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 $\texttt{(one\_step\_axis::n[F])}^+$, 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

[8] 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:

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

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]. More abstractly, the first result states that the class of ordered trees has the three variable property: every first order formula in at most three free variables is equivalent 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 [8] 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. Proofs not given in the main text are provided in the Appendix. We note that the expressive completeness result for Conditional XPath's answer sets (shown in [8]) follows from the result presented here, but not conversely. The results about Core XPath have been presented at the Twente workshop on Database Managment [9]. 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.

Let Σ be a set of atomic symbols. 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

$$\mathsf{step} ::= \mathsf{down} \mid \mathsf{up} \mid \mathsf{right} \mid \mathsf{left}.$$

We define Core XPath and Conditional XPath. They differ only in de operations allowed on path wffs. The node wffs are generated by (with $\sigma \in \Sigma$)

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

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

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

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 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 $(N, R_\downarrow, R_\rightarrow, \sigma_i)_{i \in \Sigma}$.

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

Remark 1. Unlike in most of the literature on XPath we do not restrict the class of structures to trees corresponding to XML documents (the DOM). So our trees can be infinitely deep, infinitely branching and may contain multiple atomic labels at each node. All our results apply to document object models 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*. For instance we need them to express XPath's `child::A[n]` for `n` a natural number.

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 1 contains the definition of $[\![\cdot]\!]_{\mathfrak{M}}$. The equivalence with the W3C syntax and semantics (cf., e.g., [5,14]) 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 $(\mathsf{down}?A)^+$ 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:

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

Then the next frontier node relation is definable as

$$?\mathsf{leaf}/\mathsf{right}/?\mathsf{leaf} \ \cup \ ?\mathsf{leaf}/(?\mathsf{last}/\mathsf{up})^+/\mathsf{right}/(\mathsf{down}?\mathsf{first})^*/?\mathsf{leaf}.$$

Here $(?\mathsf{last}/\mathsf{up})^+$ abbreviates $?\mathsf{last}/(\mathsf{up}?\mathsf{last})^*/\mathsf{up}$.

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. 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 $R_{\Rightarrow}$.

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

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

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

Table 1. The semantics of Core and Conditional XPath.

3.1 Answer sets

The answer set of an XPath 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 [8] stated that every first order definable set of nodes is definable as the answer set of some Conditional XPath 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. *(1) The answer sets of Core XPath 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

[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 $\langle \mathsf{down}^+/\mathsf{down}^+/?A \rangle$.

[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'
$x\, R_{\downarrow}\, y$	$\langle \mathsf{down}?A' \rangle$
$y\, R_{\downarrow}\, x$	$\langle \mathsf{up}?A' \rangle$
$x\, R_{\rightarrow}\, y$	$\langle \mathsf{right}?A' \rangle$
$y\, R_{\rightarrow}\, x$	$\langle \mathsf{left}?A' \rangle$
$x\, R_{\Rightarrow}\, y \wedge \neg x\, R_{\rightarrow}\, y$	$\langle \mathsf{right}/\mathsf{right}^{+}?A' \rangle$
$y\, R_{\Rightarrow}\, x \wedge \neg y\, R_{\rightarrow}\, x$	$\langle \mathsf{left}/\mathsf{left}^{+}?A' \rangle$
$x\, R_{\Downarrow}\, y \wedge \neg x\, R_{\downarrow}\, y$	$\langle \mathsf{down}/\mathsf{down}^{+}?A' \rangle$
$y\, R_{\Downarrow}\, x \wedge \neg y\, R_{\downarrow}\, x$	$\langle \mathsf{up}/\mathsf{up}^{+}?A'. \rangle$

Table 2. Order types and their translations

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., [13] which takes care to use only two variables). The translation is just the definition from Table 1 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 right hand side of Table 2, together with the needed translations (A' denotes the translation of A).

Remark 3. The answer sets 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 [8]. 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 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 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. Not every first order definable path set can be defined in Core XPath. As in [1], consider the relation

$$(x \,\texttt{descendant}\, y \wedge A(y) \wedge \forall z((x \,\texttt{descendant}\, z \wedge z \,\texttt{descendant}\, y) \rightarrow B(z))). \tag{1}$$

A standard argument shows that the range of this relation cannot be specified using less then three variables. Whence the relation is not expressible in Core XPath by Theorem 1. Note that the relation is expressible in Conditional XPath, as

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

It is also expressible in Core XPath expanded with a complementation operator, as

$$\mathsf{down}^+?A \;\cap\; \overline{\mathsf{down}^+/?\neg B/\mathsf{down}^+},$$

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

Being able to express complementation or the relation (1) in all four directions are both closely connected to first order expressive completeness for path sets: each of them is a sufficient condition.

We say that an XPath language $\mathcal{L}$ is first order complete 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. *Any expansion of Core XPath which is closed under complementation is first order complete.*

Proof. Let L be such an expansion. Then L can express every binary relation expressible in Tarski's relation algebras[4]. But that formalism 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 [12]. Thus it is sufficient to show that —on ordered trees and in the signature of FO^{tree}— FO^2_3 is equally expressive as FO^2_ω. This will be done in Appendix A using Ehrenfeucht–Fraïssé pebble games from [7].

Our main result, proved in Section 4, is

[4] Tarski's set 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. *id* is definable as $?\top$.

Theorem 3. *Conditional XPath is closed under complementation, whence first order complete.*

This gives the second sufficient condition for being first order complete. We finish the section with a characterization of Core XPath's path sets. First we make the connection with unions of conjunctive queries.

Definition 1. *An XPath query is a formula of the form*

$$Q(x,y) :- \bigvee_i \bigwedge (R_1^i \wedge \ldots \wedge R_n^i \wedge A_1^i \wedge \ldots \wedge A_m^i), \tag{2}$$

in which the A_j^i are of the form $\mathsf{node\_wff}(u)$ *and the R_j^i of the form u* $\mathsf{path\_wff}$ *v or $x = y$.*
It is a Core (Conditional) XPath query if the path and node wffs are from Core (Conditional) XPath.

