First order paths in ordered trees

Maarten Marx

Informatics Institute University of Amsterdam

Abstract. We give two sufficient conditions on XPath like languages for having first order expressivity, meaning that every first order definable set of paths in an ordered node-labeled tree is definable in that XPath language. They are phrased in terms of expansions of navigational (sometimes called "Core") XPath. Adding either complementation, or the more elegant conditional paths is sufficient. A conditional path is an axis relation of the form `(one_step_axis::n[F])`$^+$, denoting the transitive closure of the relation expressed by `one_step_axis::n[F]`. As neither is expressible in navigational XPath we also give characterizations in terms of first order logic of the answer sets and the sets of paths navigational XPath can define. The first in terms of a suitable two variable fragment, the second in terms of unions of conjunctive queries.

1 Introduction

[11] showed how a simple addition to Core XPath led to expressive completeness: every first order definable set of nodes in an XML tree is definable as the answer set of an expression //[`fexpr`] in which the filter expression is generated by the following grammar:

$$\begin{array}{lll} \mathsf{step} & ::= & \mathsf{self} \mid \mathsf{child} \mid \mathsf{parent} \mid \mathsf{right} \mid \mathsf{left} \\ \mathsf{locpath} & ::= & \mathsf{step{::}ntst[fexpr]} \mid (\mathsf{step{::}ntst[fexpr]})^+ \\ \mathsf{fexpr} & ::= & \mathsf{locpath} \mid \mathsf{not}\ \ \mathsf{fexpr} \mid \mathsf{fexpr\ and\ fexpr}. \end{array}$$

Here ntst is a node test consisting of a tag name or the wild card $*$. The steps correspond to the four basic steps in an ordered tree. The semantics is as with standard Core XPath [5], with $(\cdot)^+$ interpreted as the transitive closure.

Although the choice of the syntax can be motivated by its close relation to temporal logic with since and until, it may still seem rather ad hoc. Moreover the result is really about the expressive power of filter expressions, rather than about location paths. In this paper we present additional evidence for the great expressive power of the construction $(\mathsf{step{::}ntst[fexpr]})^+$, and obtain an expressive completeness result for location paths. Extensive motivation for such a result can be found in [1].

In the above definition it was not needed to close the location path expressions under composition (the $'/'$) and union (the $'|'$). This is because we dealt with filter expressions only. When defining paths in a tree they are obviously needed. So in the following, assume that the language is closed under these two operations as well. We show that

1. any extension of Core XPath which is closed under complementation can define every first order definable set of paths;
2. the above defined language (called Conditional XPath) is closed under complementation, whence first order complete for expressing paths.

The first result states a sufficient condition for an XPath dialect having full first order expressivity. The second states that very little is needed to achieve it: allow unions and compositions of path expressions, and allow transitive closure of the simplest location path step::ntst[fexpr]. The first result is a corollary of a more general result which states that the class of ordered trees has the three variable property: every first order formula in at most three free variables is equivalent (on this class) to a first order formula in at most three free and bound (possibly reused) variables.

These results are about expansions of Core XPath. This language was defined by Gottlob, Koch and Pichler [5] as the logical core of XPath 1.0. Core XPath is strictly weaker than Conditional XPath, so the question remains which fragment of first order logic is picked out by Core XPath. It turns out that this is a very natural one indeed:

1. The answer sets definable in Core XPath are exactly those definable with first order formulas $\phi(x)$ which use only two (free and bound) variables in a signature with predicates corresponding to the *child, descendant* and *following_sibling* relations.
2. The paths definable in Core XPath are exactly those which can be defined by unions of conjunctive queries consisting of the *child, descendant* and *following_sibling* relations and unary first order formulas as in item 1.
3. Core XPath is closed under intersection but not under complementation.

We thus give a precise characterization of both Core and Conditional XPath, both in terms of defining answer sets and sets of paths. For general related work, we refer to [11] and to the conclusions. Specific relations are given in the running text.

The paper is organized as follows. Section 2 introduces the needed definitions. Section 3 contains all results and some of the more easy proofs. Section 4 is devoted to the proof of the most important result: closure of Conditional XPath under complementation. We motivate our work in the conclusion. To give complete proofs of the results presented here takes at least another 15 pages, which we don't have. This work has a number of interesting connections with temporal logic. These connections, together with further related work and fairly complete proofs are in the full version of the paper, which is available from the author. For related work, see also [11,13]. We note that the expressive completeness result for Conditional XPath's answer sets (shown in [11]) follows from the result presented here, but not conversely. The results about Core XPath have been presented at the Twente workshop on Database Management [13]. They are included here in order to give a complete picture.

2 Navigational XPath

We use an XPath syntax which is better suited for mathematical manipulation and easier to read when formulas tend to get large. (And they will ...) The relation with the official W3C syntax should be clear.

