Journal of Artificial Intelligence Research 53 (2015) 271-314 Submitted 12/14; published 07/15

Model Theory of XPath on Data Trees. Part I: Bisimulation and Characterization

Diego Figueira DIEGO.FIGUEIRA@LABRI.FR
CNRS, LaBRI, France

Santiago Figueira SANTIAGO@DC.UBA.AR
Universidad de Buenos Aires and CONICET, Argentina

Carlos Areces CARLOS.ARECES@GMAIL.COM
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`, `parent`, `ancestor` and `descendant` 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 both 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.

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 (Clark & DeRose, 1999).

Core-XPath (term coined in Gottlob, Koch, & Pichler, 2005) 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 reasoning 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 Document

Type Definitions (DTDs) (Marx, 2004; Benedikt, Fan, & Geerts, 2008). 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 (Marx & de Rijke, 2005), and that it is strictly less expressive than PDL with converse over trees (Benedikt & Koch, 2008). 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 the work of Bojańczyk, Muscholl, Schwentick, and Segoufin (2009). Here, we will call this logic $\mathrm{XPath}_=$. Models of $\mathrm{XPath}_=$ 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 $\mathrm{XPath}_=$ 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 of logics evaluated over data trees. For example, satisfiability and evaluation are discussed in the works of Figueira (2010) and Bojańczyk and Parys (2011). In particular, all the logics studied in this article have a decidable satisfiability problem (Figueira & Segoufin, 2011; Figueira, 2012); 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 $\mathrm{XPath}_=$ on data trees.

In this article we focus on the basic model theory tool of *bisimulations*, which defines the structural conditions necessary for ensuring that two models coincide in all the properties expressible by a logic. Whereas the basic notion of bisimulation was introduced for the basic modal logic, one can find adequate notions of bisimulations for different logics, in the sense that they capture the notion of indistiguishability of the logic. The challenge here is to find adequate notions of bisimulation for logics such as XPath, whose navigation is akin to modal logics such as PDL, but which can also test for equality of data values in the data tree.

Contribution: $\mathrm{XPath}_=$ can navigate the data tree by means of axes like `child` (that we will note $\downarrow$), `descendant` ($\downarrow_*$), `parent` ($\uparrow$), `ancestor` ($\uparrow^*$), etc. $\mathrm{XPath}_=$ 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: $\mathrm{XPath}_=(\downarrow)$ ($\mathrm{XPath}_=$ with $\downarrow$); $\mathrm{XPath}_=(\uparrow\downarrow)$ ($\mathrm{XPath}_=$ with $\downarrow$ and $\uparrow$); $\mathrm{XPath}_=(\downarrow\downarrow_*)$ ($\mathrm{XPath}_=$ with $\downarrow$ and $\downarrow_*$); $\mathrm{XPath}_=(\downarrow\uparrow\downarrow_*\uparrow^*)$ ($\mathrm{XPath}_=$ with $\downarrow$, $\uparrow$, $\downarrow_*$ and $\uparrow^*$); and its positive fragments. Our main contributions can be summarized as follows:

- In §3 we introduce bisimulation notions for $\mathrm{XPath}_=(\downarrow)$, $\mathrm{XPath}_=(\downarrow\downarrow_*)$, $\mathrm{XPath}_=(\downarrow_*)$, $\mathrm{XPath}_=(\uparrow\downarrow)$, $\mathrm{XPath}_=(\downarrow\uparrow\downarrow_*\uparrow^*)$ and show that they precisely characterize the logical equivalence relation of the corresponding logic. We also consider fine grained versions of these bisimulations indexed by two measures of formula complexity. The first measure of formula complexity consists on the maximum number of nested axes in a formula, which we call *downward depth* in the case of $\mathrm{XPath}_=(\downarrow)$ and *vertical depth* in the case of $\mathrm{XPath}_=(\uparrow\downarrow)$. The second one is the number of nested subformulas, called *nesting depth*.

  The notion of bisimulation for $\mathrm{XPath}_=(\uparrow\downarrow)$ relies on a normal form which we also introduce. Basically, this normal form restricts the navigation of the expressions to be very simple: either going downward or going upward and then downward. Similar normal forms for langauges on trees are folklore, and our work here consists mainly in adapting them to the setup of tests for data values.

  We also 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 §4 we characterize $\mathrm{XPath}_=(\downarrow)$ 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 $\mathrm{XPath}_=(\uparrow\downarrow)$ instead the characterization fails as we show in a counter-example. However, a weaker result can still be established, namely that if a first-order formula is bisimulation-invariant, for the bisimulation notion corresponding to any fixed number of nested axes, then it is equivalent to an $\mathrm{XPath}_=(\uparrow\downarrow)$ formula.

- Using bisimulations we show some (non)expressivity results about $\mathrm{XPath}_=$ in §5. We show, for example, that formulas of $\mathrm{XPath}_=(\downarrow)$ with nesting depth $n+1$ and downward depth d have more expressive power than those of nesting and downward depth n and d respectively, as long as $n < d$.

- All results are proved both over the class of arbitrary (possibly infinite) data trees, and over the class of finite data trees.

1.1 Related Work

The notion of bisimulation was introduced independently by van Benthem (1976) in the context of modal correspondence theory, by Milner (1980) and Park (1981) in concurrency theory, and by Forti and Honsell (1983) in non-wellfounded set theory (for a historical outlook see the work of Sangiorgi, 2009). 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. XPath has been known to be closely connected to known modal languages, such as PDL and modal μ-calculus, depending on the fragments taken into account (ten Cate, Fontaine, & Litak, 2010). However, the fragments studied hitherto are "data unaware", that is, allowing to express of the structure of the model as well of the fixed set of labels.

To the best of our knowledge the present is the first work on bisimulations and invariance with logics with data tests.

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 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 (van Benthem, 1976). Van Benthem's original result over arbitrary structures was proved to hold for finite structures by Rosen (1997). The proof was then simplified and unified by Otto (2004a, 2006), and later expanded by Dawar and Otto (2009) to other classes of structures. Other formalisms of different expressive power have also been considered for querying data trees, such as first-order logic with two variables (Bojańczyk et al., 2009), tree patterns (David, 2008; Figueira & Libkin, 2014), register automata (Neven, Schwentick, & Vianu, 2004), μ-calculus with registers (Jurdziński & Lazić, 2011), or datalog programs (Abiteboul, Bourhis, Muscholl, & Wu, 2013). In the absence of data values, logics for semi-structured databases can often be seen as modal logics. In fact, structural characterizations for XPath without equality test were studied in the work of Gyssens, Paredaens, Gucht, and Fletcher (2006), and XPath is known to be captured by PDL (Harel, 1984), whose bisimulation is well-understood (Blackburn, de Rijke, & Venema, 2001). It is then natural to look for an intuitive bisimulation definition for $\text{XPath}_=$.

The first significant result concerning an algorithmic solution to the bisimulation problem for the basic modal logic was given by Hopcroft (1971), with a polynomial time algorithm for state minimization in a deterministic finite automaton. This problem is equivalent to determine a coarsest partition of a set that is stable with respect to a finite set of functions. Paige and Tarjan (1987) solved the problem for the more general case, where the restriction of stability concerns a finite set of *relations*. Kanellakis and Smolka (1990) are the first to recognize that the algorithm of Paige and Tarjan can be used to determine the maximum bisimulation for the basic modal logic in an arbitrary graph. Hence it can be decided in polynomial time whether two nodes in finite models are bisimilar for the basic modal logic —though the length of the actual formulas which distinguish two non-bisimilar nodes cannot be polynomially bounded with respect to the size of the models (Figueira & Gorín, 2010). In our case, deciding whether two nodes in finite data trees are bisimilar can also be solved in polynomial time.

A preliminary version of the present paper appeared in a work of Figueira, Figueira, and Areces (2014). Also, there is a continuation (a 'Part II'): a work by Abriola, Descotte, and Figueira (2015), after the preliminary version by Abriola, Descotte, and Figueira (2014), where they address further model theoretical questions such as definability and separation, and bisimulation notions for pairs of nodes (instead of single nodes) to capture the idea of "indistinguishability by means of path expressions" (instead of node expressions).

2. Preliminaries

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 λ for the empty string.

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.

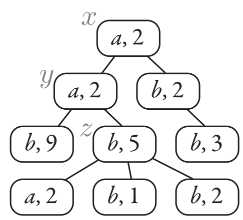


Figure 1: A data tree of *Trees*$(\mathbb{A}{\times}\mathbb{D})$ with $\mathbb{A} = \{a, b\}$ and $\mathbb{D} = \mathbb{N}$.

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, 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 $x\overset{n}{\to}y$ if y is a descendant of x at distance n. In particular, $\overset{1}{\to}$ is the same as $\to$, and $\overset{0}{\to}$ is the identity relation. We write $x\overset{*}{\to}y$ to denote that (x, y) is in the reflexive transitive closure of $\to$. $(x\overset{n}{\to})$ denotes the set of all descendants of x at distance n, and $(\overset{n}{\to}y)$ denotes the sole ancestor of y at distance n (assuming it has one).

Let P be a property on nodes of data trees. When property P is true at node u of data tree $\mathcal{T}$, we say that $(\mathcal{T}, u)$ satisfies P. For any binary relation R over nodes 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.

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:

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

A **formula** of XPath$_=(\mathcal{O})$ is either a node expression or a path expression.

We formally define the semantics of XPath$_=$ as follows, for $\mathcal{T}$ a data tree:

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

As an example, if $\mathcal{T}$ is the data tree shown in Figure 1, then

$$[\![\langle\downarrow_*[\, b \ \wedge\ \langle\downarrow[b] \neq \downarrow[b]\rangle\,]\rangle]\!]^{\mathcal{T}} = \{x, y, z\},$$

and 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 say that $\mathcal{T}, u$ is a **pointed data tree**, 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 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 $\mathcal{T}$.

The fragment of **downward XPath** is denoted XPath$_=(\downarrow)$ and **vertical XPath** is denoted XPath$_=(\uparrow\downarrow)$.

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 maximum number of nested axes or the maximum 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. In the sequel we will see that in some situations also expressions of the form $[\psi]$ can be sometimes avoided (§3.2.1), although not when we only have downward axes (Lemma 10).

2.1 Translating to First-Order Logic

In this section we show that there is a truth-preserving translation from $\text{XPath}_=(\uparrow\downarrow)$ to first-order logic over an appropriate signature. To do so, we first must interpret data trees into relational structures, and we do this in the most standard way: using a binary 'child' relation, an equivalence relation for testing data equality, and monadic relations to test for labels. 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

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

We can now give the translation from XPath to first-order logic over σ. The tranlsation function is indexed by the free variables of the formula produced —one for node expressions, and two for path expressions.

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

Proposition 1.

1. *If φ is an $\mathit{XPath}_=(\uparrow\downarrow)$ node expression, then $u \in [\![\varphi]\!]^{\mathcal{T}}$ iff $\mathcal{T} \models \mathrm{Tr}_x(\varphi)(u)$.*
2. *If α is an $\mathit{XPath}_=(\uparrow\downarrow)$ path expression, then $(u,v) \in [\![\alpha]\!]^{\mathcal{T}}$ iff $\mathcal{T} \models \mathrm{Tr}_{x,y}(\alpha)(u,v)$.*

Proof. The proof is by structural induction on φ and α. □

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 for $\odot \in \{=, \neq\}$ we have

$$\begin{aligned}
\mathrm{qr}(\mathrm{Tr}_x(\langle\alpha\rangle)) &= 1 + \mathrm{qr}(\mathrm{Tr}_x(\alpha)) \\
\mathrm{qr}(\mathrm{Tr}_x(\langle\alpha \odot \beta\rangle)) &= 2 + \max(\mathrm{qr}(\mathrm{Tr}_x(\alpha)), \mathrm{qr}(\mathrm{Tr}_x(\beta))) \\
\mathrm{qr}(\mathrm{Tr}_{x,y}(\alpha\beta)) &= 1 + \max(\mathrm{qr}(\mathrm{Tr}_{x,y}(\alpha)), \mathrm{qr}(\mathrm{Tr}_{x,y}(\beta))) \\
\mathrm{qr}(\mathrm{Tr}_{x,y}([\phi])) &= \mathrm{qr}(\mathrm{Tr}_x(\varphi)).
\end{aligned}$$

3. Bisimulation

In this section we define notions of bisimulation for the downward and vertical fragments of XPath, and we show that they coincide with the corresponding logical equivalence relation. For the case of vertical XPath the bisimulation notion relies on a normal form which we introduce for this purpose.

3.1 Downward XPath

We write $\mathrm{dd}(\varphi)$ to denote the **downward depth** of φ defined as follows:

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

where $a \in \mathbb{A}$, $\odot \in \{=, \neq\}$, and α is any path expression or the empty string λ. Let ℓ-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 node expression $\varphi \in$ 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 node expression $\varphi \in \ell$-XPath$_=(\downarrow)$, we have $\mathcal{T}, u \models \varphi$ iff $\mathcal{T}', u' \models \varphi$.[1]

We will first show that, for every ℓ, there are finitely many different formulas φ of $\mathrm{dd}(\varphi) \leq \ell$ up to logical equivalence. Formally, this is usually referred by saying that $\equiv^{\downarrow}_{\ell}$ has **finite index**.

Proposition 2. $\equiv^{\downarrow}_{\ell}$ *has finite index.*

Proof. It can be easily shown by induction that for any node expression $\varphi \in \ell$-XPath$_=(\downarrow)$ with unnecessary uses of ε (recall that $\alpha\varepsilon\beta \equiv \alpha\beta$) we have that $\mathrm{qr}(\mathrm{Tr}_x(\varphi))$ 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 3. $\{\mathcal{T}', u' \mid \mathcal{T}, u \equiv^{\downarrow}_{\ell} \mathcal{T}', u'\}$ *is definable by a node expression* $\chi_{\ell,\mathcal{T},u}$ *of* ℓ-*XPath*$_=(\downarrow)$.

Proof. Consider the conjunction of all ℓ-XPath$_=(\downarrow)$ formulas φ such that $\mathcal{T}, u \models \varphi$. By Proposition 2, up to logical equivalence, there are finitely many such φ's, and hence the conjunction is equivalent to a finite one. Define $\chi_{\ell,\mathcal{T},u}$ as this finite conjunction. □

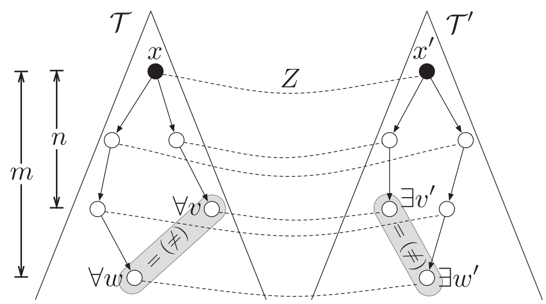
1. Two pointed data trees are 'equivalent' when they are 'indistinguishable' by the formulas of a given logic. We adopted the terminology of the literature, and so we used the first word, and not he second. The reader should not confuse this notion with that of 'equivalent' formulas.

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: $\mathcal{T}, u \underline{\leftrightarrow}^{\downarrow} \mathcal{T}', u'$) if 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 $0 \leq i < m$.

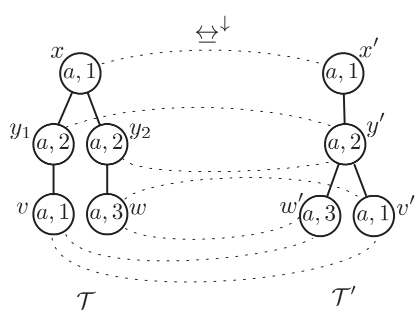
  The following picture illustrates the intended requirements.



- **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.

This bisimulation generalizes the classical bisimulation relation (Sangiorgi, 2009), in which the Zig is simply: If xZx', $x \to v$, there is v' such that $x' \to v'$ and vZv' (the Zag being symmetrical). In fact, if we restrict our Zig condition to having $n = m = 1$ and $v = w$, and the Zag condition to $n = m = 1$ and $v' = w'$, we obtain a relation that corresponds, precisely, to the classical bisimulation relation. Thus, being bisimilar for the notion we define implies being bisimlar for the classical notion.

For example, the dotted lines in the following two data trees represent a bisimulation for XPath$_=(\downarrow)$.



This notion of bisimulation can be seen, as usual, as an Ehrenfeucht-Fraïssé game, where Spoiler tries to find a difference (through the logic's glasses) between nodes u and u', while Duplicator tries to copy him, showing that u and u' are indistinguishable. To gain intuition on our notion of bisimulation, we will briefly explain its associated game (without going into the technical details). The board consists of the data trees $\mathcal{T}$ and $\mathcal{T}'$, and there are two pebbles p and p', so that all along the game, p will always be over a node of $\mathcal{T}$, and p' will always be over a node of $\mathcal{T}'$. Initially, p is over u, and p' is over u'. If u and u' do not satisfy the same label then Spoiler is declared winner and the game finishes. The game proceeds by rounds. Suppose that at some round, the pebbles p and p' are in positions x and x' satisfying the same label. A *round* consists of one move of Spoiler, followed by an answer by Duplicator, and a final decision made by spoiler.

