Basic Model Theory of XPath on Data Trees

Diego Figueira
University of Edinburgh
UK
Santiago Figueira
Universidad de Buenos Aires
and CONICET
Argentina
Carlos Areces
Universidad Nacional de Córdoba
and CONICET
Argentina

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

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

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

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

General Terms
Theory, Languages

1. INTRODUCTION

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

Core-XPath (term coined in [13]) is the fragment of XPath 1.0 containing the navigational behavior of XPath. It can express properties of the underlying tree structure of the XML document, such as the label (tag name) of a node, but it cannot express conditions on the actual data contained in the attributes. In other words, it only allows to reason about trees over a finite alphabet. Core-XPath has been well studied and its satisfiability problem is known to be decidable even in the presence of DTDs [17, 1]. Moreover, it is known that it is equivalent to FO² (first-order logic with two variables over an appropriate signature on trees) in terms of expressive power [18], and that it is strictly less expressive than PDL with converse over trees [2]. From a database perspective, however, Core-XPath fails to include the single most important construct in a query language: the join. Without the ability to relate nodes based on the actual data values of the attributes, the logic’s expressive power is inappropriate for many applications.

The extension of Core-XPath with (in)equality tests between attributes of elements in an XML document is named Core-Data-XPath in [4]. Here, we will call this logic XPath+. Models of XPath+ are data trees which can be seen as XML documents. A data tree is a tree whose nodes contain 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 XPath+ is to allow formulas of the form (α = β), 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

(c) 2014, Copyright is with the authors. Published in Proc. 17th International Conference on Database Theory (ICDT), March 24-28, 2014, Athens, Greece: ISBN 978-3-89318066-1, on OpenProceedings.org. Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0.
of logics evaluated over data trees. For example, satisfiability and evaluation are discussed in [8, 5]. In particular, all the logics studied in this article have a decidable satisfiability problem [10, 9], but tools to investigate their expressive power are still lacking. There are good reasons for this: in the presence of joins and data values, classical notions such as Ehrenfeucht-Fraïssé games or structural bisimulations are difficult to handle. In this article we take the first steps towards understanding the expressive power and model theory of XPath on data trees.

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

- In §3 and §5 we introduce bisimulation notions for XPath↓, XPath↑↓, XPath↑↑, and XPath↑↑↑ and show that they precisely characterize the logical equivalence relation of the respective logic. We also consider fine grained versions of these bisimulations that take into account the number of nested axes and subformulas. The notion of bisimulation for XPath↑↑ depends on a strong normal form which we also introduce.
- In §4 we show that the simulations associated to the defined bisimulations characterize the positive fragments of the logics: a formula is equivalent to a positive formula if and only if it is invariant under simulations.
- In §6 we characterize XPath↑↓ as the fragment of first-order logic over data trees (over a signature that includes the child relation and an equivalence relation) that is invariant under bisimulations. If we consider XPath↑↑ instead the characterization fails, but a weaker result can still be established.

Using bisimulations we show (non)expressivity results about XPath↓ in §7. We characterize, for example, in which cases increasing the nesting depth increases the expressive power of XPath↓.

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

**Related work:** The notion of bisimulation was introduced independently by Van Benthem [26] in the context of modal correspondence theory, Milner [19] and Park [23] in concurrency theory, and Forti and Honsell [11] in non-wellfounded set theory (see [25] for a historical outlook). This classical work defines a standard notion of bisimulation but this notion has to be suitably adapted for a particular, given logic. The notion of bisimulation for a given logic L defines when two models are indistinguishable for L, that is, when there is no formula of L that is true in one model but false in the other. Bisimulations can also be used to obtain model theoretic characterizations that identifies the expressive power of a logic L↑↑ in terms of the bisimulation invariant fragment of a logic L↑↑ which, hopefully, is better understood. The challenge, here, is to pinpoint both the appropriate notion of bisimulation required and the adequate ‘framework’ logic L↑↑. The classical example of a result of this kind is Van Benthem’s characterization for the basic modal logic as the bisimulation (with the standard notion of bisimulation) invariant fragment of first-order logic [26]. Van Benthem’s original result over arbitrary structures was proved to hold for finite structures by Rosen [24]. The proof was then simplified and unified by Otto [20, 22], and later expanded by Dawar and Otto [7] to other classes of structures.

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

### 2. PRELIMINARIES

#### 2.1 Notation

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

#### 2.2 Data trees

Let Trees\((A)\) be the set of ordered and unranked trees over an arbitrary alphabet \( A \). We say that \( T \) is a data tree if it is a tree from Trees\((A \times \mathbb{D})\) where \( A \) is a finite set of labels and \( \mathbb{D} \) is an infinite set of data values. Figure 1 shows an example of a (finite) data tree. A data tree is **finite branching** if every node has finitely many children. For any given data tree \( T \), we denote by \( T \) its set of nodes. We use letters \( x, y, z, v, w \) as variables for nodes. Given a node \( x \in T \), we write label\((x) \in \mathcal{A} \) to denote the node’s label, and label\((x) \in \mathbb{D} \) 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 \( x \to y \) if \( y \) is a descendant of \( x \) at distance \( n \). In particular, \( \to \) is the same as \( \sim \), and \( \to \) is the identity relation. (\( x \to \)) denotes the set of all descendants of \( x \) at distance \( n \), and (\( \to \)) denotes the sole ancestor of \( y \) at distance \( n \) (assuming it has one).