XPath languages are two sorted languages, defined by mutual recursion. There are formulas denoting sets of nodes (called *node wffs*), and formulas denoting a binary relation between nodes (called *path wffs*). An XPath *step* is one of the following four atomic relation symbols

$$\mathsf{step} ::= \mathsf{child} \mid \mathsf{parent} \mid \mathsf{right} \mid \mathsf{left}.$$

We define Core XPath and Conditional XPath. They differ only in the operations allowed on path wffs. Let some set of (propositional) variables be fixed. They are denoted by p_i. We do not put any restriction on the number of variables in our language. The node wffs are generated by

$$\mathsf{node\_wff} ::= p_i \mid \top \mid \langle \mathsf{path\_wff} \rangle \mid \neg \mathsf{node\_wff} \mid \mathsf{node\_wff} \wedge \mathsf{node\_wff}.$$

Here $\top$ denotes the predicate which always evaluates to true. The path wffs of Core XPath are generated by

$$\mathsf{path\_wff} ::= \mathsf{step} \mid \mathsf{step}^+ \mid ?\mathsf{node\_wff} \mid \mathsf{path\_wff}/\mathsf{path\_wff} \mid \mathsf{path\_wff} \cup \mathsf{path\_wff}.$$

The path wffs of Conditional XPath differ only in that we allow $(\mathsf{step}/?\mathsf{node\_wff})^+$ instead of just step^+. We call this construction a *conditional path* and the language derives its name from it. The main purpose of conditional paths is to define an until like relation. For instance, the relation between a node n and it's descendant n' at which A holds, and for which at all nodes strictly in between n and n' B holds is defined by

$$(\mathsf{child}/?B)^*/\mathsf{child}/?A.$$

Here and elsewhere we use R^* as an abbreviation of $R^+ \cup ?\top$, denoting the transitive reflexive closure of R. We use variables R, S, T for path wffs and A, B, C for node wffs. The differences with the standard XPath syntax are small. Our node wffs correspond to XPath's filter expressions. Our formulas $?\mathsf{node\_wff}$ (called *tests*) mean the same as XPath's $\mathsf{self}::*[\mathsf{node\_wff}]$. We abolished the two different tests on nodes in XPath, and capture node tests as follows:

$$\mathsf{axis}::A[F] \equiv \mathsf{axis}::*[\mathsf{self}::A \wedge F] \equiv \mathsf{axis}/?(A \wedge F).$$

To make the language context-free, we use $\langle \mathsf{path\_wff} \rangle$ inside node wffs instead of just $\mathsf{path\_wff}$. For axis one of step, step^+, $(\mathsf{step}/?A)^+$, we often write $\mathsf{axis}?\mathsf{node\_wff}$ instead of $\mathsf{axis}/?\mathsf{node\_wff}$. Just as in XPath, we consider these expressions as the basic expressions of the language. The relation between the just defined language and XPath 1.0 is exemplified in Table 1 in which some expressions in our notation are given also as equivalent XPath expressions.

$\mathsf{child} :: p_i$	$\mathsf{child}/?p_i$
$\mathsf{child} :: p_i[\mathtt{descendant} :: *]$	$\mathsf{child}/?p_i/?\langle\mathsf{child}^+\rangle$ or $\mathsf{child}/?(p_i \wedge \langle\mathsf{child}^+\rangle)$
$/\mathtt{descendant} :: p_i$	$?\neg\langle\mathsf{parent}\rangle/\mathsf{child}^+/?p_i$
$\mathsf{child} :: *$	child
$\mathtt{self} :: p_i[\mathsf{child}]$	$?(p_i \wedge \langle\mathsf{child}\rangle)$
$\mathtt{preceding} :: p_i$	$\mathsf{parent}^*/\mathsf{left}^+/\mathsf{child}^*/?p_i$.

Table 1. Equivalent XPath 1.0 and Core XPath expressions.

The semantics of XPath expressions is given with respect to *node labeled sibling ordered trees*[1] (trees 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 $(N, R_\downarrow, R_\rightarrow, P_i)_{i\in\omega}$.

Remark 1. Unlike in most of the literature on XPath we do not restrict the class of structures to trees corresponding to XML documents. So our trees can be infinitely deep, infinitely branching and may contain multiple atomic labels at each node. All our results apply to document trees as well. This is because our theorems are of the following form: for every first order formula ϕ, there is an XPath expression α such that on all trees, the denotations of ϕ and α coincide.

Remark 2. Although we borrowed the name Core XPath from [5], our language is slightly more expressive, due to the availability of the left and right axis relations. Arguably, these must be available in an XPath dialect which calls itself *navigational*.

Given a tree $\mathfrak{M}$ and an expression R, the denotation or meaning of R in $\mathfrak{M}$ is written as $[\![R]\!]_{\mathfrak{M}}$. As promised, path wffs denote sets of pairs, and node wffs sets of nodes. Table 2 contains the definition of $[\![\cdot]\!]_{\mathfrak{M}}$. The equivalence with the W3C syntax and semantics (cf., e.g., [5,18]) should be clear.

Let us spell out the semantics of the conditional axis relation, as it does not occur in standard navigational XPath. The path wff $(\mathsf{child}?A)^+$ denotes all pairs (n, n') for which there exists a finite sequence of nodes $n = n_1 \dots n_k = n'$ $(k > 1)$ such that for all i, n_{i+1} is a child of n_i and A is true at all n_j $(j > 1)$. As an example of its expressive power, consider the *next frontier node* relation which holds between leaves which are consecutive in document order. Let us use the following abbreviations:

$$\mathsf{leaf} = \neg\langle\mathsf{child}\rangle, \quad \mathsf{first} = \neg\langle\mathsf{left}\rangle, \quad \mathsf{last} = \neg\langle\mathsf{right}\rangle.$$

[1] 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 $n' = s \cdot k + 1$.