Such queries do not provide extra expressivity, since

Lemma 1. *Each Conditional (Core) XPath query is equivalent to a Conditional (Core) path wff.*

We immediately obtain the following generalization of the results in [1]:

Theorem 4. *The sets of path wffs of both Core and Conditional XPath are closed under intersection.*

We note that the result for Core XPath in Lemma 1 is essentially Theorem 6.1 in [6]. The proof of the lemma is provided in the Appendix.

Now define the following variant on the queries of the form in (2): the relations are from the signature $\{R_\downarrow, R_\rightarrow, R_\Downarrow, R_\Rightarrow\}$ and all of the A_j^i are formulas in FO_2^{tree} in one free variable. 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

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

So these are like unions of usual conjunctive queries, except that properties may contain negations. We call them first order core queries. From the definition of $[\![\cdot]\!]_{\mathfrak{M}}$, it is easy to see that every Core XPath expression is equivalent to a first order core query.

By Theorem 1 and Lemma 1 the converse also holds, yielding a first order characterization of the Core XPath definable sets of paths.

Theorem 5. *Every first order core query is equivalent to a Core XPath path wff and conversely.*

We note that Benedikt, Fan and Kuper [1] gave a characterization of positive Core XPath without the sibling axis relations as unions of conjunctive queries, in which the ϕ_i are just atomic node tag predicates. Their theorem extends to the case with sibling axis included. With negation added, one must allow all unary FO_2^{tree} formulas as the ϕ_i, reflecting the characterization of the filter expressions in Theorem 1.

4 Closure under complementation

In this section we prove Theorem 3. We first establish that path wffs have a disjunctive normal form resembling the separation property of [8].

To start a bit of terminology. An *atom* is a path wff of the form $\mathsf{step}?A$, or $(\mathsf{step}?B)^+?A$. A *test* is a path wff of the form $?A$. A *basic composition* is a test followed by a composition of atoms. 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:

$$D, \quad U, \quad U^*/R/D^*, \quad U^*/L/D^*. \tag{3}$$

Here we use $U^*/R/D^*$ as an abbreviation for the forms U/R, R, R/D, $U/R/D$, and similarly for $U^*/L/D^*$. Separated basic compositions (3) are, sub-relations of

$$\mathsf{down}^+, \quad \mathsf{up}^+, \quad \mathsf{up}^*/\mathsf{right}^+/\mathsf{down}^*, \quad \mathsf{up}^*/\mathsf{left}^+/\mathsf{down}^*, \tag{4}$$

which correspond to XPath's axis relations

$$\texttt{descendant}, \ \texttt{ancestor}, \ \texttt{following}, \ \texttt{preceding},$$

respectively. As is well known, given a tree and a node n, the answer sets of these relations evaluated at n, together with the set $\{n\}$ form a partition of the tree [3].

Lemma 2. *Every path wff is equivalent to a union of tests and separated basic compositions.*

Proof. Each path wff is equivalent to a union of basic compositions and tests, by the equivalence $?A/?B \equiv ?(A \wedge B)$ and distribution of $/$ over $\cup$. So, to prove the Lemma it is enough to describe a procedure that separates basic compositions. Table 3 shows how every composition of *two* atoms can be separated. A repeated application of the rewriting in this table combined with distributing $/$ over $\cup$ yields the desired form.

The proofs for the equivalences in the table are standard semantic arguments. As an example we consider the first case. So consider a composition of the form D/U. If one of them is just a step followed by a test, the rewriting is easy:

$$\begin{aligned}
\mathsf{down}?A/\mathsf{up}?B &\equiv ?(B \wedge \langle \mathsf{down}?A \rangle).\\
\mathsf{down}?A/(\mathsf{up}?C)^+?B &\equiv ?(C \wedge B \wedge \langle \mathsf{down}?A \rangle)\\
&\qquad \cup\\
&\qquad ?(C \wedge \langle \mathsf{down}?A \rangle)/(\mathsf{up}?C)^+?B.\\
(\mathsf{down}?C)^+?A/\mathsf{up}?B &\equiv ?(B \wedge \langle \mathsf{down}?(C \wedge A) \rangle)\\
&\qquad \cup\\
&\qquad (\mathsf{down}?C)^+?(B \wedge \langle \mathsf{down}?(C \wedge A) \rangle).
\end{aligned}$$

A having form	is separated as a union of forms
D/U	$D, T,$ or U
D/R	D
D/L	D
U/D	$D, T, U, U^*/R/D^*,$ or $U^*/L/D^*$
U/R	U/R
U/L	U/L
R/D	R/D
R/U	U
R/L	$T, L,$ or R
L/D	L/D
L/U	U
L/R	$T, L,$ or R

Table 3. Syntactical separation for compositions of two atoms.

The last case $(\mathsf{down}?C_1)^+?A/(\mathsf{up}?C_2)^+?B$ is most demanding. Suppose nodes x, y are related in this way. Then there is a z such that

$$x\ (\mathsf{down}?C_1)^+?A\ z \text{ and } z\ (\mathsf{up}?C_2)^+?B\ y.$$

Thus $x\mathsf{down}^+z$ and $y\mathsf{down}^+z$. The union depends on the position of y relative to x. There are three cases, given together with the equivalent path wff. Let E abbreviate

$$\langle\!\langle(\mathsf{down}?(C_1 \wedge C_2))^+/\mathsf{down}?(C_1 \wedge A)\rangle\!\rangle.$$

case	equivalent path wff
$x\,\mathsf{down}^+\ y$	$(\mathsf{down}?C_1)^+?(C_2 \wedge B \wedge E)$
$x = y$	$?(C_2 \wedge B \wedge E)$
$y\,\mathsf{down}^+\,x$	$?(C_2 \wedge E)/(\mathsf{up}?C_2)^+?B.$

Note that these three disjuncts are of the form D, T, and U, respectively. It is worthwhile to note that we really just reason on linear structures. Except formulas of the form U/D, all other cases in Table 3 are treated similarly: reason with linear structures. Only in the case of U/D do we really have to reason about trees, and obtain the full case distinction.

