Structural Properties of XPath Fragments

Michael Benedikt
Bell Laboratories
benedikt@research.bell-labs.com

Wenfei Fan
Bell Laboratories
wenfei@research.bell-labs.com

Gabriel M. Kuper
Università di Trento
kuper@acm.org

Abstract

We study structural properties of each of the main sublanguages of XPath [8] commonly used in practice. First, we characterize the expressive power of these language fragments in terms of both logics and tree patterns. Second, we investigate closure properties, focusing on the ability to perform basic Boolean operations while remaining within the fragment. We give a complete picture of the closure properties of these fragments, treating XPath expressions both as functions of arbitrary nodes in a document tree, and as functions that are applied only at the root of the tree. Finally, we provide sound and complete axiom systems and normal forms for several of these fragments. These results are useful for simplification of XPath expressions and optimization of XML queries.

1 Introduction

XPath [8] is a language for specifying the selection of element nodes within XML documents. It has been widely used in XML query languages (e.g., XSLT [7], XQL [22], XQuery [5]), XML specifications (e.g., XML Schema [23]), and subscription systems (e.g., [6]). It supports a number of powerful modalities and thus is rather expensive to process [3, 18, 10]. In practice, many applications do not need the excessive power of the full language; they use only a fragment of XPath. For example, XML Schema specifies integrity constraints with an XPath fragment that does not support upward modalities (the parent and ancestor axes). It is thus necessary to study the expressiveness and optimization of these XPath fragments, since their analysis and simplification are critical for efficient processing of XML documents.

These considerations motivate us to consider a variety of fragments, focusing on several dichotomies:

- **downward vs. upward:** some fragments support both downward and upward navigation, while others allow only downward traversal.
- **recursive vs. nonrecursive:** some fragments allow navigation along the ancestor and descendant axes, while others permit only parent and child axes.
- **qualified vs. non-qualified:** fragments may or may not include *qualifiers* (predicates testing properties of another expression).

The aim of our work is to show how these fragments differ from one another, in terms of expressiveness and structural properties, and to develop methods for simplifying XPath expressions in each fragment.

Our first contribution is a characterization of the expressiveness of these fragments in terms of logics as well as *tree patterns*, a class of queries fundamental in many areas of computer science [12, 14] and studied recently in connection with LDAP and XML [1]. Extending the preliminary results of [15], we show that the natural XPath fragments with qualifiers can be characterized via the positive existential fragment of first-order logic, and that these fragments also match exactly the queries formed using appropriate tree patterns. One surprising consequence of our logical characterization is that *equality qualifiers* can be added to the fragments with qualifiers, without increase in expressive power.

The second contribution is to outline the containments that hold between these fragments, and to discover what additional operations can be defined within them. Using our logical and pattern characterizations, we show that the containments holding between XPath fragments can differ significantly depending on whether their expressions are to be evaluated at arbitrary nodes of an XML document tree or are restricted to work only from the root of the tree; that is, we describe the containments that hold under *general equivalence* and *root equivalence*. In the process we show that every expression with upward modalities is root-equivalent to one without upward navigation. This is important for, among other things, processing streaming data.

Our third contribution is the delineation of the *closure properties* of XPath fragments. Knowing that a fragment F is closed under a set of operations S is not only helpful for optimization algorithms – manipulations involving S can be done while remaining in the fragment – it is also useful for XPath implementers, as it indicates that the operations in S can be built on top of the fundamental operations of the fragment. We give a complete picture of the closure properties under Boolean operations for all of our XPath fragments, making use of the logical and pattern characterizations mentioned above. As with containment, we show that the closure properties of a fragment under general equivalence may differ from those under root equivalence.

The final contribution of the paper is in connection with simplification of XPath expressions. We approach this problem based on *proof systems* for XPath using a set of *axioms* that allow simplification of expressions. This is an initial step toward establishing an algebraic framework both for directing the optimization process, and for allowing an optimizer to trade off weaker simplification for lower optimization time. Note that each individual fragment requires a separate analysis of the axiomatizability of containment/equivalence. Indeed, given fragments $F_1 \subset F_2$, F_2 may admit a simple set of simplification rules although F_1 does not, due to the lack of certain closure properties in F_1, and vice versa.

To this end, we establish some preliminary results, both positive and negative, for the axiomatizability of our XPath fragments. We show that no finite axiomatization can exist for any of our fragments; but we give sound and complete computable axiom systems for the downward non-qualified fragments. We show that some fragments actually become finitely axiomatizable when the labels are restricted to come from a fixed finite alphabet. We also provide *normal forms* for some of these fragments, which are unique up to equivalence of expressions.

Taken together, these results give a picture of the advantages and disadvantages of working within a particular fragment.

Related work: There has been considerable work on the expressiveness and complexity of XPath. One line of investigation studies formal models of XPath [3, 18, 17, 16], abstracting away from the concrete syntax of XPath into a powerful automaton model – complexity bounds are then derived from bounds on the automata. The results to date have generally given considerable insight into the expressiveness and complexity of XPath as a whole, but have shed little light on the distinction between one fragment of XPath and another. To the best of our knowledge, there is no previous work dealing directly with the expressiveness of XPath fragments.

An active area of research with direct bearing on XPath optimization has been the analysis of the *containment problem* for XPath: in a series of papers [1, 9, 15, 25], lower and upper bounds for the complexity of containment have been established for a number of XPath fragments. Note that understanding axiomatizability of containment in a fragment requires a fundamentally different analysis from the semantic methods used in these papers, since the existence of an axiom system is a property of the fragment *as a whole*. Solving containment itself is only a step toward optimization, but developing a framework for, and thorough study of, simplification in the context of XPath fragments remains an important open research issue.

Closest in spirit to our paper is [19]. It presents a set of rewrite rules for eliminating upward modalities, which is similar to some of our results in Section 5. However, [19] focuses on a single large XPath fragment, and their results do not apply to the smaller fragments considered here. Normal forms for one simple fragment of XPath are examined in [26]; for tree patterns, [1, 20] present algorithms for achieving minimization, which can be viewed as a certain normal form for tree patterns.

The issue of axiomatizing expression equivalence has been investigated for a number of formalisms related to XPath: [21] shows that there can be no finite axiom system for regular expressions, while [11] gives axiom systems for propositional dynamic logic with converse. Elimination of inverse roles (upward modality) has been studied for description logics [4]. The results of propositional dynamic logic and description logics do not carry over here since they deal with general structures (graphs) instead of trees; furthermore, since those languages support a negation operator, the expressiveness differs radically from XPath, which (as we shall see) is not negation-closed.

Organization: Section 2 defines our XPath fragments, followed by a characterization of their expressive power in Section 3. Closure properties are investigated in Sections 4 under general equivalence. Section 5 revisits expressiveness and closure properties under root equivalence. Section 6 provides axiom systems and normal forms for several fragments, and Section 7 summarizes the main results of the paper. Proofs are in the appendix.

