Semantic characterizations of navigational XPath
Marx, M.J.; de Rijke, M.

Published in:
The First Twente Data Management Workshop TDM'04 on XML databases and information retrieval

Citation for published version (APA):
Marx, M. J., & de Rijke, M. (2004). Semantic characterizations of navigational XPath. In V. Mihajlovic, & D. Hiemstra (Eds.), The First Twente Data Management Workshop TDM'04 on XML databases and information retrieval (pp. 67-73). (CTIT workshop proceedings series; No. WP04-1). Enschede: Centre for Telematics and Information Technology (CTIT), University of Twente.

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Semantics Characterizations of Navigational XPath

Maarten Marx and Maarten de Rijke
Informatics Institute
University of Amsterdam
{marx,mdr}@science.uva.nl

ABSTRACT
We give semantic characterizations of the expressive power of navigational XPath (also called Core XPath) in terms of first order logic. XPath can be used to specify sets of nodes and to specify sets of paths in a document tree. We consider both uses. For sets of nodes, we show that first order logic in two variables is equally expressive as XPath. For paths, we show that XPath can be defined using four simple connectives, which together yield the class of first order definable relations which are safe for bisimulation. Furthermore, we give a characterization of the XPath expressible paths in terms of conjunctive queries.

1. INTRODUCTION
XPath 1.0 [6] is a variable free language used for selecting nodes from XML documents. XPath plays a crucial role in other XML technologies such as XSLT [10], XQuery [9] and XML schema constraints, e.g., [8]. The recently proposed XPath 2.0 language [7] is much more expressive and is close to being a full fledged tree query language. It contains variables which are used in if-then-else, for, and quantified expressions. The available axis relations are the same in both versions of XPath. What is missing at present is a clear characterization of the expressive power of XPath, be it either semantical or with reference to some well established existing (logical) formalism. As far as we know, Benedikt, Fan and Kuper were the first and only to give characterizations, but only for positive fragments of XPath, and without considering the sibling axis relations. Their analysis can rather simply be expanded with the sibling axis, but adding negation asks for a different approach. This paper aims at filling this gap.

Characterizations of the kind we are after are useful in understanding and further designing the language. They are also useful because they allow us to transfer already known results and techniques to the world of XPath. Vianu [21] provides several examples to this effect. All characterizations we give with respect to other languages are constructive and given in terms of translations. An important issue in such comparisons is the succinctness of one language with respect to another. We only touch on this briefly.

We use the abstraction to the logical core of XPath 1.0 (called Core XPath) developed in [12, 14]. From now on we will simply speak of XPath instead of Core XPath. Core XPath is interpreted on XML document tree models. The central expression in XPath is the location path

\[
\text{axis} :: \text{node}_\text{label} \cdot \text{filter},
\]

which, when evaluated at node \(n\), yields an answer set consisting of nodes \(n'\) such that the axis relation goes from \(n\) to \(n'\), the node tag of \(n'\) is \(\text{node}_\text{label} \cdot \text{filter}\), evaluates to true at \(n'\). Alternatively, axis :: node_label[filter] can be viewed as denoting a binary relation, consisting of all nodes \((n,n')\) which stand in the above relation.

XPath serves two purposes. First and formost it is used to select nodes from a document. This use is formalized by the notion of answer set. We study the expressive power of XPath with respect to defining answer sets in Section 3. Our main result is that Core XPath is as expressive as first order logic restricted to two variables in the signature with three binary relations corresponding to the child descendant and following_sibling axis relations and unary predicates corresponding to the node tags.

The second use of XPath is as a set of binary atoms in more expressive languages with variables such as XQuery. For instance, we might want to select all nodes \(x\) satisfying

\[
\exists y (x \text{ descendant} :: A y \land
\neg x \text{ descendant} :: B/\text{descendant} :: * y).
\]

That is, the set of all points from which there starts a path without \(B\) nodes ending in an \(A\) node. We study this use in Sections 4 and 5. With respect to the expressive power of the relations expressed by Core XPath we establish the following:

1. The set of relations expressible in Core XPath is closed under intersection but not under complementation.
2. The Core XPath definable relations are exactly those that are definable as unions of conjunctive queries whose atoms correspond to the XPath axis relations and to XPath’s filter expressions.
3. The Core XPath definable relations coincide with the first order definable relations which are safe for bisimulations respecting the Core XPath axis relations.\(^1\)

\(^1\)Informally, the notion of “safety” means the following. A bisimulation relation is always defined with respect to a certain signature consisting of unary predicates and binary transition relations. A relation between the domains of two models (e.g., transition systems, or in our case trees) is a bisimulation if (1) bisimilar elements satisfy the same unary predicates and (2) moves along the transition relations in
4. All Core XPath relations can be defined from its axis and node-tag tests by composition, union, and taking the counterdomain of a relation.