$$
\begin{aligned}
[\![p_i]\!]_{\mathfrak{M}} &= \{n \mid \mathfrak{M} \models P_i(n)\} \\
[\![\top]\!]_{\mathfrak{M}} &= \{n \mid n \in \mathfrak{M}\} \\
[\![\langle R \rangle]\!]_{\mathfrak{M}} &= \{n \mid \exists n', (n, n') \in [\![R]\!]_{\mathfrak{M}}\} \\
[\![\neg A]\!]_{\mathfrak{M}} &= \{n \mid n \notin [\![A]\!]_{\mathfrak{M}}\} \\
[\![A \wedge B]\!]_{\mathfrak{M}} &= [\![A]\!]_{\mathfrak{M}} \cap [\![B]\!]_{\mathfrak{M}}.
\end{aligned}
$$

$$
\begin{aligned}
[\![\mathsf{child}]\!]_{\mathfrak{M}} &= R_{\downarrow} \\
[\![\mathsf{parent}]\!]_{\mathfrak{M}} &= [\![\mathsf{child}]\!]^{-1}_{\mathfrak{M}} \\
[\![\mathsf{right}]\!]_{\mathfrak{M}} &= R_{\rightarrow} \\
[\![\mathsf{left}]\!]_{\mathfrak{M}} &= [\![\mathsf{right}]\!]^{-1}_{\mathfrak{M}}
\end{aligned}
$$

$$
\begin{aligned}
[\![R^+]\!]_{\mathfrak{M}} &= [\![R]\!]^+_{\mathfrak{M}} \ \ (= [\![R]\!]_{\mathfrak{M}} \cup ([\![R]\!]_{\mathfrak{M}} \circ [\![R]\!]_{\mathfrak{M}}) \cup ([\![R]\!]_{\mathfrak{M}} \circ [\![R]\!]_{\mathfrak{M}} \circ [\![R]\!]_{\mathfrak{M}}) \cup \dots) \\
[\![?A]\!]_{\mathfrak{M}} &= \{(n, n) \mid n \in [\![A]\!]_{\mathfrak{M}}\} \\
[\![R/S]\!]_{\mathfrak{M}} &= [\![R]\!]_{\mathfrak{M}} \circ [\![S]\!]_{\mathfrak{M}} \\
[\![R \cup S]\!]_{\mathfrak{M}} &= [\![R]\!]_{\mathfrak{M}} \cup [\![S]\!]_{\mathfrak{M}}.
\end{aligned}
$$

Table 2. The semantics of Core and Conditional XPath.

Then the next frontier node relation is definable as the path wff

$$?\mathsf{leaf}/[?\neg\mathsf{last} \cup (?\mathsf{last}/\mathsf{parent})^+]/\mathsf{right}/(\mathsf{child}?\mathsf{first})^*/?\mathsf{leaf}. \tag{1}$$

Here $(?\mathsf{last}/\mathsf{parent})^+$ abbreviates $?\mathsf{last}/(\mathsf{parent}?\mathsf{last})^*/\mathsf{parent}$. We finish with a useful result. For R a path wff, define the *converse* of R, denoted by R^{-1} with meaning $[\![R^{-1}]\!]_{\mathfrak{M}} = \{(n', n) \mid (n, n') \in [\![R]\!]_{\mathfrak{M}}\}$.

Proposition 1. *The path wffs of both Core and Conditional XPath are closed under taking converses.*

3 First order characterizations of XPath

This section contains all our results: first order characterizations of both the node wffs and the path wffs of Core and Conditional XPath, as well as sufficient conditions for first order expressivity.

Let FO^{tree} denote the first-order language over the signature with binary predicates $\{R_{\Downarrow}, R_{\Rightarrow}\}$ and countably many unary predicates P_i. FO^{tree} is interpreted on ordered trees in the obvious way: $R_{\Downarrow}$ is interpreted by the transitive closure of the child relation, and $R_{\Rightarrow}$ is interpreted by the transitive closure of the right_sibling relation. Note that both one step relations are first order definable from $R_{\Downarrow}$ and $R_{\Rightarrow}$.

3.1 Sets of nodes

The answer set of an XPath path expression R consists of the range of R, or the nodes which are reachable from some node by R [5,3]. The main result of [11] stated that every first order definable set of nodes is definable as the answer set of some Conditional XPath path expression. Here we give a characterization

of Core XPath's expressions as the two variable fragment[2] of first order logic in an expanded signature. In FO^{tree} we can define the one step axis relations from the transitive relations using three variables[3]. With two variables this is not possible, hence we should expand the signature with relations $R_{\downarrow}$ and $R_{\rightarrow}$ corresponding to the child and to the right_sibling axis, respectively. Let FO_2^{tree} denote the restriction of FO^{tree} in this expanded signature to the two variable fragment.

Theorem 1. **[13]** *(1) The answer sets of Core XPath path expressions are exactly the sets definable in FO_2^{tree}.*
(2) FO_2^{tree} formulas in one free variable and Core XPath's node wffs are equally expressive.

The hard direction follows more or less directly from the argument used to show a similar statement for linear orders —characterizing temporal logic with only unary temporal connectives— by Etessami, Vardi and Wilke [4]. The proof shows that a similar statement holds for the version of Core XPath of Gottlob, Koch and Pichler [5] which does not have the right_ and left_sibling axis but just their transitive closures. That language can define each set definable in FO_2^{tree} without the right_sibling relation.

Proof. Because the path wffs of Core XPath are closed under taking inverses, for every path wff R there exists a node wff A such that the answer set of R equals the denotation of A in every model. Thus we need only work with the node wffs and only prove the second equivalence in the theorem. By the standard translation well known from modal logic each node wff translates into a one free variable FO_2^{tree} formula (cf., [17] which takes care to use only two variables). The translation is just the definition from Table 2 written in first order logic. This takes care of the easy direction.

