Some Modal Aspects of XPath

Balder ten Cate[1,2]

Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ, Amsterdam, The Netherlands

Gaëlle Fontaine[3]

ILLC, University of Amsterdam
Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands

Tadeusz Litak[3]

Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ, Amsterdam, The Netherlands

Abstract

This paper provides several examples of how known results in modal logic can be applied to the XML document navigation language XPath. The applications concern complete axiomatizations and expressive power.

Keywords: XPath, XML, modal logic, finite trees

XML is a standard for storage and exchange of data on the Internet. It is tailored towards hierarchically structured data, and its basic data structure is that of a (finite) *sibling-ordered tree*: a tree in which the children of each node are linearly ordered. XPath is a language used for retrieving elements from XML documents.

As has been observed by several people, XPath is closely related to modal logic. In this paper, we illustrate the connection by showing how a number of (old and new) results on XPath fragments can be derived from known results in modal logic. These results concern axiomatizations and expressivity of XPath fragments, both of which are important topics in database theory. For instance, complete axiomatizations for XPath fragments can be understood as complete sets of *rewrite rules*. Identifying effective rules for rewriting expressions to simpler but equivalent ones is important for *query optimization*.

[1] We are grateful to Maarten Marx for his advice, discussions and ideas. The authors are supported by the Netherlands Organization for Scientific Research (NWO) under grants 639.021.508, 639.073.501 and 680-50-0613, respectively.

[2] Email: balder.tencate@uva.nl

[3] Email: {gfontain, litak}@science.uva.nl

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

In Section 1, we introduce the language of Core XPath and its semantics. Section 2 shows how completeness results for the modal logic GL and the "logic of finite trees" of Blackburn et al. [3] can be transferred to the XPath setting. In Section 3, we derive characterizations of the expressive power of XPath fragments from known results in modal logic (in particular, the Janin-Walukiewicz characterization of the modal μ-calculus, the De Jongh-Sambin theorem for GL, and the expressive completeness of *Boolean Modal Logic* for FO^2).

1 Preliminaries

1.1 XML trees

If we abstract away from atomic data attached to the individual elements, then an XML document is nothing but a finite node labeled sibling ordered tree. More formally, given a countably infinite set prop of node labels, we define an *XML tree* to be a structure $T = (D, R_{\downarrow}, R_{\rightarrow}, L)$, where

- $(N, R_{\downarrow})$ is a finite tree (with N the set of nodes and $R_{\downarrow}$ the child relation),
- $R_{\rightarrow}$ is the successor relation of some linear ordering between siblings in the tree (in particular, for $R_{\rightarrow^+}$ the transitive closure of $R_{\rightarrow}$, we have that for any two distinct siblings x, y, either $xR_{\rightarrow^+}y$ or $yR_{\rightarrow^+}x$), and
- $L : \mathsf{prop} \to \wp(N)$ labels the nodes with elements of prop.

We denote by $R_{\downarrow^+}$ and $R_{\rightarrow^+}$ the transitive closures of $R_{\downarrow}$ and $R_{\rightarrow}$, i.e., the *descendant* and *following-sibling* relations. $R_{\leftarrow}$, $R_{\leftarrow^+}$, $R_{\uparrow}$ and $R_{\uparrow^+}$ are the converses of, respectively, $R_{\rightarrow}$, $R_{\rightarrow^+}$, $R_{\downarrow}$ and $R_{\downarrow^+}$. The elements of prop correspond to XML tags, such as, in the case of html, `body`, `p`, `it`, ... It is customary to require that each node satisfies precise one tag. In order to simplify the presentation, we do not make this restriction except for the discussion following Corollary 2.2 below.

1.2 Core XPath, the navigational core of XPath 1.0

XPath 1.0 was released as a W3C recommendation in 1999. Its technical specifications are approximately 30 pages long. In order to be able to study its properties from a logical point of view, Gottlob et al. [10] introduced Core XPath. It captures the part of XPath that is concerned with navigation through the tree, while excluding for instance all arithmetical and string operations. Incidentally, in January 2007, XPath 2.0 was released as a W3C recommendation. It is much more expressive than its predecessor. A "navigational core" of XPath 2.0 was identified in [7]. However, we will not say much about it in this paper.

We follow the definition of Core XPath given in [16], which differs slightly from the one in [10] as it includes the non-transitive sibling axes $\leftarrow$ and $\rightarrow$. This has no essential influence on what we discuss below.

There are two main types of expressions in Core XPath: *path expressions* and *node expressions*. Path expressions describe ways of travelling through the tree, while node expressions are used to describe properties of nodes. More precisely, the

Table 1
Semantics of Core XPath 1.0

$$
\begin{array}{ll}
[\![.]\!]_{\mathsf{PExpr}} & := \{(x,x) \mid x \in D\} \\
[\![\mathsf{a}]\!]_{\mathsf{PExpr}} & := R_{\mathsf{a}} \text{ for all } \mathsf{a} \in \mathsf{Axis} \text{ other than ``.''} \\
[\![A/B]\!]_{\mathsf{PExpr}} & := \{(a,x) \mid \exists z \text{ such that } (x,z) \in [\![A]\!]_{\mathsf{PExpr}} \text{ and } (z,y) \in [\![B]\!]_{\mathsf{PExpr}}\} \\
[\![A \cup B]\!]_{\mathsf{PExpr}} & := [\![A]\!]_{\mathsf{PExpr}} \cup [\![B]\!]_{\mathsf{PExpr}} \\
[\![A\,[\phi]]\!]_{\mathsf{PExpr}} & := \{(x,y) \mid (x,y) \in [\![A]\!]_{\mathsf{PExpr}} \text{ and } y \in [\![\phi]\!]_{\mathsf{NExpr}}\} \\
 & \\
[\![v]\!]_{\mathsf{NExpr}} & := \{x \mid x \in L(v)\} \\
[\![\langle \mathsf{PathEx} \rangle]\!]_{\mathsf{NExpr}} & := \{x \mid \exists y.(x,y) \in [\![\mathsf{PathEx}]\!]_{\mathsf{PExpr}}\} \\
[\![\neg\phi]\!]_{\mathsf{NExpr}} & := \{x \mid x \notin [\![\phi]\!]_{\mathsf{NExpr}}\} \\
[\![\phi_1 \vee \phi_2]\!]_{\mathsf{NExpr}} & := [\![\phi_1]\!]_{\mathsf{NExpr}} \cup [\![\phi_2]\!]_{\mathsf{NExpr}}
\end{array}
$$