The paper is organized as follows. We finish this introduction by recalling related work. The next section defines Core XPath. Sections 3 and 4 are about the expressive power of XPath for selecting sets of nodes, and selecting sets of paths, respectively. Section 5 establishes a minimal set of connectives for XPath.

Related work
The paper most closely related to this work is the already mentioned [2], which characterizes positive XPath without sibling axis as existential positive first order logic. Characterizations in terms of automata models have been given in [3, 19, 17, 18].

Connections with temporal logic have been observed by [12, 16] which sketch an embedding of the forward looking fragment of XPath into CTL. [1] exploits embeddings of subsets of XPath into computation tree logic to enable the use of model checking for query evaluation. [15] discusses an extension of XPath in which every first order definable set of nodes can be expressed. Several authors have considered extensions far beyond XPath 1.0, trying to capture all of monadic second order logic.

2. CORE XPATH
[14] proposes a fragment of XPath 1.0 which can be seen as its logical core, but lacks much of the functionality that accounts for little expressive power. In effect, it supports all XPath’s axis relations, except for the attribute and namespace axis relations, it allows sequencing and taking unions of path expressions and full boolean in the filter expressions. It is called Core XPath, also referred to as navigational XPath. A similar logical abstraction is made in [2]. As the focus of this paper is expressive power, we focus also XPath restricted to its logical core.

For the definition of the XPath language and its semantics, we follow the presentation of XPath in [14]. The expressions obey the standard W3C unabbreviated XPath 1.0 syntax. The semantics is as in [2, 13], which is in line with the standard XPath semantics [22].

Definition 1. The syntax of the Core XPath language is defined by the grammar

\[
\text{locpath ::= axis '::' ntst | axis '::' ntst ['[fexpr]'] | '/|locpath | locpath '/|locpath | locpath '||' locpath | fexpr ::= locpath | not fexpr | fexpr and fexpr | fexpr or fexpr
\]

\[
\text{axis ::= self | child | parent | descendant | descendant_or_self | ancestor | ancestor_or_self | following_sibling | preceding_sibling | following | preceding}
\]

where “locpath” (pronounced as location path) is the start production, “axis” denotes axis relations and “ntst” denotes tags labeling document nodes or the star * that matches all tags (these are called node tests). The “fexpr” will be called filter expressions after their use as filters in location paths. By an XPath expression we always mean a “locpath.”

The semantics of XPath expressions is given with respect to an XML document modeled as a finite node labeled sibling ordered tree (tree for short). Each node in the tree is labeled with a set of primitive symbols from some alphabet. Sibling ordered trees come with two binary relations, the child relation, denoted by $R_c$, and the immediate_right_sibling relation, denoted by $R_{\circ}$. Together with their inverses $R_i$ and $R_{\circ}$ they are used to interpret the axis relations. We denote such trees as first order structures $(N, R_c, R_{\circ}, P)_{\in \Lambda}$.

Each location path denotes a binary relation (a set of paths). The meaning of the filter expressions is given by the predicate $E(n, fexpr)$ which assigns a boolean value. Thus a filter expression $fexpr$ is most naturally viewed as denoting a set of nodes: all $n$ such that $E(n, fexpr)$ is true. For examples, we refer to [14]. Given a tree $M$ and an expression $R$, the denotation or meaning of $R$ in $M$ is written as $[R]_M$. Table 1 contains the definition of $[\cdot]_M$.

As discussed, one of the purposes of XPath is to select sets of nodes. For this purpose the notion of an answer set is defined. For $R$ an XPath expression, and $M$ a model, $\text{answer}_M(R) = \{ n \mid [\exists n.(n, n) \in [R]_M] \}$. Thus the answer set of $R$ consists of all nodes which are reachable by the path $R$ from some point in the tree.