Now Theorem 3 follows from Lemmas 1, 2 and

Lemma 3. *The complement of each separated basic composition is definable as a Conditional XPath query.*

Proof of Theorem 3. Let R be a path wff. Then by Lemma 2 $R \equiv \bigcup_i R_i$, with the R_i tests and separated basic compositions. Whence $\overline{R} \equiv \bigcap_i \overline{R_i}$. By

Theorem 4, the path wffs are closed under intersection. The complement of a test is equivalent to a path wff, as is not hard to see. By Lemmas 3 and 1 each complement of a separated basic composition is equivalent to a path wff. Hence the theorem. QED

The proof of Lemma 3 consists of an easy and a hard part, separated in the following two lemmas.

Lemma 4. *The complement of each separated basic composition is definable from path wffs and formulas of the form*

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

Lemma 5. *Each relation in (5) is definable as a Conditional XPath query.*

We start with proving the easy Lemma.

Proof of Lemma 4. Consider a separated basic composition. We may assume it has form $U^*/R^+/D^*$ or $U^*/L^+/D^*$, otherwise the lemma holds trivially. We show how to define the complement of a composition of the form U/R. The other cases follow an identical argument. Now

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

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

$$\mathsf{down}^* \cup \mathsf{up}^*/\mathsf{left}^+/\mathsf{down}^* \cup \mathsf{up}^+ \cup \mathsf{right}^+/\mathsf{down}^* \cup \mathsf{up}^+/\mathsf{right}^+/\mathsf{down}^+. \tag{7}$$

For the second disjunct, we use the following equation:

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

The left to right direction holds for all relations. For the other direction use the fact that for all x, y if $x\ U/R\ y$ then there exists a unique z such that xUz and zRy. Now both disjuncts of (6) are rewritten into the form required by the Lemma. QED

Proof of Lemma 5. Let xRy for R one of the relations defined in (5). For all four cases we just need to reason about the points in between x and y. So the argument is identical in all cases. For concreteness, we consider the case for down compositions.

To reduce the number of cases, we turn to a formalism well known from temporal logic. For A, B node wffs, define the operator $\mathsf{until}(A, B)$ with the semantics

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

Both down atoms are expressible as an until formula: $\mathsf{down}?A \equiv \mathsf{until}(A, \neg\top)$ and $(\mathsf{down}?B)^+?A \equiv \mathsf{until}(A \wedge B, B)$. Thus it is sufficient to show how to define $\mathsf{down}^+ \cap \overline{?C/R}$, 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 down^+ and down^*, respectively. We define complementation by induction on the number of /'s in R. The base case is

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

The inductive case needs an elaborate case distinction. Let $R = S/\mathsf{until}(A, B)$. Then

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

As $\models \overline{S/<} \subseteq \overline{S/\mathsf{until}(A, B)}$, the first disjunct is simply equivalent to $< \cap \overline{S/<}$. Note that S is shorter than R. So we need to be able to express $< \cap \overline{S/<}$, saying that no subinterval having the same beginning is in S. The base case is

$$\begin{aligned} < \cap \overline{\mathsf{until}(A, B)/<} \equiv\; & \mathsf{down} \;\cup \\ & \mathsf{until}(\top, \neg A) \;\cup \\ & ?\langle\mathsf{down}?\neg A\rangle \,/\, (\mathsf{until}(A, \neg A) \cap </?\neg B/<) \,/\, \leq. \end{aligned} \tag{11}$$

Note that we used $\cap$ in this definition, but that is warranted by Theorem 4. The formula $\mathsf{until}(A, \neg A) \cap </?\neg B/<$ expresses that before the first A there is already a $\neg B$. The inductive case is Lemma 8.

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

$$(x, z) \text{ is the largest subinterval in } (x, y) \text{ which is in } S, \tag{12}$$

we have defined complementation. This is shown in Lemma 7. The statement (12) 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, which we prove first.

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. For R an arbitrary until wff, $\mathsf{range}(R)$ is defined inductively as

$$\begin{aligned} \mathsf{range}(?C/\mathsf{until}(A, B)) &= \langle ?A/\mathsf{since}(C, B)\rangle \\ \mathsf{range}(R/\mathsf{until}(A, B)) &= \langle ?A/\mathsf{since}(\mathsf{range}(R), B)\rangle, \end{aligned}$$

where $\mathsf{since}(C, B)$ is the counterpart of until in the upward direction.[5]

Lemma 6. *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 1.*

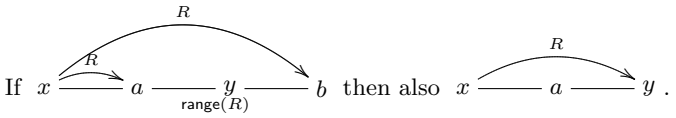


Fig. 1. Lemma 6 in a picture.

Proof. By induction on the number of /'s in R. First let $R = ?C/\mathsf{until}(A, B)$. Then Ay, because $\mathsf{range}(R)y$. As $x < y \leq b$ and xRb, C is true at x and B at all points in between x and b, hence a fortiori between x and y. Whence $x\ ?C/\mathsf{until}(A, B)\ y$ holds.