2 Notations

XPath expressions are built up from an infinite set of labels (tags, names) Σ. The largest fragment of XPath studied in this paper, denoted by $\mathcal{X}_{r,[\,]}^{\uparrow}$, is syntactically defined as follows:

$$p \ ::= \ \epsilon \ | \ \emptyset \ | \ l \ | \ \downarrow \ | \ \uparrow \ | \ \downarrow^* \ | \ \uparrow^* \ | \ p/p \ | \ p \cup p \ | \ p[q],$$

where ϵ, $\emptyset$, l denote the empty path, the empty set, and a name in Σ, respectively; '$\cup$' and '/' stand for union and concatenation, '$\downarrow$' and '$\uparrow$' for the *child*-axis and *parent*-axis, '$\downarrow^*$' and '$\uparrow^*$' for the *descendant-or-self*-axis and *ancestor-or-self*-axis, respectively; and finally, q in $p[q]$ is called a *qualifier* and defined by:

$$q \ ::= \ p \mid \text{label} = l \ ,$$

where p is an $\mathcal{X}_{r,[\,]}^{\uparrow}$ expression and l is a name in Σ.

The semantics of XPath expressions is given with respect to an XML document modeled as a node-labeled tree, referred to as an *XML tree*. Each node n in an XML tree T (denoted by $n \in T$) is labeled with a name tag from some finite alphabet $\Sigma_0 \subset \Sigma$ (we assume w.l.o.g. that Σ_0 has at least two symbols in it). It also has a (possibly empty) list of children; it is called a *leaf* of T if it has no children. A distinguished node rt is called the *root* of T; each node in T except for the root has a parent. We do not consider order of nodes in T since our XPath fragments ignore the order. In an XML tree T, an $\mathcal{X}_{r,[\,]}^{\uparrow}$ expression p is interpreted as a binary predicate on the nodes of T. That is, for any n, $n' \in T$, $T \models p(n, n')$ iff one of the following is satisfied: (1) if $p = \epsilon$, then $n = n'$; (2) if $p = \emptyset$, then $T \models p(n, n')$ is false for any n'; (3) if $p = l$, then n' is a child of n, and is labeled with l; (4) if $p = \downarrow$, then n' is a child of n, and its label does not matter; (5) if $p = \uparrow$, then n' is the parent of n; (6) if $p = \downarrow^*$, then n' is either n itself or a descendant of n; (7) if $p = \uparrow^*$, then n' is either n itself or an ancestor of n; (8) if $p = p_1/p_2$, then there exists $x \in T$ such that $T \models p_1(n, x) \wedge p_2(x, n')$; (9) if $p = p_1 \cup p_2$, then $T \models p_1(n, n') \vee p_2(n, n')$; (10) if $p = p_1[q]$, then there are two cases: when q is p_2, there exists $n'' \in T$ such that $T \models p_1(n, n') \wedge p_2(n', n'')$; when q is a label test "label $= l$", n' is labeled with l. This semantics is in the same spirit as the one given in [24].

Example. Referring to a node n in an XML tree T, some $\mathcal{X}_{r,[\,]}^{\uparrow}$ expressions and their semantics are:

- p_1: $\downarrow/A$: all the grandchildren of n that are labeled A;
- p_2: $\uparrow/A$: all the A-siblings of n, including n itself if it is labeled A;
- p_3: $\downarrow/A[\downarrow]$: all the non-leaf A-grandchildren of n;
- p_4: $\uparrow[\text{label} = A]$: the parent of n if it is labeled A, and the empty set otherwise;
- p_5: $\downarrow^*/A/\downarrow^*$: all the descendants of n that have an A-ancestor which itself is a descendant of n;
- p_6: $\downarrow^*/A[B[\downarrow^*/C]]$: all the A-descendants of n that have a B-child which in turn has a C-descendant;
- p_7: $\downarrow^*/B/\uparrow^*$: nodes having a B-descendant which is also a descendant of n; note that these nodes are not necessarily themselves descendants of n;
- p_8: $\downarrow^*/A[\uparrow^*/B]$: all the A-descendants of n that have an ancestor with a B-child.

If $T \models p(n, n')$ then we say that n' is *reachable* from n via p. We use $n[\![p]\!]$ to denote the set of all the nodes reached from n via p, i.e., $\{n' | n' \in T,\ T \models p(n, n')\}$.

Example. For any leaf n in T, $n[\![\downarrow]\!] = \emptyset$, $n[\![\downarrow^*]\!] = \{n\}$, while $rt[\![\uparrow]\!] = \emptyset$ and $rt[\![\uparrow^*]\!] = \{rt\}$ for the root rt of T.

Two XPath expressions p and p' are *generally equivalent* (or simply *equivalent* when it is clear from the context), denoted by $p \equiv p'$, iff for any XML tree T and any node $n \in T$, $n[\![p]\!] = n[\![p']\!]$. We say two expressions are *equivalent over* Σ_0 (denoted by $\equiv_{\Sigma_0}$) where Σ_0 is a fixed finite alphabet, if the above holds for any tree T whose labels are in Σ_0. For example, $\downarrow$ is equivalent to $A \cup B$ over the alphabet $\{A, B\}$, but not in general. We will usually work with the stronger notion of general equivalence $\equiv$, and specify when results also hold for restricted equivalence — equivalence w.r.t. some finite alphabet Σ_0.

We use the following notations for subclasses of $\mathcal{X}_{r,[\,]}^{\uparrow}$: the subscript '[ ]' means that the subclass allows qualifiers, i.e., expressions of the form $p[q]$; the subscript 'r' indicates the support for "recursion" $\downarrow^*$; and the superscript '$\uparrow$' denotes the support for upward modality '$\uparrow$' (and '$\uparrow^*$' in presence of the subscript r).

The subclasses of $\mathcal{X}_{r,[\,]}^{\uparrow}$ considered in this paper can be classified into two categories: with or without recursion. In the first category, $\mathcal{X}_r^{\uparrow}$ is the subclass without qualifiers; $\mathcal{X}_{r,[\,]}$ is the subclass with qualifiers but without $\uparrow$ and $\uparrow^*$; and $\mathcal{X}_r$ is the subclass with neither qualifiers, $\uparrow$ nor $\uparrow^*$. In the second category, $\mathcal{X}_{[\,]}^{\uparrow}$ is the subclass of $\mathcal{X}_{r,[\,]}^{\uparrow}$ without $\downarrow^*$ and $\uparrow^*$; $\mathcal{X}_{[\,]}$ and $\mathcal{X}^{\uparrow}$ further restrict $\mathcal{X}_{[\,]}^{\uparrow}$ by disallowing, respectively, $\uparrow$ and qualifiers; finally, $\mathcal{X}$ is the subclass of $\mathcal{X}_{[\,]}^{\uparrow}$ with neither $\uparrow$ nor qualifiers. All these fragments have been found to be useful in practice. For example, $\mathcal{X}_r$ is used by XML Schema [23] to specify integrity constraints.

