\},$$

where $h(y)$ denotes the height of y, i.e., the distance from y to the root of the corresponding tree (Z is depicted as dotted lines in Figure 9). Since $\mathcal{T}, x$ satisfies *dist*$_3$ but $\mathcal{T}', x'$ does not, from Theorem 5.1 it follows that *dist*$_3$ is not expressible in XPath$_=^{\downarrow\downarrow_*}$.

For 3, one can verify that $\mathcal{T}, x \underline{\to}^{\updownarrow} \mathcal{T}', x'$ via Z as defined above. If *dist*$_3$ were definable in XPath$_=^{\updownarrow^*+}$ via ψ and the fact that $\mathcal{T}, x \models \psi$, by Theorem 5.1(2) we would have $\mathcal{T}', x' \models \psi$, and this is a contradiction.

Let $\overline{dist_3}$ denote the complement of *dist*$_3$, i.e., $\overline{dist_3}(x)$ iff for all y, z so that $x \to y \to z$, we have that x, y, z do not have pairwise distinct data values. Now $\mathcal{T}', x'$ satisfies $\overline{dist_3}$ and $\mathcal{T}, x$ does not. Since $\mathcal{T}'$ is a substructure of $\mathcal{T}$, by an argument analog to the one used in the proof of Proposition 7.4-2, we conclude that $\overline{dist_3}$ is not expressible in XPath$_=^{\updownarrow^*+}$. $\square$

8. DISCUSSION

In this article we studied model theoretic properties of XPath over both finite and arbitrary data trees using bisimulations. One of the main results we discuss is the characterization of the downward and vertical fragments of XPath as the fragments of first-order logic which are invariant under suitable notions of bisimulation. This can be seen as a first step in the larger program of studying the model theory and expressiveness of XPath with data values and, more generally, of logics on data trees. It would be interesting to study notions of bisimulation with only descendant; or characterizations of XPath with child and descendant, as a fragment of FO with the descendant relation on data trees. We did not considered XPath with horizontal navigation between siblings, such as the axes `next-sibling` and `previous-sibling`. In fact, adding these axes results in a fragment that is somewhat less interesting since the adequate bisimulation notion on finite data trees corresponds precisely to data tree isomorphism modulo renaming of data values.

In Section 7 we show a number of concrete application of the model theoretic tools we developed, discussing both expressivity and non-expressivity results. We also show examples of operations which are safe for a given XPath fragment. It would be worthwhile to devise other model operations that preserve truth of XPath formulas as we show is the case for *subtree replication*.

An important application of bisimulation is as a minimization method: given a data tree $\mathcal{T}_1$ we want to find a data tree $\mathcal{T}_2$, as small as possible, so that $\mathcal{T}_1$ and $\mathcal{T}_2$ are bisimilar for some fragment $\mathcal{L}$ of XPath. Since $\mathcal{L}$ cannot distinguish between $\mathcal{T}_1$ and $\mathcal{T}_2$, we can use $\mathcal{T}_2$ as representative of $\mathcal{T}_1$ while the expressive power of $\mathcal{L}$ is all that is required by a given application. The complexity of several inference tasks (e.g., model checking) depends directly on the model size. This is why in some cases it may be profitable to first apply a minimization step. The existence of efficient minimization algorithms is intimately related to bisimulations: we can minimize a data tree $\mathcal{T}$ by partitioning it in terms of its coarsest auto-bisimulation. We plan to design and implement algorithms for data tree minimization using bisimulation and investigate their computational complexity.

References

[1] M. Benedikt, W. Fan, and F. Geerts. XPath satisfiability in the presence of DTDs. *Journal of the ACM*, 55(2):1–79, 2008.

[2] M. Benedikt and C. Koch. XPath leashed. *ACM Computing Surveys*, 41(1), 2008.

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

[4] 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.

[5] M. Bojańczyk and P. Parys. XPath evaluation in linear time. *Journal of the ACM*, 58(4):17, 2011.

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

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

[8] D. Figueira. *Reasoning on Words and Trees with Data.* PhD thesis, Laboratoire Spécification et Vérification, ENS Cachan, France, 2010.

[9] D. Figueira. Decidability of downward XPath. *ACM Transactions on Computational Logic*, 13(4), 2012.

[10] D. Figueira and L. Segoufin. Bottom-up automata on data trees and vertical XPath. In *International Symposium on Theoretical Aspects of Computer Science (STACS'11)*, volume 9 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 93–104. Leibniz-Zentrum für Informatik, 2011.

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

[12] V. Goranko and M. Otto. Model theory of modal logic. In J. Van Benthem P. Blackburn and F. Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, chapter 5, pages 249–329. Elsevier, 2007.

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

[14] Marc Gyssens, Jan Paredaens, Dirk Van Gucht, and George H. L. Fletcher. Structural characterizations of the semantics of xpath as navigation tool on a document. In *PODS*, pages 318–327. ACM, 2006.

[15] D. Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic. Vol. II*, volume 165 of *Synthese Library*, pages 497–604. D. Reidel Publishing Co., Dordrecht, 1984. Extensions of classical logic.

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

[17] M. Marx. XPath with conditional axis relations. In *International Conference on Extending Database Technology (EDBT'04)*, volume 2992 of *LNCS*, pages 477–494. Springer, 2004.

[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 *LNCS*. 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. Modal and guarded characterisation theorems over finite transition systems. *Annals of Pure and Applied Logic*, 130(1-3):173–205, 2004.

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

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

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

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

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