For the inductive step, let S be an until wff and let $R = S/\mathsf{until}(A, B)$. We obtain a', b', y' such that

- $y' < y$ and $y'\mathsf{until}(A, B)y$ and $\mathsf{range}(S)y'$,
- $x < a' < a$ and xSa' and $a'\mathsf{until}(A, B)a$, and
- $x < b' < b$ and xSb' and $b'\mathsf{until}(A, B)b$.

Without loss of generality, we may assume that $a' \leq b'$. Then there are three cases: (1) $y' < a'$, (2) $a' \leq y' \leq b'$, and (3) $b' < y'$.

If $y' < a'$, then from $y'\mathsf{until}(A, B)y$, we obtain $a'\mathsf{until}(A, B)y$, whence with xSa', we get $x\ S/\mathsf{until}(A, B)\ y$.

If $a' \leq y' \leq b'$, by inductive hypothesis we get xSy', whence with $y'\mathsf{until}(A, B)y$ we obtain $x\ S/\mathsf{until}(A, B)\ y$.

Let $b' < y'$. If $y = b$ we are done. Otherwise $b' < y < b$. By $y'\mathsf{until}(A, B)y$, A holds at y. As $b'\mathsf{until}(A, B)b$ (and $b' < y < b$), we thus have $b'\mathsf{until}(A, B)y$. Together with xSb' this yields $x\ S/\mathsf{until}(A, B)\ y$.

Lemma 7. *For R an until wff, $\mathsf{max}(R, x, z, y)$ is definable as a Conditional XPath query.*

Proof. We define $\mathsf{max}(R, x, z, y)$ by induction on the number of /'s in R. If $R = ?C/\mathsf{until}(A, B)$, then $\mathsf{max}(R, x, z, y)$ is the conjunction of $x < z < y$, $C(x)$, $x\mathsf{until}(A, B)z$ and a formula forbidding that there is a larger $\underline{\mathsf{until}(A, B)}$ interval in (x, y) starting in x. This is done by the formula $\neg B(z) \vee z(< \cap \overline{\mathsf{until}(A, B)/<})y$. The latter disjunct is defined in (11).

[5] $\mathsf{since}(C, B)$ is definable as $(\mathsf{up}?B)^*/\mathsf{up}?C$.

Now let $R = S/\mathsf{until}(A,B)$. Then $\mathsf{max}(R,x,z,y)$ is defined as

$$\begin{array}{c}\exists w(x<w<z<y \wedge \mathsf{max}(S,x,w,z) \wedge \mathsf{max}(\mathsf{until}(A,B),w,z,y)) \\ \vee \\ \exists w(x<z<w<y \wedge \mathsf{max}(S,x,w,y) \wedge w\overline{\mathsf{until}(A,B)/<}y \\ \wedge xRz \wedge z\mathsf{until}(\neg\mathsf{range}(R),\neg\mathsf{range}(R))w).\end{array}$$

By Lemma 6 this definition is correct.

Now we show how to define $\overline{R/<}$.

Lemma 8. *For R an until wff, $< \cap \overline{R/<}$ is definable as a Conditional XPath query.*

Proof. Again the definition is by induction on the number of /'s in R. The base case is (11). Thus let $R = S/\mathsf{until}(A,B)$. Then $< \cap \overline{R/<}$ is equivalent to

$$(< \cap S/< \cap \overline{R/<}) \cup (< \cap \overline{S/<} \cap \overline{R/<}).$$

As $R/< = S/\mathsf{until}(A,B)/< \subseteq S/</< \subseteq S/<$, the second disjunct is equivalent to $< \cap \overline{S/<}$. As S is shorter than R, this is definable by IH.

Rest us to define $x(< \cap S/< \cap \overline{S/\mathsf{until}(A,B)/<})y$ as a Conditional XPath query. The formula uses two existentially quantified variables z, z', which are ordered like $x < z \leq z' < y$. The interval (x,z) is the smallest subinterval of (x,y) which is in S. This is expressed by $x(S \cap \overline{S/<})z$. The interval (x,z') is the largest S subinterval, which we express using the max predicate. Now we must ensure that $x(\overline{S/\mathsf{until}(A,B)/<})y$ holds. So we must say that $z'(\overline{\mathsf{until}(A,B)/<})y$, and that starting at z there is no $\mathsf{until}(A,B)$ interval. Moreover, there should not be a z'' in between z and z' such that xRz''. This is done by saying that either $z = z'$ or $z\mathsf{until}(\neg\mathsf{range}(R),\neg\mathsf{range}(R))z'$. So the final formula becomes

$$\begin{array}{ll}\exists zz'(& x<z\leq z'<y \quad \wedge \\ & xSz \wedge x\overline{S/<}z \quad \wedge \\ & \mathsf{max}(S,x,z',y) \quad \wedge \\ & z'\overline{\mathsf{until}(A,B)/<}y \wedge \\ & (z=z' \vee z\mathsf{until}(\neg\mathsf{range}(R),\neg\mathsf{range}(R))z') \\ ). & \end{array}$$

By Lemma 6 this definition is correct.

Now we are prepared to finish the proof of Lemma 5, that is to show that for each until wff R, $< \cap \overline{R}$ is equivalent to a Conditional XPath query. Again, the definition is by induction on the number of /'s in R. The base case is (9). So let $R = S/\mathsf{until}(A,B)$. As in Lemma 8, $< \cap \overline{R}$ is equivalent to

$$(< \cap \overline{S/<} \cap \overline{S/\mathsf{until}(A,B)}) \ \cup \ (< \cap S/< \cap \overline{S/\mathsf{until}(A,B)}).$$

The first disjunct is equivalent to $< \cap \overline{S/<}$ and definable by Lemma 8. The last is equivalent to

$$\exists z(x<z<y \wedge \mathsf{max}(S,x,z,y) \wedge z\overline{\mathsf{until}(A,B)}y),$$

as explained in the beginning of the proof. QED

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 repitition 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 [11]. 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 [10] 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. It would be exciting to see how existing XPath algorithms can be adjusted in order to deal efficiently with conditional path expressions.