3. THE ANSWER SETS OF CORE XPATH
We show that on ordered trees, Core XPath is equally expressive as first order logic in two variables over the signature with predicates corresponding to the child, descendant, and following_sibling axis relations. More precisely, we show that for every XPath expression $R$, there exists an XPath filter expression $A$ such that, on every model $M$,

\[
\text{answer}_M(R) = \{ n \mid [\exists n.(n, n) = A] \}.
\]

Then, we show that every first order formula $\phi(x)$ in the signature just mentioned is equivalent to an XPath filter expression $A$ in the sense that for every model $M$, and for every node $n$,

\[
\text{answer}_M(R) = \{ n \mid [\exists n.(n, n) = A] \}.
\]

First, though, we fix our terminology.

$\quad$ 4A sibling ordered tree is a structure isomorphic to $(N, R_c, R_{\circ})$ where $N$ is a set of finite sequences of natural numbers closed under taking initial segments, and for any sequence $s$, if $s \cdot k \in N$, then either $k = 0$ or $s \cdot k - 1 \in N$. For $n, n' \in N$, $nR_cn'$ holds iff $n' = n \cdot k$ for a natural number; $nR_{\circ}n'$ holds iff $n = s \cdot k$ and $n' = s \cdot k + 1$. 


\[
\begin{align*}
\text{locpath} & ::= \text{axis }::\text{ntst }| \text{axis }::\text{ntst }[\text{fexpr}]
\text{axis ::= self }| \text{child }| \text{parent }| \text{descendant }| \text{descendant_or_self }| \text{ancestor }| \text{ancestor_or_self }| \text{following_sibling }| \text{preceding_sibling }| \text{following }| \text{preceding }
\end{align*}
\]
We work with first order logic over node labeled ordered trees in a signature with unary predicates from \( \Lambda = \{P_1, P_2, \ldots \} \) and with a number of binary predicates corresponding to “moves” in a tree. We use the predicates \( \text{child}, \text{descendant} \) and \( \text{following_sibling} \). For \( \text{moves} \) a subset of these three moves, we use \( \text{FO}^2[\text{moves}] \) to denote the set of first order formulas \( \phi(x) \) in which at most \( x \) occurs free, and which contain at most two variables in all of \( \phi \). Moreover, \( \phi(x) \) is written in the signature consisting of \( \Lambda \) and \( \text{moves} \). When interpreted on a tree, a \( \text{FO}^2[\text{moves}] \) formula \( \phi(x) \) denotes a set of nodes.

**Theorem 2.** For \( X \) a set of nodes in an ordered tree, the following are equivalent:

- \( X \) is the answer set of some Core XPath expression;
- \( X \) is definable by a formula in \( \text{FO}^2[\text{descendant, child, following_sibling}] \) in one free variable.

More precisely, for every formula \( \phi(x) \) in \( \text{FO}^2[\text{descendant, child, following_sibling}] \) with unary predicates from \( \Lambda \), there exists a Core XPath expression \( R \) written with node tags \( \Lambda \), such that on every tree \( M \), \( \text{answer}_M(R) = \{n \mid M \models \phi(n)\} \), and conversely.

The equivalence can be proved by translations, given in Lemmas 3-5 below. The hard direction follows more or less directly from the argument used to show a similar statement made for linear orders, characterizing temporal logic with only unary temporal connectives by Etessami, Vardi and Wilke [11].

Lemma 3 below shows that Core XPath is equally expressive as its filter expressions. Interestingly, Core XPath’s filter expressions were introduced already in [5] for exactly the same purpose as the XPath language: specifying sets of nodes in finite ordered trees. The only difference is that the language of [5] does not have the asymmetry between the vertical and the horizontal axis relations: the immediate left and right sibling relations are also present. They provide a complete axiomatization, in a logic called LOFT (Logic Of Finite Trees), which might be of interest for query rewriting.

**Lemma 3.** For any Core XPath expression \( R \) there exists a Core XPath filter expression \( A \) without \( \text{following and preceding} \) axis and without absolute expressions such that for each model \( M \),

\[
\text{answer}_M(R) = \{n \mid M \models \phi(n)\}.
\]

The size of \( A \) is linear in the size of \( R \).