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

#### 2.3 XPath

![Figure 1: A data tree of Trees\((A \times \mathbb{D})\) with \( A = \{a, b\} \) and \( \mathbb{D} = \mathbb{N} \).](image-url)
We introduce the query language XPath adapted to data trees as abstractions of XML documents. We work with a simplification of XPath that correspond to the navigational part of XPath 1.0 with data equality and inequality. XPath_0 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_0, with O ⊆ {↓, ↑, , }, is defined by mutual recursion as follows:

\[
\begin{align*}
\alpha, \beta & \vdash a | [ϕ] | \alpha β | \alpha \lor \beta \quad a \in O \cup \{ε\} \\
ϕ, ψ & \vdash a | \neg ϕ | ϕ \land ψ | ϕ γ ψ | α | \{α = β\} | \{α ≠ β\} \\
\end{align*}
\]

A formula of XPath_0 is either a node expression or a path expression. To save space, we use XPath_0 for XPath_0(↓); XPath_0 for XPath_0(↓); XPath_0↑↑ for XPath_0(↓, ↓); and XPath_0↑↑↑ for XPath_0(↓, ↓, ↓).

We formally define the semantics of XPath_0 in Table 1. As an example, if T is the data tree shown in Figure 1, then \(\{↓, \{b \land (↑b \neq ↓b)\}\}\)T = \{x, y, z\}, where the formula reads: “there is a descendant node labeled b, with two children labeled b with different data values.” For a data tree T and u ∈ T, we write T|u to denote u ∈ [ϕ]T, and we say that T, u satisfies ϕ. We say that the formulas ϕ, ψ of XPath_0 are equivalent (notation: ϕ ≡ ψ) iff \([ϕ]T = [ψ]T\) for all data trees T. Similarly, path expressions α, β of XPath_0 are equivalent (notation: α ≡ β) iff \([α]T = [β]T\) for all data trees T.

We call downward XPath to XPath_0 and vertical XPath to XPath_0↑↑.

In terms of expressive power, it is easy to see that universe: every XPath_0 node expression ϕ′ has an equivalent ϕ with no \(∪\) in its path expressions. ϕ′ can be computed in exponential time without incrementing the number of nested axes or the number of nested subformulas. It is enough to use the following equivalences to eliminate occurrences of \(∪\)

\[
\begin{align*}
\{α ∪ β\} & \equiv \{β ∪ α\} \\
\{α β\} & \equiv \{β α\} ∨ \{α β\} \\
\{γ β\} & \equiv \{γ α β\} ∨ \{γ β α\} \\
\end{align*}
\]

where \(⊙ \in \{(=, ≠)\}\). We will henceforth assume that formulas do not contain union of path expressions.

3. BISIMULATION

3.1 Downward XPath

We write d(ϕ) to denote the downward depth of ϕ, defined in Table 2. Let \(L\)-XPath_0 be the fragment of XPath_0 consisting of all formulas ϕ with d(ϕ) ≤ ℓ.

Let T and T′ be data trees, and let u ∈ T, u′ ∈ T′. We say that T, u and T′, u′ are equivalent for XPath_0 (notation: T, u ≡ T′, u′) iff for all formulas ϕ ∈ XPath_0, we have T|u ≡ T′|u′. We say that T, u and T′, u′ are \ell/-equivalent for XPath_0 (notation: T, u \equiv T′, u′) iff all ϕ ∈ \ell/XPath_0, we have T, u ≡ T′, u′. For every ℓ, there are finitely many different formulas ϕ of d(ϕ) ≤ ℓ up to logical equivalence.

**Proposition 3.1.** \(\equiv\) has finite index.

**Corollary 3.2.** \{T′, u′ | T, u \equiv T′, u′\} is definable by an \(L\)-XPath_0-formula χ_n(T, u).

3.1.1 Bisimulation and \ell/-bisimulation

Let T and T′ be two data-trees. We say that u ∈ T and u′ ∈ T′ are bisimilar for XPath_0 (notation: T, u ≡ T′, u′) iff there is a relation Z ⊆ T × T′ such that \(u Z u′\) and for all x ∈ T and x′ ∈ T′ we have

- **Harmony:** If \(x Z x′\) then \(label(x) = label(x′)\).

- **Zip (Figure 2):** If \(x Z x′, x → v \land x′ → w\) then there are \(v′, w′ \in T′\) such that \(x → v′, x′ → w′\) and

  1. data(v) = data(w) ⇔ data(v′) = data(w′),
  2. \(\rightarrow v\) \(Z \rightarrow v′\) for all 0 ≤ i < n, and
  3. \(\leftarrow w\) \(Z \leftarrow w′\) for all 0 ≤ i < m.

- **Zag:** If \(x Z x′, x → v \land x′ → w\) then there are \(v, w \in T\) such that \(x → v, x → w\) and items 2, 2, and 3 above are verified.

For a data tree T and u ∈ T, let T|u denote the subtree of T induced by \{v ∈ T | (3n) u → v\}. Observe that the root of T|u is u. The following results are straightforward consequences of the definition of bisimulation:

**Proposition 3.3.** T, u ≡ (T|u), u.

**Proposition 3.4.** If T is a subtree of T′ and u ∈ T then T, u ≡ (T′, u).

We say that u ∈ T and u′ ∈ T′ are \ell/-bisimilar for XPath_0 (notation: T, u ≡ T′, u′) if there is a family of relations \((Z_j)_{j ≤ ℓ}\) in T × T′ such that \(u Z_j u′\) and for all j ≤ ℓ, x ∈ T and x′ ∈ T′ we have

\[
\begin{align*}
\end{align*}
\]
\[ \text{dd}(a) = 0 \]
\[ \text{dd}(\varphi \land \psi) = \max\{\text{dd}(\varphi), \text{dd}(\psi)\} \]
\[ \text{dd}(\neg \varphi) = \text{dd}(\varphi) \]
\[ \text{dd}(\langle \alpha \rangle) = \text{dd}(\alpha) \]
\[ \text{dd}(\langle \alpha \land \beta \rangle) = \max\{\text{dd}(\alpha), \text{dd}(\beta)\} \]
\[ \text{dd}(\lambda) = 0 \]
\[ \text{dd}(\varepsilon \alpha) = \cdots \text{a subformula } \varphi \text{ of } \alpha \text{ and} \]
\[ k \in \{0, \ldots, n\} \text{ such that } T, x_k \models \varphi \text{ and } T', x'_k \not\models \varphi. \]
This contradicts the inductive hypothesis 1.