The proposition below justifies the lattice depicted

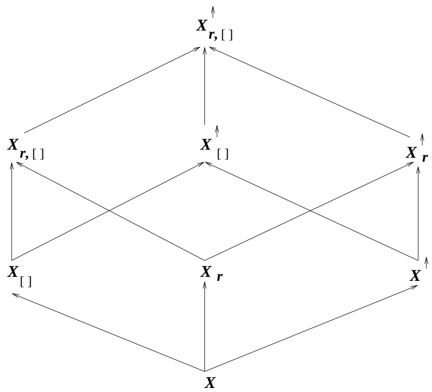

Figure 1: Fragments of XPath

in Fig. 1 (see the appendix for a proof).

Proposition 2.1: The fragments form the lattice of Fig. 1; that is, there is an edge from fragment F_1 to F_2 iff $F_1 \subseteq F_2$, i.e., if every expression of F_1 is equivalent to an expression in F_2. $\square$

Example. The XPath expressions given above can be classified as follows: p_1 is in all these fragments; p_2 is in $\mathcal{X}^{\uparrow}$, $\mathcal{X}^{\uparrow}_{[\,]}$, $\mathcal{X}^{\uparrow}_{r}$, $\mathcal{X}^{\uparrow}_{r,[\,]}$, but is not in the others; p_3 is in $\mathcal{X}_{[\,]}$, $\mathcal{X}^{\uparrow}_{[\,]}$, $\mathcal{X}_{r,[\,]}$, and $\mathcal{X}^{\uparrow}_{r,[\,]}$ only, while p_4 is in $\mathcal{X}^{\uparrow}_{[\,]}$ and $\mathcal{X}^{\uparrow}_{r,[\,]}$ only; p_5 is in $\mathcal{X}_{r}$, $\mathcal{X}^{\uparrow}_{r}$, $\mathcal{X}_{r,[\,]}$ and $\mathcal{X}^{\uparrow}_{r,[\,]}$ but is not in the others; p_6 is in $\mathcal{X}_{r,[\,]}$, $\mathcal{X}^{\uparrow}_{r,[\,]}$ only while p_7 is in $\mathcal{X}^{\uparrow}_{r}$ and $\mathcal{X}^{\uparrow}_{r,[\,]}$ only; finally, p_8 is only in $\mathcal{X}^{\uparrow}_{r,[\,]}$.

A weaker equivalence relation is defined as follows: p and p' are *root equivalent*, denoted by $p \equiv_r p'$, iff for any XML tree T, $rt[\![p]\!] = rt[\![p']\!]$, where rt is the root of T. In Section 5 we shall see that root equivalence flattens the hierarchy of Fig. 1.

The fragments studied by [15] do not support upward traversals and union, and are properly contained in $\mathcal{X}_{r,[\,]}$. Note that because the fragments considered in [15] do not deal with upward modalities such as $\uparrow$ and $\uparrow^*$, the distinction between root equivalence and general equivalence is not relevant to this prior work, since these notions are the same in the absence of upward modalities. Although [9] considers fragments similar to $\mathcal{X}^{\uparrow}_{r,[\,]}$, it does not study the closure properties and axiom systems investigated here.

3 Logic and qualified fragments

In this section we characterize the expressiveness of each of the fragments with qualifiers defined in the previous section, by means of both predicate logic and tree patterns. As we shall see, the logical characterization is not only interesting in its own right, but is also useful in the analysis of the closure properties of our fragments.

We begin with a simple observation. One might be interested in extending qualifiers in an $\mathcal{X}^{\uparrow}_{r,[\,]}$ expression $p[q]$ by including conjunction and disjunction:

$$q \quad ::= \quad p \ \mid \ \text{label} = l \ \mid \ q \wedge q \ \mid \ q \vee q,$$

with the semantics: at any node n in an XML tree T, $[q_1 \wedge q_2]$ is true iff there exist nodes n' and n'' such that both $q_1(n, n')$ and $q_2(n, n'')$, and such $[q_1 \vee q_2]$ is true iff $q_1(n, n')$ or $q_2(n, n'')$. Let us denote this extension of $\mathcal{X}^{\uparrow}_{r,[\,]}$ by $\mathcal{X}^{\uparrow}_{r,[\wedge,\vee]}$. The next proposition shows, however, that conjunction and disjunction add no expressive power over $\mathcal{X}^{\uparrow}_{r,[\,]}$ (see the appendix for a proof). This allows us to use conjunction and disjunction as shorthands in qualifiers of $\mathcal{X}^{\uparrow}_{r,[\,]}$ expressions. This also carries over to all of the other fragments with qualifiers, i.e., $\mathcal{X}_{r,[\,]}$, $\mathcal{X}^{\uparrow}_{[\,]}$, and $\mathcal{X}_{[\,]}$.

Proposition 3.1: $\mathcal{X}^{\uparrow}_{r,[\,]}$ and $\mathcal{X}^{\uparrow}_{r,[\wedge,\vee]}$ are equivalent. $\square$

We next show that each of these fragments with qualifiers can be captured using a version of *positive-existential first-order logic*, and define notions of tree patterns that capture the expressive power of each of these fragments. A precursor to this work is [15] where in analyzing the subset of $\mathcal{X}_{r,[\,]}$ consisting of expressions built up without the union operator '$\cup$', they observe that this fragment is equivalent to the queries defined using a natural notion of "unary tree pattern".

The formal definitions of our logics and pattern languages are given below.

Let $\exists^+(\mathit{child})$ be the fragment of first order logic built up from the relations *child*, label predicates $P(x)$ for each name P as well as equality '=', by closing under $\wedge$, $\vee$ and $\exists$, while $\exists^+(\mathit{child}, \mathit{desc})$ is the corresponding fragment built up from *child*, *desc*, the label predicates and '='. The semantics is the standard semantics of first-order logic over trees (with *child* and *desc* given their standard interpretation within a tree).

We use $\exists^+(\mathit{child})(c, s)$ to denote $\exists^+(\mathit{child})$ formulae with exactly the variables c and s free, and similarly for $\exists^+(\mathit{child}, \mathit{desc})(c, s)$. Note that an $\exists^+(\mathit{child})(c, s)$ formula defines a function from a node c to a set of nodes s.

A *forest pattern pc* is a forest with labels on nodes and edges, where nodes are labeled by names or the wildcard symbol '$*$' (that matches any node), and edges are labeled with $\downarrow$ or $\downarrow^*$. It has a distinguished node called the *context node*, and a distinguished subset of nodes referred to as the *selected nodes* of *pc*. A pattern *pc* thus has the form (V, E, l, c, S), where V is the underlying forest domain, E the ordering, l the labeling function, and c, S the context node and selected set, respectively.

