Semantic 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

$$\mathsf{axis} :: \mathit{node\_label}[\mathsf{filter}],$$

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 $\mathit{node\_label}$, and the expression filter evaluates to true at n'. Alternatively, $\mathsf{axis} :: \mathit{node\_label}[\mathsf{filter}]$ 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

$$(1) \qquad \exists y(x\,\mathtt{descendant} :: A\,y \wedge \neg\, x\,\mathtt{descendant} :: B/\mathtt{descendant} :: *\,y).$$

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[2] 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,[3] it allows sequencing and taking unions of path expressions and full booleans 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

$$
\begin{aligned}
\text{locpath} &::= \text{axis ‘::’ntst} \mid \text{axis ‘::’ ntst ‘[’fexpr‘]’} \mid \\
&\quad\ \text{‘/’ locpath} \mid \text{locpath ‘/’ locpath} \mid \\
&\quad\ \text{locpath ‘|’ locpath} \\
\text{fexpr} &::= \text{locpath} \mid \texttt{not}\ \text{fexpr} \mid \text{fexpr}\ \texttt{and}\ \text{fexpr} \mid \\
&\quad\ \text{fexpr}\ \texttt{or}\ \text{fexpr} \\
\text{axis} &::= \texttt{self} \mid \texttt{child} \mid \texttt{parent} \mid \\
&\quad\ \texttt{descendant} \mid \texttt{descendant\_or\_self} \mid \\
&\quad\ \texttt{ancestor} \mid \texttt{ancestor\_or\_self} \mid \\
&\quad\ \texttt{following\_sibling} \mid \texttt{preceding\_sibling} \mid \\
&\quad\ \texttt{following} \mid \texttt{preceding},
\end{aligned}
$$

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*[4] (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 $R_{\Downarrow}$, and the immediate_right_sibling relation, denoted by $R_{\Rightarrow}$. Together with their inverses $R_{\Uparrow}$ and $R_{\Leftarrow}$ they are used to interpret the axis relations. We denote such trees as first order structures $(N, R_{\Downarrow}, R_{\Rightarrow}, P_i)_{i\in\Lambda}$.

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

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 $\mathfrak{M}$ a model, $\mathit{answer}_{\mathfrak{M}}(R) = \{n \mid \exists n'(n', n) \in [\![R]\!]_{\mathfrak{M}}\}$. 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 $\mathfrak{M}$,

$$(2) \qquad \mathit{answer}_{\mathfrak{M}}(R) = \{n \mid \mathcal{E}_{\mathfrak{M}}(n, A) = \mathit{true}\}.$$

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 $\mathfrak{M}$, and for every node n,

$$(3) \qquad \mathfrak{M} \models \phi(n) \text{ if and only if } \mathcal{E}_{\mathfrak{M}}(n, A) = \mathit{true}.$$

First, though, we fix our terminology.

the signature in one model are matched by a corresponding move in the other model. We say that a binary relation α is safe for bisimulations if this second clause also holds for α. For instance, if the signature contains R and S, then $R \circ S$, R^* and even $(R \circ S)^*$ are all safe for bisimulations.

[2]The counterdomain of a binary relation R (notation: $\sim R$) is the set $\{(x, y) \mid x = y \wedge \neg\exists z\, xRz\}$.

[3]This is without loss of generality, as instead of modeling attributes as distinct axes, as in the standard XML model, we may assign multiple labels to each node, representing whether a certain attribute-value pair is true at that node.

[4]A sibling ordered tree is a structure isomorphic to $(N, R_{\Downarrow}, R_{\Rightarrow})$ 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 $n, n' \in N$, $nR_{\Downarrow}n'$ holds iff $n' = n \cdot k$ for k a natural number; $nR_{\Rightarrow}n'$ holds iff $n = s \cdot k$ and $n' = s \cdot k + 1$.