Table 2: Definitions of downward depth, vertical depth and nesting depth. \((a \in \mathcal{A}, \circ \in \{\{\}, \neq\}, '+' \text{ and 'max'} \text{ are performed component-wise}, \alpha \text{ is any path expression or the empty string } \lambda.\)

<table>
<thead>
<tr>
<th>Downward depth</th>
<th>Vertical depth</th>
<th>Nesting depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T, u \vdash_{\downarrow}^T T', u' ) implies (T, u \equiv_{\downarrow}^T T', u' ).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROPOSITION 3.6.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROOF. We actually show that if (T, u \vdash_{\downarrow}^T T', u' ) via ((Z_n)_{n \leq} ) then for all (0 \leq n \leq j \leq \ell ), for all (\varphi ) with (\text{dd}(\varphi) \leq j ), and for all (a ) with (\text{dd}(a) \leq j ):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. If (xZ_jx') then (T, x \models \varphi \iff T', x' \models \varphi; )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. If (x^n v, x'^n v' ) and ((\sim v) Z_{(j-n)+1} (\sim v')) for all (0 \leq i \leq n), then ((x, v) \in [\alpha]^T ) iff ((x', v') \in [\alpha]^T'.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>We show 1 and 2 by induction on (</td>
<td>\varphi</td>
<td>+</td>
</tr>
<tr>
<td>Let us see item 1. The base case is (\varphi = a ) for some (a \in \mathcal{A}.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>By Harmony, (label(x) = label(x')) and then (T, x \models \varphi \iff T', x' \models \varphi.) The Boolean cases for (\varphi ) are straightforward.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suppose (\varphi = (\alpha = \beta).) We show (T, x \models \varphi \Rightarrow T', x' \models \varphi, ) so assume (T, x \models \varphi.) Suppose there are (v, w \in T ) and (n, m \leq j) such that (x^n v, x^m w, (x, v) \in [\alpha]^T, (x, w) \in [\beta]^T) and (data(v) = data(w).) By Zig, there are (v', w' \in T') such that (x^n v', x^m w', (\sim v) Z_{j-n+1} (\sim v')) for all (0 \leq i \leq n, (\sim w) Z_{j-m+1} (\sim w')) for all (0 \leq i \leq m, ) and (data(v') = data(w').) By inductive hypothesis 2 (twice), ((x', v') \in [\alpha]^T') and ((x', w') \in [\beta]^T'.) Hence (T', x' \models \varphi.) The implication (T', x' \models \varphi \Rightarrow T, x \models \varphi) is analogous. The case (\varphi = (\alpha \neq \beta)) is shown similarly. The case (\varphi = (\alpha)) is similar (and simpler) to the previous case.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>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]^T'.) If (\alpha = \downarrow) then (x \rightarrow v) in (T,) and so (n = 1.) Since (x' \rightarrow v',) we have ((x', v') \in [\alpha]^T'.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>For the inductive step, let</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x_0, \ldots, x_n \in T ) and (x'_0, \ldots, x'_n \in T')</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| be such that \(x = x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_n = v\) in \(T,\) \(x' = x'_0 \rightarrow x'_1 \rightarrow x'_2 \rightarrow \cdots \rightarrow x'_n = v'\) in \(T',\) and \(x \sim Z_{(j-n)+1} x'\) for all \(0 \leq i \leq n.\) Assume, for contradiction, that \((x', v') \not\in [\alpha]^T'.\) Then, there is a subformula \(\varphi \) of \(\alpha\) and \(k \in \{0, \ldots, n\}\) such that \(T, x_k \models \varphi\) and \(T', x'_k \not\models \varphi.\) This contradicts the inductive hypothesis 1. \(\square\)\]
Proposition 3.7. \( T,u \equiv_{\downarrow} T',u' \) implies \( T,u \leftrightarrow_{\downarrow} T',u' \).