For the other direction, let $\phi(x)$ be a first order formula. We want a node wff A such that for every tree $\mathfrak{M}$, $\{n \mid \mathfrak{M} \models \phi(n)\} = [\![A]\!]_{\mathfrak{M}}$. The proof is a copy of the one for linear temporal logic in [4] (Theorem 1). The only real change needed is in the set of order types: they are given in the left hand side of Table 3, together with the needed translations (A' denotes the translation of A).

Remark 3. The answer sets of the path wffs of both Core and Conditional XPath have a first order characterization. An interesting question is how the sizes of the first order formulas and their corresponding equivalent XPath node wffs compare. For conditional XPath, the blow up is non elementary and this is unavoidable [11]. For Core XPath, it is much better. The blow up is "only" single exponential, which is also unavoidable [4]. The difference can be explained as follows. For Core XPath, we translate first order formulas in at most two variables

[2] With the two variable fragment we mean the set of formulas in which at most two variables may occur. Variables might be reused. Thus $\exists y \exists z(xR_{\Downarrow}y \wedge yR_{\Downarrow}z \wedge P(z))$ is not in the two variable fragment, but it is equivalent to $\exists y(xR_{\Downarrow}y \wedge \exists x(yR_{\Downarrow}x \wedge P(x)))$ which is equivalent to the node wff $\langle \mathsf{child}^+/\mathsf{child}^+/?P \rangle$.

[3] For instance, $x\mathsf{child}y$ is defined as $xR_{\Downarrow}y \wedge \neg\exists z(xR_{\Downarrow}z \wedge zR_{\Downarrow}y)$.

$\tau(x,y)$	$\exists y(\tau(x,y) \wedge A(y))$
$x = y$	A'
$x\, R_{\downarrow}\, y$	$\langle \mathsf{child}?A' \rangle$
$y\, R_{\downarrow}\, x$	$\langle \mathsf{parent}?A' \rangle$
$x\, R_{\rightarrow}\, y$	$\langle \mathsf{right}?A' \rangle$
$y\, R_{\rightarrow}\, x$	$\langle \mathsf{left}?A' \rangle$
$x\, R_{\Rightarrow}\, y \wedge \neg x\, R_{\rightarrow}\, y$	$\langle \mathsf{right}/\mathsf{right}^+?A' \rangle$
$y\, R_{\Rightarrow}\, x \wedge \neg y\, R_{\rightarrow}\, x$	$\langle \mathsf{left}/\mathsf{left}^+?A' \rangle$
$x\, R_{\Downarrow}\, y \wedge \neg x\, R_{\downarrow}\, y$	$\langle \mathsf{child}/\mathsf{child}^+?A' \rangle$
$y\, R_{\Downarrow}\, x \wedge \neg y\, R_{\downarrow}\, x$	$\langle \mathsf{parent}/\mathsf{parent}^+?A'. \rangle$

Table 3. Order types and their translations

into Core XPath wffs, which are again (equivalent to) first order formulas in at most two variables. For Conditional XPath, *every* first order formula (in one free variable) translates to a Conditional XPath node wff, which is (equivalent to) a first order formula in at most three variables.

3.2 Sets of paths

In the previous section we characterized the node sets that can be defined in XPath. (Defined either by means of a node wff, or equivalently as the answer set of a path wff). We next characterize path sets. It is known that on finite linear orders not every first order definable set can be defined using just two variables, even in the signature expanded with the child relation. Whence there are sets of nodes which are not definable in Core XPath, by the previous theorem. The following example goes back to Kamp [10]: the set of all nodes x such that

$$\exists y(x \ \mathtt{descendant}\ y \wedge A(y) \wedge \forall z((x \ \mathtt{descendant}\ z \wedge z \ \mathtt{descendant}\ y) \rightarrow B(z))). \quad (2)$$

Note that this set is expressible in Conditional XPath. By the node wff

$$\langle (\mathsf{child}?B)^*/\mathsf{child}?A \rangle.$$

It is also expressible in Core XPath expanded with a complementation operator $\overline{(\cdot)}$ on path wffs. The semantics of complementation is the standard boolean one: $[\![\overline{R}]\!]_{\mathfrak{M}} = \{(n,n') \mid (n,n') \notin [\![R]\!]_{\mathfrak{M}}\}$. Note that this operation is defined on *path wffs*, so it has nothing to do with the boolean negation $\neg$ which is only defined on *node wffs*. The set defined by (2) is also defined by the node wff

$$\langle \mathsf{child}^+?A \ \cap\ \overline{\mathsf{child}^+/?\neg B/\mathsf{child}^+} \rangle,$$

where $\cap$ is defined as usual from union and complementation.

Definition 1. *We say that an XPath language $\mathcal{L}$ is first order complete for path sets if for every FO^{tree} formula $\phi(x,y)$ there exists an $\mathcal{L}$ expression R such that for all trees $\mathfrak{M}$, $\{(n,n') \mid \mathfrak{M} \models \phi(n,n')\} = [\![R]\!]_{\mathfrak{M}}$.*

Theorem 2. *1. Any expansion of Core XPath which is closed under complementation is first order complete for path sets.*
2. Conditional XPath is closed under complementation, whence first order complete for path sets.

The proof of the second part will be sketched in the next section. The first part of the theorem is a corollary of a more general result:

Theorem 3. *Let C be the class of sibling ordered trees. Then, on C, every FO^{tree} formula in at most 3 free variables is equivalent to a FO^{tree} formula in the same signature which uses at most 3 free and bound variables.*

This theorem can be shown using Ehrenfeucht–Fraïssé pebble games from [9]. The first part of Theorem 2 can now be derived as follows.