$$
\begin{array}{rcl}
[\![\texttt{axis} :: P_i]\!]_{\mathfrak{M}} & = & \{(n,n') \mid n[\![\texttt{axis}]\!]_{\mathfrak{M}} n' \text{ and } P_i(n')\} \\
[\![\texttt{axis} :: P_i[e]]\!]_{\mathfrak{M}} & = & \{(n,n') \mid n[\![\texttt{axis}]\!]_{\mathfrak{M}} n' \text{ and } P_i(n') \text{ and } \mathcal{E}_{\mathfrak{M}}(n',e)\} \\
[\![/\text{locpath}]\!]_{\mathfrak{M}} & = & \{(n,n') \mid (\mathsf{root},n') \in [\![\text{locpath}]\!]_{\mathfrak{M}}\} \\
[\![\text{locpath}/\text{locpath}]\!]_{\mathfrak{M}} & = & [\![\text{locpath}]\!]_{\mathfrak{M}} \circ [\![\text{locpath}]\!]_{\mathfrak{M}} \\
[\![\text{locpath} \mid \text{locpath}]\!]_{\mathfrak{M}} & = & [\![\text{locpath}]\!]_{\mathfrak{M}} \cup [\![\text{locpath}]\!]_{\mathfrak{M}} \\
\\
[\![\texttt{self}]\!]_{\mathfrak{M}} & := & \{(x,y) \mid x = y\} \\
[\![\texttt{child}]\!]_{\mathfrak{M}} & := & R_{\Downarrow} \\
[\![\texttt{parent}]\!]_{\mathfrak{M}} & := & [\![\texttt{child}]\!]_{\mathfrak{M}}{}^{-1} \\
[\![\texttt{descendant}]\!]_{\mathfrak{M}} & := & [\![\texttt{child}]\!]_{\mathfrak{M}}{}^{+} \\
[\![\texttt{descendant\_or\_self}]\!]_{\mathfrak{M}} & := & [\![\texttt{child}]\!]_{\mathfrak{M}}{}^{*} \\
[\![\texttt{ancestor}]\!]_{\mathfrak{M}} & := & [\![\texttt{descendant}]\!]_{\mathfrak{M}}{}^{-1} \\
[\![\texttt{ancestor\_or\_self}]\!]_{\mathfrak{M}} & := & [\![\texttt{descendant\_or\_self}]\!]_{\mathfrak{M}}{}^{-1} \\
[\![\texttt{following\_sibling}]\!]_{\mathfrak{M}} & := & R_{\Rightarrow}^{+} \\
[\![\texttt{preceding\_sibling}]\!]_{\mathfrak{M}} & := & [\![\texttt{following\_sibling}]\!]_{\mathfrak{M}}{}^{-1} \\
[\![\texttt{following}]\!]_{\mathfrak{M}} & := & [\![\texttt{ancestor\_or\_self}]\!]_{\mathfrak{M}} \circ [\![\texttt{following\_sibling}]\!]_{\mathfrak{M}} \circ \\
 & & [\![\texttt{descendant\_or\_self}]\!]_{\mathfrak{M}} \\
[\![\texttt{preceding}]\!]_{\mathfrak{M}} & := & [\![\texttt{ancestor\_or\_self}]\!]_{\mathfrak{M}} \circ [\![\texttt{preceding\_sibling}]\!]_{\mathfrak{M}} \circ \\
 & & [\![\texttt{descendant\_or\_self}]\!]_{\mathfrak{M}} \\
\\
\mathcal{E}_{\mathfrak{M}}(n, \text{locpath}) = \mathit{true} & \iff & \exists n' : (n,n') \in [\![\text{locpath}]\!]_{\mathfrak{M}} \\
\mathcal{E}_{\mathfrak{M}}(n, \text{fexpr}_1 \texttt{ and } \text{fexpr}_2) = \mathit{true} & \iff & \mathcal{E}_{\mathfrak{M}}(n, \text{fexpr}_1) = \mathit{true} \text{ and } \mathcal{E}_{\mathfrak{M}}(n, \text{fexpr}_2) = \mathit{true} \\
\mathcal{E}_{\mathfrak{M}}(n, \text{fexpr}_1 \texttt{ or } \text{fexpr}_2) = \mathit{true} & \iff & \mathcal{E}_{\mathfrak{M}}(n, \text{fexpr}_1) = \mathit{true} \text{ or } \mathcal{E}_{\mathfrak{M}}(n, \text{fexpr}_2) = \mathit{true} \\
\mathcal{E}_{\mathfrak{M}}(n, \texttt{not } \text{fexpr}) = \mathit{true} & \iff & \mathcal{E}_{\mathfrak{M}}(n, \text{fexpr}) = \mathit{false}.
\end{array}
$$

Table 1: The semantics of Core Xpath.