Proof. Fix \( u \in T \) and \( u' \in T' \) such that \( T,u \equiv_{\downarrow} T',u' \). Define \( (Z_i)_{i \in \mathbb{L}} \) by
\[
x_{Z_i}x' \text{ if } T,x \equiv_{\downarrow} T',x'.
\]
We show that \( Z \) is an \( \ell \)-bisimulation between \( T,u \) and \( T',u' \). By hypothesis, \( uZ_iu' \). Fix \( h \leq \ell \), by construction, \( Z_h \) satisfies Harmony. Let us see that \( Z_h \) satisfies Zig (the case for \( Z \) is analogous). Suppose \( xZ_hx' \),
\[
x = v_0 \to v_1 \to \cdots \to v_n = v \quad \text{in } T,
\]
\[
x = w_0 \to w_1 \to \cdots \to w_m = w \quad \text{in } T',
\]
and \( data(v) = data(w) \) (the case \( data(v) \neq data(w) \) is shown in a similar way), where \( m,n \leq h \). Let \( P \subseteq T^{2h} \) be defined by
\[
P = \{ (v',w') \mid x' \xrightarrow{h} v' \land x' \xrightarrow{m} w' \land data(v') = data(w') \}.
\]
Since \( T,x \equiv_{\downarrow} T',x' \), \( dd((v \xrightarrow{h} v')) \leq h \) and \( T,x \models (v \xrightarrow{h} v') \), we conclude that \( P \neq \emptyset \). We next show that there exists \((v',w') \in P \) such that
\[
i. x' = v_0 \to v_1 \to \cdots \to v_n' = v' \quad \text{in } T',
\]
\[
ii. x' = w_0 \to w_1 \to \cdots \to w_m' = w' \quad \text{in } T',
\]
\[
iii. (\forall i \in \{0,\ldots,n\}) T,v_i \equiv_{h-i} T',v_i',
\]
\[
iv. (\forall j \in \{0,\ldots,m\}) T,w_j \equiv_{h-j} T',w_j',
\]
and hence \( Z \) is satisfied by \( Z_h \). By way of contradiction, assume that for all \((v',w') \in P \) satisfying \( i \) and \( ii \) we have either
\[
(a) \ (\exists i \in \{0,\ldots,n\}) T,v_i \not\equiv_{h-i} T',v_i',
\]
\[
(b) \ (\exists j \in \{0,\ldots,m\}) T,w_j \not\equiv_{h-j} T',w_j'.
\]
Fix \( \top \) as any tautology such that \( dd(\top) = 0 \). For each \((v',w') \in P \) we define two families of formulas,
\[
\psi_{v',w'}^0,\psi_{v',w'}^1,\ldots,\psi_{v',w'}^n \quad \text{and} \quad \psi_{v',w'}^0,\psi_{v',w'}^1,\ldots,\psi_{v',w'}^n,
\]
satisfying that \( dd(\psi_{v',w'}^i) \leq h-i \) for all \( i \in \{0,\ldots,n\} \) and \( dd(\psi_{v',w'}^j(\downarrow)) \leq h-j \) for all \( j \in \{0,\ldots,m\} \) as follows:

- Suppose that \((a) \) holds and that \( i \) is the smallest number such that \( T,v_i \not\equiv_{h-i} T',v_i' \). Let \( \psi_{v',w'}^i \) be such that \( dd(\psi_{v',w'}^i) \leq h-i \) and \( T,v_i \models \psi_{v',w'}^i \) but \( T',v_i' \not\models \psi_{v',w'}^i \). For \( k \in \{0,\ldots,n\} \setminus \{i\} \), let \( \psi_{v',w'}^k = \top \), and for \( k \in \{0,\ldots,m\} \setminus \{j\} \), let \( \psi_{v',w'}^k = \top \).

- Suppose that \((b) \) does not hold. Then \( (b) \) holds. Let \( j \) be the smallest number such that \( T,w_j \not\equiv_{h-j} T',w_j' \). Let \( \psi_{v',w'}^j \) be such that \( dd(\psi_{v',w'}^j) \leq h-j \) and \( T,w_j \models \psi_{v',w'}^j \) but \( T',w_j' \not\models \psi_{v',w'}^j \). For \( k \in \{0,\ldots,n\} \setminus \{j\} \), let \( \psi_{v',w'}^k = \top \), and for \( k \in \{0,\ldots,m\} \setminus \{j\} \), let \( \psi_{v',w'}^k = \top \).

For each \( i \in \{0,\ldots,n\} \) and \( j \in \{0,\ldots,m\} \), let
\[
\Phi^i = \bigwedge_{(v',w') \in P} \psi^i_{v',w'} \quad \text{and} \quad \Psi^j = \bigwedge_{(v',w') \in P} \psi^j_{v',w'}.
\]
Since \( dd(\psi^i_{v',w'}) \leq h-i \), by Proposition 3.1, there are finitely many non-equivalent formulas \( \psi^i_{v',w'} \): the same applies to \( \psi^j_{v',w'} \). Hence, both infinite conjunctions in \((1)\) are equivalent to finite ones, and we may assume that \( \Phi^i \) and \( \Psi^j \) are well-formed formulas. Finally, let \( \alpha = |\Phi^1| \cdot |\Phi^2| \cdots |\Phi^n| \) and \( \beta = |\Psi^1| \cdot |\Psi^2| \cdots |\Psi^m| \).

By construction, \( dd(\alpha), dd(\beta) \leq h \) and so \( dd(\alpha - \beta) \leq h \). Furthermore, \( T,x \models (\alpha = \beta) \) and \( T',x' \not\models (\alpha = \beta) \). This contradicts \( T,u \equiv_{\downarrow} T',u' \). □

3.2 Vertical XPath

We now study bisimulation for XPath\_\(z\). Interestingly, the notion we give is simpler than the one for XPath\_\(z\) due to a normal form enjoyed by the logic.

In the downward fragment of XPath\_\(z\), we used \( dd(\phi) \) to measure the maximum depth from the current point of evaluation that the formula can access. For the vertical fragment of XPath\_\(z\), we need to define both the maximum distance \( r \) going downward and the maximum distance \( s \) going upward that the formula can reach. We call the pair \((r,s)\) the vertical depth of a formula. Formally, the vertical depth of a formula \( \phi \) (notation: \( vd(\phi) \)) is the pair \( vd(\phi) \in \mathbb{Z}_\geq 0^2 \) defined in Table 2 for the formal definition.

Let \((r,s,k)\)-XPath\_\(z\) be the set of all formulas \( \phi \in \text{XPath}_{z} \) with \( vd(\phi) \leq (r,s) \) and \( nd(\phi) \leq k \).