**Proof.** Consider an arbitrary XPath expression. First perform the following substitutions, from left to right:

1. \( \text{following} :: P_1[A] \equiv \text{ancestor_or_self} :: * \\
   \text{following_sibling} :: * \\
   \text{descendant_or_self} :: P_1[A] \)
2. \( \text{preceding} :: P_1[A] \equiv \text{ancestor_or_self} :: * \\
   \text{preceding_sibling} :: * \\
   \text{descendant_or_self} :: P_1[A] \)
3. \( R \equiv \text{ancestor_or_self} :: * [\text{not parent :: *}] / R \)

The result is an equivalent formula without \( \text{following and preceding} \) axis and without absolute expressions. Let \( R \) be
the result. Now obtain \( A \) by applying the converse operator \((\cdot)^{-1}\) as follows:

\[
(S \mid T)^{-1} \equiv S^{-1} \mid T^{-1} \\
(S/T)^{-1} \equiv T^{-1}/S^{-1} \\
(axis :: P_i[B])^{-1} \equiv self :: P_i[B]/axis^{-1} :: *,
\]

with \( axis^{-1} \) having the obvious meaning. Note that it is crucial that the Core XPath axis relations are closed under taking converses. Then, for each node \( n, n \in \text{answer}_M(R) \) iff \( \mathcal{E}(n, A) \) equals true. Whence the lemma. \( \square \)

**Lemma 4.** The answer set of any Core XPath expression can be defined by a formula of \( \text{FO}^2[\text{descendant}, \text{child}, \text{following_sibling}] \) in one free variable.

**Proof.** Let \( R \) be a Core XPath expression. Let \( A \) be the filter expression obtained in Lemma 3. Apply the standard translation well known from modal logic to \( A \) to obtain the desired first order formula (cf., Vardi [20] which takes care to use only two variables). The translation is just the definition of \( \mathcal{E} \) from Table 1 written in first order logic. \( \square \)

With this we have shown the easy side of Theorem 2. Now we discuss the other side, following [11].

**Lemma 5.** Every set of nodes defined by a formula in \( \text{FO}^2[\text{descendant}, \text{child}, \text{following_sibling}] \) with one free variable can be defined as the answer set of an absolute Core XPath expression.

We note that, as in [11], the size of the filter expression is exponential in the size of the first order formula. [11] show that on finite linear structures already this is unavoidable, so also on trees this bound is tight.

**Proof.** Let \( \phi(x) \) be the first order formula. We will provide an XPath filter expression \( A \) such that (3) holds. Whence \( \text{descendant or self} :: *|A \) is the desired absolute XPath expression. The proof is a copy of the one for linear temporal logic in Theorem 1 in [11]. The only real change needed is in the set of order types: they are given in the right hand side of Table 2, together with the needed translations (\( A' \) denotes the translation of \( A \)). The other change is rather cosmetic. For \( A \) an atom, \( A(x) \) needs to be translated using the self axis as \( \text{self} :: A \). For instance, \( \forall y(y \text{child} x \land A(x)) \) translates to \( \text{parent} :: *| \text{self} :: A \). Translating \( \phi(x) \), the result of this process is a filter expression \( A \) for which in any model, for every node \( n, M, \mathcal{E}(n, A) \) equals true iff \( M \models \phi(n) \). \( \square \)

**Remark 6.** Just as in [11], it is straightforward to apply the argument to XPath fragments with less or more axis relations, as long as the axis are closed under taking converses. The argument in Table 2 is modular in the operators. Care has to be taken with the axis used in defining away the following and preceding axis and the absolute expressions. In some cases, the signature of the first order language has to be expanded.

\( ^5 \)Because \( n \in \text{answer}_M(R) \) iff by definition there exists \( n' \) such that \( n'Rn' \) iff there exists \( n' \) such that \( nR^{-1}n' \) iff, by definition, \( \mathcal{E}(n, A) \) equals true.

### 4. The Paths of Core XPath

In the previous section we characterized the answer sets of XPath. We now turn to the sets of paths that can be defined in XPath; they too admit an elegant characterization which we provide here. First, we define the appropriate first order language.

A conjunctive path query is a conjunctive query of the form

\[
Q(x, y) \equiv \neg R_1 \land \ldots \land R_m \land \phi_1 \land \ldots \land \phi_m,
\]

in which the \( R_i \) are relations from the signature \{\text{descendant}, \text{child}, \text{following_sibling}\} and all of the \( \phi_i \) are formulas in \( \text{FO}^2[\text{descendant}, \text{child}, \text{following_sibling}] \) in one free variable. An example is provided by

\[
Q(x, y) \equiv \neg z \text{descendant} x \land z \text{following_sibling} z',
\]

\[
\neg \text{descendant} y, P_1(z), P_2(y),
\]

which is equivalent to the XPath expression

\[
\text{ancestor} :: P_1/\text{following_sibling} :: *|\text{descendant} :: P_2.
\]