syntax of Core XPath is defined as follows:

$$
\begin{array}{rl}
\mathsf{Axis} := & . \mid \downarrow \mid \leftarrow \mid \uparrow \mid \rightarrow \mid \downarrow^+ \mid \leftarrow^+ \mid \uparrow^+ \mid \rightarrow^+ \\
\mathsf{PathEx} := & \mathsf{a} \mid \mathsf{PathEx}/\mathsf{PathEx} \mid \mathsf{PathEx} \cup \mathsf{PathEx} \mid \mathsf{PathEx}\,[\mathsf{NodeEx}] \quad (\mathsf{a} \in \mathsf{Axis}) \\
\mathsf{NodeEx} := & v \mid \langle \mathsf{PathEx} \rangle \mid \neg \mathsf{NodeEx} \mid \mathsf{NodeEx} \vee \mathsf{NodeEx} \quad (v \in \mathsf{prop})
\end{array}
$$

The axes correspond to basic moves one can make in the tree. Just like in Unix directory paths, "." corresponds to staying at the current node. The axes can be composed into path expressions by using *composition* ($/$), *union* ($\cup$), and *node tests* ($\cdot[\cdot]$); the symbol for composition again originates in Unix directory path notation. The node expression $\langle \mathsf{PathEx} \rangle$ expresses that the current node belongs to the domain of the binary relation defined by PathEx. We abbreviate $\neg(\neg\phi \vee \neg\psi)$ as $\phi \wedge \psi$ and $.\,[\neg\,\langle . \rangle]$ as $\bot$. and use the following syntactic conventions: v, v' range over elements of prop, Greek letters $\phi, \psi \ldots$ over elements of NodeEx, Roman capitals $A, B, C, \ldots$ over elements of PathEx and $P, Q, R \ldots$ over arbitrary elements of NodeEx$\cup$PathEx.

The semantics of Core XPath expressions is given by two functions, $[\![\cdot]\!]_{\mathsf{PExpr}}$ and $[\![\cdot]\!]_{\mathsf{NExpr}}$. For any path expression A and XML tree T, $[\![A]\!]^T_{\mathsf{PExpr}}$ denotes a binary relation over the domain of T, and for any node expression ϕ and XML tree T, $[\![\phi]\!]^T_{\mathsf{NExpr}}$ denotes a subset of the domain of T. The definition of $[\![\cdot]\!]_{\mathsf{PExpr}}$ and $[\![\cdot]\!]_{\mathsf{NExpr}}$ is given in Table 1, where, for readability, the superscript T is left out.

We will denote by Core XPath($\downarrow^+$) the fragment of Core XPath in which the only allowed axes are $\downarrow^+$ and ".".

1.3 Connections with modal logic

The syntax of Core XPath is similar to that of two-sorted modal formalisms such as PDL, node expressions being counterparts of PDL formulas and path expressions corresponding to PDL programs. A conceptual difference is that, in XPath, path expressions are the primary object of interest, while in PDL, it is usually the formulas that are considered primary. A technical difference is of course the more restrictive intended semantics (finite sibling-ordered trees as opposed to arbitrary Kripke structures). Nevertheless, if we restrict the attention to node expressions, under the intended semantics, Core XPath and its fragments are closely related to certain modal logics. In particular, Core XPath($\downarrow^+$) corresponds to the modal logic

Table 2
Axioms for Core XPath($\downarrow^+$)

Path Axiom Schemes for Idempotent Semirings			
ISAx1	$(A \cup B) \cup C$	$\equiv$	$A \cup (B \cup C)$
ISAx2	$A \cup B$	$\equiv$	$B \cup A$
ISAx3	$A \cup A$	$\equiv$	A
ISAx4	$A/(B/C)$	$\equiv$	$(A/B)/C$
ISAx5	$./A$	$\equiv$	A
	$A/.$	$\equiv$	A
ISAx6	$A/(B \cup C)$	$\equiv$	$A/B \cup A/C$
	$(A \cup B)/C$	$\equiv$	$A/C \cup B/C$
Path Axiom Schemes for Predicates			
PrAx1	$A\,[\neg\,\langle B\rangle]\,/B$	$\equiv$	$\bot$
PrAx2	$(A/B)\,[\phi]$	$\equiv$	$A/B\,[\phi]$
PrAx3	$.\,[\langle . \rangle]$	$\equiv$	$.$
PrAx4	$A\,[\phi \vee \psi]$	$\equiv$	$A\,[\phi] \cup A\,[\psi]$
Node Axiom Schemes			
NdAx1	ϕ	$\equiv$	$\neg(\neg\phi \vee \psi) \vee \neg(\neg\phi \vee \neg\psi)$
NdAx2	$\langle . \,[\phi]\rangle$	$\equiv$	ϕ
NdAx3	$\langle A \cup B\rangle$	$\equiv$	$\langle A\rangle \vee \langle B\rangle$
NdAx4	$\langle A/B\rangle$	$\equiv$	$\langle A\,[\langle B\rangle]\rangle$
NdAx5	$\langle\downarrow^+ [\phi]\rangle$	$\equiv$	$\langle\downarrow^+ [\phi \wedge \neg\langle\downarrow^+ [\phi]\rangle]\rangle$

GL, whereas Core XPath itself is closely related to the "modal logic of finite trees" of Blackburn et al. [3]. Throughout this paper, we will exploit these connections.