We have that if \( T,u \equiv_{\downarrow} T',u' \) then \( T,u \leftrightarrow_{\downarrow} T',u' \).

3.2.1 Normal form

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

A path expression \( \alpha \) of XPath\_\(z\) is downward [resp. upward] if it is of the form \( \downarrow^n[\varphi] \) [resp. \( \varphi\uparrow^n \)] for some \( n \geq 0 \) with \( \varphi \in \text{XPath}_{z} \). For example, \( \downarrow[(\downarrow)] \) is a downward expression whereas \( \downarrow[(\uparrow)] \) is not. An upward expression is any expression of the form \( \alpha',\beta \), \( \alpha' \alpha \) or \( \alpha' \beta' \) where \( \alpha' \) is upward and \( \alpha \) is downward. Henceforth we will use \( \alpha',\beta \), \( \gamma \) to denote upward expressions and \( \alpha',\beta',\gamma \) to denote downward expressions and \( \alpha',\beta',\gamma' \) to denote up-down expressions. Note that in particular any downward or upward expression is an up-down expression. An XPath\_\(z\) formula or expression is in up-down normal form if every path expression contained in it is up-down and every data test is of the form \( (\varphi \land \alpha') \) with \( \alpha' \in \{\varphi \} \).

Proposition 3.8. Let \( \varphi \in (r,s,k)\)-XPath\_\(z\). There is \( \varphi^{r+s+2} \in \text{XPath}_{z} \) in up-down normal form such that

- \( \varphi^{r+s+2} \equiv \varphi \);
- \( vd(\varphi^{r+s+2}) = (r,s) \); and
- \( nd(\varphi^{r+s+2}) \leq k \cdot (r + s + 2) \).
3.2.2 Finite index

Contrary to the case of XPath\(_i\) (cf., Proposition 3.1), the logical equivalence relation restricted to XPath\(_i\)-formulas of bounded vertical depth has infinitely many equivalence classes.

**Proposition 3.9.** If \( r + s \geq 2 \) then \( \equiv^{\uparrow}\_\,_{s,k} \) has infinite index.

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

**Proposition 3.10.** \( \equiv^{\uparrow}\_\,_{s,k} \) has finite index.

**Corollary 3.11.** \( \{ T', u' \mid T, u \equiv^{\uparrow}\_\,_{s,k} T', u' \} \) is definable by an \((r, s, k)\)-XPath\(_i\)-formula.

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

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

\[
\equiv^{\uparrow}\_\,_{r,s,k} \subseteq \equiv^{\downarrow}\_\,_{r,s,k} \subseteq \equiv^{\downarrow}\_\,_{r,s,k} - (r+s+2) \cdot \]

Let \( T \) and \( T' \) be two data-trees. We say that \( u \in T \) and \( u' \in T' \) are **bisimilar for XPath\(_i\)** (notation: \( T, u \equiv^{\uparrow}\_\,_{s,k} T', u' \)) 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 \( \text{label}(x) = \text{label}(x') \).
- **Zig (Figure 3):** If \( xZx', y^n_x \) and \( y^m_z \) then there are \( y', z' \in T' \) such that \( y^\gamma y'z', y^\delta z', \text{data}(z) = \text{data}(z') \) and \( zZz' \).
- **Zag:** If \( xZx', y^\gamma y'z' \) then there are \( y, z \in T \) such that \( y^\gamma yx, y^\delta z \), \( \text{data}(z) = \text{data}(z') \) and \( zZz' \).

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

We say that \( u \in T \) and \( u' \in T' \) are **XPath\(_i\)-bisimilar** (notation: \( T, u \equiv^{\uparrow}\_\, T', u' \)) if there is a family of relations \( (Z^{\uparrow}_{r,s,k})_{r+s\leq r+s+k \leq k} \) in \( T \times T' \) such that \( uZ^\uparrow_{r,s,k}u' \) and for all \( r \leq s \leq r+s \leq k \), \( x \in T \) and \( x' \in T' \) we have that the following conditions hold.

- **Harmony:** If \( xZ^\uparrow_{r,s,k}x' \) then \( \text{label}(x) = \text{label}(x') \).
- **Zig:** If \( xZ^\uparrow_{r,s,k}x', y^n_x \) and \( y^m_z \) then there are \( y', z' \in T' \) such that \( y^\gamma y'z', y^\delta z' \), and the following hold
  1. \( \text{data}(z) = \text{data}(x) \leftrightarrow \text{data}(z') = \text{data}(x) \).
  2. if \( \hat{k} > 0 \), \( zZ^{\hat{k}-1}_{r',s',z'} \) for \( r' = r+n-m, s' = \hat{s} - n + m \).
- **Zag:** If \( xZ^\uparrow_{r,s,k}x', y^n_x \) and \( y^m_z \) then it follows that \( yZ^{\hat{k}-1}_{r',s',z'} \) for \( r' = r+n, s' = \hat{s} - n \). The same occurs with \( Z \) instead of \( Z^{\hat{k}} \) for the case of bisimilarity.

**Observation 3.12.** If \( xZ^\uparrow_{r,s,k}x', y^n_x \) and \( y^m_z \) then it follows that \( yZ^{r+k-1}_{n+m, y^n_z} \) for \( r' = r+n, s' = \hat{s} - n \). The same occurs with \( Z \) instead of \( Z^{\hat{k}} \) for the case of bisimilarity.

For a data tree \( T \) and \( u \in T \), let \( \mathcal{T}[u] \) denote the subtree of \( T \) induced by \( \{ v \in T \mid (\exists m \leq s) (\exists n \leq r+m) (\exists w \in T) w^{-m}u \land w^{-n}v \} \).