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, Petrucio Viana and in particular to Yde Venema.

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 *Proceedings 12th Annual IEEE Symposium on Logic in Computer Science*, pages 228–235, Warsaw, Poland, 1997. IEEE.
5. 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.
6. G. Gottlob, C. Koch, and K. Schulz. Conjunctive queries over trees. In *Proceedings of PODS*, pages 189–200, 2004.
7. 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.
8. M. Marx. Conditional XPath, the first order complete XPath dialect. In *Proceedings of PODS'04*, 2004.
9. 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.
10. A. Palm. Propositional tense logic for trees. In *Sixth Meeting on Mathematics of Language*. University of Central Florida, Orlando, Florida, 1999.
11. J. Rogers. *A Descriptive Approach to Language Theoretic Complexity*. CSLI Press, 1998.
12. A. Tarski and S. Givant. *A Formalization of Set Theory without Variables*, volume 41. AMS Colloquium publications, Providence, Rhode Island, 1987.
13. 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.
14. P. Wadler. Two semantics for XPath. Technical report, Bell Labs, 2000.

A Ordered trees have the three variable property

A first order language $\mathcal{L}$ is said to have the *three variable property* over a class of structures C if every $\mathcal{L}$ formula in at most three free variables is equivalent within the class C to an $\mathcal{L}$ formula in at most three (free and bound) variables. Thus the next theorem finishes the proof of Theorem 2.

Theorem 6. *The class of ordered trees has the three variable property.*

We prove the theorem using k pebble n round Ehrenfeucht–Fraïssé games as in [7]. We recall some of the terminology. Let L be a first order language with variables $x_1, x_2, \ldots$. A partial assignment over a structure A for L is a partial function $u : \{x_1, x_2, \ldots\} \longrightarrow A$. The domain of u is denoted ∂u. The cardinality of ∂u is denoted $|u|$. A k-configuration over A, B is a pair (u, v), where u, v are partial assignments over A, B, respectively such that $\partial u = \partial v \subseteq \{x_1, \ldots, x_k\}$. For A, B structures and (u, v) a k–configuration, $G(u, v, k, n)$ denotes the n–round, k–pebble game played on structures A, B starting at configuration (u, v). The fundamental result on games is that for L a first order language in a finite signature without function symbols it holds that Player II (assumed to be female) has a winning strategy in $G(u, v, k, n)$ if and only if A, u and B, v satisfy exactly the same L formulas of quantifier rank at most n written in at most k (free and

bound) variables. The theorem now follows by a standard argument[6] from the following lemma.

Lemma 9. *For any two ordered trees A, B, for any 3-configuration (u, v), if player II has a winning strategy in $G(u, v, 3, 7n)$ (played on A, B) , then she has a winning strategy in $G(u, v, k, n)$, for all k.*

Proof. Let A, B be ordered trees and (u, v) a 3-configuration. Suppose player II has a winning strategy in $G(u, v, 3, 7n)$. We describe her winning strategy in $G(u, v, k, n)$ and prove the lemma by induction on the number of rounds. By standard game arguments, the lemma holds for 0 rounds, and also that

(*) if $|u| = |v| < 3$ and she has a winning strategy in $G(u, v, 3, 7n+1)$, then she also has a winning strategy in $G(u, v, k, n+1)$, for all k.

So assume $|u| = |v| = 3$, she has a winning strategy for $G(u, v, 3, 7 \cdot (n+1))$ and no two pebbles occupy the same position (otherwise one of them can be removed and we obtain the result by (*)). We must show that she has a winning strategy in $G(u, v, k, n+1)$, for all k. By standard game arguments, we may assume that $k > n + 1$, so that Player I never needs to remove a pebble.

Renumber the variables if needed so that $u(x_1) \ll u(x_2) \ll u(x_3)$. Here $\ll$ denotes the document order defined as $\mathsf{down}^+ \cup \mathsf{up}^*/\mathsf{right}^+/\mathsf{down}^*$. On each ordered tree $\ll$ is a total linear order, so the assumption that $u(x_1) \ll u(x_2) \ll$

[6] Let $\phi(\bar{x})$ be a first order formula in three free variables. Let n be its quantifier rank. Let C_ϕ be all structures $(A, \bar{a})$ such that A is a tree and $A \models \phi(\bar{a})$. Let $\overline{C_\phi} = C_{\neg\phi}$. First observe that for any $(A, \bar{a}) \in C_\phi$, for any $(B, \bar{b}) \notin C_\phi$, there exists a formula $\delta_{AB}(\bar{x})$ of quantifier rank at most $7n$ written in three free and bound variables such that

$$A \models \delta_{AB}(\bar{a}) \text{ and } B \not\models \delta_{AB}(\bar{b}).$$

For, suppose to the contrary. Then $(A, \bar{a})$ and $(B, \bar{b})$ satisfy the same formulas in three variables of quantifier rank $7n$. But then, by the fundamental result on games, with some abuse of notation, Player II has a winning strategy in the game $G(\bar{a}, \bar{b}, 3, 7n)$. But by the lemma she then has a winning strategy for the game $G(\bar{a}, \bar{b}, k, n)$, for any k. But that means that the two structures satisfy the same L formulas of quantifier rank at most n. In particular, as $A \models \phi(\bar{a})$, also $B \models \phi(\bar{b})$, a contradiction.
Now define

$$\delta(\bar{x}) = \bigvee_{(A,\bar{a}) \in C_\phi} \bigwedge_{(B,\bar{b}) \in \overline{C_\phi}} \delta_{A\bar{a}B\bar{b}}(\bar{x}).$$