2 Axiomatization

In this section, we discuss *complete equational axiomatizations* for Core XPath and Core XPath($\downarrow^+$): finite sets of equivalences such that every other valid equivalence can be derived from them using the standard rules of equational reasoning. For both languages, such complete axiomatizations can be derived from known results in modal logic, at least for *node expressions*. The axiomatizations can be extended to *path expressions* using a *non-standard inference rule*. Whether such rules can be avoided, we leave as an open question.

2.1 Core XPath($\downarrow^+$)

Recall that GL is the modal logic of the class of all finite trees, where the modal operator is interpreted by the (strict) descendant relation. Using the known axioms of GL, it is easy to obtain an axiomatization for Core XPath($\downarrow^+$). The axioms are given in Table 2. Those from the first group are familiar from arrow logic and PDL: their models are called *idempotent semirings*. The second group of axioms governs the behavior of *predicates* (i.e., of the $\cdot[\cdot]$ operator, the analogue of *tests* in PDL). Together with NdAx1, these two groups of axioms define the *dynamic relation algebras* of [11]. Incidentally, the choice of NdAx1 as an axiom forcing the boolean character of node expressions is justified by a famous result in automated theorem proving [18], which implies that NdAx1 together with (node equivalences derivable from) ISAx1 and ISAx2 allow to derive all equations of Boolean algebra.

Table 3
Translation of Core XPath($\downarrow^+$) node expressions into simple node expressions

v^{s}	:=	v	$.^{\mathsf{s}}(\phi)$	:=	ϕ
$(\neg\phi)^{\mathsf{s}}$	:=	$\neg\phi^{\mathsf{s}}$	$\downarrow^{+\mathsf{s}}(\phi)$	:=	$\langle\downarrow^+[\phi]\rangle$
$(\phi\vee\psi)^{\mathsf{s}}$	:=	$\phi^{\mathsf{s}}\vee\psi^{\mathsf{s}}$	$(A\cup B)^{\mathsf{s}}(\phi)$	:=	$A^{\mathsf{s}}(\phi)\vee B^{\mathsf{s}}(\phi)$
$\langle A\rangle^{\mathsf{s}}$	:=	$A^{\mathsf{s}}(\langle.\rangle)$	$(A[\psi])^{\mathsf{s}}(\phi)$	:=	$A^{\mathsf{s}}(\psi^{\mathsf{s}}\wedge\phi)$
			$(A/B)^{\mathsf{s}}(\phi)$	:=	$A^{\mathsf{s}}(B^{\mathsf{s}}(\phi))$

Of course, less impressive and less concise boolean axiom(s) would also do.

The axioms NdAx4, NdAx2 and NdAx3 are counterparts of PrAx2, PrAx3 and PrAx4, respectively. This slight redundancy is the price we have to pay for working in a two-sorted signature. NdAx5, finally, is just a notational variant of the Löb axiom from GL [2,22].

For P, Q both path expressions or both node expressions, we say that $P\equiv Q$ is *derivable* from the axioms in Table 2 if it can be obtained from them using the standard rules of equational logic:

- $P\equiv P$
- If $P\equiv Q$ then $Q\equiv P$
- If $P\equiv Q$ and $Q\equiv R$, then $P\equiv R$
- If $P\equiv Q$ and R' is obtained from R by replacing some occurrences of P by Q, then $R\equiv R'$

Theorem 2.1 *All axioms in Table 2 are valid, and every valid equivalence between* Core XPath($\downarrow^+$) *node expressions is derivable from the axioms in Table 2.*

Proof. The soundness claim is left as an exercise to the reader. The completeness proof is based on the completeness of GL with respect to the class of finite trees with the descendant relation. First, we identify a subclass of the Core XPath($\downarrow^+$) node expression, which we call *simple node expressions*, that can be identified with the modal formulas of the GL language. The are generated by the following recursive definition:

$$\mathsf{siNode} := \langle.\rangle \mid v \mid \left\langle\downarrow^+[\mathsf{siNode}]\right\rangle \mid \neg\mathsf{siNode} \mid \mathsf{siNode}\vee\mathsf{siNode}$$

where $v\in\mathsf{prop}$. These expressions are indeed isomorphic variants of the formulas in the basic modal language. For instance, our $\left\langle\downarrow^+[v]\right\rangle$ is just the modal formula $\Diamond v$ and $\langle.\rangle$ is the truth constant.

The first step in the proof is to show that every Core XPath($\downarrow^+$) node expression is provably equivalent to a simple node expression. Hereto, we provide a translation $(\cdot)^{\mathsf{s}} : \mathsf{NodeEx}\to\mathsf{siNode}$ which is constant for elements of siNode. This mapping uses an auxiliary mapping $(\cdot)^{\mathsf{s}} : \mathsf{PathEx}\to(\mathsf{siNode}\to\mathsf{siNode})$ assigning to every path expression a unary function defined on simple node expressions. As domains of both mappings are disjoint, we use the same symbol with no risk of confusion. Their definitions are given in Table 3. It is easy to show from the axioms that this translation always yields a provably equivalent node expression.