3.2.4 Equivalence and bisimulation

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

**Theorem 3.13.**

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

**Corollary 3.14.** \( \equiv^{\downarrow}\_\,_{r,s,k} \) has finite index.

4. SIMULATION

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

We say that an XPath\(_i\)-formula is **positive** if it contains no negation \( \neg \) and no inequality data tests \( (\alpha \neq \beta) \). For \( \mathcal{L} \) one of XPath\(_i\), XPath\(_j\), XPath\(_i\)-bisim., or XPath\(_j\)-bisim., we write \( \mathcal{L}^+ \) for the positive fragment of \( \mathcal{L} \).

A **simulation for XPath\(_i\)** [resp. XPath\(_j\)] is simply a bisimulation from which the Zig 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\(_i\) (notation: \( T, u \rightarrow^+ T', u' \)) 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 $xx'x'$ then $\text{label}(x) = \text{label}(x')$.

• Zig: If $xx', x_m \rightarrow v$ and $x_m \rightarrow w$, then there are $v', w' \in T'$ such that $x' \rightarrow v', x' \rightarrow w'$ and
  1. $\text{data}(v) = \text{data}(w) \Rightarrow \text{data}(v') = \text{data}(w')$,
  2. $(\rightarrow v') Z (\rightarrow v')$ for all $0 \leq i < n$, and
  3. $(\rightarrow w') Z (\rightarrow w')$ for all $0 \leq i < m$.

$u \in T$ is similar to $u' \in T'$ for XPath $^+$ (notation: $T, u \rightarrow v$ $T', u'$) 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 $xx'$ then $\text{label}(x) = \text{label}(x')$.

• Zig: If $xx', y \rightarrow z$ and $y \rightarrow z'$, then there are $v', w' \in T'$ such that $x' \rightarrow v', y \rightarrow z$, and if $\text{data}(z) = \text{data}(x) \Rightarrow \text{data}(z') = \text{data}(x')$.

Relations $\rightarrow_3$ and $\rightarrow_4$, $s, k$, are defined accordingly. We define one-way (non-symmetric) logical implication between models as follows. We write $T, u \models^1 T', u'$ for

$$(\forall \phi \in \text{XPath}^+)(\exists \psi)(\exists \bar{x})(\exists \bar{y})(\forall \bar{z})(\exists \bar{w})(\exists \bar{v})(\forall \bar{u})(\forall \bar{z}') (T, \bar{x}, \bar{y}, \bar{z}) = (T', \bar{u}, \bar{z}')$$

Define $\models^1$, $\models^2$, and $\models^3, s, k$, in an analogous way for $\ell$-XPath $^+$, XPath $^+$, $(r, s, k)$-XPath $^+$, respectively. As for bisimulation, we have that $\models^1$ coincides with $\models_\bar{y}$.

**Theorem 4.1.**

1. Let $\dagger \in \{\dagger, \dagger^*, \dagger^+\}$. $T, u \rightarrow \dagger T', u'$ implies $T, u \models^1 T', u'$.
   The converse holds when $T'$ is finitely branching.

2. $T, u \rightarrow_{\ell} T', u'$ if $T, u \models^1 T', u'$.

3. $T, u \rightarrow_{r, s, k}(r+s+2) T', u'$ implies $T, u \models^1 T', u'$.

4. $T, u \rightarrow_{r, s, k} T', u'$ implies $T, u \models^1 T', u'$.

**Theorem 4.3.**

1. $\phi \in \text{XPath}^+$ is $\rightarrow_3$-invariant [resp. $\rightarrow_4$-invariant] if it is equivalent to a formula of XPath $^+$ [resp. $\ell$-XPath $^+$].

2. $\phi \in \text{XPath}^+$ is $\rightarrow_3$-invariant if it is equivalent to a formula of XPath $^+$.

3. If $\phi \in \text{XPath}^+_s$ is $\rightarrow_3$-invariant then it is equivalent to a formula of $(r, s, k)$-XPath $^+$.

4. If $\phi \in \text{XPath}^+_s$ is equivalent to a formula of $(r, s, k)$-XPath $^+$ then $\phi$ is $\rightarrow_3$-invariant, for $k = k'(r+s+2)$.

**5. ADDING TRANSITIVITY**

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

Let $\equiv_{\dagger}$ and $\equiv_{\uparrow}$ be the logical equivalence relation for XPath $\dagger$ and XPath $\uparrow$, respectively, and let $\models_{\dagger}$ and $\models_{\uparrow}$ be the logical implication for XPath $\dagger$ and XPath $\uparrow$, respectively.

**Theorem 5.1.** Let $\dagger \in \{\dagger, \dagger^*, \dagger^+\}$.

1. $T, u \equiv_{\dagger} T', u'$ implies $T, u \models_{\dagger} T', u'$.
   The converse also holds when $T'$ is finitely branching.

2. $T, u \models_{\dagger} T', u'$ implies $T, u \equiv_{\dagger} T', u'$.
   The converse also holds when $T'$ is finitely branching.

**6. CHARACTERIZATION**

In §6.1 we show that there is a truth-preserving translation from XPath to first-order logic over an appropriate signature. In §6.2 we characterize XPath $^+$, as the fragment of first-order logic $\rightarrow_3$-invariant over data trees. In §6.3 we show that this result fails for XPath $^+$ in general, but a weaker result can still be proved.