As the $\delta_{A\bar{a}B\bar{b}}$ are of quantifier rank at most $7n$ and in a finite signature, δ is finite modulo logical equivalence. We claim that $\delta(\bar{x})$ is the desired formula equivalent to $\phi(\bar{x})$. For suppose $A \models \phi(\bar{a})$, for $A, \bar{a}$ arbitrary. Then $(A, \bar{a})$ in C_ϕ. But then $A \models \bigwedge_{B \in \overline{C_\phi}} \delta_{A\bar{a}B\bar{b}}(\bar{a})$, and a fortiori also δ. Conversely, suppose $B \not\models \phi(\bar{b})$. Then $(B, \bar{b}) \in \overline{C_\phi}$. Now suppose to the contrary that $B \models \delta(\bar{b})$. Then for some $A, \bar{a}$, $B \models \bigwedge_{(B,\bar{b}) \in \overline{C_\phi}} \delta_{A\bar{a}B\bar{b}}(\bar{x})$. In particular $B \models \delta_{A\bar{a}B\bar{b}}(\bar{x})$, a contradiction with the fact that $(B, \bar{b}) \in \overline{C_\phi}$ and the definition of $\delta_{A\bar{a}B\bar{b}}$.

$u(x_3)$ can always be made. Here and below we use the XPath notation for binary relations as it is easier to follow. Thus $x\mathsf{down}^+y$ iff $xR_{\Downarrow}y$, $x\mathsf{up}^*y$ iff $x = y \vee yR_{\Downarrow}x$, etc.

As $\ll$ consists of five disjuncts we have quite a large number of cases. The simplest case is when

$$u(x_1)\mathsf{down}^+u(x_2)\mathsf{down}^+u(x_3).$$

Now just as in the linear case discussed in [7], we break up the game into two subgames on disjoint regions of the respective structures, each containing two pebbles. Using (*) we may then argue that she has a winning strategy in the games $G(u_{\restriction(x_1,x_2)}, v_{\restriction(x_1,x_2)}, k, n+1)$ and $G(u_{\restriction(x_2,x_3)}, v_{\restriction(x_2,x_3)}, k, n+1)$ played on the respective regions. Then it is a matter of combining these two winning strategies into one for the whole game $G(u, v, k, n+1)$.

We now describe these subgames. Consider the pair of corresponding regions

$$\{a \in A \mid u(x_2)((\mathsf{left}^* \cup \mathsf{right}^*)/\mathsf{down}^*)a\} \text{ and } \{b \in B \mid v(x_2)((\mathsf{left}^* \cup \mathsf{right}^*)/\mathsf{down}^*)b\}.$$

Associate with this pair of regions the game $G(u_{\restriction(x_2,x_3)}, v_{\restriction(x_2,x_3)}, 3, 7 \cdot (n+1))$. We call these the regions below, and their complements the regions above $u(x_2)$ and $v(x_2)$, respectively. Associate with the pair of complements the game $G(u_{\restriction(x_1,x_2)}, v_{\restriction(x_1,x_2)}, 3, 7 \cdot (n+1))$.

As each restricted game contains just two pebbles, by (*), she has a winning strategy in the games $G(u_{\restriction(x_2,x_3)}, v_{\restriction(x_2,x_3)}, k, n+1)$ and $G(u_{\restriction(x_1,x_2)}, v_{\restriction(x_1,x_2)}, k, n+1)$, played on the respective subdomains. The crucial property of this partition is that for all a in the region above and b in the region below $u(x_2)$ it holds that

$$\text{neither } aR_{\Rightarrow}b \text{ nor } bR_{\Rightarrow}a \text{ nor } bR_{\Downarrow}a \text{ holds, and } aR_{\Downarrow}b \text{ only if } aR_{\Downarrow}u(x_2). \quad (13)$$

And of course similarly for $v(x_2)$.

Now take the combined strategy for Player II in the game $G(u, v, k, n+1)$ played on the whole structures as described[7] in [7]. The property (13) ensures that the global configuration is a local isomorphism.

[7] We repeat its construction for completeness, and we also need it later on. We describe a strategy for Player II in the game $G(u, v, k, n+1)$. Assume $k > n+1$ so Player I never needs to remove a pebble from the board. The result follows for smaller k. Whenever Player I moves in one of the designated regions of either A or B, Player II responds using her winning strategy in the game associated to that region. Player II will then move in the corresponding region in the other structure. She knows where it is since there is always a pebble of $u(x_2)$. If (u', v') is any subsequent (global) configuration, the restriction of (u', v') to either of the two pairs of regions is a local isomorphism, since Player II has a winning strategy in the game associated to that region. Now suppose $u'(x_i) = a$ and $u'(x_j) = b$ and a, b come from the two different regions. By construction of the strategy, then also $v'(x_i)$ and $v'(x_j)$ come from the corresponding (different) regions. By property (13) they cannot be $R_{\Rightarrow}$ related. So (u', v') is a local isomorphism for $R_{\Rightarrow}$. Now suppose $u'(x_i)R_{\Downarrow}u'(x_j)$. Then by (13), $u'(x_i)R_{\Downarrow}u'(x_2)$ and by construction $u'(x_2) = u'(x_j)$ or $u'(x_2)R_{\Downarrow}u'(x_j)$. But then $v'(x_i)R_{\Downarrow}v'(x_2)$ and $v'(x_2) = v'(x_j)$ or $v'(x_2)R_{\Downarrow}v'(x_j)$, because (u', v') restricted to the regions is a local isomorphism. But then also $v'(x_i)R_{\Downarrow}v'(x_j)$. Thus (u', v') is a local isomorphism.