Next, we have that, for every two equivalent simple node expressions ϕ,ψ, their equivalence is derivable from the axioms in Table 2. This follows from the fact that

all axioms of GL (when written as simple node expressions) are derivable from our axioms, something the reader may easily verify. Putting the two together, we obtain that every valid equivalence between Core XPath($\downarrow^+$) is derivable. □

The above completeness result can be lifted to path expressions by introducing an extra inference rule with syntactic side conditions, which we call the *Name* rule, and which is closely related to the *separability rule* in [19].

(Name) If $\langle A\,[v]\rangle \equiv \langle B\,[v]\rangle$ and v does not occur in A and B, then $A \equiv B$.

Corollary 2.2 *For every two equivalent* Core XPath($\downarrow^+$) *path expressions* α, β*, their equivalence is derivable from the axioms in Table 2 using the standard rules of equational logic plus the* Name *rule.*

Proof. Suppose $\alpha \equiv \beta$. Pick any v not occurring in α and β. Then the node expressions $\langle\alpha[v]\rangle$ and $\langle\beta[v]\rangle$ are also equivalent. Hence, by Theorem 2.1, their equivalence is derivable from the axioms in Table 2. A single application of the *Name* rule now yields a derivation of $\alpha \equiv \beta$. □

Recall that in "real" XML trees, unlike the ones that we are using, each node satisfies precisely one label. In order to obtain completeness with respect to this more restricted class of structures, one could add a further axiom scheme:

$$v \wedge v' \equiv \bot \qquad \text{for } v, v' \in \mathsf{prop} \text{ distinct}$$

It is not hard to show that, adding this to the axioms in Table 2, we indeed obtain completeness for node expressions under the intended semantics. However, a slight complication now arises, namely that the *Name* rule is no longer sound: its soundness relies on the fact that nodes can satisfy multiple labels. This problem can be fixed, but at the cost of further complicating the formulation of the rule. It is one of reasons why elimination of *Name* would be a desirable step—see the discussion below in Section 2.3 for other reasons. Another solution is to leave the axiomatization unchanged and to pre-process the input: given any two expression A, B, collect all labels $v_1 ... v_n \in \mathsf{prop}$ occurring in A and/or B, and then uniformly replace each v_i by $v_i \wedge \bigwedge_{j\leq i} \neg v_j$. The resulting A' and B' are equivalent iff the original ones were equivalent on uni-labeled trees.

2.2 Full Core XPath

We will now extend the axiomatization to full Core XPath. This time, we exploit another known result from modal logic, namely the axiomatization of the "logic of finite trees" in [3]. We will skip over most of the details, see [5] for more information.

The new group of axioms we add are *tree axioms*, analogous to those used in [3]. They are given in Table 4. TreeAx1 is a Kleene algebra axiom [13,14], well-known to anyone familiar with PDL, denoted as *Mix* in [9]. It simply forces a^+ to be a (not necessarily least) fixed-point containing a. TreeAx2, TreeAx3 and TreeAx4 force that $\rightarrow$ is the converse of $\leftarrow$, $\uparrow$ is the converse of $\downarrow$ and that $\uparrow$, $\leftarrow$ and $\rightarrow$ are partial functions. TreeAx3 together with TreeAx4 in addition ensure proper interplay between horizontal and vertical axes. Finally, the Löb axiom NdAx5 is now postulated for all transitive axes, not just for $\downarrow^+$.

Table 4
Additional Axioms for Core XPath

TreeAx1	$\mathsf{a}^+/\mathsf{a} \cup \mathsf{a}$	$\equiv$	a^+	for $\mathsf{a} \in \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$
	$\mathsf{a}/\mathsf{a}^+ \cup \mathsf{a}$	$\equiv$	a^+	
TreeAx2	$\mathsf{a}\,[\phi]\,/\mathsf{a}^{-1}$	$\equiv$	$.\,[\langle \mathsf{a}\,[\phi]\rangle]$	for $\mathsf{a} \in \{\leftarrow, \rightarrow, \downarrow\}$
TreeAx3	$\uparrow[\phi]/\downarrow$	$\equiv$	$(\leftarrow^+ \cup \rightarrow^+ \cup .)\,[\langle\uparrow[\phi]\rangle]$	
TreeAx4	$\leftarrow^+$	$\equiv$	$\leftarrow^+ [\langle\uparrow\rangle]$	
	$\rightarrow^+$	$\equiv$	$\rightarrow^+ [\langle\uparrow\rangle]$	
Generalization of the Löb axiom				
NdAx5'	$\langle \mathsf{a}^+\,[\phi]\rangle$	$\equiv$	$\langle \mathsf{a}^+\,[\phi \wedge \neg\langle \mathsf{a}^+\,[\phi]\rangle]\rangle$	for $\mathsf{a} \in \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$

Theorem 2.3 *All axioms in Tables 2 and 4 are valid, and every valid equivalence between* Core XPath *node expressions is derivable from these axioms.*

Proof. Analogous to the proof of Theorem 2.1. All references to the completeness theorem for GL are replaced by references to the completeness result for the logic of finite trees in [3]. It then suffices to shows that all axiom of this logic, when written as simple node expressions, are derivable. Here, among other things, we make use of a result from [22], according to which *Ind* [9], one of the standard axioms of PDL, used in [3], is derivable from NdAx5'. See [5] for more details. □

Using the *Name* rule, we can again lift the completeness result to path expressions:

Corollary 2.4 *For any two equivalent* Core XPath *path expressions* α, β*, their equivalence is derivable from the axioms in Tables 2 and 4 using the standard rules of equational logic plus the* Name *rule.*

Concerning the fact that, in "real" XML trees each node satisfies precisely one label, the same considerations apply as in the case of Core XPath($\downarrow^+$).