Proof. Let L be an expansion as in the Theorem. Then L can express every binary relation expressible in Tarski's relation algebras. Tarski's relation algebras are algebras of the form $(A, \cup, \overline{(\cdot)}, \circ, (\cdot)^{-1}, \epsilon)$ with A a set of binary relations, and the operators have the standard set theoretic meaning. As the atoms of Core XPath are closed under $^{-1}$, the language is closed under it. ϵ is definable as $?\top$. The formalism of Tarski's relation algebras is equally expressive as FO^2_3, first order logic in a signature with at most binary relations symbols in which every formula contains at most three free and bound (possibly reused) variables and at most two free variables [16]. The desired result now follows from Theorem 3.

Remark 4. It is tempting to think that the three variable property for ordered trees is derivable from results about trees and CTL like languages, perhaps in case of finite trees. This is partly due to inconsistent terminology, partly because all notions are closely related. [8] give a clear picture of these notions and their relations and especially the relations that do not exist. In particular they show that the three variable property is strictly stronger than the property which says that every *sentence* is equivalent to a sentence in just three variables (called H-dimension). Also, the fact that there is an finite number of "one dimensional temporal connectives" with which we can express every first order formula $\phi(x)$ is independent from the three (in fact k) variable property. Such a set was established implicitly in [11]. That result does indeed imply that every FO^{tree} formula in *one* free variable is equivalent to a Conditional XPath node wff, whence to an FO^{tree} formula in at most three variables. Unfortunately, from this we cannot derive the same thing for formulas in *two* free variables.

We finish the section with a characterization of Core XPath's path wffs. For *positive* Core XPath (without negation but with disjunction of node wffs), such a characterization is provided in [1] and in [6]. It is exactly positive existential first order logic (of course in the signature expanded with the child and right-sibling relation).[4] The node wffs of Core XPath *with* negation have been characterized in Theorem 1. We can combine these results using a relaxation of positive existential first order logic, reminiscent of the Carin language [7].

[4] [1] does not consider the horizontal axis relations, but their proof is easily adjusted.

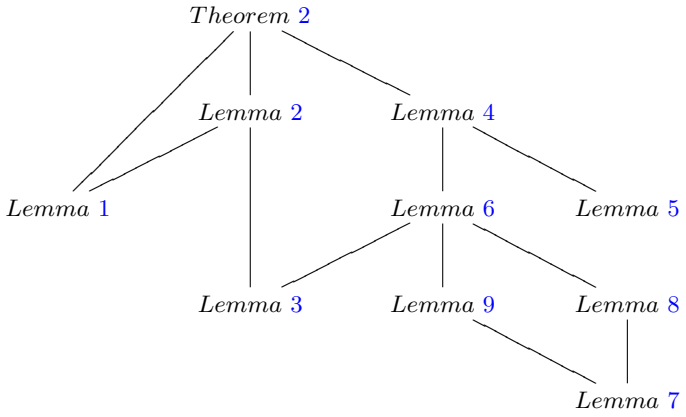

Fig. 1. Dependencies within the proof of Theorem 2.

Definition 2. *A first order Core XPath query is a formula of the form*

$$Q(x,y) :- \bigvee_i \bigwedge (R_1^i \wedge \ldots \wedge R_n^i \wedge A_1^i \wedge \ldots \wedge A_m^i), \tag{3}$$

in which the A_j^i are FO_2^{tree} formulas in one free variable, and the R_j^i are atomic formulas in the signature $\{R_{\Downarrow}, R_{\Rightarrow}, R_{\downarrow}, R_{\rightarrow}, =\}$

An example is

$$Q(x,y) : -z\, R_{\Downarrow}\, x, z\, R_{\Rightarrow}\, z', z'\, R_{\Downarrow}\, y, P_1(z), \forall x(y\, R_{\downarrow}\, x \rightarrow P_2(x)),$$

which is equivalent to the XPath expression

$$\mathsf{parent}^+?P_1/\mathsf{right}^+/\mathsf{child}^+?\neg\langle\mathsf{child}?\neg P_2\rangle.$$

So these are like unions of usual conjunctive queries, except that properties may contain negations.

Theorem 4. [13] *Every first order Core XPath query is equivalent to a Core XPath path wff and conversely.*

4 Closure under complementation

In this section we prove Theorem 2. Recall that in order to prove Theorem 2, we must find, given an arbitrary Conditional XPath path wff R, a Conditional XPath path wff R' which is equivalent to the complement of R. The proof is divided into a number of lemmas. The proof itself is not very difficult, but consists of a great number of small steps. The dependencies between the different lemmas are given in Figure 1.

In the first lemma, R is brought into a shape which is easier to handle. We need a bit of terminology. An *atom* is a path wff of the form $\mathsf{step}?A$, or

case	equivalent path wff
$a\,\mathsf{right}^+\,b$	$(\mathsf{right}?A_1)^+?(A_2 \wedge B_2 \wedge E)$
$a = b$	$?(A_2 \wedge B_2 \wedge E)$
$b\,\mathsf{right}^+\,a$	$?(A_2 \wedge E)/(\mathsf{left}?A_2)^+?B_2.$

Table 4. Separating $(\mathsf{right}?A_1)^+?B_1/(\mathsf{left}?A_2)^+?B_2$.