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 `child`, `descendant` and `following_sibling`. For `moves` a subset of these three moves, we use $\mathrm{FO}^2[\texttt{moves}]$ 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 ϕ. Moreover, $\phi(x)$ is written in the signature consisting of Λ and `moves`. When interpreted on a tree, a $\mathrm{FO}^2[\texttt{moves}]$ 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 $\mathrm{FO}^2[\texttt{descendant}, \texttt{child}, \texttt{following\_sibling}]$ in one free variable.*

More precisely, for every formula $\phi(x)$ in $\mathrm{FO}^2[\texttt{descendant}, \texttt{child}, \texttt{following\_sibling}]$ with unary predicates from Λ, there exists a Core XPath expression R written with node tags Λ, such that on every tree $\mathfrak{M}$, $answer_{\mathfrak{M}}(R) = \{n \mid \mathfrak{M} \models \phi(n)\}$, 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 `following` and `preceding` axis and without absolute expressions such that for each model $\mathfrak{M}$,*

$$answer_{\mathfrak{M}}(R) = \{n \mid \mathcal{E}_{\mathfrak{M}}(n, A) = \mathit{true}\}.$$

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:

$$\texttt{following} :: P_i[A] \equiv \texttt{ancestor\_or\_self} :: * \; /\texttt{following\_sibling} :: * \; /\texttt{descendant\_or\_self} :: P_i[A] \tag{4}$$

$$\texttt{preceding} :: P_i[A] \equiv \texttt{ancestor\_or\_self} :: * \; /\texttt{preceding\_sibling} :: * \; /\texttt{descendant\_or\_self} :: P_i[A] \tag{5}$$

$$/R \equiv \texttt{ancestor\_or\_self} :: *[\texttt{not parent} :: *]/R \tag{6}$$

The result is an equivalent formula without `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:

$$\begin{array}{rcl} (S \mid T)^{-1} & \equiv & S^{-1} \mid T^{-1} \\ (S/T)^{-1} & \equiv & T^{-1}/S^{-1} \\ (\texttt{axis} :: P_i[B])^{-1} & \equiv & \texttt{self} :: P_i[B]/\texttt{axis}^{-1} :: *, \end{array}$$

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, $n \in answer_{\mathfrak{M}}(R)$ iff[5] $\mathcal{E}(n, A)$ equals true. Whence the lemma. QED

LEMMA 4. *The answer set of any Core XPath expression can be defined by a formula of* FO^2[`descendant`, `child`, `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. QED

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* FO_2[`descendant`, `child`, `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 `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 `self` $:: A$. Thus, for instance, $\exists y(y\,\texttt{child}\,x \wedge A(x))$ translates to `parent` $:: *$ [`self` $:: A$]. Translating $\phi(x)$, the result of this process is a filter expression A for which in any model, for every node n, $\mathfrak{M}$, $\mathcal{E}(n, A)$ equals true iff $\mathfrak{M} \models \phi(n)$. QED

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 argumentation 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 $n \in answer_{\mathfrak{M}}(R)$ 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

$$Q(x, y) :- R_1 \wedge \ldots \wedge R_n \wedge \phi_1 \wedge \ldots \wedge \phi_m,$$

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

$$\begin{array}{rcl} Q(x,y) & :- & z\,\texttt{descendant}\,x, z\,\texttt{following\_sibling}\,z', \\ & & z'\,\texttt{descendant}\,y, P_1(z), P_2(y), \end{array}$$

which is equivalent to the XPath expression

$$\texttt{ancestor} :: P_1/\texttt{following\_sibling} :: */\texttt{descendant} :: P_2.$$

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,

$$\texttt{descendant} :: P_2 \mid \texttt{parent} :: */\texttt{ancestor} :: P_1$$

is equivalent to the union of the two queries

$$Q(x, y) : -x\,\texttt{descendant}\,y, P_2(y)$$

and

$$Q(x, y) : -z\,\texttt{child}\,x, z\,\texttt{ancestor}\,y, p_1(y).$$

¿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* $\mathfrak{M}$, $\{(n, n') \mid \mathfrak{M} \models Q(n, n')\} = [\![R]\!]_{\mathfrak{M}}$.

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* $\mathfrak{M}$, $[\![A]\!]_{\mathfrak{M}} \cap [\![B]\!]_{\mathfrak{M}} = [\![C]\!]_{\mathfrak{M}}$.