2.3 Further remarks

The above axiomatizations are not very satisfactory as axiomatizations for path expressions, because of their use of the non-standard rule *Name*. Purely equational axiomatizations would be preferable. An interesting challenge is to find such axiomatizations. Temporal logic is another area where much energy was devoted to the elimination of such *Gabbay-style* rules, see, e.g., [21] and [23]. These papers provide also an extensive discussion *why* it is important to find such an axiomatization.

From the point of XML applications, another important question is whether axiomatization such as these can be used to obtain effective rewrite strategies.

Finally, let us mention some other completeness results for XPath fragments. Benedikt, Fan and Kuper [1] axiomatize path expression equivalence for a fragment of Core XPath in which only the axes ., $\downarrow$ and $\downarrow^+$ are allowed, and where no node expressions are allowed except for tests on atomic labels. Ten Cate and Marx [7] axiomatize the navigational core of XPath 2.0: a very expressive language, whose signature is in fact a variant of that of Tarski's relation algebras. Neither of these axiomatizations use non-standard rules.

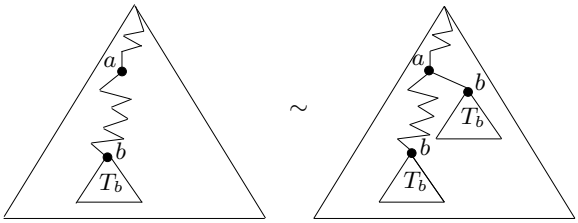

Fig. 1. The trees T and $C_{a,b}(T)$ from Definition 3.1.

3 Expressivity

In this section, we investigate the expressive power of Core XPath and Core XPath($\downarrow^+$). In determining the expressive power of these languages, there are at least two natural yardsticks. One is *first-order logic*, which is probably the most well known logical language. The second, even more attractive one, is *monadic second order logic*. On trees, monadic second-order is a very well-behaved language, and its expressive power can be characterized in many ways (e.g., in terms of finite automata). A straightforward adaptation of the *standard translation* [2] shows that Core XPath is a fragment of first-order logic. However, if we can characterize (fragments of) it in terms of MSO, this means we have a stronger result.

To be more precise, let us indicate exactly which relation symbols are allowed. For any $X \subseteq \{\downarrow, \rightarrow, \downarrow^+, \rightarrow^+\}$, we will denote by FO$[X]$ and MSO$[X]$ the first-order and monadic second-order language with a binary relation R_{a} for each $\mathsf{a} \in X$, as well as a unary predicate P_v for each node label $v \in$ prop (note that we do not need to consider the inverses of these axes, as they are definable in FO and MSO).

3.1 Core XPath($\downarrow^+$)

In this section, we discuss an elegant characterization of Core XPath($\downarrow^+$) node expressions in terms of MSO$[\downarrow]$. Moreover, this characterization is effective: it yields a decision procedure for establishing whether a given MSO$[\downarrow]$ formula is equivalent to a Core XPath($\downarrow^+$) node expression, and if so, it allows us to compute one.

Two characteristic features of Core XPath($\downarrow^+$) are that (i) whether a node expression holds at a node depends only on the subtree below it, and (ii) Core XPath($\downarrow^+$) expressions cannot see the difference between children and descendants. It turns out that, in some sense, these two properties characterize Core XPath($\downarrow^+$) as a fragment of monadic second-order logic.

Before we state the characterization, we introduce some further terminology. Throughout this subsection, we restrict attention to unordered trees, i.e., we forget about the sibling order. Given any node a in a tree T, we denote by T_a the subtree of T rooted by a.

Definition 3.1 Let T be a finite tree and let a, b nodes such that b is a descendant of a. We define $C_{a,b}(T)$ as the tree that is obtained by adding an isomorphic copy of T_b to the tree T, plus an edge from a to the copy of b, cf. Figure 1.

This definition allows us to make precise what it means not to distinguish children from descendants: it means that T and $C_{a,b}(T)$ are indistinguishable. We can now state the characterization of Core XPath($\downarrow^+$) precisely:

Theorem 3.2 *An* MSO[↓] *formula* $\varphi(x)$ *is equivalent to a* Core XPath($\downarrow^+$) *node expression iff*

(i) for all trees T *and nodes* a, $T, a \models \varphi$ *iff* $T_a, a \models \varphi$

(ii) for all trees T *with root* r*, and with nodes* a, b *such that* b *is a descendant of* a*,* $T, r \models \varphi$ *iff* $C_{a,b}(T), r \models \varphi$.

The proof will be based on two known expressivity results. The first theorem we use is the bisimulation characterization of the modal μ-calculus, due to Janin and Walukiewicz. The modal μ-calculus is an extension of modal logic with a *least fixed point* operator. We briefly recall its syntax and semantics. The formulas of the modal μ-calculus are given by the following recursive definition:

$$\varphi ::= \top \mid v \mid \varphi \vee \varphi \mid \neg\varphi \mid \Diamond\varphi \mid \mu v.\varphi.$$

where, in the case of $\mu v.\varphi$, we require that the variable v appears only under an even number of negations in φ. The semantics is given in terms of Kripke models, as for the basic modal language. The interpretation of the Boolean connectives is as usual. For the modal operator, we have $[\![\Diamond\varphi]\!] = \{x \mid \exists y \text{ s.t. } Rxy \text{ and } y \in [\![\varphi]\!]\}$. Finally, if T is a finite tree, we define $[\![\mu v.\varphi]\!]^T$ as the set $\bigcap\{X \subseteq T \mid [\![\varphi]\!]^{T[v \mapsto X]} = X\}$, where $T[v \mapsto X]$ is as T, except that the nodes labelled with v are the ones in X.