- Step 1. Spoiler's first move: he chooses a pebble, two integers n and m, and two paths, one of length m and the other of length n. These paths start in x if he chose p or in x' if he chose p'. Suppose the chose p (the case for p' is analogous but over data tree $\mathcal{T}'$ instead of $\mathcal{T}$). His first path is represented by some $w \in T$ such that $x \xrightarrow{m} w$, and his second path is represented by some $v \in T$ such that $x \xrightarrow{n} v$.

- Step 2. Duplicator's answer: he shows two paths in $\mathcal{T}'$ (or in $\mathcal{T}$ in case duplicator had chosen p' instead of p), one of length m and the other of length n, both starting on x', such that $data(u) = data(v)$ iff $data(u') = data(v')$. If there are no such paths, then Spoiler is declared the winner and the game finishes.

- Step 3. Final move by Spoiler: he chooses
  - either $i \in \{0 \dots n-1\}$, and places pebble p over $(\xrightarrow{i} v)$ and pebble p' over $(\xrightarrow{i} v')$
  - or $i \in \{0 \dots m-1\}$, and places pebble p on $(\xrightarrow{i} w)$ and pebble p' over $(\xrightarrow{i} w')$

  If now the pebbles are over two nodes which satisfy the same label, the game proceeds. Else, Spoiler is declared the winner and the game finishes.

Duplicator wins if spoiler is never declared winner in a game at infinitely many rounds. The resemblance of the game rules with **Harmony**, **Zig** and **Zag** is evident. One can see that spoiler has a winning strategy in the game whose initial pebbles are placed over x and x' if and only if $\mathcal{T}, u \underline{\leftrightarrow}^{\downarrow} \mathcal{T}', u'$.

It is interesting to compare this game to the one for capturing *bisimulation* for the basic modal logic. In the latter, Spoiler just chooses a successor of x (or x') following the accessibility relation. This is also the case for Core-XPath or PDL. But in our case, because of adding comparison of data values, Spoiler has to choose a *whole* path (in fact, two paths), making the game 'less local' that the one for the basic modal logic (or Core-XPath or PDL). An analogous view for the game of XPath$_=(\downarrow)$ would be to allow Spoiler to build his paths in a step-by-step fashion, that is, extending the paths constructed so far with a new *single* node, thus giving Duplicator less certainty. Is it possible to change the rules of the game so that in the second step of the round Spoiler builds his paths in a step-by-step fashion? No. Even if Spoiler has the freedom to extend step-by-step *only one* of his paths, this would be too unfair for Duplicator. Indeed, consider our last example of bisimilar $\mathcal{T}, x$ and $\mathcal{T}', x'$. Spoiler would initially declare that his first path will be x' of length 0, and so

Duplicator can only define his first path to be x of length 0. Then Spoiler will construct, step-by-step, his second path. Initially he shows the path $x' \to y'$ to Duplicator. Duplicator has two possible answers: either the path $x \to y_1$ or $x \to y_2$. Suppose the chooses $x \to y_1$ (the case for $x \to y_2$ is analogous). Now Spoiler extends his path $x' \to y'$ to $x' \to y' \to w'$. Since y_1 has only one child, Spoiler has one possible answer for extending his own path: $x \to y_1 \to v$. Next, Spoiler declares that he is done: he has constructed two paths starting in x', one is of length 0 and the other one is $x' \to y' \to w'$, of length 2. Duplicator has constructed one path of length 0 and the other as $x \to y_1 \to v$. Duplicator looses, as $data(x') \neq data(w')$ but $data(x) = data(v)$. This example shows that the game fairness is established when spoiler tells duplicator *in advance* which are the two paths he chooses.

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:

Observation 4. $\mathcal{T}, u \,\underline{\leftrightarrow}^{\downarrow}\, (\mathcal{T}|u), u$.

Observation 5. *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 $u Z_\ell u'$ and for all $j \leq \ell$, $x \in T$ and $x' \in T'$ we have

- **Harmony:** If $x Z_j x'$ then $label(x) = label(x')$.
- **Zig:** If $x Z_j x'$, $x \xrightarrow{n} v$ 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. $data(v) = data(w) \Leftrightarrow data(v') = data(w')$,
  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 $x Z_j x'$, $x' \xrightarrow{n} v'$ 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.

This notion of ℓ-bisimulation corresponds to the game between spoiler and duplicator at ℓ rounds.

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.

Observation 6. *Suppose $\mathcal{T}$ and $\mathcal{T}'$ have height at most ℓ, $u \in T$, and $u' \in \mathcal{T}'$. Then $\mathcal{T}, u \,\underline{\leftrightarrow}^{\downarrow}_{\ell}\, \mathcal{T}', u'$ iff $\mathcal{T}, u \,\underline{\leftrightarrow}^{\downarrow}\, \mathcal{T}', u'$.*

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

Observation 7. $\mathcal{T}, u \,\underline{\leftrightarrow}^{\downarrow}_{\ell}\, (\mathcal{T}|_\ell u), u$.

3.1.2 Equivalence and Bisimulation

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

Theorem 8.

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

The above theorem will be shown as a consequence of Propositions 9 and 11:

Proposition 9. *$\mathcal{T}, u \leftrightarrow^{\downarrow}_{\ell} \mathcal{T}', u'$ implies $\mathcal{T}, u \equiv^{\downarrow}_{\ell} \mathcal{T}', u'$.*

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

1. If $xZ_j x'$ 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 $0 \le i \le n$, 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, $\mathit{label}(x) = \mathit{label}(x')$ 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 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 $n, m \le j$ such that $x \xrightarrow{n} v$, $x \xrightarrow{m} w$, $(x, v) \in [\![\alpha]\!]^{\mathcal{T}}$, $(x, w) \in [\![\beta]\!]^{\mathcal{T}}$ and $\mathit{data}(v) = \mathit{data}(w)$. By Zig, there are $v', w' \in T'$ such that $x' \xrightarrow{n} v'$, $x' \xrightarrow{m} w'$, $(\xrightarrow{i} v)\, Z_{j-n+i}\, (\xrightarrow{i} v')$ for all $0 \le i \le n$, $(\xrightarrow{i} w)\, Z_{j-m+i}\, (\xrightarrow{i} w')$ for all $0 \le i \le m$, and $\mathit{data}(v') = \mathit{data}(w')$. 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 $x_0, \dots, x_n \in T$ and $x'_0, \dots, x'_n \in T'$ 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 $0 \le i \le n$. Assume, for contradiction, that $(x', v') \notin [\![\alpha]\!]^{\mathcal{T}'}$. Then, there is a subformula φ of α and $k \in \{0, \dots, n\}$ such that $\mathcal{T}, x_k \models \varphi$ and $\mathcal{T}', x'_k \not\models \varphi$ as the next Lemma shows.

> **Lemma 10.** *Let α be a path expression of $XPath_=(\downarrow\downarrow_*)$. Let $x \xrightarrow{n} v$ and $x' \xrightarrow{n} v'$ such that $(x, v) \in [\![\alpha]\!]^{\mathcal{T}}$ and $(x', v') \notin [\![\alpha]\!]^{\mathcal{T}'}$. Then there is a subformula φ of α and $k \in \{0, \dots, n\}$ such that $\mathcal{T}, (\xrightarrow{k} v) \models \varphi$ and $\mathcal{T}', (\xrightarrow{k} v') \not\models \varphi$.*

Proof of Lemma. Let $x = v_0 \to v_1 \to \cdots \to v_n = v$ and $x' = v'_0 \to v'_1 \to \cdots \to v'_n = v'$. We proceed by induction on $|\alpha|$. If $\alpha = \varepsilon$ then $x = v$ and so $n = 0$. Hence $x' = v'$ and $(x', v') \in [\![\alpha]\!]^{\mathcal{T}'}$, which contradicts the hypothesis, and thus the statement is trivially true. If $\alpha = \downarrow$ then $x \to v$ and so $n = 1$. Hence $x' \to v'$ and $(x', v') \in [\![\alpha]\!]^{\mathcal{T}'}$. This case is also trivial. The case $\alpha = \downarrow_*$ is similar.

Suppose $\alpha = [\psi]$. Since $(x', v') \notin [\![\alpha]\!]^{\mathcal{T}'}$, we have $x' = v'$ and $\mathcal{T}', v' \not\models \psi$. Taking $k = 0$ and $\varphi = \psi$ the statement holds. Observe that ψ is a subformula of α.

Suppose $\alpha = \beta\gamma$. Then there is k such that $(x, v_k) \in [\![\beta]\!]^{\mathcal{T}}$ and $(v_k, v) \in [\![\gamma]\!]^{\mathcal{T}}$. Since $(x', v') \notin [\![\alpha]\!]^{\mathcal{T}'}$, we have $(x', v'_k) \notin [\![\beta]\!]^{\mathcal{T}'}$ or $(v'_k, v') \notin [\![\gamma]\!]^{\mathcal{T}'}$. In either case, apply inductive hypothesis straightforwardly. ■

But this contradicts the inductive hypothesis 1. □

Proposition 11. *$\mathcal{T}, u \equiv^{\downarrow}_{\ell} \mathcal{T}', u'$ implies $\mathcal{T}, u \leftrightarrows^{\downarrow}_{\ell} \mathcal{T}', u'$.*

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',

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

and $data(v) = data(w)$ (the case $data(v) \neq data(w)$ 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 $P \neq \emptyset$. 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. $(\forall i \in \{0, \ldots, n\})$ $\mathcal{T}, v_i \equiv^{\downarrow}_{h-i} \mathcal{T}', v'_i$, and

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

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

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

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

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

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

- Assume (a) 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 $k \in \{0, \ldots, n\} \setminus \{i\}$, let $\varphi^k_{v',w'} = \top$, and for $k \in \{0, \ldots, m\}$, let $\psi^k_{v',w'} = \top$.

- Suppose (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 $k \in \{0, \ldots, m\} \setminus \{j\}$, let $\psi^k_{v',w'} = \top$, and for $k \in \{0, \ldots, n\}$, let $\varphi^k_{v',w'} = \top$.

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

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

However these conjunctions could be potentially infinite the if trees are infinitely branching. Since $\mathrm{dd}(\varphi^i_{v',w'}) \le h - i$, by Proposition 2 there are finitely many non-equivalent node expressions $\varphi^i_{v',w'}$ and 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 $\alpha = [\Phi^0]{\downarrow}[\Phi^1]{\downarrow} \cdots {\downarrow}[\Phi^n]$ and $\beta = [\Psi^0]{\downarrow}[\Psi^1]{\downarrow} \cdots {\downarrow}[\Psi^m]$. By construction, $\mathrm{dd}(\alpha), \mathrm{dd}(\beta) \le h$ and so $\mathrm{dd}(\langle \alpha = \beta \rangle) \le h$.

It is clear that by construction $(x, v) \in [\![\alpha]\!]^{\mathcal{T}}$ and $(x, w) \in [\![\beta]\!]^{\mathcal{T}}$, and therefore $\mathcal{T}, x \models \langle \alpha = \beta \rangle$. We see next that $\mathcal{T}', x' \not\models \langle \alpha = \beta \rangle$. This contradicts $\mathcal{T}, x \equiv^{\downarrow}_h \mathcal{T}', x'$, and hence we are done. Suppose that $\mathcal{T}', x' \models \langle \alpha = \beta \rangle$. Then there is $(v', w') \in P$ such that $(x', v') \in [\![\alpha]\!]^{\mathcal{T}'}$ and $(x', w') \in [\![\beta]\!]^{\mathcal{T}'}$. In particular, i and ii are true, and then either (a) or (b) hold. In the first case, we have, by construction, that $(x', v') \notin [\![\alpha]\!]^{\mathcal{T}'}$, and in the second it is clear that $(x', w') \notin [\![\beta]\!]^{\mathcal{T}'}$. In either case, we arrive to a contradiction. $\square$

Proof of Theorem 8. Item 2 is a direct consequence of Propositions 9 and 11.

The left-to-right argument for item 1 can be seen as a consequence of item 2. Indeed, $\mathcal{T}, u \leftrightarroweq^{\downarrow} \mathcal{T}', u'$ implies $\mathcal{T}, u \leftrightarroweq^{\downarrow}_{\ell} \mathcal{T}', u'$ for all ℓ, which by item 2 implies $\mathcal{T}, u \equiv^{\downarrow}_{\ell} \mathcal{T}', u'$ for all ℓ, which in turn entails $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'$.

The right-to-left argument for item 1 is similar to that of Proposition 11, but defining Z by xZx' iff $\mathcal{T}, x \equiv^{\downarrow} \mathcal{T}', x'$. The conjunctions in (1) are then finite because $\mathcal{T}'$ is finitely branching, and so P is finite (the fact that $\mathcal{T}$ is finitely branching is used to show that Zag is satisfied). $\square$

3.2 Vertical XPath

We now study bisimulation for $\mathrm{XPath}_=(\uparrow\downarrow)$. 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 (notation: $\mathrm{vd}(\varphi)$). The **nesting depth** of a formula φ (notation: $\mathrm{nd}(\varphi)$) is the maximum number of nested [ ] appearing in φ.

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

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

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

Let (r, s, k)-$\mathrm{XPath}_=(\uparrow\downarrow)$ be the set of formulas φ in $\mathrm{XPath}_=(\uparrow\downarrow)$ 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 **equivalent for** $\mathbf{XPath}_=(\uparrow\downarrow)$ (notation: $\mathcal{T}, u \equiv^{\downarrow\uparrow} \mathcal{T}', u'$) iff for all $\varphi \in \mathrm{XPath}_=(\uparrow\downarrow)$, 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** $\mathbf{XPath}_=(\uparrow\downarrow)$ (notation: $\mathcal{T}, x \equiv^{\downarrow\uparrow}_{r,s} \mathcal{T}', x'$ [resp. $\mathcal{T}, x \equiv^{\downarrow\uparrow}_{r,s,k} \mathcal{T}', x'$]) if they satisfy the same node expressions φ of $\mathrm{XPath}_=(\uparrow\downarrow)$ so that $\mathrm{vd}(\varphi) \leq (r, s)$ [resp. $\mathrm{vd}(\varphi) \leq (r, s)$ and $\mathrm{nd}(\varphi) \leq k$].

3.2.1 Normal Form

We define a normal form for $\mathrm{XPath}_=(\uparrow\downarrow)$ that will be implicitly used in the definition of bisimulation in this section. For $n \geq 0$, 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 $\mathrm{XPath}_=(\uparrow\downarrow)$ is **downward** [resp. **upward**] if it is of the form $\downarrow^n[\varphi]$ [resp. $[\varphi]\uparrow^n$] for some $n > 0$ with $\varphi \in \mathrm{XPath}_=(\uparrow\downarrow)$. 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 $\mathrm{XPath}_=(\uparrow\downarrow)$ formula 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 $\odot \in \{=, \neq\}$. Next, we show that for every $\mathrm{XPath}_=(\uparrow\downarrow)$ formula there is an equivalent one in up-down normal form. Of course, the idea of replacing child axes with parent axes is by no means novel, and there have been a number of works on rewriting expressions into equivalent ones to improve performance or streamability, such as the works of Olteanu (2007) and Olteanu, Meuss, Furche, and Bry (2002). However, these rewrite systems aim at removing backward axes (parent, ancestor, etc) maintaining equivalence at the root, different to our end, which is to maintain equivalence of path expressions over any pair of nodes. Further, we doubt our normal form can be useful in such scenarios since from a computational or streamability perspective the resulting formula in up-down normal from seems more complex. Although being more 'regular', it has more nested modalities and introduces parent axes, as it will be shown later. Our motivation for this normal form is to simplify the defintion of bisimulation by means of rendering the formula as simple as possible in form.

Given a path expression α, the **navigation of** α (notation: $\mathrm{nav}(\alpha)$) is the string of $\{\uparrow,\downarrow\}^*$ that results from removing all node expressions $[\psi]$ and ε from α. For example, $\mathrm{nav}(\downarrow[\langle\uparrow\rangle]\downarrow[\langle\downarrow = \uparrow\rangle]\uparrow[b]) = \downarrow\downarrow\uparrow$.

Proposition 12. *Let $\varphi \in (r,s,k)$-XPath$_=(\uparrow\downarrow)$, then there is $\varphi^{\uparrow\downarrow} \in$ XPath$_=(\uparrow\downarrow)$ in up-down normal form such that the following hold*

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

Proof. The idea is that we can factorize any path in the tree going down and up as a node test in the expression. Consider for instance the expression $\alpha = \uparrow\downarrow[a]\uparrow\uparrow\downarrow$. It is immediate that α is equivalent to the up-down expression $[\langle\uparrow[\langle\downarrow[a]\rangle]\rangle]\uparrow\uparrow\downarrow$, which is in up-down normal form.