A forest pattern can be given semantics by translating it into $\exists^+(\mathit{child}, \mathit{desc})$. The pattern (V, E, l, c, S) is translated as follows: Letting $V = v_1 \dots v_n$ be the elements of V, assume, w.l.o.g., that v_0 is the context node, $v_1 \dots v_k$ are the selected nodes, and $v_{k+1} \dots v_n$ are the remaining nodes of V. Then let $\phi(x_{v_0}, \dots, x_{v_k})$ be the formula:

$$\exists x_{v_{k+1}} \ \dots \ x_{v_n} \ (\bigwedge_P \ \bigwedge_{v \in V,\ l(v)=P} P(x_v) \ \wedge \ \bigwedge_{(v,v') \in E,\ l(v,v')=\mathit{child}} \mathit{child}(x_v, x_{v'}) \ \wedge \ \bigwedge_{(v,v') \in E,\ l(v,v')=\mathit{desc}} \mathit{desc}(x_v, x_{v'}))$$

Special kinds of patterns we consider are:

- A *tree pattern* is a forest pattern consisting of a single tree.
- A *child pattern* is one in which all edges are labeled with $\downarrow$.
- A *unary pattern* is one in which the selected set S has exactly one node in it.

A *forest pattern query* is a finite set of forest patterns, with all patterns having the same cardinality for the selected set S (designed to model the arity of the query). A forest pattern query t returns, given a tree and a distinguished context node, the union of all outputs returned by the patterns in it, i.e., $[t] = \bigcup_{tp \in t}[tp]$, where $[tp]$ is the relation defined by tp. By convention we take the empty forest pattern query to be equivalent to *false*. We likewise talk about a child pattern query, unary tree pattern query, etc (note that a tree pattern query is a *finite set* of tree patterns). We let $\bigcup\mathrm{TP}(\mathit{child})$ be the set of unary child tree pattern queries and $\bigcup\mathrm{TP}(\mathit{child}, \mathit{desc})$ be the set of unary tree pattern queries.

It is easy to see that a tree pattern query can be expressed in $\exists^+(\mathit{child}, \mathit{desc})$, and that a child pattern query can be expressed in $\exists^+(\mathit{child})$. A unary pattern can be translated as above to a formula $\phi(x_{v_0}, x_{v_n})$: renaming x_{v_0} as c and x_{v_n} as s, we get a formula $\phi(c, s)$.

The first result is for the fragment $\mathcal{X}^{\uparrow}_{r,[\,]}$.

Theorem 3.2: The following languages are equivalent in expressive power

- $\mathcal{X}^{\uparrow}_{r,[\,]}$,
- $\bigcup\mathrm{TP}(\mathit{child}, \mathit{desc})$,
- $\exists^+(\mathit{child}, \mathit{desc})(c, s)$. □

Proof sketch: The proof consists of three parts.

1. $\bigcup\mathrm{TP}(\mathit{child}, \mathit{desc}) \subseteq \mathcal{X}^{\uparrow}_{r,[\,]}$.

It suffices to show that each unary tree pattern $p = (V, E, l, c, s)$ has an equivalent formula in $\mathcal{X}^{\uparrow}_{r,[\,]}$. We handle the case where c is neither a descendant nor an ancestor of s; the other two cases are similar but simpler. We first define a mapping Q from nodes $v \in V$ to qualifiers. The mapping is defined inductively (starting from the leaves) as follows:

$$\begin{array}{rcll} Q(v) & = & (\mathit{label} = l(v)) & \texttt{--}\ \texttt{if}\ l(v) \neq \text{`}*\text{'} \\ & & \wedge \displaystyle\bigwedge_{(v,v') \in E,\, l(v,v')=\mathit{desc}} & \downarrow^*/[Q(v')] \\ & & \wedge \displaystyle\bigwedge_{(v,v) \in E,\, l(v,v')=\mathit{child}} & \downarrow/[Q(v')] \end{array}$$

Now let rt be the root node of (V, E). Let $c = v_0, \dots, v_n = rt$ be the path from c to rt, and let $rt = y_0, \dots, y_m = s$ be the path from rt to s. Let u_i: $0 \leq i \leq n-1$ be defined by: $u_i = \uparrow$ if $l(v_i, v_{i+1}) = \mathit{child}$ and $\uparrow^*$ otherwise. Let $d_i = \downarrow$ if $l(y_i, y_{i+1}) = \mathit{child}$ and $\downarrow^*$ otherwise.

Then we translate the pattern p to

$$\epsilon[Q(v_0)]/u_0[Q(v_1)] \cdots /u_{n-2}[Q(v_{n-1})]/u_{n-1}[Q(rt)]/ \\ d_0[Q(y_1)]/\cdots/d_m[Q(s)] \ .$$

That is, starting at the context node c, the $\mathcal{X}^{\uparrow}_{r,[\,]}$ expression first goes upward to the root rt and then downward to the selected nodes s. One can verify that this translation is correct.

2. $\mathcal{X}^{\uparrow}_{r,[\,]} \subseteq \exists^+(\mathit{child}, \mathit{desc})(c, s)$.

This is immediate from the definition (logic translation) of the semantics of $\mathcal{X}^{\uparrow}_{r,[\,]}$ expressions given in the previous section.

3. $\exists^+(\mathit{child}, \mathit{desc})(c, s) \subseteq \bigcup\mathrm{TP}(\mathit{child}, \mathit{desc})$.

We prove the more general fact that an *arbitrary* formula $\phi(c, s_1, \dots, s_n)$ in $\exists^+(\mathit{child}, \mathit{desc})$ is equivalent to a tree pattern query. Let ϕ be a formula. We can assume that it is of the form $\exists x_1 \dots x_n \ \gamma$ where γ is quantifier-free. We now create a graph whose vertices are the variables of γ and whose edges correspond to the atomic formulae in γ (with nodes and edges labeled by the corresponding unary and binary atomic formulae satisfied by these variables). By combining strongly-connected components into a single node and identifying any two variables that are forced by γ to be the same, we can get an equivalent formula for which the corresponding graph is a tree. From this tree it is easy to generate a tree pattern query. Details are in the appendix.

This completes the proof of the theorem. $\square$

We now consider the situation for $\mathcal{X}^{\uparrow}_{[\,]}$. Part of the previous result goes through as before. To capture the expressive power of $\mathcal{X}^{\uparrow}_{[\,]}$ precisely, let $\exists^+(\mathit{child})[\mathit{loc}]$ be the first-order logic built up from *child* and the label predicates via conjunction, disjunction and the quantification $\exists x \in B(c, n)$ (*local quantification*), for every integer n. Here $\exists x \in B(c, n)\ \phi(x, \vec{y})$ holds iff there is a connected set of nodes of size at most n in the tree that contains x and c (we say "x is in the ball of radius n around c" in this case). We let $\exists^+(\mathit{child})[\mathit{loc}](c, \vec{s})$ be the fragment of $\exists^+(\mathit{child})[\mathit{loc}]$ with the further restriction that each s_i is restricted to be in the ball of radius n around c. It is clear that every property expressible in $\exists^+(\mathit{child})[\mathit{loc}]$ can be expressed in $\exists^+(\mathit{child})$ (by making the chain leading to c explicit). Then we have (see the appendix for a proof):

Theorem 3.3: The following languages are equivalent in expressive power

- $\mathcal{X}^{\uparrow}_{[\,]}$,
- $\bigcup\mathrm{TP}(\mathit{child})$,
- $\exists^+(\mathit{child})[\mathit{loc}](c, s)$. $\square$

As an immediate result of Theorems 3.2 and 3.3, it follows that equality test in qualifiers adds no expressive power over $\mathcal{X}^{\uparrow}_{[\,]}$ and $\mathcal{X}^{\uparrow}_{r,[\,]}$. XPath allows qualifiers of the form $[q_1 = q_2]$ with the following semantics: at any node n in an XML tree T, $[q_1 = q_2]$ is true iff there exists a node n' such that both $q_1(n, n')$ and $q_2(n, n')$ hold, i.e., $n[\![q_1]\!]$ and $n[\![q_2]\!]$ are not disjoint. Note that the semantics of equality test in XPath is quite different from that of qualifier conjunction.