A beautiful characterization of the modal μ-calculus was obtained by D. Janin and I. Walukiewicz [12], as the bisimulation invariant fragment of monadic second-order logic. This characterization works on arbitrary Kripke models, but also on the restricted class of finite trees, which is important for us. Moreover, in the case of finite trees, the characterization is effective: it is decidable whether an MSO formula is invariant for bisimulations, and for bisimulation invariant MSO formulas an equivalent formula of the modal μ-calculus can be effectively computed. First, let us recall what bisimulations are. We focus on the special case of finite trees.

Definition 3.3 Let T and T' be two finite trees. A relation $Z \subseteq T \times T'$ is a *bisimulation* if for all (x, x') in Z, we have

- the same node labels hold at x and x',
- if $xR_{\downarrow}y$, there exists $y' \in T'$ such that $x'R_{\downarrow}y'$ and $(y, y') \in Z$,
- if $x'R_{\downarrow}y'$, there exists $y \in T$ such that $xR_{\downarrow}y$ and $(y, y') \in Z$.

A formula $\varphi(x)$ is said to be *bisimulation invariant* on finite trees if for all finite trees T, T', bisimulations $Z \subseteq T \times T'$, and pairs $(x, y) \in Z$, $T, x \models \varphi$ iff $T', y \models \varphi$.

Note that we used $R_{\downarrow}$, and not $R_{\downarrow^+}$, in the definition of bisimulations. If $R_{\downarrow}$ is replaced with $R_{\downarrow^+}$ in Definition 3.3, then we say Z is a *transitive bisimulation*. Also, note that whenever $\varphi(x)$ is (transitive) bisimulation invariant, then, in particular, for all trees T and nodes a, $T, a \models \varphi$ iff $T_a, a \models \varphi$.

Theorem 3.4 (The Janin-Walukiewicz theorem for finite trees [12]) *On finite trees with the child relation, an* MSO[↓] *formula* $\varphi(x)$ *is equivalent to a formula of the* μ*-calculus iff* $\varphi(x)$ *is invariant for bisimulations. Moreover, on these structures bisimulation invariance of* MSO[↓] *formulas is decidable, and for bisimulation invariant* MSO[↓] *formulas an equivalent formula of the modal*

μ-calculus can be effectively computed.

The second result we use concerns the modal logic GL. Recall that GL is the modal logic of the class of finite trees with the descendant relation. The following fact about GL is an easy consequence of a theorem first proved independently by D. de Jongh and G. Sambin (cf. for instance [20,24]).

Theorem 3.5 (Fixed point theorem for GL) *Let φ be a modal formula in which v only occurs in the scope of an even number of negations. Then, on the class of finite trees with the descendant relation, $\mu v.\varphi$ is equivalent to a modal formula δ. Moreover, δ can be effectively computed from φ.*

We are now ready to prove Theorem 3.2. First, we will show that the two conditions from Theorem 3.2 can be reformulated in terms of transitive bisimulations. Next, we show that Core XPath($\downarrow^+$) is the transitive bisimulation invariant fragment of MSO[$\downarrow$], by putting together Theorem 3.4 and 3.5.

Our first observation is that transitive bisimulations are closely related to the operation $C_{a,b}$ we introduced before. Formally, let us write $T \Rightarrow T'$ if T' is isomorphic to $C_{a,b}(T)$ for some $a, b \in T$ (with a an ancestor of b), and let $\sim$ be the reflexive, transitive and symmetric closure of $\Rightarrow$.

Lemma 3.6 *For all finite trees T and T' with nodes a, a', (T, a) and (T', a') are transitive bisimilar iff $T_a \sim T'_{a'}$.*

Proof. The proof is by induction on the depth of a. The idea is to construct two chains $T = T_0 \Rightarrow T_1 \cdots \Rightarrow T_n$ and $T' = T'_0 \Rightarrow T'_1 \cdots \Rightarrow T'_{n'}$ such that T_n and $T'_{n'}$ are isomorphic. We leave out the details. □

Proposition 3.7 *An MSO[$\downarrow$] formula $\varphi(x)$ is equivalent to a Core XPath($\downarrow^+$) node expression iff $\varphi(x)$ is invariant for transitive bisimulations.*

Proof. We restrict ourselves to a sketch of difficult direction. Let $\varphi(x)$ be an MSO[$\downarrow$] formula that is invariant for transitive bisimulations. We need to find a node expression χ of Core XPath($\downarrow^+$) such that for all finite trees T and all nodes a, χ holds at a iff $\varphi(a)$ is true.

Since $\varphi(x)$ is invariant for transitive bisimulations, in particular $\varphi(x)$ is invariant for ordinary bisimulations. It follows for Theorem 3.4 that there exists some formula ψ of the μ-calculus such that $\varphi(x)$ and ψ are equivalent on finite trees (with the child relation).

Using the fact that $\phi(x)$ and ψ are both invariant for transitive bisimulations, it is possible to deduce that $\varphi(x)$ and ψ are equivalent on finite trees with the descendant relation as well. Finally, it follows from Theorem 3.5 that ψ is equivalent on finite trees with the descendant relation, to a modal formula χ. Or in other words, ψ is equivalent on XML trees to a simple node expression of Core XPath($\downarrow^+$). □

Using Proposition 3.7 and Lemma 3.6, we obtain immediately Theorem 3.2. Putting everything together, it is also not hard to show that there is an effective procedure to check whether an MSO[$\downarrow$] formula defines a transitive bisimulation closed class of finite trees. Or to put it in a different way, it is decidable whether an MSO[$\downarrow$] formula is equivalent on finite trees to a (simple) node expression of Core XPath($\downarrow^+$).