We use the following directed equivalences to translate any path expression into an equivalent up-down expression.

$$\varepsilon\gamma \equiv^{\downarrow\uparrow} \gamma \tag{ε}$$
$$\alpha[\psi_1][\psi_2]\beta \equiv^{\downarrow\uparrow} \alpha[\psi_1 \wedge \psi_2]\beta \tag{\textit{merge}}$$
$$\alpha\xi_{-n}\downarrow\cdots\downarrow\xi_{-1}\downarrow\xi_0\uparrow\xi_1\uparrow\cdots\uparrow\xi_n\beta \equiv^{\downarrow\uparrow} \alpha[\langle\xi_{-n}\xi_n\downarrow\cdots\downarrow\xi_{-1}\xi_1\downarrow\xi_0\rangle]\beta \tag{\textit{factor}}$$
$$\alpha\xi_n\downarrow\xi_{n-1}\downarrow\cdots\downarrow\xi_0 \equiv^{\downarrow\uparrow} \alpha\downarrow^n[\langle\xi_0\uparrow\xi_1\uparrow\ldots\uparrow\xi_n\rangle] \tag{\textit{shift-r}}$$
$$\xi_0\uparrow\xi_1\uparrow\cdots\uparrow\xi_n\beta \equiv^{\downarrow\uparrow} [\langle\xi_0\uparrow\xi_1\uparrow\cdots\uparrow\xi_n\rangle]\uparrow^n\beta \tag{\textit{shift-l}}$$

In the expressions above, each ξ_i is the empty string, or it is of the form ε or $[\varphi_1][\varphi_2]\ldots[\varphi_n]$, α and β can be any path expression, or the empty string, and γ is any path expression. The idea is that (*factor*) converts an expression that goes down n times and then up n times into a node expression, and when doing this, any test done in the i-th node when going down is merged with the $(n-i)$-th test when going up. For example, $\downarrow[\neg a]\downarrow[c]\uparrow[\neg b]\uparrow \equiv^{\downarrow\uparrow} [\langle\downarrow[\neg a][\neg b]\downarrow[c]\rangle]$. On the other hand, (*shift-r*) and (*shift-l*) group all the node tests in the lowest node in the expression, making use of the fact that the parent relation is functional. Thus, for example $[a]\downarrow[b]\downarrow \equiv^{\downarrow\uparrow} \downarrow\downarrow[\langle\uparrow[b]\uparrow[a]\rangle]$ and $\uparrow[a]\uparrow[b] \equiv^{\downarrow\uparrow} [\langle\uparrow[a]\uparrow[b]\rangle]\uparrow\uparrow$. It is thus clear that the left and right expressions above are semantically equivalent.

The following lemma treats the case of path expressions:

Lemma 13. *Let α be an XPath$_=(\uparrow\downarrow)$-path expression with* $\mathrm{vd}(\alpha) = (r, s)$ *and* $\mathrm{nd}(\alpha) = k$*, then there is an up-down path expression $\alpha^{\uparrow\!\downarrow}$ such that:*

1. $\alpha^{\uparrow\!\downarrow} \equiv^{\downarrow\uparrow} \alpha$
2. $\mathrm{vd}(\alpha^{\uparrow\!\downarrow}) = (r, s)$*, and*
3. $\mathrm{nd}(\alpha^{\uparrow\!\downarrow}) \leq k + r + s + 1$.

Proof of Lemma. We first apply rule (*factor*) as many times as possible. It is clear that if $\mathrm{nav}(\alpha)$ is of the form $\uparrow^n\downarrow^m$ for some $n, m \geq 0$ then rule (*factor*) cannot be applied and we are done. Hence, suppose $\mathrm{nav}(\alpha)$ contains the pattern $\downarrow\uparrow$. Let

$$\begin{aligned} \alpha &= \gamma_\uparrow \alpha_1 \gamma_\downarrow \\ \alpha_1 &= \gamma_1 \underbrace{\xi^1_{-n_1}\downarrow \dots \downarrow \xi^1_0 \uparrow \dots \uparrow \xi^1_{n_1}}_{\text{matches } (\textit{factor})} \\ &\quad\ \gamma_2 \underbrace{\xi^2_{-n_2}\downarrow \dots \downarrow \xi^2_0 \uparrow \dots \uparrow \xi^2_{n_2}}_{\text{matches } (\textit{factor})} \\ &\quad\ \vdots \\ &\quad\ \gamma_{m-1} \underbrace{\xi^m_{-n_m}\downarrow \dots \downarrow \xi^m_0 \uparrow \dots \uparrow \xi^m_{n_m}}_{\text{matches } (\textit{factor})} \gamma_m, \end{aligned}$$

where $\mathrm{nav}(\gamma_\uparrow), \mathrm{nav}(\gamma_m) \in \uparrow^*$, $\mathrm{nav}(\gamma_\downarrow), \mathrm{nav}(\gamma_1) \in \downarrow^*$, and ξ^i_j are the empty string, ε or $[\varphi^{i,j}_1][\varphi^{i,j}_2] \dots [\varphi^{i,j}_{h_{i,j}}]$. Furthermore, assume that m is maximal (i.e., it is impossible to apply (*factor*) in any of the γ_i's) and that the length of each γ_i is minimal (i.e., it is not the case that $\mathrm{nav}(\gamma_i)$ ends with $\downarrow$ and that $\mathrm{nav}(\gamma_{i+1})$ begins with $\uparrow$). Observe that $\mathrm{nav}(\gamma_i) \in \uparrow^*\downarrow^*$. We apply rule (*factor*) in the $m - 1$ marked places and obtain

$$\begin{aligned} \alpha_2 &= \gamma_1 \underbrace{[\langle \xi^1_{-n_1} \xi^1_{n_1} \downarrow \cdots \downarrow \xi^1_{-1} \xi^1_1 \downarrow \xi^1_0 \rangle]}_{(\textit{factor}) \text{ applied}} \\ &\quad\ \gamma_2 \underbrace{[\langle \xi^2_{-n_2} \xi^2_{n_2} \downarrow \cdots \downarrow \xi^2_{-1} \xi^2_1 \downarrow \xi^2_0 \rangle]}_{(\textit{factor}) \text{ applied}} \\ &\quad\ \vdots \\ &\quad\ \gamma_{m-1} \underbrace{[\langle \xi^m_{-n_m} \xi^m_{n_m} \downarrow \cdots \downarrow \xi^m_{-1} \xi^m_1 \downarrow \xi^m_0 \rangle]}_{(\textit{factor}) \text{ applied}} \gamma_m, \end{aligned}$$

Let $\mathrm{vd}(\mathrm{nav}(\alpha_1)) = (r_1, s_1)$. Since $\mathrm{nav}(\alpha) = \mathrm{nav}(\gamma_\uparrow \alpha_1 \gamma_\downarrow)$ contains the pattern $\downarrow\uparrow$, we have that $r_1 > 0$. It can be shown that $\mathrm{vd}(\gamma_\uparrow \alpha_2 \gamma_\downarrow) = (r, s)$, $\mathrm{nd}(\alpha_2) \leq \mathrm{nd}(\alpha_1) + 1$, and $\mathrm{vd}(\mathrm{nav}(\alpha_2)) \leq (r_1 - 1, s_1)$. If we repeat this procedure with α_2 and so on until we can no longer apply rule (*factor*), we end up with an up-down path expression α_f so that

1. $\alpha_f \equiv^{\downarrow\uparrow} \alpha_1$,
2. $\mathrm{vd}(\gamma_\uparrow \alpha_f \gamma_\downarrow) = (r, s)$, and
3. $\mathrm{nd}(\alpha_f) \leq \mathrm{nd}(\alpha_1) + r_1$.

After applying (ε) and (*merge*) to $\gamma_\uparrow \alpha_f \gamma_\downarrow$ as many times as possible, we obtain an equivalent α', of the same vertical and nesting depth as $\gamma_\uparrow \alpha_f \gamma_\downarrow$, of the form

$$\alpha' = \chi_1 \uparrow \chi_2 \uparrow \dots \uparrow \chi_n \downarrow \chi_{n+1} \downarrow \chi_{n+2} \downarrow \dots \downarrow \chi_{n+m},$$

where χ_i is a (possibly empty) string of the form $[\varphi_i^1] \dots [\varphi_i^{n_i}]$. We apply (*shift-l*) and (*shift-r*) to α' to obtain an equivalent α'' of the same vertical depth (i.e. $\mathrm{vd}(\alpha'') = \mathrm{vd}(\alpha') = (r, s)$) and of nesting depth equal to $\mathrm{nd}(\alpha') + 1$ of the form

$$\alpha^{\uparrow\!\downarrow} = \chi' \uparrow^n \downarrow^m \chi'',$$

where χ' and χ'' are the empty string or of the form $[\varphi]$. Observe that $\mathrm{nd}(\alpha^{\uparrow\!\downarrow}) = \mathrm{nd}(\alpha') + 1 \leq k + r_1 + 1 \leq r + s + 1$, and so $\alpha^{\uparrow\!\downarrow}$ satisfies all requirements of the lemma. This concludes the proof of Lemma 13. ∎

The following lemma treats the case of data tests:

Lemma 14. *Let $\alpha^{\uparrow\!\downarrow}, \beta^{\uparrow\!\downarrow}$ be up-down path expressions and let $\varphi = \langle \alpha^{\uparrow\!\downarrow} \odot \beta^{\uparrow\!\downarrow} \rangle$ (for $\odot \in \{=, \neq\}$) with $\mathrm{vd}(\varphi) = (r, s)$ and $\mathrm{nd}(\varphi) = k$. Then there is an up-down path expression $\gamma^{\uparrow\!\downarrow}$ such that:*

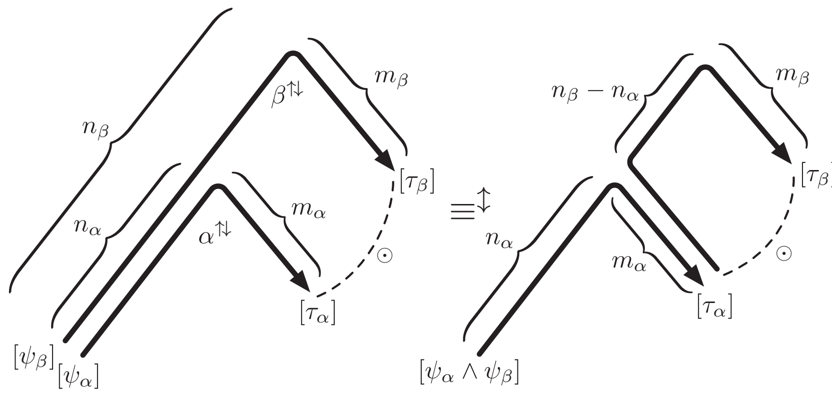
1. $\langle \gamma^{\uparrow\!\downarrow} \rangle \equiv^{\downarrow\uparrow} \varphi$,
2. $\mathrm{vd}(\gamma^{\uparrow\!\downarrow}) = (r, s)$, *and*
3. $\mathrm{nd}(\gamma^{\uparrow\!\downarrow}) \leq k + 1$.

Proof of Lemma. Let us analyse the case where $\alpha^{\uparrow\!\downarrow} = [\psi_\alpha] \uparrow^{n_\alpha} \downarrow^{m_\alpha} [\tau_\alpha]$ and $\beta^{\uparrow\!\downarrow} = [\psi_\beta] \uparrow^{n_\beta} \downarrow^{m_\beta} [\tau_\beta]$, where $n_\alpha + m_\alpha > 0$, $n_\beta + m_\beta > 0$, and ψ_α, ψ_β, τ_α, τ_β are in up-down normal form (the remaining cases where $\alpha^{\uparrow\!\downarrow} = \varepsilon$ or $\beta^{\uparrow\!\downarrow} = \varepsilon$ being simpler). Suppose, without loss of generality, that $n_\alpha \leq n_\beta$.

Hence, we have $\langle \alpha^{\uparrow\!\downarrow} \odot \beta^{\uparrow\!\downarrow} \rangle \equiv^{\downarrow\uparrow} \langle \gamma^{\uparrow\!\downarrow} \rangle$, where

$$\gamma^{\uparrow\!\downarrow} = [\psi_\alpha \wedge \psi_\beta] \uparrow^{n_\alpha} \downarrow^{m_\alpha} [\tau_\alpha \wedge \langle \varepsilon \odot \uparrow^{m_\alpha} \uparrow^{n_\beta - n_\alpha} \downarrow^{m_\beta} [\tau_\beta] \rangle].$$

It is clear that the formulas are equivalent (*cf.* the picture below).



Moreover, the right-hand formula has at most one more nesting than the left-hand formula, and its vertical depth is at most (r, s). This concludes the proof of Lemma 14. ■

By induction on φ, and using Lemmas 13 and 14, one can show that there is $\varphi^{\uparrow\downarrow}$ as desired. □

3.2.2 Finite Index

Contrary to the case of XPath$_=(\downarrow)$ (*cf.*, Proposition 2), the logical equivalence relation restricted to XPath$_=(\uparrow\downarrow)$-formulas of bounded vertical depth has infinitely many equivalence classes.

Proposition 15. *If $r + s \geq 2$ then $\equiv^{\downarrow\uparrow}_{r,s}$ has infinite index.*

Proof. We show that for every r, s so that $r + s = 2$ there is an infinite set $\{\psi^i_{r,s}\}_{i\geq 0}$ of non-equivalent node expressions with vertical depth (r, s). It thus follows that for every r, s so that $r + s \geq 2$, $\equiv^{\downarrow\uparrow}_{r,s}$ has infinite index.

Consider the following formulas.

$$\begin{aligned}
\psi^0_{1,1} &= \langle \varepsilon = \uparrow\downarrow\downarrow \rangle & \psi^{i+1}_{1,1} &= \langle \varepsilon = \uparrow\downarrow[\psi^i_{1,1}]\downarrow \rangle \\
\psi^0_{0,2} &= \langle \uparrow = \uparrow\uparrow\downarrow\downarrow \rangle & \psi^{i+1}_{0,2} &= \langle \uparrow = \uparrow\uparrow\downarrow[\psi^i_{1,1}]\downarrow \rangle \\
\psi^0_{2,0} &= \langle \downarrow = \downarrow\downarrow \rangle & \psi^{i+1}_{2,0} &= \langle \downarrow = \downarrow[\psi^i_{1,1}]\downarrow \rangle
\end{aligned}$$

Note that $\mathrm{vd}(\psi^n_{r,s}) = (r, s)$ and $\mathrm{nd}(\psi^n_{r,s}) = n$ for every n. The formula $\psi^n_{r,s}$ intuitively says that there is a chain of length n as depicted in Figure 2.

In the data tree $\mathcal{T}_n$ of the figure, we have that $\mathcal{T}_n, x_{r,s} \models \psi^n_{r,s}$ but $\mathcal{T}_n, x_{r,s} \not\models \psi^{n'}_{r,s}$ for any $n' > n$. Therefore, $\{\psi^i_{r,s}\}_{i\geq 0}$ is an infinite set of non-equivalent formulas of vertical depth (r, s). □

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.

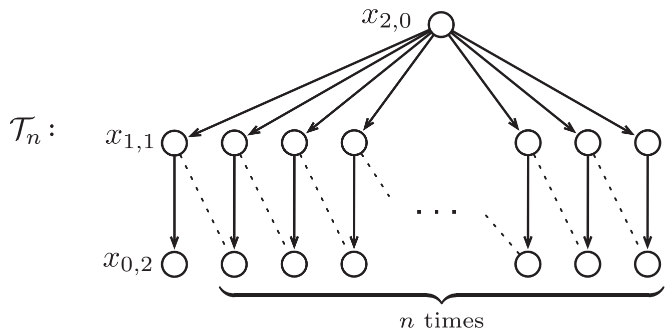


Figure 2: Model verifying ψ_i^j for all $i \geq n$ and not verifying φ_l for no $l < n$. Dotted lines represent equal data values.

Proposition 16. $\equiv_{r,s,k}^{\downarrow\uparrow}$ *has finite index.*

Proof. Same argument as in the proof of Proposition 2. $\square$

Corollary 17. $\{\mathcal{T}', u' \mid \mathcal{T}, u \equiv_{r,s,k}^{\downarrow\uparrow} \mathcal{T}', u'\}$ *is definable by a node expression of* (r, s, k)*-XPath*$_=(\uparrow\downarrow)$.