Corollary 3.4: The extensions of $\mathcal{X}^{\uparrow}_{[\,]}$ and $\mathcal{X}^{\uparrow}_{r,[\,]}$ to allow the equality test $q_1 = q_2$ in qualifiers are the same as $\mathcal{X}^{\uparrow}_{[\,]}$ and $\mathcal{X}^{\uparrow}_{r,[\,]}$, respectively. $\square$

Proof sketch: This follows from the fact that the statements of the form $[q_1 = q_2]$ can clearly be expressed in $\exists^+(\mathit{child})$ and $\exists^+(\mathit{child}, \mathit{desc})$: let $\phi_1(s_1)$ and $\phi_2(s_2)$ be the $\exists^+(\mathit{child})$ (resp. $\exists^+(\mathit{child}, \mathit{desc})$) formulae representing q_1 and q_2, with free variables s_1 and s_2 denoting the selected nodes; then $[q_1 = q_2]$ is equivalent to $\exists x(\phi_1(x) \wedge \phi_2(x))$. $\square$

For example, the expression

$$[A/{\downarrow^*}/B/{\downarrow^*} = A/{\downarrow^*}/C/{\downarrow^*}]$$

can be converted to

$$[A/{\downarrow^*}/B/{\downarrow^*}/C/{\downarrow^*} \cup A/{\downarrow^*}/C/{\downarrow^*}/B/{\downarrow^*}].$$

There is an additional equivalence between the full logic $\exists^+(\mathit{child})$ and forest patterns, which can be verified along the same lines as Theorems 3.2 and 3.3.

Theorem 3.5: Every formula in $\exists^+(\mathit{child})$ is equivalent to a child forest query, and vice versa. $\square$

We now turn to the downward fragments. Let us define $\bigcup\mathrm{TP}(\mathit{child}, \mathit{desc})[\mathit{down}]$ to be tree pattern queries where the node labeled c is restricted to be at the root of each query. Let $\exists^+(\mathit{child}, \mathit{desc})[\mathit{down}](c, s)$ be the fragment of $\exists^+(\mathit{child}, \mathit{desc})(c, s)$ in which every bound variable as well as s is syntactically restricted to be a descendant-or-self of the node c. Then similarly to Theorem 3.2 we have (see the appendix for a proof):

Theorem 3.6: The following languages are equivalent in expressive power:

- $\mathcal{X}_{r,[\,]}$,
- $\bigcup\mathrm{TP}(\mathit{child}, \mathit{desc})[\mathit{down}]$,
- $\exists^+(\mathit{child}, \mathit{desc})[\mathit{down}](c, s)$. $\square$

Now we give the characterizations for $\mathcal{X}_{[\,]}$. We define $\bigcup\mathrm{TP}(\mathit{child})[\mathit{down}]$ to be the fragment of $\bigcup\mathrm{TP}(\mathit{child})$ with the node c at the root of each tree pattern, and let $\exists^+(\mathit{child})[\mathit{loc}, \mathit{down}]$ be the intersection of the languages $\exists^+(\mathit{child})[\mathit{loc}]$ and $\exists^+(\mathit{child}, \mathit{desc})[\mathit{down}]$, i.e., every variable and also s are syntactically restricted to be a fixed descendant of c. Then similar to Theorem 3.3, we have (see the appendix for a proof):

Theorem 3.7: The following languages are equivalent in expressive power

- $\mathcal{X}_{[\,]}$,
- $\bigcup\mathrm{TP}(\mathit{child})[\mathit{down}]$,
- $\exists^+(\mathit{child})[\mathit{loc}, \mathit{down}](c, s)$. $\square$

4 Closure properties

An XML query frequently involves union, intersection and complementation of XPath expressions. This motivates the study of the closure properties of XPath under these Boolean operations, i.e., whether the Boolean operations preserve our XPath fragments. This is important for, among other things, query optimization. The XPath 2.0 draft [2] proposes adding all these operations;

however closure properties of the fragments studied here will remain relevant, given that most applications will use only a portion of XPath 2.0, and that XPath and XQuery implementations will focus their optimization efforts on particular subsets of the language.

Formally, a fragment F of XPath is *closed under intersection* iff for any expressions p_1, p_2 in F, there exists an expression p in F, denoted by $p_1 \cap p_2$, such that $n[\![p]\!] = n[\![p_1]\!] \cap n[\![p_2]\!]$ for any XML tree T and any node $n \in T$. Similarly, F is *closed under complementation* iff for any p in F, there exists p' in F, denoted by $\neg p$, such that $n[\![p']\!] = \{n' \mid n \in T,\ n \notin n[\![p]\!]\}$ for any XML tree T and $n \in T$. Closure under union can be defined similarly; all of our fragments are closed under union since they all contain the operator '$\cup$'.

Theorem 4.1: The fragments $\mathcal{X}$, $\mathcal{X}_{[\,]}$, $\mathcal{X}^{\uparrow}_{[\,]}$, $\mathcal{X}_r$, $\mathcal{X}_{r,[\,]}$, and $\mathcal{X}^{\uparrow}_{r,[\,]}$ are closed under intersection, whereas $\mathcal{X}^{\uparrow}$ and $\mathcal{X}^{\uparrow}_r$ are not. □

Proof sketch: Theorems 3.2, 3.3, 3.6 and 3.7 characterize the qualified fragments $\mathcal{X}^{\uparrow}_{r,[\,]}$, $\mathcal{X}^{\uparrow}_{[\,]}$, $\mathcal{X}_{r,[\,]}$ and $\mathcal{X}_{[\,]}$ with the logics $\exists^+(\mathit{child}, \mathit{desc})(c,s)$ and $\exists^+(\mathit{child})(c,s)$ as well as their restrictions $\exists^+(\mathit{child}, \mathit{desc})[\mathit{down}](c,s)$ $\exists^+(\mathit{child})[\mathit{loc}, \mathit{down}](c,s)$, respectively. In particular, for expressions p_1 and p_2 in any of these fragments, their intersection can be expressed in the corresponding logic as $\exists y_1 \phi_1(x,y_1) \wedge \exists y_2 \phi_2(x,y_2)$, where ϕ_1 and ϕ_2 are the codings of p_1 and p_2 respectively. From this, it immediately follows that these fragments are closed under intersection since the logics are clearly closed under conjunction.