Incidentally, Proposition 3.7 tells us in particular that every $\mathsf{MSO}[\downarrow]$-formula invariant for transitive bisimulations is equivalent to a formula of $\mathsf{FO}[\downarrow^+]$.

To summarize, we gave an effective characterization of node expressions of $\mathsf{Core\ XPath}(\downarrow^+)$, using the expressiveness result of the modal μ-calculus and the fact that GL is closed under fixpoints. It is in fact possible to adapt the method in order to obtain similar characterizations for *path expressions*, but this is still work in progress (in collaboration with M. Bojańczyk).

3.2 Full Core XPath

Whether there is a similar decidable characterization for full Core XPath in terms of monadic second-order logic is an important open question. There is, however, a syntactic characterization. Marx and De Rijke [17] showed that Core XPath node expressions have the same expressive power as the two-variable fragment of $\mathsf{FO}[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$.

Theorem 3.8 ([17]) *For every formula $\psi(x) \in \mathsf{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$, there is a node expression $\phi \in \mathsf{NodeEx}$ such that the two are equivalent: for any XML tree T, $[\![\phi]\!]^T_{\mathsf{NExpr}} = \{n \mid T, n \vDash \psi(x)\}$. Likewise in the other direction.*

Incidentally, note that, while $\downarrow$ and $\rightarrow$ are definable from $\downarrow^+$ and $\rightarrow^+$ in first-order logic, they are not so in the two variable fragment. This is the reason why they need to be explicitly added to the signature here.

Theorem 3.8 is a natural generalization of a characterization of temporal logic (without the "until" connective) on linearly ordered sets [8]. The proof in [17] closely follows the one in [8].[4] Rather than sketching the same proof again, we will give a slightly different presentation, by combining some observations from the literature.

The translatoin from Core XPath node expressions to $\mathsf{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ is straightforward: it is essentially the same as the *standard translation* of modal formulas into $\mathsf{FO}^2(x)$ [2]. The interesting part of Theorem 3.8 is the converse. Our proof will be based on the following two facts: (i) the expressive completeness of *Boolean modal logic* (BML) for FO^2 [15], and (ii) the fact that Core XPath is closed under path intersections.

Boolean modal logic is an extension of the basic modal language in which one can use complex modalities built up from the atomic ones using the Boolean connectives and the *converse* operator $(\cdot^\smile)$. Formally, given a set Mod of atomic modalities, we define *complex modalities* as terms generated by

$$\mathsf{CMod}[Mod] ::= Mod \mid \mathsf{id} \mid -\mathsf{CMod} \mid \mathsf{CMod} \cup \mathsf{CMod} \mid \mathsf{CMod} \cap \mathsf{CMod} \mid \mathsf{CMod}^\smile$$

Whenever the set Mod of atomic modalities is clear from the context, we omit this

[4] Incidentally, for readers who are interested in this, in Table 3 of [17] the following row is missing:

$\tau(x,y)$	$\exists y.(\tau(x,y) \wedge A(y))$
$\neg xR_{\downarrow^+}y \wedge \neg xR_{\rightarrow^+}y \wedge \neg yR_{\downarrow^+}x \wedge$ $\wedge \neg yR_{\rightarrow^+}x \wedge \neg x = y$	other $[A']$

where other is as defined in the proof of Theorem 3.8

Table 5
Complements of basic axes are Core XPath definable

A	Complement of A
$\downarrow$	$\downarrow/\downarrow^+ \cup\, . \cup \leftarrow^+ \cup \rightarrow^+ \cup \uparrow^+ \cup$ other
$\downarrow^+$	$. \cup \leftarrow^+ \cup \rightarrow^+ \cup \uparrow^+ \cup$ other
$\rightarrow$	$\rightarrow/\rightarrow^+ \cup\, . \cup \leftarrow^+ \cup \downarrow^+ \cup \uparrow^+ \cup$ other
$\rightarrow^+$	$. \cup \leftarrow^+ \cup \downarrow^+ \cup \uparrow^+ \cup$ other

parameter. The formulas of Boolean modal logic are then defined as follows:

$$\varphi ::= \top \mid v \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle \mathsf{CMod} \rangle \varphi,$$

The definition of semantics should be clear to the reader. In particular, the formulas are interpreted in Kripke models with a binary relation for each element of Mod, and id is interpreted as the identity relation.

Theorem 3.9 ([15]) *On any class of Kripke models, the formulas of BML have the same expressive power as* FO^2*-formulas in one free variable.*

In particular, it holds on the class of XML trees, where the atomic modalities are the axes of Core XPath. The second fact we use is the following result about Core XPath:

Theorem 3.10 *For every two* Core XPath *path expressions* A, B*, there is a* Core XPath *path expression* C *such that, in any XML tree,* C *that defines the intersection of the binary relations defined by* A *and* B*.*

This was proved in [17], as a corollary of the expressive completeness of Core XPath. However, here, we are using it as a lemma in the proof of the expressive completeness, so, clearly, we cannot refer to [17]. However, a direct proof of Theorem 3.10 may be found for instance in [6]. We are now ready to proceed with the proof of Theorem 3.8.

Proof of Theorem 3.8. Let $\psi(x)$ be any formula of $\mathsf{FO}^2[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$. By Theorem 3.9, $\psi(x)$ is equivalent on XML trees to a a BML formula χ, where the atomic modalities are $\downarrow$, $\downarrow^+$, $\rightarrow$, and $\rightarrow^+$. We will show that every element of CMod[Axis] is equivalent to a path expression of Core XPath. The result then follows by a straightforward inductive translation from Boolean modal formulas to Core XPath node expressions (cf. also the *simple node expressions* described in Section 2.1).