Proof. Similar to the proof of Corollary 3. $\square$

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

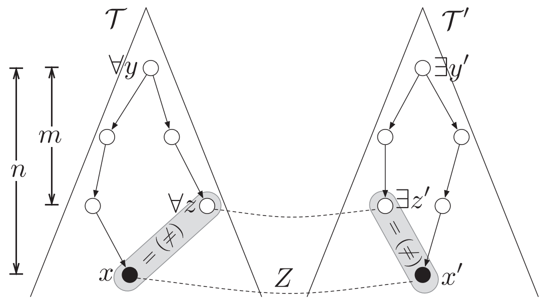
The advantage of the normal form presented in Section 3.2.1 is that it makes it possible to use a very simple notion of bisimulation. The disadvantage is that, since it does not preserve nesting depth, $\underline{\leftrightarrow}_{r,s,k}^{\downarrow\uparrow}$ does not correspond precisely to $\equiv_{r,s,k}^{\downarrow\uparrow}$,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

$$\underline{\leftrightarrow}_{r,s,k} \subseteq \equiv_{r,s,k}^{\downarrow\uparrow} \subseteq \underline{\leftrightarrow}_{r,s,k\cdot(r+s+2)}^{\downarrow\uparrow}.$$

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

  The following picture illustrates the intended requirements



- **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 above definitions heavily rely on the normal form of Proposition 12. In fact, the normal form is not strictly necessary for giving a notion of bisimulation for the vertical fragment. For the case of the downward fragment, no normal form was used, as every path expression is essentially a repetition of node test and 'child' relation. On the contrary, for the case of the vertical fragment, the use of a normal form would be very beneficial. A notion of bisimulation not taking into account the existence of normal form would have a rule **Zig** of the form: If If xZx' and $n_1, \dots, n_k$, $m_1, \dots, m_k$, $\tilde{n}_1, \dots, \tilde{n}_{\tilde{k}}$ and $\tilde{m}_1, \dots, \tilde{m}_{\tilde{k}}$ are such that

- $v^i_1 \rightarrow \dots \rightarrow v^i_{n_i}$ in $\mathcal{T}$ and $w^i_1 \rightarrow \dots \rightarrow w^i_{m_i}$ in $\mathcal{T}$ for $i \in \{1, \dots, k\}$,
- $\tilde{v}^i_1 \rightarrow \dots \rightarrow \tilde{v}^i_{\tilde{n}_i}$ in $\mathcal{T}$ and $\tilde{w}^i_1 \rightarrow \dots \rightarrow \tilde{w}^i_{\tilde{m}_i}$ in $\mathcal{T}$ for $i \in \{1, \dots, \tilde{k}\}$,
- $v^i_1 = w^i_1$ and $\tilde{v}^i_1 = \tilde{w}^i_1$
- $w^i_{m_i} = v^{i+1}_{n_{i+1}}$ and $\tilde{w}^i_{\tilde{m}_i} = \tilde{v}^{i+1}_{\tilde{n}_{i+1}}$
- $x = v^1_{n_1}$ and $x = \tilde{v}^1_{\tilde{n}_1}$

then there are $v'^i_1, \dots, v'^i_{n_i}$ in $\mathcal{T}'$, $w'^i_1, \dots, w'^i_{m_i}$ for $i \in \{1, \dots, k\}$, and $\tilde{v}'^i_1, \dots, \tilde{v}'^i_{\tilde{n}_i}$, $\tilde{w}'^i_1, \dots, \tilde{w}'^i_{\tilde{m}_i}$ in $\mathcal{T}'$ for $i \in \{1, \dots, \tilde{k}\}$ such that

- $v'^i_1 \rightarrow \dots \rightarrow v'^i_{n_i}$ in $\mathcal{T}'$ and $w'^i_1 \rightarrow \dots \rightarrow w'^i_{m_i}$ in $\mathcal{T}'$,
- $\tilde{v}'^i_1 \rightarrow \dots \rightarrow \tilde{v}'^i_{\tilde{n}_i}$ in $\mathcal{T}'$ and $\tilde{w}'^i_1 \rightarrow \dots \rightarrow \tilde{w}'^i_{\tilde{m}_i}$ in $\mathcal{T}'$,
- $v'^i_1 = w'^i_1$ and $\tilde{v}'^i_1 = \tilde{w}'^i_1$
- $w'^i_{m_i} = v'^{i+1}_{n_{i+1}}$ and $\tilde{w}'^i_{\tilde{m}_i} = \tilde{v}'^{i+1}_{\tilde{n}_{i+1}}$
- $x = v'^1_{n_1}$ and $x = \tilde{v}'^1_{\tilde{n}_1}$
- $v^i_j Z v'^i_j$, $w^i_j Z w'^i_j$, $\tilde{v}^i_j Z \tilde{v}'^i_j$, $\tilde{w}^i_j Z \tilde{w}'^i_j$ and
- $data(w^k_{n_k}) = data(\tilde{w}^k_{n_k})$ iff $data(w'^{\tilde{k}}_{n_{\tilde{k}}}) = data(\tilde{w}'^{\tilde{k}}_{n_{\tilde{k}}})$

Intuitively, this definition establishes that for every paths p and $\tilde{p}$ going up and down, many times in $\mathcal{T}$, there has to be a similar paths p' and $\tilde{p}'$ in $\mathcal{T}$, going up and down many times, and respecting the 'shape' of p and $\tilde{p}$ respectively such that the j-th node of p is connected to the j-the node of p' via Z, and the same for the nodes along $\tilde{p}$, and such that the data values of the last node of p and $\tilde{p}$ in $\mathcal{T}$ are equal if and only if the data values of the last nodes of p' and $\tilde{p}'$ are so.

This definition is the 'natural' extension of the downward bisimulation, but instead of considering downward paths, it considers more general ones. As it happens with the downward bisimulation, every intermediate node of $\mathcal{T}$ in both paths have to be related with the corresponding node in $\mathcal{T}'$. As one can immediately see, this definition is quite long and checks too many conditions. Working with normal forms allow us to get a much simpler definition of bisimulation, which has two main advantages: *a)* there is only one path in each data tree, and *b)* we do not require intermediate nodes to be related by Z. These two features are direct consequences of the up-down normal form, as the normal form *a)* only compares values against the root (so there is only one non-empty path expression in each 'diamond' node expression) and *b)* it only make tests at the beginning and at the end of each path expressions (but not in intermediate nodes). However, notice that due to the normal form both definitions denote *the same* bisimulation relation.

We say that $u \in T$ and $u' \in T'$ are (r, s, k)**-bisimilar for XPath**$_=(\uparrow\downarrow)$ (notation: $\mathcal{T}, u \leftrightarroweq^{\downarrow\uparrow}_{r,s,k} \mathcal{T}', u'$) 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 18. *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

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

Observation 19. $\mathcal{T}, u \leftrightarroweq^{\downarrow\uparrow}_{r,s,k} (\mathcal{T}|^s_r u), u$.

3.2.4 Equivalence and Bisimulation

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

Theorem 20.

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

The above theorem will be shown as a consequence of the following Propositions 21 and 22.

Proposition 21. *$\mathcal{T}, u \leftrightarrows^{\downarrow\uparrow}_{r,s,k\cdot(r+s+2)} \mathcal{T}', u'$ implies $\mathcal{T}, u \equiv^{\downarrow\uparrow}_{r,s,k} \mathcal{T}', u'$.*

Proof. We show that if $\mathcal{T}, u \leftrightarrows^{\downarrow\uparrow}_{r,s,k} \mathcal{T}', u'$ via

$$(Z^{\hat{k}}_{\hat{r},\hat{s}})_{\hat{r}+\hat{s}\leq r+s, \hat{k}\leq k}$$

then for all $n \leq \hat{s}$ and $m \leq \hat{r} + n$, for all φ in up-down normal form with $\mathrm{vd}(\varphi) \leq (\hat{r}, \hat{s})$, $\mathrm{nd}(\varphi) \leq \hat{k}$, for all upward expression $\alpha^{\uparrow}$ in up-down normal form, and for all downward expression $\alpha^{\downarrow}$ in up-down normal form with $\mathrm{vd}(\alpha^{\uparrow}), \mathrm{vd}(\alpha^{\downarrow}) \leq (\hat{r}, \hat{s})$, $\mathrm{nd}(\alpha^{\uparrow}), \mathrm{nd}(\alpha^{\downarrow}) \leq \hat{k}$:

1. If $x Z^{\hat{k}}_{\hat{r},\hat{s}} x'$ then $\mathcal{T}, x \models \varphi$ iff $\mathcal{T}', x' \models \varphi$.
2. If $y \xrightarrow{n} x$, $y' \xrightarrow{n} x'$, $x\, Z^{\hat{k}-1}_{\hat{r},\hat{s}}\, x'$, then $(x, y) \in [\![\alpha^{\uparrow}]\!]^{\mathcal{T}}$ iff $(x', y') \in [\![\alpha^{\uparrow}]\!]^{\mathcal{T}'}$.
3. If $y \xrightarrow{m} z$, $y' \xrightarrow{m} z'$, $z\, Z^{\hat{k}-1}_{\hat{r}',\hat{s}'}\, z'$ for $\hat{r}' = \hat{r} + n - m$, $\hat{s}' = \hat{s} - n + m$, then $(y, z) \in [\![\alpha^{\downarrow}]\!]^{\mathcal{T}}$ iff $(y', z') \in [\![\alpha^{\downarrow}]\!]^{\mathcal{T}'}$.

Hence, by Proposition 12, the main statement follows. We simultaneously show 1, 2 and 3 by induction on $|\varphi| + |\alpha^{\uparrow}| + |\alpha^{\downarrow}|$.

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 $\varphi = \langle \varepsilon = \alpha^{\uparrow}\alpha^{\downarrow} \rangle$. We show $\mathcal{T}, x \models \varphi \Rightarrow \mathcal{T}', x' \models \varphi$, so assume $\mathcal{T}, x \models \varphi$. Suppose there are $y, z \in \mathcal{T}$ and $n \leq \hat{s}$, $m \leq \hat{r} + n$ such that $y \xrightarrow{n} x$, $y \xrightarrow{m} z$, $(x, y) \in [\![\alpha^{\uparrow}]\!]^{\mathcal{T}}$, $(y, z) \in [\![\alpha^{\downarrow}]\!]^{\mathcal{T}}$ and *data*(x) = *data*(z). By Zig, there are $y', z' \in T'$ such that $z Z^{\hat{k}-1}_{\hat{r}',\hat{s}'} z'$ for $\hat{r}' = \hat{r} + n - m$, $\hat{s}' = \hat{s} - n + m$, and *data*$(x')$ = *data*(z'). By inductive hypothesis 2 and 3, $(x', y') \in [\![\alpha^{\uparrow}]\!]^{\mathcal{T}'}$ and $(y', z') \in [\![\alpha^{\downarrow}]\!]^{\mathcal{T}'}$. Hence $\mathcal{T}', x' \models \varphi$. The implication $\mathcal{T}', x' \models \varphi \Rightarrow \mathcal{T}, x \models \varphi$ is analogous. The cases $\varphi = \langle \varepsilon \neq \alpha^{\uparrow\!\downarrow} \rangle$, and $\varphi = \langle \varepsilon \odot \alpha^{\uparrow} \rangle$, $\varphi = \langle \varepsilon \odot \alpha^{\downarrow} \rangle$ ($\odot \in \{=, \neq\}$) and $\varphi = \langle \alpha \rangle$ (for α in up-down normal form) are shown in a similar way. The cases $\varphi = \langle \varepsilon \odot \varepsilon \rangle$ ($\odot \in \{=, \neq\}$) are trivial.

Let us now analyze item 2. Let $\alpha^{\uparrow} = [\psi]\uparrow^n$ $(n \geq 0)$, and let $x_0, \ldots, x_n \in T$ and $x'_0, \ldots, x'_n \in T'$ be such that

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

and $x Z^{\hat{k}-1}_{\hat{r},\hat{s}} x'$. By Observation 18, we have $x_0 Z^{\hat{k}-1}_{\hat{r}',\hat{s}'} x'_0$, for $\hat{r}' = \hat{r} + n$, $\hat{s}' = \hat{s} - n$. Assume by contradiction that $(x', y') \notin [\![\alpha^{\uparrow}]\!]^{\mathcal{T}'}$. This necessarily means that $\mathcal{T}, x_0 \models \psi$ but $\mathcal{T}', x'_0 \not\models \psi$. But ψ is a subformula of $\alpha^{\uparrow}$, $\mathrm{nd}(\psi) \leq \hat{k} - 1$ and $\mathrm{nd}(\psi) \leq (\hat{r}', \hat{s}')$ and this contradicts inductive hypothesis 1.

Item 3 is shown in a similar way. Let $\alpha^{\downarrow} = \downarrow^m[\psi]$ $(m \geq 0)$, and let $z_0, \ldots, z_m \in T$ and $z'_0, \ldots, z'_m \in T'$ be such that

$$\begin{aligned} y = z_0 \to z_1 \to \cdots \to z_m = z \qquad & \text{in } \mathcal{T}, \\ y' = z'_0 \to z'_1 \to \cdots \to z'_m = z' \qquad & \text{in } \mathcal{T}', \end{aligned}$$

and $z Z^{\hat{k}-1}_{\hat{r}',\hat{s}'} z'$. Assume by contradiction that $(y', z') \notin [\![\alpha^{\downarrow}]\!]^{\mathcal{T}'}$. This necessarily means that $\mathcal{T}, x_m \models \psi$ but $\mathcal{T}', x'_m \not\models \psi$. But ψ is a subformula of $\alpha^{\downarrow}$, $\mathrm{nd}(\psi) \leq \hat{k} - 1$ and $\mathrm{nd}(\psi) \leq (\hat{r}', \hat{s}')$ and this contradicts inductive hypothesis 1. $\square$

Proposition 22. *$\mathcal{T}, u \equiv^{\downarrow\uparrow}_{r,s,k} \mathcal{T}', u'$ implies $\mathcal{T}, u \leftrightarroweq^{\downarrow\uparrow}_{r,s,k} \mathcal{T}', u'$.*

Proof. Fix $u \in T$ and $u' \in T'$ such that $\mathcal{T}, u \equiv^{\downarrow\uparrow}_{r,s,k} \mathcal{T}', u'$. Define $(Z^{\hat{k}}_{\hat{r},\hat{s}})_{\hat{r}+\hat{s} \leq r+s, \hat{k} \leq k}$ by

$$x Z^{\hat{k}}_{\hat{r},\hat{s}} x' \quad \text{iff} \quad \mathcal{T}, x \equiv^{\downarrow\uparrow}_{\hat{r},\hat{s},\hat{k}} \mathcal{T}', x'.$$

We show that $Z^k_{r,s}$ is a (r, s, k)-bisimulation between $\mathcal{T}, u$ and $\mathcal{T}', u'$. By hypothesis, $u Z^k_{r,s} u'$. Now fix $\hat{r} + \hat{s} \leq r + s$, $\hat{k} \leq k$. By construction, $Z^{\hat{k}}_{\hat{r},\hat{s}}$ satisfies Harmony. Let us see that $Z^{\hat{k}}_{\hat{r},\hat{s}}$ satisfies Zig (the case for Zag is analogous). Suppose $x Z^{\hat{k}}_{\hat{r},\hat{s}} x'$,

$$\begin{aligned} y = x_0 \to x_1 \to \cdots \to v_n = x \qquad & \text{in } \mathcal{T}, \\ y = z_0 \to z_1 \to \cdots \to z_m = z \qquad & \text{in } \mathcal{T}, \end{aligned}$$

and $\mathit{data}(x) = \mathit{data}(z)$ (the case $\mathit{data}(x) \neq \mathit{data}(z)$ is shown in a similar way), where $m \leq \hat{r} + n$. Let $P \subseteq T'^2$ be defined by

$$P = \{(y', z') \mid y' \xrightarrow{n} x' \wedge y' \xrightarrow{m} z' \wedge \mathit{data}(x') = \mathit{data}(z')\}.$$

Since $\mathcal{T}, x \equiv^{\downarrow\uparrow}_{r,s,k} \mathcal{T}', x'$, $\mathrm{vd}(\langle \varepsilon = \uparrow^n \downarrow^m \rangle) \leq (r, s)$, $\mathrm{nd}(\langle \varepsilon = \uparrow^n \downarrow^m \rangle) = 0$, and $\mathcal{T}, x \models \langle \varepsilon = \uparrow^n \downarrow^m \rangle$, we conclude that $P \neq \emptyset$. We next show that there exists $(y', z') \in P$ such that

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

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

iii. $\mathcal{T}, x \equiv^{\downarrow\uparrow}_{\hat{r},\hat{s},\hat{k}-1} \mathcal{T}', x'$, and

iv. $\mathcal{T}, z \equiv^{\downarrow\uparrow}_{\hat{r}',\hat{s}',\hat{k}-1} \mathcal{T}', z'$, where $\hat{r}' = \hat{r} + n - m$, $\hat{s}' = \hat{s} - n + m$,

and hence, by inductive hypothesis, Zig is satisfied by $Z^{\hat{k}}_{\hat{r},\hat{s}}$. By way of contradiction, assume that for all $(y', z') \in P$ satisfying i and ii we have either

(a) $\mathcal{T}, x \not\equiv^{\downarrow\uparrow}_{\hat{r},\hat{s},\hat{k}-1} \mathcal{T}', x'$; or

(b) $\mathcal{T}, z \not\equiv^{\downarrow\uparrow}_{\hat{r}',\hat{s}',\hat{k}-1} \mathcal{T}', z'$ for $\hat{r}' = \hat{r} + n - m$, $\hat{s}' = \hat{s} - n + m$.

Fix $\top$ as any tautology such that $\mathrm{vd}(\top) = (0, 0)$, $\mathrm{nd}(\top) = 0$. For each $(y', z') \in P$ we define node expressions, $\varphi_{y',z'}$ and $\psi_{y',z'}$, satisfying that $\mathrm{vd}(\varphi_{y',z'}) \leq (\hat{r}, \hat{s})$, $\mathrm{nd}(\varphi_{y',z'}) < \hat{k}$ and $\mathrm{vd}(\psi_{y',z'}) \leq (\hat{r}', \hat{s}')$, $\mathrm{nd}(\psi_{y',z'}) < \hat{k}$ as follows:

- Suppose (a) holds. Let $\varphi_{y',z'}$ be such that $\mathrm{vd}(\varphi_{v',w'}) \leq (\hat{r}, \hat{s})$, $\mathrm{nd}(\varphi_{v',w'}) < \hat{k}$, and such that $\mathcal{T}, x \models \varphi_{y',z'}$ but $\mathcal{T}', x' \not\models \varphi_{y',z'}$; and let $\psi_{v',w'} = \top$.
- Suppose (a) does not hold. Then (b) holds. Let $\psi_{y',z'}$ be such that $\mathrm{vd}(\psi_{y',z'}) \leq (\hat{r}', \hat{s}')$, $\mathrm{nd}(\psi_{y',z'}) < \hat{k}$ and such that $\mathcal{T}, z \models \psi_{y',z'}$ but $\mathcal{T}', z' \not\models \psi_{y',z'}$; and let $\varphi_{y',z'} = \top$.