$(\mathsf{step}?B)^+?A$. A *test* is a path wff of the form $?A$. A *basic composition* is a test followed by a sequence of atoms separated by /'s. We call an atom *down* if it is of the form $\mathsf{down}?A$, or $(\mathsf{down}?B)^+?A$. Analogously, we define atoms being *up, right*, and *left*. A path wff *has form* T if it is a test. It has form D, U, R, L if it is a basic composition of down, up, right or left atoms, respectively. We say that a basic composition is *separated* if it has one of the following forms:

$$D, \quad U, \quad U^*/R/D^*, \quad U^*/L/D^*. \tag{4}$$

Here we use $U^*/R/D^*$ as an abbreviation for the forms U/R, R, R/D, $U/R/D$, and similarly for $U^*/L/D^*$.

The syntactic notion of separated composition has a semantic counterpart. On every tree, every wff of the form D is a subrelation of child^+ (or `descendant` in XPath terminology). Similarly, wffs of the form U, $U^*/R/D^*$ and $U^*/L/D^*$ are subrelations of parent^+ (`ancestor`), $\mathsf{parent}^*/\mathsf{right}^+/\mathsf{child}^*$ (`following`) and $\mathsf{parent}^*/\mathsf{left}^+/\mathsf{child}^*$ (`preceding`), respectively.

Lemma 1. *Every path wff is equivalent to a union of tests and separated basic compositions.*

The proof of the lemma consists of an case analysis of all compositions of two atoms. A representative example is

$$(\mathsf{right}?A_1)^+?B_1 \;/\; (\mathsf{left}?A_2)^+?B_2.$$

Suppose, for nodes a, b in some tree, $a\,(\mathsf{right}?A_1)^+?B_1/(\mathsf{left}?A_2)^+?B_2\,b$ holds. Then there is a node c such that $a\,(\mathsf{right}?A_1)^+?B_1\,c$ and $c\,(\mathsf{left}?A_2)^+?B_2\,b$. It follows that $a\,\mathsf{right}^+\,c$ and $b\,\mathsf{right}^+\,c$. There are three cases, depending on the relation between a and b. The corresponding path wffs are given in Table 4 for each case with E an abbreviation of the node wff

$$\langle\!\langle(\mathsf{right}?(A_1 \wedge A_2))^+/\mathsf{right}?(B_1 \wedge A_1)\rangle\!\rangle.$$

Thus the equivalent wff is a union of a T, an L and an R wff.

Lemma 1 is very helpful. It provides for a quick proof of two results which are useful later on.

Lemma 2. *Conditional XPath path wffs are closed under intersection.*

Lemma 3. *Let $Q(x,y)$ be a conjunctive query consisting of down atoms and a test $x?Ax$. $Q(x,y)$ is such that it implies $x\,\mathsf{child}^+\,y$ and for all existentially quantified variables z, $x\,\mathsf{child}^+\,z$ and $z\,\mathsf{child}^+\,y$. Then there exists an equivalent Conditional XPath path wff which is a union of basic compositions of form D. The same result holds for up, left and right atoms.*

Lemmas 1 and 2 reduce the task of proving Theorem 2 to showing

Lemma 4. *The complement of each separated basic composition is definable as a Conditional XPath path wff.*

The proof of Lemma 4 consists of an easy and a hard part, separated in Lemma 5 and 6, which are shown below.

Proof of Theorem 2.2. Let R be a path wff. Then by Lemma 1 $R \equiv \bigcup_i R_i$, with the R_i tests and separated basic compositions. Whence $\overline{R} \equiv \bigcap_i \overline{R_i}$. By Lemma 2, the path wffs are closed under intersection. The complement of a test $?A$ is equivalent to $?\neg A \cup \mathsf{not\_equal}$ with the latter abbreviating

$$\mathsf{child}^+ \cup \mathsf{parent}^+ \cup \mathsf{parent}^*/\mathsf{left}^+/\mathsf{child}^* \cup \mathsf{parent}^*/\mathsf{right}^+/\mathsf{child}^*.$$

By Lemma 4 each complement of a separated basic composition is equivalent to a path wff. Hence the theorem. QED

We rewrote the path wffs into separated basic compositions because it helps to reduce the reasoning to "lines" or "strings". For example, consider a path wff of the form U. Now if in a tree $a\,\overline{U}\,b$ holds, we can break into two cases:

- a is not below b;
- a is below b, but not $a\,U\,b$.

The first case is easy to express (using the partition again). The second is harder but, as U is a composition of up atoms, *we only need to reason about the elements in between a and b*. That is, we need to reason about a line segment. But not all separated path wffs are of this simple form, consisting of one direction. The next lemma however states that complements of these can be defined using complements of the uni-directed forms.

Lemma 5. *The complement of each separated basic composition is definable from path wffs and formulas of the form*

$$(\mathsf{child}^+ \cap \overline{D}),\ (\mathsf{parent}^+ \cap \overline{U}),\ (\mathsf{right}^+ \cap \overline{R}),\ \textit{and}\ (\mathsf{left}^+ \cap \overline{L}). \tag{5}$$

As an example, consider a path wff of the form U/R. The other forms are handled using the same argument. Then

$$\overline{U/R} \equiv (\overline{\mathsf{parent}^+/\mathsf{right}^+} \cap \overline{U/R}) \ \cup\ (\mathsf{parent}^+/\mathsf{right}^+ \cap \overline{U/R}). \tag{6}$$

Because $\models U/R \subseteq \mathsf{parent}^+/\mathsf{right}^+$, the first disjunct is equivalent to $\overline{\mathsf{parent}^+/\mathsf{right}^+}$, which is equivalent to

$$\mathsf{child}^* \cup \mathsf{parent}^*/\mathsf{left}^+/\mathsf{child}^* \cup \mathsf{parent}^+ \cup \mathsf{right}^+/\mathsf{child}^* \cup \mathsf{parent}^+/\mathsf{right}^+/\mathsf{child}^+. \tag{7}$$

For the second disjunct, we use the following equation:

$$\mathsf{parent}^+/\mathsf{right}^+ \cap\ \overline{U/R} \equiv (\mathsf{parent}^+ \cap \overline{U})/\mathsf{right}^+\ \cup\ \mathsf{parent}^+/(\mathsf{right}^+ \cap \overline{R}). \quad (8)$$

The left to right direction is a validity for all relations. The other direction is not, but it holds because the models are trees.