**6.1 Translating to first-order logic**

We say that an XPath $^+$-path expression $\phi$ is in simple normal form if it is of the form

$$[\phi_0] \alpha_1[\phi_1] \alpha_2 \cdots \alpha_n[\phi_n],$$

for $n \geq 0$, $\phi_i \in \text{XPath}^+$, and $\alpha_i \in \{\dagger, \dagger^*, \dagger^+\}$. For any XPath $^+$-[resp. XPath $^*$]-path expression $\alpha$ there is an equivalent XPath $^+$-[resp. XPath $^*$]-path expression $\alpha'$ in simple normal form. Further, $\alpha'$ can be computed in polynomial time from $\alpha$.

We say that an XPath $^+$-formula $\phi$ is in simple normal form if each path expression $\alpha$ occurring in $\phi$ is in simple normal form.

Fix the signature $\sigma$ with binary relations $\sim$ and $\approx$, and a unary predicate $P_a$ for each $\alpha \in A$. Any data tree $T$ can be seen as a first-order $\sigma$-structure such that

$$\sim_T = \{(x, y) \in T^2 \mid y \text{ is a child of } x\};$$

$$\approx_T = \{(x, y) \in T^2 \mid \text{data}(x) = \text{data}(y)\};$$

$$P_a^T = \{x \in T \mid \text{label}(x) = a\}.$$
6.3 Vertical XPath

The analog of Theorem 6.1 fails for XPath\(_E^L\):

**Lemma 6.3.** The FO(\(\sigma\))-formula

\[
(\exists x) \, P_\sigma(x)
\]

is \(\mathcal{L}^{-}\)-invariant though not logically equivalent over \([finite]\) data-trees to any XPath\(_E^L\)-formula.

Hence XPath\(_E^L\) is not the fragment of FO(\(\sigma\)) which is \(\mathcal{L}^{-}\)-invariant over \([finite]\) data-trees. However, the following proposition still holds for the case of XPath\(_E^L\):

**Proposition 6.4.** Let \(k = k \cdot (r + s + 2)\). If \(\varphi(x) \in FO(\sigma)\) is \(\mathcal{L}^{-}\)-invariant over \([finite]\) data-trees, then there is \(\psi \in (r, s, k)\)-XPath\(_E^L\) such that Tr\(_E\)(\(\psi\)) is logically equivalent to \(\varphi\) over \([finite]\) data-trees.

Notice that the counterexample in Lemma 6.3 is an un-restricted, existential formula. One may wonder if it might be possible to extend the expressive power of XPath\(_E^L\) to account for unrestricted quantification. The natural candidate would be the modal operator \(E\) (usually known as the existential modality) which, intuitively, let us express that there is some node in the model where a formula holds. But even with the additional expressive power provided by \(E\) the analog of Theorem 6.1 fails. Formally, consider the logic XPath\(_E^L\), which results from adding the operator \(E\) to XPath\(_E^L\) with the following semantics: \([E\psi]^T = T\) if \(\psi^T \neq 0\) and \([E\psi]^T = \emptyset\) otherwise.

The following lemma shows a counterexample to the analog of Theorem 6.1, showing that XPath\(_E^L\) is not the fragment of FO(\(\sigma\)) \(\mathcal{L}^{-}\)-invariant over \([finite]\) data-trees.
Lemma 6.5. The FO(σ)-formula

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

is \(\leftrightarrow\)-invariant though not logically equivalent over [finite] data-trees to any XPath\(\subseteq\) formula.

7. APPLICATIONS

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

7.1 Expressiveness hierarchies

Define \(\equiv^{1}_{\ell,k}\) as the equivalence \(\equiv^1\) restricted to formulas of nesting depth at most \(k\), that is, \(\mathcal{T}, u \equiv^{1}_{\ell,k} \mathcal{T}', u'\) iff for all \(\varphi \in \text{XPath}^\subseteq\) such that \(\text{dil}(\varphi) \leq \ell\) and \(\text{nd}(\varphi) \leq k\) 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 \(u \in \mathcal{T}\) and \(u' \in \mathcal{T}'\) are \((\ell, k)\)-bisimilar for XPath\(\subseteq\) (notation: \(\mathcal{T}, u \equiv^{1}_{\ell,k} \mathcal{T}', u'\)) if there is a family of relations \((Z_{j,t})_{j \leq t \leq k}\) in \(\mathcal{T} \times \mathcal{T}'\) such that \(uZ_{j,t}u'\) and for all \(j \leq t \leq k\), \(x \in \mathcal{T}\) and \(x' \in \mathcal{T}'\) we have

- **Harmony**: If \(xZ_{j,t}x'\) then \(\text{label}(x) = \text{label}(x')\).
- **Zig**: If \(xZ_{j,t}x', x \rightarrow v\) and \(x' \rightarrow w\) with \(n, m \leq j\) then there are \(v', w' \in \mathcal{T}'\) such that \(x \rightarrow v'\), \(x' \rightarrow w'\) and
  1. \(\text{data}(v) = \text{data}(w) \Rightarrow \text{data}(v') = \text{data}(w')\),
  2. if \(t > 0\), \((\rightarrow v)Z_{j-n+i,t-1} (\rightarrow v')\) for all \(0 < i < n\), and
  3. if \(t > 0\), \((\rightarrow w)Z_{j-m+i,t-1} (\rightarrow w')\) for all \(0 < i < m\).
- **Zag**: If \(xZ_{j,t}x', x' \leftarrow v'\) and \(x \leftarrow w\) with \(n, m \leq j\) then there are \(v, w \in \mathcal{T}\) such that \(x \rightarrow v\), \(x \rightarrow w\) and items 1, 2 and 3 above are verified.

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

**Proposition 7.1.** \(\mathcal{T}, u \equiv^{1}_{\ell,k} \mathcal{T}', u'\) iff \(\mathcal{T}, u \equiv^{\ell,k}_{\ell,k} \mathcal{T}', u'\).