$\tau(x,y)$	$\exists y(\tau(x,y) \wedge A(y))$
$x = y$	$\texttt{self} :: * [A']$
$x\ \texttt{child}\ y$	$\texttt{child} :: * [A']$
$y\ \texttt{child}\ x$	$\texttt{parent} :: * [A']$
$x\ \texttt{following\_sibling}\ y$	$\texttt{following\_sibling} :: * [A']$
$y\ \texttt{following\_sibling}\ x$	$\texttt{preceding\_sibling} :: * [A']$
$x\ \texttt{descendant}\ y \wedge \neg x\ \texttt{child}\ y$	$\texttt{child} :: */\texttt{descendant} :: * [A']$
$y\ \texttt{descendant}\ x \wedge \neg y\ \texttt{child}\ x$	$\texttt{parent} :: */\texttt{ancestor} :: * [A']$.

Table 2: Order types and their translation

Proof. This follows immediately from Theorem 8 and Proposition 7. QED

Theorem 10. *Core XPath is not closed under complementation.*

Proof. Suppose it was. We will derive a contradiction. Then (1) would be expressible. (1) is equivalent to the first-order formula

(7) $\exists y(x\ \texttt{descendant}\ y \wedge A(y) \wedge$
$\forall z((x\ \texttt{descendant}\ z \wedge z\ \texttt{descendant}\ y) \to \neg B(z)))$.

A standard argument shows that this set cannot be specified using less then 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

5. THE CONNECTIVES OF XPATH

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 ('|') 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) $$R ::= \texttt{axis} \mid ?P_i \mid R/R \mid R|R \mid \sim R,$$

for $\texttt{axis}$ one of Core XPath's axis relations, for P_i a tagname, and the following meaning for the two new connectives:

$$\begin{aligned} [\![?P_i]\!]_{\mathfrak{M}} &= \{(x,x) \mid \text{the tag of } x \text{ is } P_i\} \\ [\![\sim R]\!]_{\mathfrak{M}} &= \{(x,y) \mid x = y \text{ and } \neg\exists z\ x[\![R]\!]_{\mathfrak{M}} z\}. \end{aligned}$$

We call this language SCX (short for *short core xpath*). $?P_i$ simply tests whether a node has tag P_i. Thus $\texttt{child} :: P_i$ can be written as $\texttt{child}/?P_i$. The unary operator $\sim$ is sometimes called *counterdomain*. For instance $\sim\texttt{child}$ defines the set of all pairs (x,x) for x a leaf, and $\sim\texttt{parent}$ 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 : Core XPath $\longrightarrow$ SCX and t_2 : SCX $\longrightarrow$ Core XPath such that for all models $\mathfrak{M}$, the following hold:*

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

Proof. Because the counterdomain of a relation R is definable in XPath as $\texttt{self} :: *[\text{not } R]$, every relation defined in (8) can be expressed as a Core XPath formula. For the other side, first observe that $\texttt{axis} :: P_i$ and $\texttt{axis}/?P_i$ 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 ? to all filter expressions (cf. Lemma 2.82 in [4]):

$$\begin{aligned} ?(\texttt{axis} :: P_i) &\equiv \sim\sim(\texttt{axis}/?P_i) \\ ?(\texttt{axis} :: *) &\equiv \sim\sim(\texttt{axis}/(?P_i \cup \sim?P_i) \\ ?(\texttt{axis} :: P_i[A]) &\equiv \sim\sim(\texttt{axis}/?P_i/?A) \\ ?(\text{not } A) &\equiv \sim?A \\ ?(A \text{ and } B) &\equiv ?A/?B \\ ?(A \text{ or } B) &\equiv ?A \mid ?B. \end{aligned}$$

A simple semantic argument shows the correctness of these equations. QED

So we can conclude that the "true" set of XPath connectives consists of testing a node tag, composition, union and counterdomain. This set of connectives between binary relations is closely connected to the notion of bisimulation, as exemplified in Theorem 12 below. Before we state it we need a couple of definitions.

For P a set of tag names, and R a set of relation names, let $B_{P,R}$ denote the P,R bisimulation relation. Let D, D' be XML tree models and $B_{P,R}$ a non-empty binary relation between the nodes of D and D'. We call $B_{P,R}$ a *P R bisimulation* if, whenever $xB_{P,R}y$, then the following conditions hold, for all relations $S \in R$,

tag x and y have the same tag names, for all tag names in P;

forth if there exists an $x' \in D$ such that xSx', then there exists a $y' \in D'$ such that ySy' and $x'By'$;

back similarly for $y' \in D'$.