Let

$$\Phi = \bigwedge_{(y',z') \in P} \varphi_{y',z'} \quad \text{and} \quad \Psi = \bigwedge_{(y',z') \in P} \psi_{y',z'}. \tag{2}$$

Since $\mathrm{vd}(\varphi_{y',z'}) \leq (\hat{r}, \hat{s})$, $\mathrm{nd}(\varphi_{y',z'}) < \hat{k}$, by Proposition 16, there are finitely many non-equivalent formulas $\varphi_{y',z'}$. The same applies to formulas $\psi_{y',z'}$. Hence both infinite conjunctions in (2) are equivalent to finite ones, and therefore without loss of generality we may assume that Φ and Ψ are well-formed formulas.

Finally, let

$$\alpha^{\uparrow} = [\Phi]{\uparrow^n} \quad \text{and} \quad \alpha^{\downarrow} = {\downarrow^m}[\Psi].$$

By construction, $\mathrm{vd}(\alpha^{\uparrow}\alpha^{\downarrow}) \leq (\hat{r}, \hat{s})$, $\mathrm{nd}(\alpha^{\uparrow}\alpha^{\downarrow}) \leq \hat{k}$. Furthermore, $\mathcal{T}, x \models \langle \varepsilon = \alpha^{\uparrow}\alpha^{\downarrow} \rangle$ and $\mathcal{T}', x' \not\models \langle \varepsilon = \alpha^{\uparrow}\alpha^{\downarrow} \rangle$, but this contradicts the fact that $\mathcal{T}, x \equiv^{\downarrow\uparrow}_{\hat{r},\hat{s},\hat{k}} \mathcal{T}', x'$. □

Proof of Theorem 20. Items 2 and 3 are shown in Propositions 21 and 22.

The left-to-right argument for item 1 can be seen as a consequence of item 2. Indeed, $\mathcal{T}, u \leftrightarrows^{\downarrow\uparrow} \mathcal{T}', u'$ implies $\mathcal{T}, u \leftrightarrows^{\downarrow\uparrow}_{(r,s)} \mathcal{T}', u'$ for all r, s, which by item 2 implies $\mathcal{T}, u \equiv^{\downarrow\uparrow}_{(r,s)} \mathcal{T}', u'$ for all r, s, which in turn entalis $\mathcal{T}, u \equiv^{\downarrow\uparrow} \mathcal{T}', u'$.

The right-to-left argument for item 1 is similar to that of Proposition 22, but working with a single Z instead of $(Z^{\hat{k}}_{\hat{r},\hat{s}})_{\hat{r},\hat{s},\hat{k}}$. For the converse implication, define Z by xZx' iff $\mathcal{T}, x \equiv^{\downarrow\uparrow} \mathcal{T}', x'$. The conjunctions in (2) are then finite because $\mathcal{T}'$ is finitely branching, and so P is finite (the fact that $\mathcal{T}$ is finitely branching is used for showing that Zag is satisfied). □

Corollary 23. $\leftrightarrows^{\downarrow\uparrow}_{r,s,k}$ *has finite index.*

Proof. Immediate from Theorem 20 and Proposition 16. □

3.3 Simulation

In this section we define notions of directed (non-symmetric) simulations for XPath$_=(\downarrow)$ and XPath$_=(\uparrow\downarrow)$, as it is done, e.g., in the works of Kurtonina and de Rijke (1997) or Lutz, Piro, and Wolter (2011) for some modal logics. We obtain results similar to Theorems 8 and 20 but relating each simulation notion with the corresponding logical implication.

We say that an 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 XPath$_=(\downarrow)$, XPath$_=(\uparrow\downarrow)$, XPath$_=(\downarrow\downarrow_*)$, or XPath$_=(\downarrow\uparrow\downarrow_*\uparrow^*)$, we write $\mathcal{L}^+$ for the positive fragment of $\mathcal{L}$.

A **simulation for XPath$_=(\downarrow)$** [resp. for XPath$_=(\uparrow\downarrow)$] 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 XPath$_=(\downarrow)$** (notation: $\mathcal{T}, u \rightharpoonup^{\downarrow} \mathcal{T}', u'$) 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. $data(v) = data(w) \Rightarrow data(v') = data(w')$,
  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 $0 \leq i < m$.

$u \in T$ is **similar to** $u' \in T'$ **for XPath$_=(\uparrow\downarrow)$** (notation: $\mathcal{T}, u \rightharpoonup^{\downarrow\uparrow} \mathcal{T}', u'$) 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 $y' \xrightarrow{n} x'$, $y' \xrightarrow{m} z'$, zZz', and if $data(z) = data(x)$ then $data(z') = data(x')$.

Relations $\rightharpoonup^{\downarrow}_{\ell}$ and $\rightharpoonup^{\downarrow\uparrow}_{r,s,k}$ 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

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

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

Theorem 24.

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

4. $\mathcal{T}, u \Rrightarrow^{\downarrow\uparrow}_{r,s,k} \mathcal{T}', u'$ *implies* $\mathcal{T}, u \underrightarrow{\to}^{\downarrow\uparrow}_{r,s,k} \mathcal{T}', u'$.

Proof. The proofs are straightforward adaptations of the proofs of Propositions 9 and 11 and Propositions 21 and 22 respectively, and are ommitted here. In particular, for the 'if' part, in the adaptation of the proofs of Propositions 11 and 22, the simulations are defined by

$$\begin{array}{rcl} xZ_ix' & \text{iff} & \mathcal{T}, x \Rrightarrow^{\downarrow}_{i} \mathcal{T}', x \\ xZ^{\hat{k}}_{\hat{r},\hat{s}}x' & \text{iff} & \mathcal{T}, x \Rrightarrow^{\downarrow\uparrow}_{\hat{r},\hat{s},\hat{k}} \mathcal{T}', x \end{array}$$

respectively, and the conditions (a) and (b) on page 283 become now

(a) $[\exists i \in \{0, \dots, n\}\ \exists \varphi \in \text{XPath}_=(\downarrow)^+]\ \text{dd}(\varphi) \leq h - i \wedge \mathcal{T}, v_i \models \varphi \wedge \mathcal{T}', v'_i \not\models \varphi$; or

(b) $[\exists j \in \{0, \dots, m\}\ \exists \varphi \in \text{XPath}_=(\downarrow)^+]\ \text{dd}(\varphi) \leq h - j \wedge \mathcal{T}, w_j \models \varphi \wedge \mathcal{T}', w'_j \not\models \varphi$,

and

(a) $[\exists i \in \{0, \dots n\}\ \exists \varphi \in \text{XPath}_=(\downarrow\uparrow)^+]\ \text{vd}(\varphi) \leq (\hat{r} + i, \hat{s} - i) \wedge \text{nd}(\varphi) \leq k - 1 \wedge \mathcal{T}, v_i \models \varphi \wedge \mathcal{T}', v'_i \not\models \varphi$; or

(b) $[\exists j \in \{0, \dots m\}\ \exists \varphi \in \text{XPath}_=(\downarrow\uparrow)^+]\ \text{vd}(\varphi) \leq (\hat{r} + j', \hat{s} - j')$ for $j' = n - m + j$ $\wedge \text{nd}(\varphi) \leq k - 1 \wedge \mathcal{T}, w_j \models \varphi \wedge \mathcal{T}', w'_j \not\models \varphi$

respectively. □

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) $\mathit{label}(u)$ on $\mathcal{T}'$ equals $\mathit{label}(u)$ on $\mathcal{T}$; and 3) $\mathit{data}(u)$ on $\mathcal{T}'$ equals $\mathit{data}(u)$ 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 $\text{XPath}_=(\uparrow\downarrow)$ from $\mathcal{T}'$ to $\mathcal{T}$.

Lemma 25. *If $\mathcal{T}'$ is a substructure of $\mathcal{T}$ and $u' \in T'$ then $\mathcal{T}', u' \underrightarrow{\to}^{\downarrow\uparrow} \mathcal{T}, u'$.*

Lemma 26.

(1) $\{\mathcal{T}', u' \mid \mathcal{T}, u \underrightarrow{\to}^{\downarrow}_{\ell} \mathcal{T}', u'\}$ is definable by a node expression $\chi^+_{\ell,u,\mathcal{T}}$ of $XPath_=(\downarrow)^+$ with downward depth $\leq \ell$.

(2) $\{\mathcal{T}', u' \mid \mathcal{T}, u \underrightarrow{\to}^{\downarrow\uparrow}_{r,s,k} \mathcal{T}', u'\}$ is definable by a node expression $\chi^+_{r,s,k,u,\mathcal{T}}$ of $XPath_=(\downarrow\uparrow)^+$ with vertical depth $\leq (r, s)$ and nesting depth $\leq k$.

Proof. For item (2), let $\mathit{sim}^{\downarrow\uparrow}_{r,s,k}(\mathcal{T}, u) = \{\mathcal{T}', u' \mid \mathcal{T}, u \underrightarrow{\to}^{\downarrow\uparrow}_{r,s,k} \mathcal{T}', u'\}$. Let $\Phi_{\mathcal{T}',u'}$ be the set of all positive node expressions $\varphi \in \text{XPath}_=(\downarrow\uparrow)^+$ of vertical depth at most (r, s) and nesting depth at most k so that $\mathcal{T}', u' \models \varphi$. Let Ψ be

$$\Psi = \bigvee_{\mathcal{T}',u' \in \mathit{sim}^{\downarrow\uparrow}_{r,s,k}(\mathcal{T},u)} \bigwedge \Phi_{\mathcal{T}',u'}.$$

Since every $\Phi_{\mathcal{T}',u'}$ is finite up to logical equivalence by Proposition 16, it follows that Ψ is a valid node expression. We show that it defines $sim^{\downarrow\uparrow}_{r,s,k}(\mathcal{T},u)$.

Let $\mathcal{T}',u' \in sim^{\downarrow\uparrow}_{r,s,k}(\mathcal{T},u)$. Then, $\mathcal{T}',u' \models \bigwedge \Phi_{\mathcal{T}',u'}$ and thus $\mathcal{T}',u' \models \Psi$. If on the other hand $\mathcal{T}',u' \models \Psi$ we have that $\mathcal{T}',u' \models \bigwedge \Phi_{\mathcal{T}'',u''}$ for some $\mathcal{T}'',u'' \in sim^{\downarrow\uparrow}_{r,s,k}(\mathcal{T},u)$ and then $\mathcal{T}',u' \equiv^{\downarrow\uparrow}_{r,s,k} \mathcal{T}'',u''$. By Theorem 20-3 we then have that $\mathcal{T}',u' \leftrightarrows^{\downarrow\uparrow}_{r,s,k} \mathcal{T}'',u''$, and in particular $\mathcal{T}'',u'' \rightrightarrows^{\downarrow\uparrow}_{r,s,k} \mathcal{T}',u'$. Since $\mathcal{T},u \rightrightarrows^{\downarrow\uparrow}_{r,s,k} \mathcal{T}'',u''$ and $\mathcal{T}'',u'' \rightrightarrows^{\downarrow\uparrow}_{r,s,k} \mathcal{T}',u'$, then $\mathcal{T},u \rightrightarrows^{\downarrow\uparrow}_{r,s,k} \mathcal{T}',u'$ (by transitivity of $\rightrightarrows^{\downarrow\uparrow}_{r,s,k}$) and thus $\mathcal{T}',u' \in sim^{\downarrow\uparrow}_{r,s,k}(\mathcal{T},u)$.

Item (1) is shown in a similar way, using Proposition 2 and Theorem 8-2. $\square$

We obtain that the node expressions of XPath$_=$ invariant under simulations are, precisely, the positive ones.

Theorem 27.

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

Proof. We start with item (1), for the case of $\rightrightarrows^{\downarrow}_{\ell}$. The 'if' part is straightforward from Theorem 24-2, and here we focus on the 'only if' part. Let φ be preserved under $\rightrightarrows^{\downarrow}_{\ell}$. Let $\{(\mathcal{T}_i,u_i)\}_{i\leq n}$ be the set of all pointed models of φ modulo $\leftrightarrows^{\downarrow}_{\ell}$ (which is finite due to Theorem 8-2 together with Proposition 2). We claim that

$$\mathcal{T},u \models \varphi \text{ iff } \mathcal{T}_i,u_i \rightrightarrows^{\downarrow}_{\ell} \mathcal{T},u \text{ for some } i \leq n. \tag{3}$$

On the one hand, if $\mathcal{T},u \models \varphi$ then there is $i \leq n$ such that $\mathcal{T}_i,u_i \leftrightarrows^{\downarrow}_{\ell} \mathcal{T},u$, and so $\mathcal{T}_i,u_i \rightrightarrows^{\downarrow}_{\ell} \mathcal{T},u$. On the other hand, suppose $\mathcal{T}_i,u_i \rightrightarrows^{\downarrow}_{\ell} \mathcal{T},u$ for some i. Since φ is preserved under $\rightrightarrows^{\downarrow}_{\ell}$ and $\mathcal{T}_i,u_i \models \varphi$, we conclude $\mathcal{T},u \models \varphi$.

Let $\chi_{\ell,u_i,\mathcal{T}_i} \in$ XPath$_=(\downarrow)^+$, $\mathrm{dd}(\psi_i) \leq \ell$, be as in Lemma 26-(1). Using (3) one shows that $\bigvee_{i\leq n} \chi_{\ell,u_i,\mathcal{T}_i} \equiv \varphi$.

For the case of $\rightrightarrows^{\downarrow}$ of item (1), the 'if' direction follows from Theorem 24-1. For the 'only if' direction, let φ be preserved under $\rightrightarrows^{\downarrow}$. It is easy to see that φ is preserved under $\rightrightarrows^{\downarrow}$ iff it is preserved under $\rightrightarrows^{\downarrow}_{\mathrm{dd}(\varphi)}$. We can then apply the same reasoning as before and the statement follows.

Item (3) follows the same argument as item (1) but this time using Corollary 23 and Lemma 26-(2).

Item (4) is straightforward from Theorem 24-3.

Item (2) follows from items (3) and (4) and the observation that φ is preserved under $\rightrightarrows^{\downarrow\uparrow}$ iff it is preserved under $\rightrightarrows^{\downarrow\uparrow}_{r,s,k\cdot(r+s+2)}$ for $\mathrm{vd}(\varphi) = (r,s)$ and $\mathrm{nd}(\varphi) = k$. $\square$

3.4 Transitive Axes

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 also 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 $\uparrow$. Hence the notions of bisimulations for $\mathrm{XPath}_=(\downarrow\downarrow_*)$ and $\mathrm{XPath}_=(\downarrow\uparrow\downarrow_*\uparrow^*)$ (denoted $\underline{\leftrightarrow}^{\downarrow\downarrow_*}$ and $\underline{\leftrightarrow}^{\downarrow\uparrow\downarrow_*\uparrow^*}$ respectively) coincide with $\underline{\leftrightarrow}^{\downarrow}$ and $\underline{\leftrightarrow}^{\downarrow\uparrow}$, respectively.

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

Theorem 28. *Let $\dagger \in \{\downarrow\downarrow_*, \downarrow\uparrow\downarrow_*\uparrow^*\}$.*

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

Proof. The proof that $\mathcal{T}, u \underline{\leftrightarrow}^{\downarrow} \mathcal{T}', u' \Rightarrow \mathcal{T}, u \equiv^{\downarrow\downarrow_*} \mathcal{T}', u'$ follows from a simple adaptation of Proposition 9 to the logic $\mathrm{XPath}_=(\downarrow\downarrow_*)$ and Lemma 10. The fact that for finitely branching, $\mathcal{T}, u \equiv^{\downarrow\downarrow_*} \mathcal{T}', u' \Rightarrow \mathcal{T}, u \underline{\leftrightarrow}^{\downarrow} \mathcal{T}', u'$ is straightforward from Theorem 8-1 since $\equiv^{\downarrow\downarrow_*} \subseteq \equiv^{\downarrow}$. The cases for $\mathrm{XPath}_=(\downarrow\uparrow\downarrow_*\uparrow^*)$, $\mathrm{XPath}_=(\downarrow\downarrow_*)$ and $\mathrm{XPath}_=(\downarrow\uparrow\downarrow_*\uparrow^*)^+$ are analogous. ☐

It is not hard to see that the adequate notion of (bi)simulation of any other 'intermediate' fragment between $\mathrm{XPath}_=(\uparrow\downarrow)$ and $\mathrm{XPath}_=(\downarrow\uparrow\downarrow_*\uparrow^*)$ —such as $\mathrm{XPath}_=(\downarrow\uparrow\downarrow_*)$ and $\mathrm{XPath}_=(\downarrow\uparrow\uparrow^*)$— also corresponds to that of $\mathrm{XPath}_=(\uparrow\downarrow)$ in the sense of the statement above.