Now we consider the most complicated case. When

$$u(x_1)\mathsf{up}^+/\mathsf{right}^+/\mathsf{down}^+u(x_2)\mathsf{up}^+/\mathsf{right}^+/\mathsf{down}^+u(x_3).$$

Let a_4, a_5, a_6, a_7 be the unique nodes such that

$$u(x_1)\mathsf{up}^+\, a_4\, \mathsf{right}^+\, a_5\, \mathsf{down}^+u(x_2)\mathsf{up}^+\, a_6\, \mathsf{right}^+\, a_7\, \mathsf{down}^+u(x_3).$$

There are three ways in which a_5 and a_6 can be related: $a_5\mathsf{down}^+a_6$, $a_5 = a_6$ and $a_6\mathsf{down}^+a_5$. We consider the first, depicted in Figure 2. The other two are treated similarly. As in the simplest case, we partition the trees into different

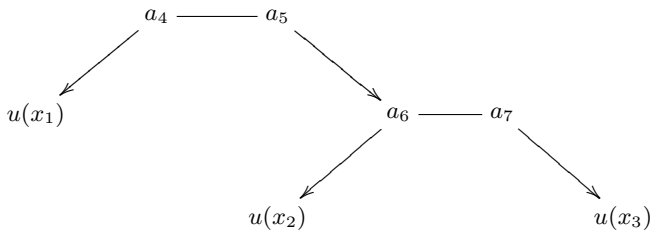
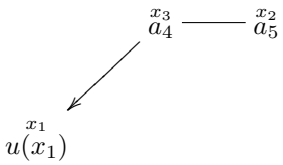


Fig. 2. Location of the pebbles and their corner points.

corresponding regions, though this time we need six of them. We assumed she has a winning strategy in the game $G(u, v, 3, 7 \cdot (n + 1))$. She uses the extra moves to put the pebbles on the corner points a_i and make the partition. Stated differently she uses the extra moves to let the other player help her.

First she lets Player I move pebbles x_3 to a_4 and x_2 to a_5 and uses her winning strategy to move accordingly in the other structure. Thus she has a winning strategy in the game with $7n + 5$ rounds when the pebbles are as in



and correspondingly in the other structure. Partition both structures into three corresponding parts as follows:

$$\begin{aligned} I &= \{a \mid a_4\mathsf{down}^+a\} \\ II &= A \setminus (I \cup \mathrm{Rest}_1) \\ \mathrm{Rest}_1 &= \{a \mid a_5\mathsf{down}^+a\} \end{aligned}$$

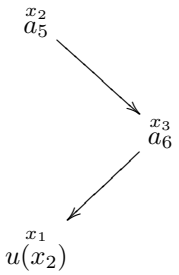
The important property of the partition given in the previous case was (13). It stated that points in different regions are related only if they were both related to the corner point $u(x_2)$. Here we are after a similar property, and we need the extra points $a_4, \ldots, a_7$ as additional corner points. Observe that for all a, b which are *not* in the same region it holds that

$$\text{not } aR_{\Rightarrow}b \tag{14}$$

$$aR_{\Downarrow}b \text{ iff } a\mathsf{down}^*a_4\mathsf{down}^+b \text{ or } a\mathsf{down}^*a_5\mathsf{down}^+b. \tag{15}$$

Let (u', v') be any 2-configuration such that the pebbles are placed as in the latest picture. She has a winning strategy in the games $G(u', v', 3, 7n+5)$ restricted to each of the three regions. So by (*) she also has a winning strategy in the games $G(u', v', k, n+1)$ restricted to those regions. For regions I and II we are done. She plays on in order to partition Rest_1.

She lets Player I move x_3 to a_6 and x_1 to $u(x_2)$ (as in the original game in Figure 2), obtaining



Partition the restriction to $\text{Rest}_1 = \{a \mid a_5\mathsf{down}^+a\}$ of both structures into three corresponding parts as follows:

$$\begin{aligned} III &= \{a \mid a_5\mathsf{down}^+a \wedge a_6\mathsf{up}^+/(\mathsf{left}^* \cup \mathsf{right}^*)/\mathsf{down}^*a\} \\ IV &= \{a \mid a_6\mathsf{down}^+a\} \\ \text{Rest}_2 = \text{Rest}_1 \setminus (III \cup IV) &= \{a \mid a_6\,(\mathsf{left}^* \cup \mathsf{right}^*)/\mathsf{down}^*\,a\} \end{aligned}$$

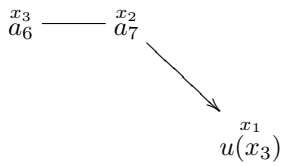
We obtain that for all $a, b \in \text{Rest}_1$ which are *not* in the same region it holds that

$$\text{not } aR_{\Rightarrow}b \tag{16}$$

$$aR_{\Downarrow}b \text{ iff } a\mathsf{down}^*a_6(\mathsf{left}^* \cup \mathsf{right}^*)/\mathsf{down}^*b \tag{17}$$

In all the subgames restricted to the regions and the appropriate two pebbles she has a winning strategy for $7n+3$ rounds. Thus, by (*), also for the k pebble $n+1$ round game on those regions.

Finally she partitions Rest_2. Now she let Player I move x_2 to a_7 and x_1 to $u(x_3)$, obtaining



and she can partition Rest_2 into two parts

$$\begin{aligned} V &= \{a \mid a_7 \mathsf{down}^+ a\} \\ VI &= \mathrm{Rest}_2 \setminus V. \end{aligned}$$

We obtain that for all $a, b \in \mathrm{Rest}_2$ which are *not* in the same region it holds that

$$\text{not } aR_{\Rightarrow}b \tag{18}$$

$$aR_{\Downarrow}b \text{ iff } a = a_7 \text{ and } a_7R_{\Downarrow}b \tag{19}$$

She still has a winning strategy in the 3 pebble $7n+1$ round games restricted to these two regions and their corresponding two pebbles. So by (*) she also has a winning strategy for the k pebble $n+1$ round games on these two regions.