All the preparation has been done, we can start the real work. We just have to define the relations in (5) as Conditional XPath path wffs.

Lemma 6. *Each relation in* (5) *is definable as a Conditional XPath path wff.*

We will define these relations as conjunctive queries of the form specified in Lemma 3. This is sufficient by that Lemma. We use the fact (Proposition 1) that Conditional XPath is closed under conversion (denoted by R^{-1}) to reduce the number of cases to two. Consider $\mathsf{parent}^+ \cap \overline{U}$. Then

$$\mathsf{parent}^+ \cap \overline{U} \equiv ((\mathsf{parent}^+ \cap \overline{U})^{-1})^{-1} \equiv ((\mathsf{parent}^+)^{-1} \cap \overline{U^{-1}})^{-1} \equiv \bigcap(\mathsf{child}^+ \cap \overline{D})^{-1}.$$

The last equivalence holds because the converse of a composition of form U is a union of compositions of form D. We can similarly relate the L and R forms. By Proposition 1 path wffs are closed under $(\cdot)^{-1}$, thus it is sufficient to show the lemma for $\mathsf{child}^+ \cap \overline{D}$ and $\mathsf{right}^+ \cap \overline{R}$. The argument is identical in both cases. For concreteness, we consider the case for down compositions.

Let R be of the form $\mathsf{child}^+ \cap \overline{D}$. All our arguments are semantical, thus assume xRy holds, for x, y nodes in an arbitrary model.

To reduce the number of cases, we use a notion well known from temporal logic. For A, B node wffs, define the path wff $\mathsf{until}(A, B)$ with the semantics

$$x\,\mathsf{until}(A, B)\,y \iff xR_{\Downarrow}y \wedge A(y) \wedge \forall z(x\,R_{\Downarrow}\,z\,R_{\Downarrow}\,y \to B(z)).$$

Please note that $\mathsf{until}(A, B)$ is a path wff, whence denotes a set of pairs, unlike its use in temporal logic. Temporal logic is a *one-sorted* formalism, containing only node wffs. The until formula from temporal logic, denoting a set of points is of course expressed in our formalism as $\langle\mathsf{until}(A, B)\rangle$.

Both down atoms are expressible as an until formula: $\mathsf{child}?A \equiv \mathsf{until}(A, \neg\top)$ and $(\mathsf{child}?B)^+?A \equiv \mathsf{until}(A \wedge B, B)$. Thus it is sufficient to show how to define $\mathsf{child}^+ \cap \overline{?C/R}$, for R a composition of until formulas, and C an arbitrary test. We call such formulas *until wffs*. In order to increase readability we use $<$ and $\leq$ instead of child^+ and child^*, respectively. We define complementation by a case distinction. The first case is when there is only one atom:

$$< \cap\, \overline{?C/\mathsf{until}(A, B)}\ \equiv\ ?\neg C/<\ \cup\ ?C/</?\neg A\ \cup\ ?C/</?\neg B/</?A. \quad (9)$$

For the case with more atoms we make a further case distinction. Let $R = S/\mathsf{until}(A, B)$, where S is a composition of until wffs. Then

$$< \cap\, \overline{R} \equiv (\overline{S/<} \cap < \cap\ \overline{R})\ \cup\ (S/< \cap < \cap\ \overline{R}). \quad (10)$$

As $\models \overline{S/<} \subseteq \overline{S/\mathsf{until}(A, B)}$, the first disjunct is simply equivalent to $< \cap\ \overline{S/<}$. Lemma 9 shows how to define such expressions.

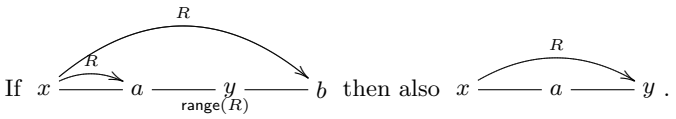

Fig. 2. Lemma 7 in a picture.

Now we explain how to define $S/{<} \cap {<} \cap \overline{R}$, the second disjunct in (10). Suppose x and y stand in this relation. Then $x < y$ and there is a z such that xSz and $z < y$. Let z' be the *last* between x and y such that xSz'. Then we must enforce $z'\overline{\mathsf{until}(A, B)}y$, which we can by (9). But that is enough, because suppose to the contrary that there is a z such that xSz and $z\mathsf{until}(A, B)y$ and $z < z' < y$. From the last two conjuncts we obtain that $z'\mathsf{until}(A, B)y$, a contradiction. So if we can express that

$$(x, z) \text{ is the largest subinterval in } (x, y) \text{ which is in } S, \tag{11}$$

we have defined the second disjunct.

To summarize, for $R = S/\mathsf{until}(A, B)$, the expression ${<} \cap \overline{R}$ is equivalent to the union of ${<} \cap \overline{S/{<}}$ and a formula expressing $\exists z((11) \wedge z\,({<} \cap \overline{\mathsf{until}(A, B)})\,y)$. (9) defines ${<}\cap\overline{\mathsf{until}(A, B)}$ as a path wff. Lemmas 8 and 9 define (11) and ${<} \cap \overline{S/{<}}$, respectively.