We now show that $\mathcal{X}_r$ is closed under intersection using an automata-theoretic approach. A similar argument also works for $\mathcal{X}$.

Let p be an expression in $\mathcal{X}_r$. Acceptance of a node n by such a p depends only on the labels on a path from the context node c to n, hence p can be modeled by an automaton that accepts *strings* with c being its start state. The automaton corresponding to p can be taken to be acyclic, with the exception of self-loops which can be followed regardless of the alphabet symbol (corresponding to the symbol $\downarrow^*$). Conversely, any such automaton can be converted to an $\mathcal{X}_r$-expression by taking the union of all paths from the start to an accepting state, introducing $\downarrow^*$ wherever such a self-loop occurs. We now show that these restricted automata are closed under the standard product construction. Let M and M' be two such automata, and let M'' be the result of the standard Cartesian product construction of the intersection [13]. We show that M'' corresponds to an $\mathcal{X}_r$-expression. To see this, assume by contradiction that the transition graph of M'' has a cycle which is not a self-loop. Let (q_1, q'_1), ... , (q_n, q'_n) be a cycle in M'' with $n > 1$ with all pairs distinct. Then q_1, ... , q_n, q_1 and q'_1, ... , q'_n, q'_1 are cycles in M and M', respectively. But then, $q_1 = q_2 = \cdots = q_n$ and $q'_1 = q'_2 = \cdots = q'_n$, using the fact that M and M' have no non-self-loop cycles. From this we get a contradiction.

To show the negative results, consider the intersection $\epsilon \cap \uparrow/A/\uparrow/\downarrow$ of two $\mathcal{X}^{\uparrow}$ expressions – this expression returns the context node alone, but only if it has a sibling labeled 'A'. We shall show that this intersection cannot be expressed in $\mathcal{X}^{\uparrow}_r$, and thus not in $\mathcal{X}^{\uparrow}$ either. To see this, suppose that p were such an $\mathcal{X}^{\uparrow}_r$ expression. Let T be the tree with root rt, having children n_1 (labeled B), n_2 (labeled A), and n_3 (labeled B). Let n_1 be the context node; clearly $n_1[\![p]\!] = \{n_1\}$. Then there must exist a *simple path* p', i.e., a sequence of labels and $\uparrow$, such that $n_1[\![p']\!] = \{n_1\}$ and for every tree T and node n, $n[\![p']\!] \subseteq n[\![p]\!]$. Let $v_1 \ldots v_k$ be a sequence of nodes starting and ending n_1 that witnesses $n_1 \in n_1[\![p']\!]$. If none of the v_i is rt, then n_1 will be in $n_1[\![p]\!]$ even if we delete the node n_2 from the tree, a contradiction. If some v_j is rt, then replace each subsequent occurrence of n_1 in the path by n_3, and it follows that $n_1[\![p']\!]$ must include the node n_3, once more a contradiction. This concludes the proof. □

For complementation, on the other hand:

Theorem 4.2: None of the fragments is closed under complementation. □

Proof sketch: We consider here the case of equivalence over a fixed finite alphabet Σ_0 – the proofs will also work for general equivalence, but in such a case one could simply show that the complement of the simple path "A" cannot be represented in any of the fragments.

We distinguish between two groups of fragments – those with recursion ($\downarrow^*$ and possibly $\uparrow^*$), and those without.

First, consider a non-recursive fragment. Let p_1 be the simple path "A". We claim that $p = \neg(p_1)$ cannot be represented in this fragment. To see this, let m be the size of p. It is easy to see that p cannot match any path of length $> m$, a contradiction.

In the case of recursive fragments, when fragments are considered w.r.t. equivalence over a fixed finite alphabet Σ_0, this argument does not work: if the alphabet is $\{A, A_2, \ldots, A_n\}$, $\neg A$ can be represented as $\epsilon \cup A_2 \cup \cdots \cup A_n \cup \downarrow/\downarrow/\downarrow^*$. We use a different approach,

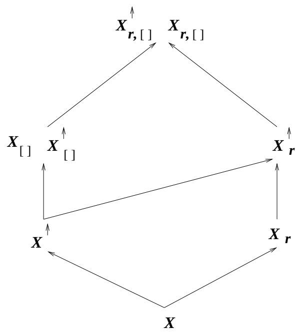

Figure 2: XPath fragments under root equivalence

and let $p_1 = \downarrow^*/A/\downarrow^*$. Note that p_1 does not make use of $\uparrow$ or of qualifiers, and hence is in all of the recursive fragments. We claim that the complement of p_1 cannot be expressed in $\mathcal{X}^{\uparrow}_{r,[\,]}$, and hence not in any of the smaller recursive fragments.

We show this by proving that the complement of p_1 cannot be expressed in $\exists^{+}(\mathit{child}, \mathit{desc})(c, s)$. Let p be a representation of the complement of p_1 in this logic, of size m. Consider two structures T_1 and T_2. The structure T_1 consists of a root rt followed by a chain of length 2^{m+1}, with all these nodes labeled B (and hence the end of the chain is in the complement with respect to context node rt). The structure T_2 is similar, except that a node in the middle of the chain is labeled A (and hence the node at the end of the chain is not in the complement). A simple argument shows that every m-quantifier $\exists^{+}(\mathit{child}, \mathit{desc})(c, s)$ sentence holding in T_1 also holds in T_2, a contradiction. $\square$

5 Root equivalence

Recall the notion of root equivalence defined in Section 2. Under this weaker equivalence relation, the hierarchy of Fig. 1 collapses: the XPath fragments form the lattice of Fig. 2. This follows from:

Theorem 5.1: Under root equivalence,

1. $\mathcal{X}^{\uparrow}_{r} \subset \mathcal{X}_{r,[\,]}$ but $\mathcal{X}_{r,[\,]} \not\subseteq \mathcal{X}^{\uparrow}_{r}$;
2. $\mathcal{X}^{\uparrow} \subseteq \mathcal{X}_{[\,]}$; moreover, in the absence of label tests, i.e., qualifiers of the form $[\mathit{label} = l]$, $\mathcal{X}^{\uparrow} = \mathcal{X}_{[\,]}$;
3. $\mathcal{X}_{r,[\,]} = \mathcal{X}^{\uparrow}_{r,[\,]}$;
4. $\mathcal{X}_{[\,]} = \mathcal{X}^{\uparrow}_{[\,]}$. $\square$

As an immediate consequence, any XPath expression with upward modalities in any of these fragments is root-equivalent to an XPath expression with neither $\uparrow$ nor $\uparrow^*$. This is important for, among other things, processing streaming data.

Proof sketch:

1. $\mathcal{X}^{\uparrow}_{r} \subset \mathcal{X}_{r,[\,]}$.