On the other hand, if we restrict formulas to have only transitive axes, we obtain that the notion of bisimulation for $\mathrm{XPath}_=(\downarrow_*)$ has a coarser bisimulation notion, as we define next.

Let $\mathcal{T}$ and $\mathcal{T}'$ be two data-trees. We say that $u \in T$ and $u' \in T'$ are **bisimilar for $\mathbf{XPath}_=(\downarrow_*)$** (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 $label(x) = label(x')$.
- **Zig:** If xZx' and there are $x \overset{*}{\to} v_1 \overset{*}{\to} \cdots \overset{*}{\to} v_n$, and $x \overset{*}{\to} w_1 \overset{*}{\to} \cdots \overset{*}{\to} w_m$ in $\mathcal{T}$ then there are $x' \overset{*}{\to} v'_1 \overset{*}{\to} \cdots \overset{*}{\to} v'_n$, and $x' \overset{*}{\to} w'_1 \overset{*}{\to} \cdots \overset{*}{\to} w'_m$ in $\mathcal{T}'$ such that
  1. $data(v_n) = data(w_m) \Leftrightarrow data(v'_n) = data(w'_m)$,
  2. $v_i \, Z \, v'_i$ for all $1 \le i \le n$, and
  3. $w_i \, Z \, w'_i$ for all $1 \le i \le m$.
- **Zag:** If xZx' and there are $x' \overset{*}{\to} v'_1 \overset{*}{\to} \cdots \overset{*}{\to} v'_n$, and $x' \overset{*}{\to} w'_1 \overset{*}{\to} \cdots \overset{*}{\to} w'_m$ in $\mathcal{T}'$ then there are $x \overset{*}{\to} v_1 \overset{*}{\to} \cdots \overset{*}{\to} v_n$, and $x \overset{*}{\to} w_1 \overset{*}{\to} \cdots \overset{*}{\to} w_m$ in $\mathcal{T}$ such that items 1, 2 and 3 above are verified.

As before, one can define ℓ-bisimilarity for XPath$_=(\downarrow_*)$, notated $\mathcal{T}, u \underline{\leftrightarrow}_\ell^{\downarrow_*} \mathcal{T}', u'$, and also the notions of equivalence $\equiv^{\downarrow_*}$ and $\equiv_\ell^{\downarrow_*}$ as expected.

These notions of bisimulation coincide with the corresponding logical equivalences:

Theorem 29.

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 finite.*

2. *$\mathcal{T}, u \underline{\leftrightarrow}_\ell^{\downarrow_*} \mathcal{T}', u'$ iff $\mathcal{T}, u \equiv_\ell^{\downarrow_*} \mathcal{T}', u'$.*

For the case of XPath$_=(\downarrow_*\uparrow^*)$ it is not obvious how to adapt the bisimulation of vertical XPath, since the normal form results does not hold for XPath$_=(\downarrow_*\uparrow^*)$.

4. Characterization

In Section 4.1 we characterize XPath$_=(\downarrow)$ as the fragment of first-order logic $\underline{\leftrightarrow}^{\downarrow}$-invariant over data trees. In Section 4.2 we show that this result fails for XPath$_=(\uparrow\downarrow)$ in general, but a weaker result holds: Any first-order formula $\underline{\leftrightarrow}_{r,s,k}^{\downarrow\uparrow}$-invariant for some r, s, k is equivalent to a XPath$_=(\uparrow\downarrow)$ formula.

4.1 Downward XPath

Recall that a data tree $\mathcal{T}$ and $u \in T$, let $\mathcal{T}|_\ell u$ denote the subtree of $\mathcal{T}$ induced by $\{v \in T \mid (\exists n \leq \ell)\ u \overset{n}{\to} v\}$. Any data tree can be regarded as a σ-structure, as explained in §2.1. An FO(σ)-formula $\varphi(x)$ is **ℓ-local** if for all data trees $\mathcal{T}$ and $u \in T$, we have $\mathcal{T} \models \varphi(u) \Leftrightarrow \mathcal{T}|_\ell u \models \varphi(u)$. Recall that $\varphi \in \mathrm{FO}(\sigma)$, $\mathrm{qr}(\varphi)$ is the quantifier rank of φ.

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

Theorem 30 (Characterization)**.** *Let $\varphi(x) \in \mathrm{FO}(\sigma)$. The following are equivalent:*

1. *φ is $\underline{\leftrightarrow}^{\downarrow}$-invariant over [finite] data-trees;*

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

The proof of this theorem, whose proof is afterwards, will be a consequence of the following three propositions:

Proposition 31. *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 the proof by Otto (2004a). 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) $\mathcal{T}', u' \underline{\leftrightarrow}^{\downarrow} \mathcal{T}, u$,

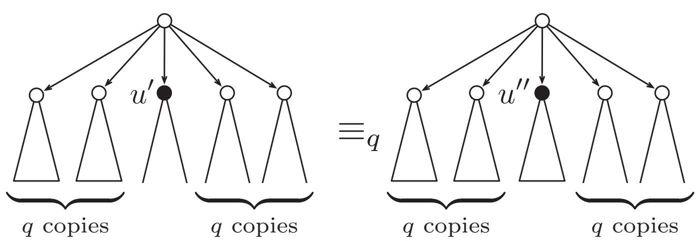
(b) $\mathcal{T}'', u'' \underline{\leftrightarrow}^{\downarrow} (\mathcal{T}|_\ell u), u$, and

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

Indeed, from the above conditions it follows that

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

and hence φ is ℓ-local. By Observation 4 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 $\mathcal{T}|_\ell u$. Consider the structures $\mathcal{T}'$ and $\mathcal{T}''$ in the diagram below,



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. By Observation 5, 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 as the one used in the paper by Otto (2004a). □

Proposition 32. *Any $\underline{\leftrightarrow}^{\downarrow}$-invariant $\varphi(x) \in \mathrm{FO}(\sigma)$ over [finite] data-trees that is ℓ-local, is $\underline{\leftrightarrow}^{\downarrow}_\ell$-invariant.*

Proof. Let $\varphi(x)$ be ℓ-local and $\underline{\leftrightarrow}^{\downarrow}$-invariant. Suppose $\mathcal{T}, u \underline{\leftrightarrow}^{\downarrow}_\ell \mathcal{T}', u'$ and $\mathcal{T} \models \varphi(u)$. By ℓ-locality, $\mathcal{T}|_\ell u \models \varphi(u)$. Now

$$\begin{aligned} \mathcal{T}, u \underline{\leftrightarrow}^{\downarrow}_\ell \mathcal{T}', u' &\text{ iff } (\mathcal{T}|_\ell u), u \underline{\leftrightarrow}^{\downarrow}_\ell (\mathcal{T}'|_\ell u'), u' && \text{(Obs. 7)} \\ &\text{ iff } (\mathcal{T}|_\ell u), u \underline{\leftrightarrow}^{\downarrow} (\mathcal{T}'|_\ell u'), u'. && \text{(Obs. 6)} \end{aligned}$$

By $\underline{\leftrightarrow}^{\downarrow}$-invariance, $\mathcal{T}'|_\ell u' \models \varphi(u')$ and by ℓ-locality again, $\mathcal{T}' \models \varphi(u')$. □

Proposition 33. *If $\varphi(x) \in \mathrm{FO}(\sigma)$ is $\underline{\leftrightarrow}^{\downarrow}_\ell$-invariant over [finite] data-trees, then there is $\psi \in \ell$-XPath$_=(\downarrow)$ such that $\mathrm{Tr}_x(\psi)$ is logically equivalent to φ over [finite] data-trees.*

Proof. By Corollary 3, for every data tree $\mathcal{T}$ and $u \in T$ there is a node expression $\chi_{\ell,\mathcal{T},u}$ of ℓ-XPath$_=(\downarrow)$ such that $\mathcal{T}, u \equiv^{\downarrow}_{\ell} \mathcal{T}', u'$ iff $\mathcal{T}', u' \models \chi_{\ell,\mathcal{T},u}$. Let

$$\psi = \bigvee_{\mathcal{T} \models \varphi(u)} \chi_{\ell,\mathcal{T},u}.$$

Since $\chi_{\ell,\mathcal{T},u} \in \ell$-XPath$_=(\downarrow)$ and, by Proposition 2, $\equiv^{\downarrow}_{\ell}$ has finite index, it follows that ψ is equivalent to a finite disjunction.

We now show that $\varphi \equiv \mathrm{Tr}_x(\psi)$. Let us see that $\varphi \models \mathrm{Tr}_x(\psi)$. Suppose $\mathcal{T} \models \varphi(u)$. Since $\mathcal{T}, u \models \chi_{\ell,\mathcal{T},u}$, we have $\mathcal{T}, u \models \psi$ and so $\mathcal{T} \models \mathrm{Tr}_x(\psi)(u)$. Let us now see that $\mathrm{Tr}_x(\psi) \models \varphi$. Assume $\mathcal{T} \models \mathrm{Tr}_x(\psi)(u)$, and so $\mathcal{T}, u \models \psi$. Then there exists $\mathcal{T}', u'$ such that $\mathcal{T}' \models \varphi(u')$ and $\mathcal{T}, u \models \chi_{\ell,\mathcal{T}',u'}$. By the property of $\chi_{\ell,\mathcal{T}',u'}$, we have $\mathcal{T}, u \equiv^{\downarrow}_{\ell} \mathcal{T}', u'$ and since φ is $\underline{\leftrightarrow}^{\downarrow}_{\ell}$-invariant (and hence $\equiv^{\downarrow}_{\ell}$-invariant by Theorem 8-2) we conclude $\mathcal{T} \models \varphi(u)$. $\square$

Proof of theorem 30. The implication $2 \Rightarrow 1$ follows straightforwardly from Theorem 8. The proof of $1 \Rightarrow 2$ is 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 31). 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 (see Proposition 32 below). Finally, we show that any FO(σ)-definable property which is $\underline{\leftrightarrow}^{\downarrow}_{\ell}$-invariant is definable in ℓ-XPath$_=(\downarrow)$ (see Proposition 33 below). $\square$

4.2 Vertical XPath

The analog of Theorem 30 fails for XPath$_=(\uparrow\downarrow)$. This basically because the property $p(x) =$"the tree where x belongs contains a label a" is $\underline{\leftrightarrow}^{\downarrow\uparrow}$-invariant while it is not expressible in XPath$_=(\uparrow\downarrow)$. Notice that, however, p is not $\underline{\leftrightarrow}^{\downarrow}$-invariant. This is as expected, since otherwise p would be expressed by a XPath$_=(\downarrow)$ formula. Indeed, consider two data trees with only two nodes, the root and a leaf, labeled respectively a, b in one tree, and b, b in the other tree. Note that the leafs of these trees are $\underline{\leftrightarrow}^{\downarrow}$-bisimilar although they don't coincide in the property p.

Lemma 34. *The* FO(σ)*-formula* $(\exists x)\ P_a(x)$ *is* $\underline{\leftrightarrow}^{\downarrow\uparrow}$*-invariant though not logically equivalent over [finite] data-trees to any node expression of XPath*$_=(\uparrow\downarrow)$.

Proof. Let $\varphi(x)$ be the FO(σ)-formula for *there is a node labeled a in the tree*, i.e., $\varphi(x) = (\exists y)\ P_a(y)$. We prove that φ is $\underline{\leftrightarrow}^{\downarrow\uparrow}$-invariant over [finite] data-trees, though it is not logically equivalent over [finite] data-trees to any node expression of XPath$_=(\uparrow\downarrow)$.

To see that φ is $\underline{\leftrightarrow}^{\downarrow\uparrow}$-invariant over [finite] data-trees, take $\mathcal{T}, u$ and $\mathcal{T}', u'$ such that $\mathcal{T}, u \,\underline{\leftrightarrow}^{\downarrow\uparrow}\, \mathcal{T}', u'$ and $\mathcal{T} \models \varphi(u)$. Furthermore, suppose that $\mathcal{T}, u \models \uparrow^m\downarrow^n a$ for adequate n and m. By Theorem 20, $\mathcal{T}', u' \models \uparrow^n\downarrow^m a$ and so $\mathcal{T}' \models \varphi(u')$.

Assume by contradiction that there is $\psi \in$ XPath$_=(\uparrow\downarrow)$ such that $\mathcal{T}, u \models \psi$ iff $\mathcal{T} \models \varphi(u)$ for all data-tree $\mathcal{T}$ and $u \in T$. Suppose $\mathrm{vd}(\psi) = (r, s)$ and $\mathrm{nd}(\psi) = k$. Let $\mathcal{T}$ be a data tree formed by a chain of length $r+1$ starting from the root u with all its nodes containing a label b except the leave, which has label a (the data values are irrelevant). By Observation 19 we have $\mathcal{T}, u \,\underline{\leftrightarrow}^{\downarrow\uparrow}_{r,s,k}\, (\mathcal{T}|^s_r u), u$. Since $\mathcal{T}, u \models \psi$, by Theorem 20, we have $(\mathcal{T}|^s_r u), u \models \psi$, and so $\mathcal{T}|^s_r u \models \varphi(u)$. This last fact is a contradiction because no node of $\mathcal{T}|^s_r u$ is labeled with a. $\square$

Hence $\text{XPath}_=(\uparrow\downarrow)$ is not the fragment of $\text{FO}(\sigma)$ which is $\underline{\leftrightarrow}^{\downarrow\uparrow}$-invariant over [finite] data-trees. However, the following analog of Proposition 33 (needed for the proof of Theorem 30) still holds for the case of $\text{XPath}_=(\uparrow\downarrow)$: For any r, s, k, every first-order formula $\underline{\leftrightarrow}^{\downarrow\uparrow}_{r,s,k}$-invariant is equivalent to a $\text{XPath}_=(\uparrow\downarrow)$ formula.

Proposition 35. *Let $k' = k \cdot (r + s + 2)$. If $\varphi(x) \in \text{FO}(\sigma)$ is $\underline{\leftrightarrow}^{\downarrow\uparrow}_{r,s,k'}$-invariant over [finite] data-trees, then there is $\psi \in (r, s, k)$-XPath$_=(\uparrow\downarrow)$ such that $\text{Tr}_x(\psi)$ is logically equivalent to φ over [finite] data-trees.*

Proof. By Corollary 17, for every data tree $\mathcal{T}$ and $u \in T$ there is a node expression $\chi_{r,s,k,\mathcal{T},u}$ of (r, s, k)-$\text{XPath}_=(\uparrow\downarrow)$ such that $\mathcal{T}, u \equiv^{\downarrow\uparrow}_{r,s,k} \mathcal{T}', u'$ iff $\mathcal{T}', u' \models \chi_{r,s,k,\mathcal{T},u}$. Let

$$\psi = \bigvee_{\mathcal{T} \models \varphi(u)} \chi_{r,s,k,\mathcal{T},u}.$$

As $\chi_{r,s,k,\mathcal{T},u} \in (r, s, k)$-$\text{XPath}_=(\uparrow\downarrow)$ and, by Proposition 16, $\equiv^{\downarrow\uparrow}_{r,s,k}$ has finite index, it follows that ψ is equivalent to a finite disjunction. The proof that $\varphi(x) \equiv \text{Tr}_x(\psi)$ is similar to Proposition 33, as we show next. Let us see that $\varphi \models \text{Tr}_x(\psi)$. Suppose $\mathcal{T} \models \varphi(u)$. Since $\mathcal{T}, u \models \chi_{r,s,k,\mathcal{T},a}$, we have $\mathcal{T}, u \models \psi$ and so $\mathcal{T} \models \text{Tr}_x(\psi)(u)$. Let us see that $\text{Tr}_x(\psi) \models \varphi$. Assume $\mathcal{T} \models \text{Tr}_x(\psi)(u)$, and so $\mathcal{T}, u \models \psi$. Then there exists $\mathcal{T}', u'$ such that $\mathcal{T}' \models \varphi(u')$ and $\mathcal{T}, u \models \chi_{r,s,k,\mathcal{T}',u'}$. By the property of $\chi_{r,s,k,\mathcal{T}',u'}$, we have $\mathcal{T}, u \equiv^{\downarrow\uparrow}_{r,s,k} \mathcal{T}', u'$ and since φ is $\underline{\leftrightarrow}^{\downarrow}_{r,s,k\cdot(r+s+2)}$-invariant (and hence $\equiv^{\downarrow}_{r,s,k}$-invariant by Theorem 20-2) we conclude $\mathcal{T} \models \varphi(u)$. $\square$

Notice that the counterexample in Lemma 34 is an unrestricted, existential formula. One may wonder if it might be possible to extend the expressive power of $\text{XPath}_=(\uparrow\downarrow)$ 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 30 fails. Formally, consider the logic $\text{XPath}_=(\updownarrow E)$, which results from adding the operator E to $\text{XPath}_=(\uparrow\downarrow)$ 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 30, showing that $\text{XPath}_=(\updownarrow E)$ is not the fragment of $\text{FO}(\sigma)$ $\underline{\leftrightarrow}^{\downarrow\uparrow}$-invariant over [finite] data-trees.