Take any $A \in$ CMod[Axis]. We may assume without loss of generality that A is a positive Boolean combination of terms of the form a, $-\mathsf{a}$, $\mathsf{a}^\smile$ or $-\mathsf{a}^\smile$, with $\mathsf{a} \in \{\downarrow, \downarrow^+, \rightarrow, \rightarrow^+\}$. Since the path expressions of Core XPath are closed under union and intersection (cf. Theorem 3.10), it therefore only remains to be shown that each of these four is equivalent to a Core XPath path expression.

But this follows directly from Table 5, where other is used as shorthand for

$$\uparrow^+/(\leftarrow^+ \cup \rightarrow^+)/\downarrow^+ \cup \uparrow^+/(\leftarrow^+ \cup \rightarrow^+) \cup (\leftarrow^+ \cup \rightarrow^+)/\downarrow^+ ,$$

together with the fact that the Core XPath path expressions are closed under taking converse. □

In [17], Theorem 3.8 has also been combined with results of [1] to give a characterization of Core XPath *path expressions*: they correspond to *unions of conjunctive queries* in which $\mathsf{FO}[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$-formulas with one free variable may be used as atomic predicates.

3.3 Further notes

Theorem 3.2 has already been proved independently by M. Bojańczyk and I. Walukiewicz (p.c.), using the framework of *forest algebras* (and without reference to modal logic). They also found similar characterizations for the fragments Core XPath$(\downarrow)$ (that is, the only axes allowed are $\downarrow$ and ".") and Core XPath$(\downarrow, \downarrow^+)$ (that is, the only axes allowed are $\downarrow$, $\downarrow^+$, and "."). See [4] for more details on forest algebras (in the case of binary branching trees).

Concerning more "syntactic" characterizations, such as the one we described in Section 3.2, we mention two further examples. The negation-free Core XPath$(\downarrow, \downarrow^+)$ path expressions correspond to existential positive formulas of $\mathsf{FO}[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ in two free variables [1]. Marx [16] investigates Core XPath 1.0 with "until" operator and shows that—analogously to temporal logic for linear orders—this language has the same expressive power as $\mathsf{FO}[\downarrow^+, \rightarrow^+]$, both for node and path expressions.

References

[1] M. Benedikt, W. Fan, and G. Kuper. Structural properties of XPath fragments. *Theoretical Computer Science*, 336(1):3–31, 2005.

[2] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[3] P. Blackburn, W. Meyer-Viol, and M. de Rijke. A proof system for finite trees. In H. Kleine Büning, editor, *Computer Science Logic*, volume 1092 of *LNCS*, pages 86–105. Springer, 1996.

[4] M. Bojańczyk and I. Walukiewicz. Characterizing EF and EX tree logics. *Theoretical Computer Sciences*, 358:255–272, 2006.

[5] B. ten Cate, T. Litak, and M. Marx. A complete axiomatization for Core XPath 1.0. In Jan Van den Bussche, editor, *Liber Amicorum for Jan Paredaens*. September 2007. Full technical report available online at http://staff.science.uva.nl/~litak.

[6] B. ten Cate and C. Lutz. The complexity of query containment in expressive fragments of xpath 2.0. In *PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 73–82, New York, NY, USA, 2007. ACM.

[7] B. ten Cate and M. Marx. Axiomatizing the logical core of XPath 2.0. In Th. Schwentick and D. Suciu, editors, *Proceedings ICDT 2007*, 2007.

[8] K. Etessami, M. Vardi, and Th. Wilke. First-order logic with two variables and unary temporal logic. In *Proc. LICS'97*, pages 228–235, 1997.

[9] R. Goldblatt. *Logics of Time and Computation*, volume 7 of *Lecture Notes*. CSLI Publications, 1987.

[10] G. Gottlob, C. Koch, and R. Pichler. Efficient algorithms for processing XPath queries. In *VLDB'02*, 2002.

[11] M. Hollenberg. An equational axiomatization of dynamic negation and relational composition. *Journal of Logic, Language and Information*, 6(4):381–401, 1997.

[12] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional μ-calculus with respect to monadic second order logic. In *Lecture Notes In Computer Science, Proceedings of the 7th International Conference on Concurrency Theory*, volume 1119, pages 263–277, 1996.

[13] S. Kleene. *Representation of Events in Nerve Nets and Finite Automata*, pages 3–42. Princeton University Press, Princeton, N.J., 1956.

[14] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Inf. Comput.*, 110(2):366–390, 1994.

[15] C. Lutz, U. Sattler, and F. Wolter. Modal logic and the two-variable fragment. *Lecture Notes in Computer Science*, 2141:262–276, 2001.

[16] M. Marx. Conditional XPath. *ACM Transactions on Database Systems (TODS)*, 30(4):929–959, 2005.

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

[18] W. McCune. Solution of the Robbins problem. *J. Autom. Reason.*, 19(3):263–276, 1997.

[19] V. Pratt. Dynamic algebras: Examples, constructions, applications. *Studia Logica*, 50, 3–4:571–605, 1991.

[20] L. Reidhaar-Olson. A new proof of the fixed-point theorem for provability logic. *Notre Dame Journal of Formal Logic*, 31(1):37–43, 1989.

[21] M. Reynolds. An axiomatization for Until and Since over the reals without the IRR rule. *Studia Logica*, 51(2):165–193, 1992.

[22] J. van Benthem. Modal frame correspondence and fixed-points. *Studia Logica*, 83(1):133–155, 2006.

[23] Y. Venema. Completeness via completeness: Since and Until. Synthese Library vol. 229, pages 279–286. Kluwer Academic Publishers, 1993.

[24] A. Visser. *Löb's logic meets the μ-calculus*, volume 3838 of *Lecture Notes in Computer Science*. Springer, 2005.