First note that $p/(p_1 \cup p_2)/p' \equiv_r p/p_1/p' \cup p/p_2/p'$. Thus, w.l.o.g., we assume that any $\mathcal{X}^{\uparrow}_{r}$ expression is of the form $p_1 \cup \cdots \cup p_k$, where each p_i $(1 \le i \le k)$ has the form $\eta_1/\cdots/\eta_m$, and each η_j is one of $\emptyset$, ϵ, l, $\downarrow$, $\downarrow^*$, $\uparrow$, or $\uparrow^*$. It suffices to show that each p_i can be converted to an equivalent $\mathcal{X}_{r,[\,]}$ expression. The conversion is done from left to right, starting with the first occurrence of $\uparrow$ or $\uparrow^*$, using the following rewriting rules (we omit the trivial rules for $\emptyset$ and ϵ):

$$
\begin{aligned}
\uparrow/p \quad &\Rightarrow \quad \emptyset, \text{ if } \uparrow \text{ is the first symbol of } p_i\,;\\
\uparrow^*/p \quad &\Rightarrow \quad p, \text{ if } \uparrow^* \text{ is the first symbol of } p_i\,;\\
p/\eta/\uparrow \quad &\Rightarrow \quad p[\eta], \text{ where } \eta \text{ is } \downarrow \text{ or a label}\,;\\
\eta_1/\cdots/\eta_n/\uparrow^* \quad &\Rightarrow \quad \eta_1[\eta_2/\cdots/\eta_n] \cup \cdots\\
&\qquad \cup\, \eta_1/\cdots/\eta_i[\eta_{i+1}/\cdots/\eta_n] \cup \cdots\\
&\qquad \cup\, \eta_1/\cdots/\eta_n,\\
&\qquad \text{where } \eta_i \text{ is } \downarrow, \downarrow^* \text{ or a label.}
\end{aligned}
$$

In other words, if η_1 is $\uparrow$ (resp. $\uparrow^*$), the first (resp. second) rule is applied to eliminate it; otherwise we eliminate $\uparrow$ and $\uparrow^*$ by introducing qualifiers using the other rules. In these rules, p, p_1, and p_2 denote $\mathcal{X}^{\uparrow}_{r}$ expressions without upward modalities, and the left-to-right conversion implies that a rewriting rule is applied only if its left-hand side matches the prefix of an $\mathcal{X}^{\uparrow}_{r}$ expression. Although the rewriting may introduce '$\cup$', one needs only to consider p_i's that do not contain '$\cup$' because of the rewriting rules for '$\cup$' referred to earlier. It is straightforward to verify that each rewriting rule is root-equivalence preserving and that p_i can be converted to an $\mathcal{X}_{r,[\,]}$ expression in a finite number of steps, concluding the proof of containment.

The containment is proper because the $\mathcal{X}_{r,[\,]}$ expression $\downarrow^*/A[\downarrow^*/B]$ is not root-equivalent to any $\mathcal{X}^{\uparrow}_{r}$ expression. See the appendix for a proof.

2. $\mathcal{X}^{\uparrow} \subseteq \mathcal{X}_{[\,]}$, and furthermore, $\mathcal{X}^{\uparrow} = \mathcal{X}_{[\,]}$ in the absence of label tests.

The proof that $\mathcal{X}^{\uparrow}_{r} \subseteq \mathcal{X}_{r,[\,]}$ given above also shows $\mathcal{X}^{\uparrow} \subseteq \mathcal{X}_{[\,]}$. For the other direction, it suffices to show that without label tests in qualifiers, each $\mathcal{X}_{[\,]}$ expression can be rewritten to an equivalent $\mathcal{X}^{\uparrow}_{r}$ expression. This can be done by eliminating the qualifiers using the rewriting rules, along the same lines as the proof of part *1* above (see the appendix).

Note that in the presence of label tests, $\mathcal{X}_{[\,]} \not\subseteq \mathcal{X}^{\uparrow}$.

A concrete example of this is $\epsilon[label = A]$, i.e., testing whether the root is labeled A. It is easy to show that this $\mathcal{X}_{[\,]}$ expression is not expressible in $\mathcal{X}^{\uparrow}$.

3. $\mathcal{X}_{r,[\,]} = \mathcal{X}^{\uparrow}_{r,[\,]}$.

From Theorems 3.2 and 3.6, $\mathcal{X}_{r,[\,]}$ and $\mathcal{X}^{\uparrow}_{r,[\,]}$ are equivalent to the tree pattern queries $\bigcup$TP(*child*, *desc*) and $\bigcup$TP(*child*, *desc*)[*down*] respectively. Recall that $\bigcup$TP(*child*, *desc*)[*down*] is defined to be the fragment of $\bigcup$TP(*child*, *desc*) where the context node labeled c is restricted to be at the root of each query. Under root equivalence, the context node c is always explicitly restricted to be at the root. Hence, $\bigcup$TP(*child*, *desc*) and $\bigcup$TP(*child*, *desc*)[*down*] have the same expressive power under root equivalence; the same holds for $\mathcal{X}^{\uparrow}_{r,[\,]}$ and $\mathcal{X}_{r,[\,]}$.

4. $\mathcal{X}_{[\,]} = \mathcal{X}^{\uparrow}_{[\,]}$.

In a similar way to part *3*, we use Theorems 3.3 and 3.7 to show that $\mathcal{X}_{[\,]}$ and $\mathcal{X}^{\uparrow}_{[\,]}$ are root equivalent to each other. □

As a consequence of Theorem 5.1 the following can be easily verified along the same lines as Proposition 2.1.

Corollary 5.2: Under root equivalence, the fragments form the lattice of Fig. 2; that is, there is an edge from fragment F_1 to F_2 iff $F_1 \subseteq F_2$, i.e., if every expression of F_1 is root-equivalent to an expression in F_2. □

Recall that $\mathcal{X}^{\uparrow}$ is not closed under intersection as far as general equivalence is concerned. The situation is different when we consider root equivalence:

Corollary 5.3: Under root equivalence, all of our fragments except for $\mathcal{X}^{\uparrow}_{r}$ are closed under intersection. However, they remain not closed under complementation. □

This can be verified using Theorems 4.1, 4.2 and 5.1 (see the appendix for a proof).

6 Axiom systems and normal forms

We next study axiomatizability and normal forms for XPath. In particular, we present preliminary results for two fragments, namely, $\mathcal{X}_r$ and $\mathcal{X}$. Although the results of this section are established under full equivalence, they also hold under root equivalence.

6.1 Axiom systems

An XPath term over a fragment F is built up from the constructors of F, but supplementing the base case of labels (names) and ϵ with a set of *expression variables* (ranged over by $p_1, \ldots, p_n$). For a term τ over F, the variables used in constructing τ are called the *free variables of* τ. All the XPath expressions that we have seen before are just terms with no free variables, and will also be referred to as *ground terms*, or simply *expressions*.