Lemma 36. *The $\text{FO}(\sigma)$-formula $(\exists y, z)\ [y \approx z \wedge P_a(y) \wedge P_b(z)]$ is $\underline{\leftrightarrow}^{\downarrow\uparrow}$-invariant though not logically equivalent over [finite] data-trees to any node expression of $\text{XPath}_=(\updownarrow E)$.*

Proof. Let $\varphi(x)$ be the $\text{FO}(\sigma)$-formula for *there are two nodes with same data value and labels a and b respectively*, i.e., $\varphi(x) = (\exists y, z)\ [y \approx z \wedge P_a(y) \wedge P_b(z)]$. We show that φ cannot be expressed in $\text{XPath}_=(\downarrow, \uparrow, E)$. Suppose, by means of contradiction, that there is a node expression $\psi \in \text{XPath}_=(\downarrow, \uparrow, E)$ expressing φ, with $\text{vd}(\psi) = (r, s)$ ($\text{vd}(\cdot)$ for $\text{XPath}_=(\downarrow, \uparrow, E)$ is defined as before together with the clause $\text{vd}(E\varphi) = \text{vd}(\varphi)$). Let $n = r + s$, and let $\mathcal{T}$ be the chain-like data-tree $u_0 \to u_1 \to \cdots \to u_n$ such that $label(u_0) = a$, $label(u_n) = b$, $label(u_i) = c$ for $i \in \{1, \dots n - 1\}$ and $data(u_i) = i$ for $i \in \{0, \dots, n\}$.

Let $\mathcal{T}'$ be the chain-like data-tree $u'_0 \to u'_1 \to \cdots \to u'_n$ such that $label(u'_i) = label(u_i)$ for $i \in \{0, \dots n\}$, $data(u'_i) = data(u_i)$ for $i \in \{0, \dots, n - 1\}$ and $data(u'_n) = 0$. Note that

$\mathcal{T} \not\models \varphi(u_0)$ and $\mathcal{T}' \models \varphi(u'_0)$. However, one can show that for all $i \in \{0, \ldots, n\}$ we have $\mathcal{T}, u_i \models \psi$ iff $\mathcal{T}', u'_i \models \psi$. Hence, ψ does not express φ and thus φ is not expressible in XPath$_=(\downarrow, \uparrow, E)$. $\square$

5. Applications

We devote this section to exemplify how the model theoretic tools we developed can be used to show expressiveness results for 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.

5.1 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. Observation 4, 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.

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 3 gives a schematic representation of this operation.

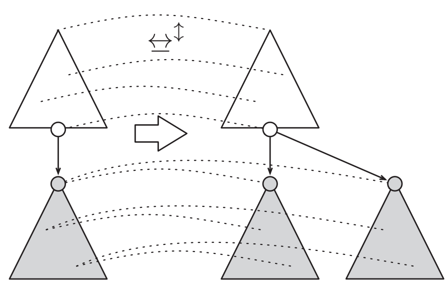


Figure 3: Closure under subtree replication.

Proposition 37. *XPath$_=(\downarrow\uparrow\downarrow_*\uparrow^*)$ is closed under subtree replication, i.e. if $\mathcal{T}'$ is a subtree replication of $\mathcal{T}$, and $u \in T$ then $\mathcal{T}', u \equiv^{\downarrow\uparrow\downarrow_*\uparrow^*} \mathcal{T}, u$.*

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 28, it suffices to verify that $\mathcal{T}, u \underline{\leftrightarrow}^{\downarrow\uparrow} \mathcal{T}', u$ 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 3. $\square$

5.2 Non-expressivity Results

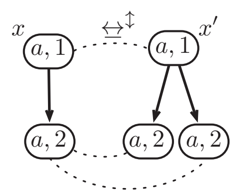
Finally, we will use bisimulation to show the expressivity limits of different fragments of XPath. Let *key*(a) be the property stating that every node with label a has a different data value. Let *fk*(a, b) (for *foreign key*) be the property $(\forall x)[P_a(x) \Rightarrow (\exists y)[P_b(y) \wedge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

Proposition 38.

1. *key*(a) *is not expressible in* $XPath_=(\downarrow\uparrow\downarrow_*\uparrow^*)$.

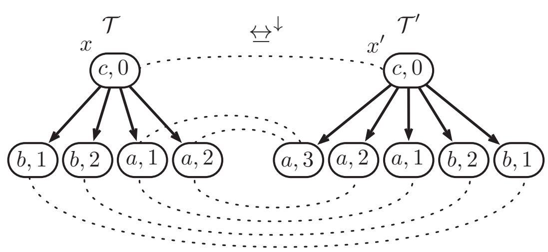
2. *fk*(a, b) *is expressible in* $XPath_=(\downarrow\uparrow\downarrow_*\uparrow^*)$ *but it is not expressible in* $XPath_=(\downarrow\downarrow_*)$ *or* $XPath_=(\downarrow\uparrow\downarrow_*\uparrow^*)^+$.

Proof. The first item follows from Proposition 37. Since the logic is closed under subtree replication, the trees below 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 $\text{XPath}_=(\downarrow\downarrow_*)$ because the models $\mathcal{T}$ and $\mathcal{T}'$ below are bisimilar for $\text{XPath}_=(\downarrow)$ via Z, depicted as dotted lines.



Since $\mathcal{T}, x$ satisfies *fk*(a, b) but $\mathcal{T}', x'$ does not, from Theorem 28 it follows that *fk*(a, b) is not expressible in $\text{XPath}_=(\downarrow\downarrow_*)$.

Finally, suppose there exists $\psi \in \text{XPath}_=(\downarrow\uparrow\downarrow_*\uparrow^*)^+$ expressing *fk*(a, b). Since $\mathcal{T}$ is a substructure of $\mathcal{T}'$ we have $\mathcal{T}, x \,\underline{\rightarrow}^{\downarrow\uparrow}\, \mathcal{T}', x$ by Lemma 25. By Theorem 28(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 $x \rightarrow y \rightarrow z$ and x, y, z have pairwise distinct data values.

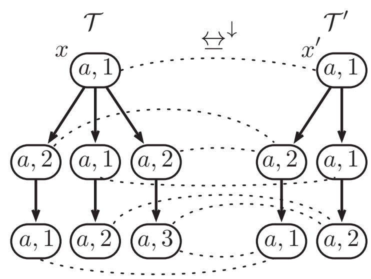
Proposition 39.

1. *$dist_3$ is expressible in XPath$_=(\uparrow\downarrow)$;*
2. *$dist_3$ is not expressible in XPath$_=(\downarrow\downarrow_*)$;*
3. *neither $dist_3$ nor its complement can be expressed in XPath$_=(\downarrow\uparrow\downarrow_*\uparrow^*)^+$.*

Proof. For 1, one can check that $\mathcal{T}, x \models \varphi$ iff $\mathcal{T}, x$ satisfies $dist_3$, for

$$\varphi = \langle \varepsilon \neq \downarrow\downarrow[\langle \varepsilon \neq \uparrow[\langle \varepsilon \neq \uparrow\rangle]\rangle]\rangle.$$

Let us see 2. Consider the data trees $\mathcal{T}, x$ and $\mathcal{T}', x'$ depicted below. 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

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

where $h(y)$ is the height of y, i.e., the distance from y to the root of the corresponding tree (Z is depicted as dotted lines in picture above). Since $\mathcal{T}, x$ satisfies $dist_3$ but $\mathcal{T}', x'$ does not, from Theorem 28 it follows that $dist_3$ is not expressible in XPath$_=(\downarrow\downarrow_*)$.

For 3, one can verify that $\mathcal{T}, x \underline{\rightarrow}^{\downarrow\uparrow} \mathcal{T}', x'$ via Z as defined above. If $dist_3$ were definable in XPath$_=(\downarrow\uparrow\downarrow_*\uparrow^*)^+$ via ψ and the fact that $\mathcal{T}, x \models \psi$, by Theorem 28(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 $x \rightarrow y \rightarrow z$, 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 38-2, $\overline{dist_3}$ is not expressible in XPath$_=(\downarrow\uparrow\downarrow_*\uparrow^*)^+$. $\square$

5.3 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 $\varphi \in$ 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 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 $u Z_{\ell,k} u'$ and for all $j \leq \ell$, $t \leq k$, $x \in T$ and $x' \in T'$ we have

$$
\begin{array}{ccccccccccc}
\equiv^{\downarrow}_{0,0} & = & \equiv^{\downarrow}_{0,1} & = & \equiv^{\downarrow}_{0,2} & = & \equiv^{\downarrow}_{0,3} & = & \equiv^{\downarrow}_{0,4} & \cdots & \\
\rotatebox{-90}{$\supsetneq$} & & \rotatebox{-90}{$\supsetneq$} & & \rotatebox{-90}{$\supsetneq$} & & \rotatebox{-90}{$\supsetneq$} & & \rotatebox{-90}{$\supsetneq$} & & \\
\equiv^{\downarrow}_{1,0} & \supsetneq & \equiv^{\downarrow}_{1,1} & = & \equiv^{\downarrow}_{1,2} & = & \equiv^{\downarrow}_{1,3} & = & \equiv^{\downarrow}_{1,4} & \cdots & \\
\rotatebox{-90}{$\supsetneq$} & & \rotatebox{-90}{$\supsetneq$} & & \rotatebox{-90}{$\supsetneq$} & & \rotatebox{-90}{$\supsetneq$} & & \rotatebox{-90}{$\supsetneq$} & & \\
\equiv^{\downarrow}_{2,0} & \supsetneq & \equiv^{\downarrow}_{2,1} & \supsetneq & \equiv^{\downarrow}_{2,2} & = & \equiv^{\downarrow}_{2,3} & = & \equiv^{\downarrow}_{2,4} & \cdots & \\
\rotatebox{-90}{$\supsetneq$} & & \rotatebox{-90}{$\supsetneq$} & & \rotatebox{-90}{$\supsetneq$} & & \rotatebox{-90}{$\supsetneq$} & & \rotatebox{-90}{$\supsetneq$} & & \\
\equiv^{\downarrow}_{3,0} & \supsetneq & \equiv^{\downarrow}_{3,1} & \supsetneq & \equiv^{\downarrow}_{3,2} & \supsetneq & \equiv^{\downarrow}_{3,3} & = & \equiv^{\downarrow}_{3,4} & \cdots & \\
\vdots & & \vdots & & \vdots & & \vdots & & \vdots & \ddots &
\end{array}
$$

Figure 4: Hierarchy of $\mathrm{XPath}_=(\downarrow)$.

- **Harmony:** If $xZ_{j,t}x'$ then $label(x) = label(x')$.
- **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. $data(v) = data(w) \Leftrightarrow data(v') = data(w')$,
  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.

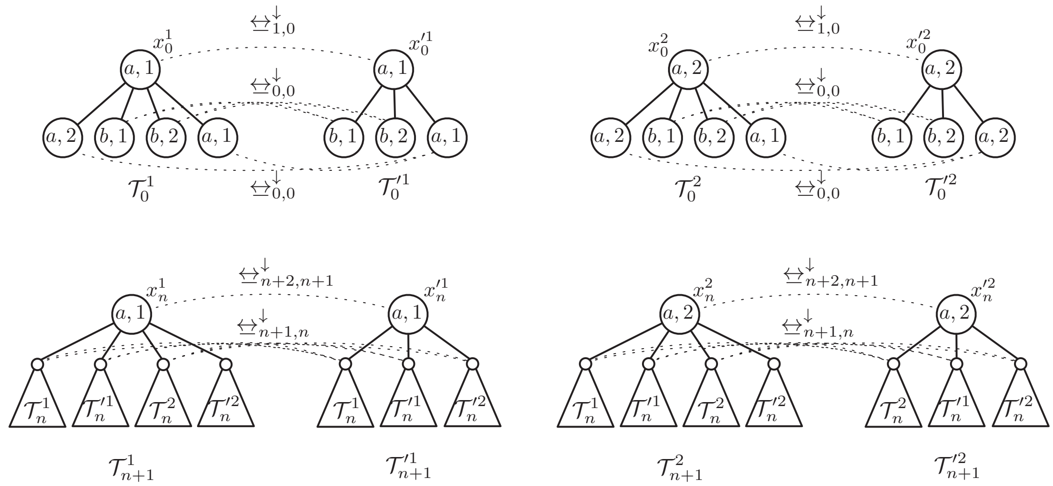
Following the same ideas used in Propositions 9 and 11, it is easy to show that (ℓ, k)-bisimulations characterize (ℓ, k)-equivalence.

Proposition 40. *$\mathcal{T}, u \mathrel{\underline{\leftrightarrow}}^{\downarrow}_{\ell,k} \mathcal{T}', u'$ 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 4). We speculate that a similar hierarchy holds in the absence of data values, but this is not a direct consequence of our result.

Theorem 41. *For all $\ell, k \geq 0$, $i \geq 1$, $\equiv^{\downarrow}_{\ell,0} \supsetneq \equiv^{\downarrow}_{\ell,1} \supsetneq \cdots \supsetneq \equiv^{\downarrow}_{\ell,\ell} = \equiv^{\downarrow}_{\ell,\ell+i}$, and $\equiv^{\downarrow}_{\ell,k} \supsetneq \equiv^{\downarrow}_{\ell+i,k}$.*

Proof. Consider the data trees $\mathcal{T}^i_n$, $\mathcal{T}'^i_n$ $(n \geq 0, i \in \{1, 2\})$ defined for every k.



Note that $\equiv^{\downarrow}_{\ell,k+1} \subseteq \equiv^{\downarrow}_{\ell,k}$ and $\equiv^{\downarrow}_{\ell+1,k} \subseteq \equiv^{\downarrow}_{\ell,k}$ by definition. We show that $\equiv^{\downarrow}_{\ell,k} \neq \equiv^{\downarrow}_{\ell,k+1}$ for all $\ell \geq k+1$. For this purpose, we show that $\mathcal{T}_k^1, x_k^1 \equiv^{\downarrow}_{k+1,k} \mathcal{T}_k'^1, x_k'^1$ but $\mathcal{T}_k^1, x_k^1 \not\equiv^{\downarrow}_{k+1,k+1} \mathcal{T}_k'^1, x_k'^1$.

That $\mathcal{T}_k^1, x_k^1 \not\equiv^{\downarrow}_{k+1,k+1} \mathcal{T}_k'^1, x_k'^1$ results from the fact that the property "there is a path of length $k+1$ ending with a label a whose every pair of consecutive nodes have distinct data value" is definable with the following formula φ_{k+1} of depth $k+1$ and nesting depth $k+1$,

$$\varphi_1 = \langle \varepsilon \neq \downarrow[a] \rangle$$
$$\varphi_{i+1} = \langle \varepsilon \neq \downarrow[\varphi_i] \rangle \quad \text{for } i > 0.$$

Since $\mathcal{T}_k^1, x_k^1 \models \varphi_{k+1}$ but $\mathcal{T}_k'^1, x_k'^1 \not\models \varphi_{k+1}$, it follows that $\mathcal{T}_k^1, x_k^1 \not\equiv^{\downarrow}_{k+1,k+1} \mathcal{T}_k'^1, x_k'^1$.

To show $\mathcal{T}_k^1, x_k^1 \equiv^{\downarrow}_{k+1,k} \mathcal{T}_k'^1, x_k'^1$ we use Proposition 40 and show $\mathcal{T}_k^1, x_k^1 \underline{\leftrightarrow}^{\downarrow}_{k+1,k} \mathcal{T}_k'^1, x_k'^1$. Note that $\mathcal{T}_k^1$ and $\mathcal{T}_k^2$ (resp. $\mathcal{T}_k'^1$ and $\mathcal{T}_k'^2$) are equal modulo renaming of data values, so we are also showing that the roots of any two data trees with subindex k are $(k+1, k)$-bisimilar.

Observation 42. *Note that the set of immediate subtrees of the roots of* $\mathcal{T}_k^1, \mathcal{T}_k'^1, \mathcal{T}_k^2, \mathcal{T}_k'^2$ *are the same as those of* $\mathcal{T}_k'^1, \mathcal{T}_k^2, \mathcal{T}_k'^2$ *(and of* $\mathcal{T}_k^1, \mathcal{T}_k'^1, \mathcal{T}_k'^2$*) by construction.*

We show $\mathcal{T}_k^1, x_k^1 \underline{\leftrightarrow}^{\downarrow}_{k+1,k} \mathcal{T}_k'^1, x_k'^1$. For every $j \leq k+1$, $t \leq k$, let $Z_{j,t}$ be the set of all pairs $(x, y) \in T_k^1 \times T_k'^1$ so that x and y are some $x_{k'}^i$ or $x_{k'}'^i$ for $i \in \{1, 2\}$ and $k' \geq t$ (the notation $x_{k'}^i$ and $x_{k'}'^i$ does not necessarily identify a unique node but, possibly, many; the intended meaning is that they can refer to *any* of them). Observe that

$$Z_{j+1,t} \subseteq Z_{j,t} \quad \text{for all } j, t \leq k. \tag{4}$$

We show that $(Z_{j,t})_{j \leq k+1, t \leq k}$ verify the bisimulation conditions. We proceed by induction on $j + t$. The base case, $j = t = 0$, is trivial. The case $l > 0$, $t = 0$ is also straightforward.

Suppose then that $t > 0$. Let $(u, u') \in Z_{j,t}$. Again, Harmony is met since $Z_{l,t}$ relates only nodes with label a. Let us suppose that u is some $x_{t'}^1$ and u' is $x_{t'}'^1$ for some $t' \leq t$, the other cases being similar or simpler.