With a union of conjunctive path queries we mean a disjunction of such queries with all of them the same two free variables \( x \) and \( y \). For example,

\[
\text{descendant} :: P_2 \mid \text{parent} :: *|\text{ancestor} :: P_1
\]

is equivalent to the union of the two queries

\[
Q(x, y) \equiv \neg x \text{descendant} y, P_2(y)
\]

and

\[
Q(x, y) \equiv \neg z \text{child} x \land z \text{ancestor} y, P_1(y).
\]

From Lemma 4 and some simple syntactic manipulation we immediately obtain

**Proposition 7.** Every XPath expression is equivalent to a union of conjunctive path queries.

The converse also holds, which gives us a characterization of the XPath definable sets of paths.

**Theorem 8.** For every union of conjunctive path queries \( Q(x, y) \) there exists a Core XPath expression \( R \) such that for every model \( M, \{ (n, n') \mid M \models Q(n, n') \} = [R]'_M \).

For lack of space we can only give a sketch of the proof. The theorem can be shown using (3) and an extension of the argument as used in [2], where Benedikt, Fan, and Kuper show that positive XPath without sibling axis is equivalent to positive existential first order logic.

### 4.1 Structural properties of XPath

Benedikt, Fan and Kuper [2] have given an in depth analysis of a number of structural properties of fragments of XPath. Their fragments are all positive (no negations inside the filters) and restricted to the “vertical” axis relations defined along the tree order. All their fragments allowing filter expressions are closed under intersection, while none is closed under complementation. Here, we show that this is also true for full XPath.

**Theorem 9.** Core XPath is closed under intersections. That is, for every two Core XPath expressions \( A, B \), there exists a Core XPath expression \( C \) such that on every model \( M, [A]'_M \cap [B]'_M = [C]'_M \).
There exist linear translations \( t_1, t_2 \) with \( t_1 : \text{Core XPath} \rightarrow \text{SCX} \) and \( t_2 : \text{SCX} \rightarrow \text{Core XPath} \) such that for all models \( \mathcal{M} \), the following hold:

- for every XPath expression \( R \), \( [R]_\mathcal{M} = [t_1(R)]_\mathcal{M} \),
- for every SCX expression \( R \), \( [R]_\mathcal{M} = [t_2(R)]_\mathcal{M} \).

**PROOF.** Because the counterdomain of a relation \( R \) is definable in XPath as \( \text{self} :: \ast \{ A \} \), every relation defined in (8) can be expressed as a Core XPath formula. For the other side, first observe that \( \text{axis} :: P_i \) and \( \text{axis}/P_j \) are equivalent. As both languages are closed under composition and union, we only have to show that all filter expressions are expressible. With the following equivalences we can extend \( \tau \) to all filter expressions (cf. Lemma 2.82 in [4]):

\[
\begin{align*}
\tau(x, y) &= \exists y (x \text{ descendant } y \land A(y)) \land \\
\exists y (x \text{ following sibling } y) \\
\tau(x, y) &= \exists y (x \text{ following sibling } y \land \exists z ((x \text{ descendant } z \land z \text{ descendant } y) \rightarrow \sim B(z))).
\end{align*}
\]

A standard argument shows that this set cannot be specified using less than three variables. This contradicts Theorem 2 which states that the answer set of every XPath expression is equivalent to a first order formula in two variables. QED

### Table 2: Order types and their translation

<table>
<thead>
<tr>
<th>( \tau(x, y) )</th>
<th>( \exists y (\tau(x, y) \land A(y)) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = y )</td>
<td>self :: ( \ast { A } )</td>
</tr>
<tr>
<td>( x \text{ child } y )</td>
<td>child :: ( \ast { A } )</td>
</tr>
<tr>
<td>( y \text{ child } x )</td>
<td>parent :: ( \ast { A } )</td>
</tr>
<tr>
<td>( x \text{ following sibling } y )</td>
<td>following sibling :: ( \ast { A } )</td>
</tr>
<tr>
<td>( y \text{ following sibling } x )</td>
<td>preceding sibling :: ( \ast { A } )</td>
</tr>
<tr>
<td>( x \text{ descendant } y \land \sim x \text{ child } y )</td>
<td>child :: ( \ast / \text{ descendant } :: { A } )</td>
</tr>
<tr>
<td>( y \text{ descendant } x \land \sim y \text{ child } x )</td>
<td>parent :: ( \ast / \text{ ancestor } :: { A } ).</td>
</tr>
</tbody>
</table>

In this section we look at the connectives of XPath and argue that they are very well chosen. We disregard the following and preceding axis relations as well as absolute expressions (those are expressions starting with a \( / \) as they are definable anyway (cf. Lemma 3 above). What are the connectives of XPath? This question is not trivial. Clearly, there is composition (\(' /'\)) and union (\('\lor'\)) of paths. Then there is composition with a filter expression (\('('F'\)'). And inside the filter expressions all boolean connectives are allowed. It turns out that this rather messy set can be streamlined.