The statement (11) is a first order formula in three free variables. As we need it quite a lot, we make an abbreviation. For S an until wff, define $\mathsf{max}(S, x, z, y)$ as the ternary relation $x < z < y \wedge xSz \wedge \neg\exists w(z < w < y \wedge xSw)$. In defining both $\overline{S/{<}}$ and the max predicate we use a crucial lemma. For R a path wff, let $\mathsf{range}(R)$ be the node wff which is true at a point x iff there exists a point y such that yRx holds. These node wffs are definable in Conditional XPath, using conversion: $\mathsf{range}(R) \equiv \langle R^{-1}\rangle$.

Lemma 7. *Let R be an until wff. For all points x, y, a, b, such that $x < a \leq y \leq b$, if xRa and xRb and $\mathsf{range}(R)y$, then also xRy. See Figure 2.*

The proof is by induction on the number of /'s in R.

Lemma 8. *For R an until wff, $\mathsf{max}(R, x, z, y)$ is definable as a Conditional XPath path wff.*

Lemma 9. *For R an until wff, ${<} \cap \overline{R/{<}}$ is definable as a Conditional XPath path wff.*

For both lemmas, the definitions are given inductively on the number of /'s in R. Lemma 7 is used in the inductive case.

5 Conclusion

The results make us conclude that both Core and Conditional XPath are very natural languages for talking about ordered trees. Their simplicity and visual attractiveness make them suitable candidates for a user-friendly alternative to first order logic. The expressive completeness result for paths is very important, as arguably the relations in Conditional XPath are still "drawable". With drawable we mean that one can make an intuitive picture which exactly captures the meaning of the query. Composition and union are obviously drawable, whereas intersection and negation are not. The conditional step $(\mathsf{step}?A)^+$ is also drawable using ellipsis. Of course one should not draw the filter expressions, but just indicate them with formulas attached to nodes in the drawings.

In this context it is interesting to note a repetition in history. The natural class of models in computational linguistics is the class of finite ordered trees. In the beginning of the field of model theoretic syntax Monadic Second Order Logic was invariably used to reason about these structures [15]. Later, formalisms based on modal logic were proposed as alternatives. Arguments for the alternatives were both based on computational complexity (which is lower both for model checking and theorem proving) and on "naturalness" of expressing properties (in this case of grammars). In fact, both Core and Conditional XPath have their roots in the nineties: [2] and [14] define isomorphic variants of the filter expressions of Core and Conditional XPath, respectively.

From a theoretical point of view, Conditional XPath is not harder than Core XPath: the query evaluation problem is still solvable in time $O(|Q| \cdot |D|)$, with $|Q|$, $|D|$, the sizes of the query and the data, respectively [12].

Its easy syntax, visual attractiveness, and low complexity combined with its expressive completeness make Conditional XPath an excellent candidate for succeeding XPath 1.0.

Acknowledgments

Maarten Marx was supported by the Netherlands Organization for Scientific Research (NWO), under project number 612.000.106. Thanks are due to Loredana Afanasiev, David Gabelaia, Evan Goris, Jan Hidders, Sanjay Modgil, Maarten de Rijke, Thomas Schwentick, Yde Venema and Petrucio Viana.

References

1. M. Benedikt, W. Fan, and G. Kuper. Structural properties of XPath fragments. In *Proceedings. ICDT 2003*, 2003.
2. 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.
3. World-Wide Web Consortium. XML path language (XPath): Version 1.0. `http://www.w3.org/TR/xpath.html`.

4. 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.
5. G. Gottlob, C. Koch, and R. Pichler. Efficient algorithms for processing XPath queries. In *VLDB'02*, 2002.
6. G. Gottlob, C. Koch, and K. Schulz. Conjunctive queries over trees. In *Proceedings of PODS*, pages 189–200, 2004.
7. A. Halevy and M. Rousset. Combining horn rules and description logics in CARIN. *Artificial Intelligence*, 104:165–209, 1998.
8. I. Hodkinson and A Simon. The k-variable property is stronger than H-dimension k. *Journal of Philosophical Logic*, 26:81–101, 1997.
9. N. Immerman and D. Kozen. Definability with bounded number of bound variables. In *Proceedings of the Symposium of Logic in Computer Science*, pages 236–244, Washington, 1987. Computer Society Press.
10. J.A.W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968.
11. M. Marx. Conditional XPath, the first order complete XPath dialect. In *Proceedings of PODS'04*, pages 13–22, 2004.
12. M. Marx. XPath with conditional axis relations. In *Proceedings of EDBT'04*, volume 2992 of *LNCS*, pages 477–494, 2004.
13. M. Marx and M. de Rijke. Semantic characterizations of XPath. In *TDM'04 workshop on XML Databases and Information Retrieval.*, Twente, The Netherlands, June 21, 2004.
14. A. Palm. Propositional tense logic for trees. In *Sixth Meeting on Mathematics of Language*. University of Central Florida, Orlando, Florida, 1999.
15. J. Rogers. *A Descriptive Approach to Language Theoretic Complexity*. CSLI Press, 1998.
16. A. Tarski and S. Givant. *A Formalization of Set Theory without Variables*, volume 41. AMS Colloquium publications, Providence, Rhode Island, 1987.
17. 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.
18. P. Wadler. Two semantics for XPath. Technical report, Bell Labs, 2000.