Let $\alpha(x,y)$ be a first order formula in the signature with unary predicates P and binary relations R. We say that $\alpha(x,y)$ is *safe for P, R bisimulations* if the back and forth clauses of the bisimulation definition hold for $\alpha(x,y)$, for all P,R bisimulations. In words, if $\alpha(x,y)$ is safe for bisimulations, it acts lack a morphism with respect to bisimulations. It is easy to see that all relations defined in (8) are safe for bisimulations respecting the node tags and the atomic axis relations. The other direction is known as van Benthem's safety theorem (see [4] Theorem 2.83):

THEOREM 12 (VAN BENTHEM). *Let $\alpha(x,y)$ be as above. If $\alpha(x,y)$ is safe for P, R bisimulations it can be defined by the grammar in (8).*

Why is this result so important? XPath is a language in which we can specify relations between nodes, and in several applications (like XQuery) it is used in this way. Theorems 12 and 11 together guarantee that XPath is in a well defined sense *complete*: every relation which is safe for bisimulations respecting node tags and XPath's axis relations can be defined in XPath.

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[6] yields expressive completeness for answer sets. In an unpublihed 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

[1] L. Afanasiev, M. Francheschet, M. Marx, and M. de Rijke. CTL Model Checking for Processing Simple XPath Queries. In *Proceedings Temporal Representation and Reasoning (TIME 2004)*, 2004.

[2] M. Benedikt, W. Fan, and G. Kuper. Structural properties of XPath fragments. In *Proceedings. ICDT 2003*, 2003.

[3] G. Bex, S. Maneth, and F. Neven. A formal model for an expressive fragment of XSLT. *Information Systems*, 27(1):21–39, 2002.

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

[5] 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.

[6] World-Wide Web Consortium. XML path language (XPath): Version 1.0. http://www.w3.org/TR/xpath.html.

[7] World-Wide Web Consortium. XML path language (XPath): Version 2.0. http://www.w3.org/TR/xpath20/.

[8] World-Wide Web Consortium. XML schema part 1: Structures. http://www.w3.org/TR/xmlschema-1.

[9] World-Wide Web Consortium. XQuery 1.0: A query language for XML. http://www.w3.org/TR//xquery/.

[10] World-Wide Web Consortium. XSL transformations language (XSLT): Version 2.0. http://www.w3.org/TR/xslt20/.

[11] K. Etessami, M. Vardi, and Th. Wilke. First-order logic with two variables and unary temporal logic. In *Proceedings 12th Annual IEEE Symposium on Logic in Computer Science*, pages 228–235, Warsaw, Poland, 1997. IEEE.

[12] G. Gottlob and C. Koch. Monadic queries over tree-structured data. In *Proc. LICS*, Copenhagen, 2002.

[13] G. Gottlob, C. Koch, and R. Pichler. Efficient algorithms for processing XPath queries. In *Proc. of the 28th International Conference on Very Large Data Bases (VLDB 2002)*, 2002.

[14] G. Gottlob, C. Koch, and R. Pichler. The complexity of XPath query evaluation. In *PODS 2003*, pages 179–190, 2003.

[15] M. Marx. Conditional XPath, the first order complete XPath dialect. In *Proceedings of PODS'04*, 2004.

[16] G. Miklau and D. Suciu. Containment and equivalence for an XPath fragment. In *Proc. PODS'02*, pages 65–76, 2002.

[17] T. Milo, D. Suciu, and V. Vianu. Typechecking for XML transformers. In *Proceedings PODS*, pages 11–22. ACM, 2000.

[18] M. Murata. Extended path expressions for XML. In *Proceedings PODS*, 2001.

[19] F. Neven and T. Schwentick. Expressive and efficient pattern languages for tree-structured data. In *Proc. PODS*, pages 145–156. ACM, 2000.

[20] M. Vardi. Why is modal logic so robustly decidable? In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science 31*, pages 149–184. American Math. Society, 1997.

[21] V. Vianu. A Web odyssey: from Codd to XML. In *Proc. PODS*, pages 1–15. ACM Press, 2001.

[22] P. Wadler. Two semantics for XPath. Technical report, Bell Labs, 2000.

[6]A conditional axis relation is of the form $(\texttt{child} :: \mathrm{ntst}[\mathrm{fexpr}])^*$ which denotes the reflexive and transitive closure of the relations denoted by $\texttt{child} :: \mathrm{ntst}[\mathrm{fexpr}]$. Using this we can express the set of nodes in (1) by

$$\texttt{self} :: *[(\texttt{child} :: *[\texttt{not self} :: B])^*/\texttt{child} :: A].$$