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

**Theorem 7.2.** For all \(\ell, k \geq 0, i \geq 1\),

\[
\equiv^1_{\ell,0} \supseteq \equiv^1_{\ell,1} \supseteq \cdots \supseteq \equiv^1_{\ell,i} = \equiv^1_{\ell,\ell+i}, \quad \text{and} \\
\equiv^1_{\ell,k} \supseteq \equiv^1_{\ell,i+k}.
\]

7.2 Safe operations on models

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

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

**Proposition 7.3.** XPath\(\subseteq\) is closed under subtree replication, i.e., if \(\mathcal{T}'\) is a subtree replication of \(\mathcal{T}\), and \(u \in \mathcal{T}\) then \(\mathcal{T}', u \equiv^{\mathcal{T}'\subseteq} \mathcal{T}, u\).

**Proof.** Suppose that \(x \in \mathcal{T}\) 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 \(T_{x}\) 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 \(T_{x}\). 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 \(T_{x}\).

By Theorem 5.1, it suffices to verify that \(\mathcal{T}, u \equiv \mathcal{T}', u\) via \(Z \subseteq \mathcal{T} \times \mathcal{T}'\) defined by:

\[
Z = \{(y, y) \mid y \in \mathcal{T}\} \cup \{(v, v_{x}) \mid v \in X\}
\]

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

**7.3 Non-expressivity results**

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

**Proposition 7.4.**
1. key(a) is not expressible in XPath\textsuperscript{++}.

2. fk(a, b) is expressible in XPath\textsuperscript{++} but it is not expressible in XPath\textsuperscript{++} or XPath\textsuperscript{++}.

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

For the second item, it is easy to see that fk(a, b) is expressible with the formula $\neg(\lceil a \rceil \land \neg(\lceil b \rceil \land [b]))$. However, this property cannot be expressed in XPath\textsuperscript{++} because the models T and T′ in Figure 8 are bisimilar for XPath\textsuperscript{+} via Z, depicted as dotted lines. Since T, x satisfies fk(a, b) but T′, x′ does not, from Theorem 5.1 it follows that fk(a, b) is not expressible in XPath\textsuperscript{++}.

Finally, suppose there exists $\psi \in$ XPath\textsuperscript{++} expressing fk(a, b). Since $\mathcal{T}$ is a substructure of $\mathcal{T}'$, we have $\mathcal{T}, x \models \psi$ if and only if $\mathcal{T}', x' \models \psi$. By Lemma 4.2 and the fact that $\mathcal{T}, x \models \psi$, we have $\mathcal{T}', x' \models \psi$, which is a contradiction.

Let dist\textsubscript{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 7.5.**

1. dist\textsubscript{3} is expressible in XPath\textsuperscript{+};

2. dist\textsubscript{3} is not expressible in XPath\textsuperscript{++};

3. neither dist\textsubscript{3} nor its complement can be expressed in XPath\textsuperscript{++}.

Proof. For 1, one can check that $\mathcal{T}, x \models \psi$ iff $\mathcal{T}, x$ satisfies dist\textsubscript{3} for $\psi = (\varepsilon \neq \cld(\varepsilon \neq \cld))$.

Let us see 2. Consider the data trees $\mathcal{T}, x$ and $\mathcal{T}', x'$ depicted in Figure 9. It is straightforward that $\mathcal{T}, x$ satisfies dist\textsubscript{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 $\mathcal{T}, x \models \psi$ via $Z \subseteq T \times T'$ defined by

$Z = \{ (u, u') \mid h(u) = h(u') \land \text{data}(u) = \text{data}(u') \} \cup \{ (v, v'_1), (v, v'_2) \},$

where $h(y)$ denotes the height of y, i.e., the distance from y to the root of the corresponding tree (Z is depicted as dotted lines in Figure 9). Since $\mathcal{T}, x$ satisfies dist\textsubscript{3} but $\mathcal{T}', x'$ does not, from Theorem 5.1 it follows that dist\textsubscript{3} is not expressible in XPath\textsuperscript{++}.

For 3, one can verify that $\mathcal{T}, x \models \psi$ via Z as defined above. If dist\textsubscript{3} were definable in XPath\textsuperscript{++} then $\mathcal{T}, x \models \psi$, and the fact that $\mathcal{T}, x \models \psi$, by Theorem 5.1(2) we would have $\mathcal{T}', x' \models \psi$, and this is a contradiction.

Let dist\textsubscript{3} denote the complement of dist\textsubscript{3}, i.e., $\overline{\text{dist}}_3(x) \text{iff for all } y, z \text{ so that } x \rightarrow y \rightarrow z, \text{we have that } x, y, z \text{ do not have pairwise distinct data values.}$ Now $\mathcal{T}', x' \models \overline{\text{dist}}_3$ and $\mathcal{T}, x \not\models \overline{\text{dist}}_3$, and $\mathcal{T}, x$ does not. Since $\mathcal{T}'$ is a substructure of $\mathcal{T}$, by an argument analog to the one used in the proof of Proposition 7.4-2, we conclude that dist\textsubscript{3} is not expressible in XPath\textsuperscript{++}.

8. DISCUSSION

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

In Section 7 we showed 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 $T_1$ we want to find a data tree $T_2$, as small as possible, so that $T_1$ and $T_2$ are bisimilar for some fragment $\mathcal{L}$ of XPath. Since $\mathcal{L}$ cannot distinguish between $T_1$ and $T_2$, we can use $T_2$ as representative of $T_1$ 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 $T$ by partitioning it in terms of its coarsest auto-bisimulation. We plan to design and implement algorithms for data tree minimization using bisimulation and investigate their computational complexity.

References