Consider the following definition of path formulas:

\[(8) \quad \tau ::= \text{axis} \mid \?P_i \mid \tau/R \mid \tau/R \mid \sim \tau, \]

for \( \text{axis} \) one of Core XPath’s axis relations, for \( P \), a tagname, and the following for the meaning of the two new connectives:

\[
\begin{align*}
[\tau]_\mathcal{M} &= \{(x, x) \mid \text{the tag of } x \text{ is } P_i \} \\
[\sim \tau]_\mathcal{M} &= \{(x, y) \mid x = y \land \sim \exists z [\tau/\mathcal{M}](x, z) \}.
\end{align*}
\]

We call this language SCX (short for short core xpath). \( \?P_i \) simply tests whether a node has tag \( P_i \). Thus child :: \( P_i \) can be written as \( \text{child}/\?P_i \). The unary operator \( \sim \) is sometimes called counterdomain. For instance \( \sim \text{child} \) defines the set of all pairs \( (x, x) \) for \( x \) a leaf, and \( \sim \text{parent} \) the singleton \( \{ \text{root}, \text{root} \} \).

Below we explain why this set of connectives is so nice. First we show that this definition is equivalent in a very strong sense to that of Core XPath.

**THEOREM 11.** There exist linear translations \( t_1, t_2 \) with \( t_1 : \text{Core XPath} \rightarrow \text{SCX} \) and \( t_2 : \text{SCX} \rightarrow \text{Core XPath} \) such that for all models \( \mathcal{M} \), the following hold:

\[
\begin{align*}
\tau(x, y) &= \exists y (\tau(x, y) \land A(y)) \\
\exists y (\tau(x, y) \land A(y)) &= \text{self} :: \ast \{ A \} \\
\exists y (\tau(x, y) \land A(y)) &= \text{child} :: \ast \{ A \} \\
\exists y (\tau(x, y) \land A(y)) &= \text{parent} :: \ast \{ A \} \\
\exists y (\tau(x, y) \land A(y)) &= \text{following sibling} :: \ast \{ A \} \\
\exists y (\tau(x, y) \land A(y)) &= \text{preceding sibling} :: \ast \{ A \} \\
\exists y (\tau(x, y) \land A(y)) &= \text{child} :: \ast / \text{ descendant} :: \ast \{ A \} \\
\exists y (\tau(x, y) \land A(y)) &= \text{parent} :: \ast / \text{ ancestor} :: \ast \{ A \}.
\end{align*}
\]
6. CONCLUSIONS

We have given semantic characterizations of navigational XPath in terms of natural fragments of first order logic. Besides that, we looked at the connectives of XPath and also argued that they are nicely chosen. We can conclude that the navigational part of XPath is a very well designed language. On ordered trees it corresponds to a natural fragment of first order logic. This holds both for the sets of nodes and the sets of paths definable in XPath. The characterization in terms of conjunctive queries seems especially useful, as this is a very natural and user-friendly way to specify relations.

The only negative aspect we discovered concerning XPath is that it is not closed under complementation. We turn to that now, by considering two obvious points we did not address here yet:

- Is every first order definable set of nodes definable in XPath?
- Is every first order definable set of paths definable in XPath?

By Theorem 10, both questions are answered negatively. [15] showed that expanding XPath with conditional axis relations yields expressiveness completeness for answer sets. In an unpublished manuscript we have also shown that the same language is complete for expressing every first order definable set of paths.

Further investigations are needed for fragments not containing all the Booleans and we are working on that.

7. ACKNOWLEDGMENTS

We want thank Loredana Afanasiev, Jan Hidders, and Petrucio Viana for valuable feedback.

Maarten Marx was supported by the Netherlands Organization for Scientific Research (NWO), under project number 612.000.106. Maarten de Rijke was supported by grants from NWO, under project numbers 365-20-005, 612.069.006, 612.000.106, 220-80-001, 612.000.207, and 612.066.302.

8. REFERENCES