Let us now show Zig. Let v, w be so that $x_{t'}^1 \xrightarrow{n} v$ and $x_{t'}'^1 \xrightarrow{m} w$ with $n, m \leq j$.

- If v is inside the subtree $\mathcal{T}^2_{t'-1}$ of $\mathcal{T}^1_{t'}$, but it is not $x^2_{t'-1}$, then we choose v' as the corresponding node inside the subtree $\mathcal{T}^1_{t'-1}$ of $\mathcal{T}'^1_{t'}$. Remember that $\mathcal{T}^1_{t'-1}$ and $\mathcal{T}^2_{t'-1}$ are isomorphic modulo a renaming of data values, so by *corresponding* we mean the node in the same position in the tree. $data(v) = data(v')$ by Observation 42. Furthermore, since every node of $\mathcal{T}^1_{t'-1}$ is in a $Z_{j,t-1}$-relation with the corresponding node in $\mathcal{T}^2_{t'-1}$ by construction of $Z_{j,t-1}$, it follows that $(\stackrel{i}{\to}v)Z_{j,t-1}(\stackrel{i}{\to}v')$ for all $i \leq n$. Thus, by (4), $(\stackrel{i}{\to}v)Z_{j-n+i,t-1}(\stackrel{i}{\to}v')$ for all $i \leq n$.

- If, on the other hand, v is $x^2_{t'-1}$, we choose v' as the root of $\mathcal{T}'^2_{t'-1}$, $x'^2_{t'-1}$. Again, we have that $data(v') = data(v)$ and by construction that $vZ_{j,t-1}v'$. Thus, by (4), $vZ_{j-1,t-1}v'$.

- Finally, if v falls outside $\mathcal{T}'^2_{t'-1}$, we choose v' as the same node in $\mathcal{T}'^1_{t'}$, where of course we will have that $data(v) = data(v')$ and that $(\stackrel{i}{\to}v)Z_{j,t-1}(\stackrel{i}{\to}v')$ for all $i \leq n$. Thus, by (4), $(\stackrel{i}{\to}v)Z_{j-n+i,t-1}(\stackrel{i}{\to}v')$ for all $i \leq n$.

We do the same with w and w'. Since in every case we can reach a node with the same data value and so that the corresponding nodes in the path are $Z_{j,t-1}$-related, it follows that the Zig condition is satisfied. The Zag condition is only easier, and hence we conclude that $\mathcal{T}^1_k, x \leftrightarroweq^{\downarrow}_{k+1,k} \mathcal{T}'^1_k, x'$ for every k.

We therefore have that $\equiv^{\downarrow}_{\ell,k+1} \subsetneq \equiv^{\downarrow}_{\ell,k}$ for all $\ell \geq k+1$.

The fact that $\equiv^{\downarrow}_{\ell+1,k} \subsetneq \equiv^{\downarrow}_{\ell,k}$ is of course trivial, formulas of depth $\ell+1$ can express "the tree has at least depth $\ell+1$", which cannot be expressed by formulas of depth ℓ.

It remains to show that $\equiv^{\downarrow}_{\ell,k} = \equiv^{\downarrow}_{\ell,k+1}$ for all $\ell \leq k$. To show this, we prove $\mathcal{T}, x \leftrightarroweq^{\downarrow}_{\ell,k+1} \mathcal{T}', x'$ for every $\mathcal{T}, \mathcal{T}'$ so that $\mathcal{T}, x \leftrightarroweq^{\downarrow}_{\ell,k} \mathcal{T}', x'$. We prove it by induction on $\ell + k$. The base case is easy.

For the inductive case, let $Z_{j,t} = \leftrightarroweq^{\downarrow}_{j,t}$ for all $j \leq \ell, t \leq k$. Hence, $(Z_{j,t})_{j \leq \ell, t \leq k}$ verify the bisimulation conditions. Let $Z_{\ell,k+1} = \{(x, x')\}$. We show that $Z_{\ell,k+1}$ together with $(Z_{j,t})_{j \leq \ell, t \leq k}$ verifies the bisimulation conditions. Harmony follows from $xZ_{\ell,k}x'$. We show Zig since Zag is equivalent. Suppose $x \stackrel{n}{\to} v$, $x \stackrel{m}{\to} w$ with $n, m \leq \ell$. Then, since $Z_{\ell,k}$ verifies Zig, there are $x' \stackrel{n}{\to} v'$, $x' \stackrel{m}{\to} w'$ where

(1) $data(v) = data(w')$ iff $data(v') = data(w')$,

(2) $(\stackrel{i}{\to}v)Z_{\ell-n+i,k-1}(\stackrel{i}{\to}v')$ for all $i \in \{0, \ldots, n-1\}$, and

(3) $(\stackrel{i}{\to}w)Z_{\ell-m+i,k-1}(\stackrel{i}{\to}w')$ for all $i \in \{0, \ldots, m-1\}$.

Since $\ell \leq k$, then $\ell - n + i \leq k - 1$. Further, $\ell - n + i + k < \ell + k$, which means that we can apply the inductive hypothesis. Hence, by inductive hypothesis, $\mathcal{T}, (\stackrel{i}{\to}v) \leftrightarroweq^{\downarrow}_{\ell-n+i,k} \mathcal{T}', (\stackrel{i}{\to}v')$ and thus $(\stackrel{i}{\to}v)Z_{\ell-n+i,k}(\stackrel{i}{\to}v')$. By an indentical reasoning, $\mathcal{T}, (\stackrel{i}{\to}w) \leftrightarroweq^{\downarrow}_{\ell-n+i,k} \mathcal{T}', (\stackrel{i}{\to}w')$ and thus $(\stackrel{i}{\to}w)Z_{\ell-n+i,k}(\stackrel{i}{\to}w')$. Thus, the Zig condition for $\leftrightarroweq^{\downarrow}_{\ell,k+1}$ is verified. The Zag condition holds by symmetry. □

With respect to vertical XPath, note that since $\equiv^{\downarrow\uparrow}_{r,s,k} \subseteq \equiv^{\downarrow\uparrow}_{r',s',k'}$ for all $(r, s, k) \leq (r', s', k')$, as a consequence of Proposition 15 we obtain that for every r, s, k with $r + s \geq 2$ there is some $k' > k$ so that $\equiv^{\downarrow\uparrow}_{r,s,k} \supsetneq \equiv^{\downarrow\uparrow}_{r,s,k'}$. In fact, we conjecture that $\equiv^{\downarrow\uparrow}_{r,s,k} \supsetneq \equiv^{\downarrow\uparrow}_{r,s,k+1}$ for every k. We argue that this can be proven through the models $(\mathcal{T}_n)_n$ in the proof of Proposition 15, by showing that $\mathcal{T}_k, x_{r',s'} \equiv^{\downarrow\uparrow}_{r,s,k} \mathcal{T}_{k+1}, x_{r',s'}$ but $\mathcal{T}_k, x_{r',s'} \not\equiv^{\downarrow\uparrow}_{r,s,k+1} \mathcal{T}_{k+1},, x_{r',s'}$ for every $(r, s) \geq (r', s')$. The fact that $\equiv^{\downarrow\uparrow}_{r,s,k} \supsetneq \equiv^{\downarrow\uparrow}_{r+1,s,k}$ and $\equiv^{\downarrow\uparrow}_{r,s,k} \supsetneq \equiv^{\downarrow\uparrow}_{r,s+1,k}$ are straightforward. We would then obtain the following.

Conjecture 43. $\equiv^{\downarrow\uparrow}_{r,s,k} \supsetneq \equiv^{\downarrow\uparrow}_{r',s',k'}$ *for all* $(r, s, k) < (r', s', k')$, $r + s \geq 2$.

6. 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` ($\rightarrow$) and `previous-sibling` ($\leftarrow$). 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. Next we explain why this is so. Consider the following notation for denoting positions of nodes in a tree. Positions are elements of $(\mathbb{N} \times \mathbb{N})^*$, where the root's position is the empty string ϵ, and the position of any other node in the tree is the concatenation of the position of its parent and the pair (l, r), where l is the number of siblings to the left of the node and r the number of siblings to its right. For every position $p \in (\mathbb{N} \times \mathbb{N})^*$ of a tree, there is a path expression α_p of XPath$_=(\downarrow\rightarrow)$ that can access the node in this position (and only this position) from the root. The α_p's are defined as $\alpha_\epsilon = \varepsilon$ for the root; $\alpha_{(l,r)} = \downarrow\rightarrow^{l+1}[\langle\rightarrow^k\rangle \wedge \neg\langle\rightarrow^{k+1}\rangle]$ for $(l, r) \in \mathbb{N} \times \mathbb{N}$; and $\alpha_{p\cdot(l,r)} = \alpha_p\alpha_{(l,r)}$ for $p \in (\mathbb{N} \times \mathbb{N})^*$, $(l, r) \in \mathbb{N} \times \mathbb{N}$. It is easy to check that testing $\bigwedge_p \langle\alpha_p[\neg\langle\downarrow\rangle]\rangle$ where p ranges over all leaf positions of a given tree $\mathcal{T}$ tests the property that the subtree hanging from the node where the formula is evaluated is structurally isomorphic to $\mathcal{T}$. Further, by also adding the tests $\langle\alpha_p[a] = \alpha_{p'}[a']\rangle$ for every pair of positions p, p' with label a, a' of $\mathcal{T}$ having the same data value, as well as $\langle\alpha_p[a] \neq \alpha_{p'}[a']\rangle$ for those having different data value, yields the property of being *equal* to $\mathcal{T}$, up to isomorphism of data values.

In Section 5 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 maximum auto-bisimulation (observe that the identity is always an auto-bisimulation, and that the union of two bisimulations is a bisimulation; therefore there is a 'maximum', often called 'coarsest', bisimulation).

The idea is to find the coarsest auto-bisimulation Z over a given data-tree $\mathcal{T}$. One cannot simply make the quotient of $\mathcal{T}$ over Z, as the result is not necessarily a tree and it is not clear how to assign data values to each class in the quotient. However, one can make a quotient of $\mathcal{T}$ over the equivalence relation 'Z and same data value'. If we do so, we obtain a smaller structure, and then evaluate all queries here.

Determining the maximum auto-(bi)simulation, either downward or vertical, of a finite data $\mathcal{T}$ tree can be done in polynomial time. A naive algorithm starts by defining Z as the set of all nodes which satisfy **Harmony**. Each time **Zig** or **Zag** is not satisfied, it removes from Z the pair responsible for **Zig** or **Zag** not being true. It repeats this until a fixed point of Z is found; this is the maximum auto-bisimulation in $\mathcal{T}$. If one is interested in deciding if two nodes of different finite data trees are bisimilar, one can use the same idea: the answer is 'yes' if and only if the fixed point is not the empty set. Since checking the validity of **Zig** or **Zag** is polynomial time computable (because there are linearly many paths in a tree), and at each step the algorithm decreases the size of Z, the whole process has polynomial time. Better implementations, based on more sophisticated ideas, such as the works of Henzinger, Henzinger, and Kopke (1995) or Dovier, Piazza, and Policriti (2004), can lead to more efficient polynomial time algorithms. We plan to design and implement algorithms for data tree minimization using bisimulation and investigate their computational complexity.

Acknowledgements

This work was partially supported by grant ANPCyT-PICT-2013-2011, 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".

References

Abiteboul, S., Bourhis, P., Muscholl, A., & Wu, Z. (2013). Recursive queries on trees and data trees. In *International Conference on Database Theory*, pp. 93–104.

Abriola, S., Descotte, M. E., & Figueira, S. (2014). Definability for downward and vertical XPath on data trees. In *Workshop on Logic, Language, Information and Computation*, Vol. 6642 of *Lecture Notes in Computer Science*, pp. 20–34.

Abriola, S., Descotte, M. E., & Figueira, S. (2015). Model theory of XPath on data trees. Part II: Binary bisimulation and definability. To appear in *Information and Computation*. `http://www.glyc.dc.uba.ar/santiago/papers/xpath-part2.pdf`.

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

Benedikt, M., & Koch, C. (2008). XPath leashed. *ACM Computing Surveys*, *41*(1).

Blackburn, P., de Rijke, M., & Venema, Y. (2001). *Modal Logic*, Vol. 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

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

Bojańczyk, M., & Parys, P. (2011). XPath evaluation in linear time. *Journal of the ACM*, *58*(4), 17.

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

David, C. (2008). Complexity of data tree patterns over XML documents. In *Mathematical Foundations of Computer Science*, Vol. 5162 of *Lecture Notes in Computer Science*, pp. 278–289. Springer.

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

Dovier, A., Piazza, C., & Policriti, A. (2004). An efficient algorithm for computing bisimulation equivalence. *Theor. Comput. Sci*, *311*, 221–256.

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

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

Figueira, D., & Segoufin, L. (2011). Bottom-up automata on data trees and vertical XPath. In *International Symposium on Theoretical Aspects of Computer Science*, Vol. 9 of *LIPIcs*, pp. 93–104. Leibniz-Zentrum für Informatik.

Figueira, D., Figueira, S., & Areces, C. (2014). Basic model theory of XPath on data trees. In *International Conference on Database Theory*, pp. 50–60.

Figueira, D., & Libkin, L. (2014). Pattern logics and auxiliary relations. In *Logic in Computer Science*, pp. 40:1–40:10.

Figueira, S., & Gorín, D. (2010). On the size of shortest modal descriptions.. In *Advances in Modal Logic*, Vol. 8, pp. 114–132.

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

Goranko, V., & Otto, M. (2007). Model theory of modal logic. In P. Blackburn, J. V. B., & Wolter, F. (Eds.), *Handbook of Modal Logic*, Vol. 3 of *Studies in Logic and Practical Reasoning*, chap. 5, pp. 249–329. Elsevier.

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

Gyssens, M., Paredaens, J., Gucht, D. V., & Fletcher, G. (2006). Structural characterizations of the semantics of XPath as navigation tool on a document. In *Principles of Database Systems*, pp. 318–327. ACM.

Harel, D. (1984). Dynamic logic. In Gabbay, D., & Guenthner, F. (Eds.), *Handbook of Philosophical Logic. Vol. II*, Vol. 165 of *Synthese Library*, pp. 497–604. D. Reidel Publishing Co., Dordrecht. Extensions of classical logic.

Henzinger, M. R., Henzinger, T. A., & Kopke, P. W. (1995). Computing simulations on finite and infinite graphs. In *Proc. of 36th Annual Symposium on Foundations of Computer Science*, pp. 453–462. IEEE Computer Society Press.

Hopcroft, J. (1971). *An nlog(n) algorithm for minimizing states in a finite automaton*. In Z. Kohave, editor, *Theory of Machines and Computations*. Academic Press.

Jurdziński, M., & Lazić, R. (2011). Alternating automata on data trees and xpath satisfiability. *ACM Transactions on Computational Logic*, *12*(3), 19.

Kanellakis, P., & Smolka, S. (1990). CCS expressions, finite state processes, and three problems of equivalence. *Inf. Comput.*, *86*(1), 43–68.

Kurtonina, N., & de Rijke, M. (1997). Simulating without negation. *Journal of Logic and Computation*, *7*, 503–524.

Lutz, C., Piro, R., & Wolter, F. (2011). Description logic tboxes: Model-theoretic characterizations and rewritability. In *International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pp. 983–988.

Marx, M. (2004). XPath with conditional axis relations. In *International Conference on Extending Database Technology*, Vol. 2992 of *Lecture Notes in Computer Science*, pp. 477–494. Springer.

Marx, M., & de Rijke, M. (2005). Semantic characterizations of navigational XPath. *SIGMOD Record*, *34*(2), 41–46.

Milner, R. (1980). *A Calculus of Communicating Systems*, Vol. 92 of *Lecture Notes in Computer Science*. Springer.

Neven, F., Schwentick, T., & Vianu, V. (2004). Finite state machines for strings over infinite alphabets. *ACM Transactions on Computational Logic*, *5*(3), 403–435.

Olteanu, D. (2007). Forward node-selecting queries over trees. *ACM Transactions on Database Systems*, *32*(1), 3.

Olteanu, D., Meuss, H., Furche, T., & Bry, F. (2002). XPath: Looking forward. In *International Conference on Extending Database Technology*, pp. 109–127.

Otto, M. (2004a). Elementary proof of the van Benthem-Rosen characterisation theorem. Tech. rep. 2342, Fachbereich Mathematik, Technische Universität Darmstadt.

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

Otto, M. (2006). Bisimulation invariance and finite models. In *Logic Colloquium'02*, Vol. 27 of *Lecture Notes in Logic*, pp. 276–298.

Paige, R., & Tarjan, R. (1987). Three partition refinement algorithms. *SIAM J. Comput.*, *16*(6), 973–989.

Park, D. (1981). Concurrency and automata on infinite sequences. In *Theoretical Computer Science*, Vol. 104 of *Lecture Notes in Computer Science*, pp. 167–183. Springer.

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

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

ten Cate, B., Fontaine, G., & Litak, T. (2010). Some modal aspects of XPath. *Journal of Applied Non-Classical Logics*, *20*(3), 139–171.

van Benthem, J. (1976). *Modal Correspondence Theory.* PhD thesis, Universiteit van Amsterdam.