Thus she has winning strategies for the k pebble $n+1$ round games, restricted to the regions $I \ldots VI$. Now we put all six winning strategies together and show that she has a winning strategy for the game $G(u, v, k, n+1)$ with u as in Figure 2. But that should now be rather obvious. Given the location of the pebbles x_1, x_2, x_3, the corner points a_4, a_5, a_6, a_7 are uniquely determined. As we may assume that $k > n+1$, these three pebbles are never removed. Her combined winning strategy consists as before of the winning strategies corresponding to the regions. If Player I plays a pebble x_4 say in region X, she responds according to her winning strategy in that region. As the pebbles x_1, x_2, x_3 are not removed, she can play in the corresponding region X in the other structure. Now let (u', v') be any subsequent (global) configuration. The restriction of (u', v') to any of the six regions is a local isomorphism since she has a winning strategy in the game according to that region. Moreover, by the properties (14)–(19) the relations $R_{\Downarrow}$ and $R_{\Rightarrow}$ between elements in different regions are uniquely determined by their relation to one of the corner points $a_5 \ldots a_7$. Thus (u', v') is a local isomorphism, hence she has a winning strategy in $G(u, v, k, n+1)$.

The other cases for $u(x_1) \ll u(x_2) \ll u(x_3)$ are all treated similarly. Whence the lemma.

B Proof of Lemma 1

Proof. We need a general result first. Let's say a structure $\mathfrak{M} = (N, R_{\Downarrow}, R_{\Rightarrow}, P_i)_{i \in \omega}$ is *connected* if for all $n, n' \in N$, n and n' are connected by a path of the form

$$=,\ R_{\Downarrow},\ R_{\Rightarrow},\ R_{\Uparrow}/R_{\Rightarrow},\ R_{\Rightarrow}/R_{\Downarrow},\ R_{\Uparrow}/R_{\Rightarrow}/R_{\Downarrow}, \tag{20}$$

with $R_{\Uparrow}$ shorthand for $R_{\Downarrow}^{-1}$. Obviously all ordered trees are connected. For $\mathfrak{M}$ a model and $X \subseteq N$, we say that X is *path-closed* if $\mathfrak{M}_{\restriction X}$ is connected. The set of points in Figure 2 is path closed. The set $\{u(x_1), u(x_2), u(x_3)\}$ is not. A query is said to be *path-closed* if for $\phi(\bar{x})$ the body of the query, for each model $\mathfrak{M}$, if $\mathfrak{M} \models \phi(\bar{a})$, then $\bar{a}$ is path closed. A query is said to be *determined* if in every model of the query, each two variables in the body are related by the same relation from (20). For instance, $Q(x, y) :- x\mathsf{down}^+y,\ w\mathsf{down}^+y$ is connected but not determined. The following claim has a straightforward proof

Claim. Every query is equivalent to a union of path closed determined conjunctive queries, in which the binary relations are of the form $x = y$ and

- step and step^+, in the case of Core XPath;
- step and $(\mathsf{step}?A)^+$, in the case of Conditional XPath.

Thus we may restrict our attention to these conjunctive queries.

Consider the graph of the query, in which the nodes are variables, labeled with the node wffs and the edges are labeled with the path wffs (in the case the query contains $x = y$, there is just one node for both x and y). The node wffs are not important to our argument. The graph is generally a multigraph. It is easy to see that if it is a tree, then the query is expressible as a path wff (cf. e.g., [1]). Thus our goal is to turn the multigraph into an equivalent tree, achieving that for each node v, there exists at most one node u such that uRv is part of the query, for R one of the edge labels.

As the query is path closed, there is a path as in (20) between each two nodes, and none of these is the identity. If nodes u and v are connected by a path as in (20) containing a / then each conjunct uRv is inconsistent, so the whole query is equivalent to $\bot$ (expressible as the path wff $?\neg\top$). Now we describe how to remove other edges, first for the simple case of Core XPath. If $u\mathsf{down}^+v$ and uRv for some edge label R, then either it is inconsistent or uRv implies $u\mathsf{down}^+v$. In the last case, the latter can be removed. If the graph contains $u\mathsf{down}^+v$ and $w\mathsf{down}^+v$, then either it is inconsistent, or (because the query is determined) it implies one of $u\mathsf{down}^+w, w\mathsf{down}^+u$. But then one of $u\mathsf{down}^+v$ and $w\mathsf{down}^+v$ can be removed. Similarly when $u\mathsf{right}^+v$. Now suppose we have $u\mathsf{down}^+v$ and wRv, for R one of $\mathsf{right}, \mathsf{right}^+$. Then replace $u\mathsf{down}^+v$ by $u\mathsf{down}^+w$. Do the same when $u\mathsf{down}v$. Repeating this process the final graph has the shape of a tree.

For Conditional XPath we have to be more careful. If $u(\mathsf{down}?A)^+v$ and $u(\mathsf{down}?B)^+v$, then replace both by $u(\mathsf{down}?(A \wedge B))^+v$. If $u(\mathsf{down}?A)^+v$ and there exists a w in between u and v, then replace $u(\mathsf{down}?A)^+v$ by $u(\mathsf{down}?A)^+w$ and $w(\mathsf{down}?A)^+v$. Similarly for the sibling labels. Repeating this process leads to a graph in which every node has at most one incoming down and at most one incoming sibling edge. So the only case left is when a node has both, as in $u(\mathsf{down}?A)^+v$ and $w\mathsf{right}^+v$. There are three cases:

- u is the parent of v. Remove $u(\mathsf{down}?A)^+v$, and add $u\mathsf{down}w$ and $A(v)$.
- The parent z of v and w is a node in the graph. Remove $u(\mathsf{down}?A)^+v$, and add $u(\mathsf{down}?A)^+z$ and $A(v)$.

– The parent is not part of the graph. Add a new node z and add $u(\mathsf{down}?A)^+z$, $A(v)$, $z\mathsf{down}w$.

Repeating this process, the final graph has the shape of a tree.