An *equation* for a fragment F is an equality of terms. Given a set of equations A in F, the equivalence relation *equivalence modulo* A, $\equiv_A$, is the smallest equivalence relation between terms that contains the symmetric closure of A (considering A as a binary relation between terms) and closed under the inference rules:

1. if $\tau \equiv_A \tau'$ then $\sigma(\tau) \equiv_A \sigma(\tau')$ for any substitution σ that maps the free variables in τ or τ' to expressions of F;
2. if $\tau \equiv_A \tau'$ then $\gamma \equiv_A \gamma'$, where γ' is obtained from γ by replacing some occurrence of subexpression τ in γ with τ'.

A set of equations A is *sound* for a fragment F of XPath if for every expressions τ_1 and τ_2 of F, $\tau_1 \equiv_A \tau_2$ implies $\tau_1 \equiv \tau_2$, and is *complete* if for every such τ_1 and τ_2, $\tau_1 \equiv \tau_2$ implies $\tau_1 \equiv_A \tau_2$. A *(resp. finite) axiom system* for F is a (resp. finite) set of equations that is sound and complete. A fragment of XPath is said to be *axiomatizable* iff it has an axiom system, and *finitely axiomatizable* iff it has a finite axiom system.

A (finite) axiom system for a fragment of XPath is useful in, among other things, optimizing and normalizing the XPath expressions.

Unfortunately, none of the fragments considered in this paper is finitely axiomatizable.

Proposition 6.1: None of the fragments is finitely axiomatizable. □

Proof sketch: We show that $\mathcal{X}$ is not finitely axiomatizable. The same proof also applies to other fragments.

Assume, by contradiction, that there were a finite axiomatization A for $\mathcal{X}$. Since A has finitely many axioms, there are only finitely many labels of Σ mentioned in A. Let c be a name that does not appear in A. Observe that $c \cup {\downarrow} \equiv {\downarrow}$ holds. Since A is complete, there must be a proof S of $c \cup {\downarrow} \equiv_A {\downarrow}$ using the inference rules and axioms in A. Since c is not in any of these axioms, and c appears in the proof, it can only be introduced by substituting some $\mathcal{X}$ expression E containing

c for some variable p using the first inference rule above. Note that this proof will still be a proof if, in each such E, one substitutes a path c/c for each occurrence of c. This yields a proof S' which is the same as S, except that c/c appears in S' wherever c appears in S. Thus it is a proof for $c/c \cup \downarrow \ \equiv_A \ \downarrow$, which does not hold. Hence A cannot be sound. Therefore, it is not a finite axiomatization for $\mathcal{X}$. □

6.2 Axiom systems for $\mathcal{X}_r$ and $\mathcal{X}$

We next present a natural set of axiom schemas for two fragments, namely, $\mathcal{X}_r$ and $\mathcal{X}$. We show that when equivalence over a finite alphabet Σ_0 is considered, $\mathcal{X}$ is finitely axiomatizable. One application of these axiom systems, deriving a normal form for expressions in these fragments, will be shown later.

It is worth mentioning that since the equivalence problem for XPath expressions in all of our fragments is decidable (e.g. by appealing to [16]), a trivial computable axiomatization can be taken for any of our fragments by simply enumerating all ground equalities. Clearly such an axiomatization would not assist simplification of XPath, and would also not help in providing finite axiomatizations for restricted equivalence.

Fragment $\mathcal{X}_r$. Table 1 provides an axiom system for $\mathcal{X}_r$, denoted by $\mathcal{I}_r$. The system is infinite since for each label l in the alphabet Σ, there is an l-child rule in $\mathcal{I}_r$. When considered with respect to equivalence over any fixed finite alphabet Σ_0, $\mathcal{I}_r$ is still sound, but it is no longer complete. Note that by the *concat-associativity* axiom in $\mathcal{I}_r$, we can write $\tau_1/\tau_2/\tau_3$ for $(\tau_1/\tau_2)/\tau_3$ and $\tau_1/(\tau_2/\tau_3)$.

Theorem 6.2: For any $\mathcal{X}_r$ expressions τ and τ', $\tau \equiv \tau'$ iff $\tau \equiv_{\mathcal{I}_r} \tau'$. □

Example. Using $\mathcal{I}_r$, $\downarrow^*/\downarrow^* \ \equiv \ \downarrow^*$ can be verified as follows:

$\downarrow^*/\downarrow^*$	$\equiv_{\mathcal{I}_r} \downarrow^*/(\downarrow^* \cup \ \epsilon)$	by the *descendants* axiom
	$\equiv_{\mathcal{I}_r} \downarrow^*/\downarrow^* \cup \downarrow^*$	by *left-distribution*
	$\equiv_{\mathcal{I}_r} \downarrow^*$	by *descendants-union*

To prove Theorem 6.2, we define a containment relation. An XPath expression p_1 is *contained* in p_2 (written $p_1 \subseteq p_2$) iff for any XML tree T and any node n in T, $n[\![p_1]\!] \subseteq n[\![p_2]\!]$. If we want containment to hold only for trees over a fixed alphabet Σ_0, we write $p_1 \subseteq_{\Sigma_0} p_2$.

The following lemma shows that $\mathcal{I}_r$ also suffices to determine containment of $\mathcal{X}_r$ expressions.

$\epsilon/p \ \equiv \ p \ \equiv \ p/\epsilon$	(empty-path)
$\emptyset \ \cup \ p \ \equiv \ p$	(empty-set-union)
$p_1/\emptyset/p_2 \ \equiv \ \emptyset$	(empty-set-concat)
$l \ \cup \downarrow \ \equiv \ \downarrow$	(l-child)
$\downarrow^* \ \equiv \ \epsilon \ \cup \ \downarrow/\downarrow^*$	(descendants)
$p \ \cup \ \downarrow^* \ \equiv \ \downarrow^*$	(descendants-union)
$\downarrow/\downarrow^* \ \equiv \ \downarrow^*/\downarrow$	(child-descendants)
$p_1/(p_2/p_3) \ \equiv \ (p_1/p_2)/p_3$	(concat-associativity)
$p_1 \cup (p_2 \cup p_3) \ \equiv \ (p_1 \cup p_2) \cup p_3$	(union-associativity)
$p_1 \ \cup \ p_2 \ \equiv \ p_2 \ \cup \ p_1$	(union-commutativity)
$p \ \cup \ p \ \equiv \ p$	(union-idempotent)
$p/(p_1 \ \cup \ p_2) \ \equiv \ p/p_1 \ \cup \ p/p_2$	(left-distributivity)
$(p_1 \ \cup \ p_2)/p \ \equiv \ p_1/p \ \cup \ p_2/p$	(right-distributivity)

Table 1: $\mathcal{I}_r$**: axioms for** $\mathcal{X}_r$

Lemma 6.3: For any $\mathcal{X}_r$ expressions τ_1 and τ_2, $\tau_1 \subseteq \tau_2$ iff $\tau_2 \cup \tau_1 \equiv_{\mathcal{I}_r} \tau_2$. □

If this holds, then so does Theorem 6.2. Indeed, if $\tau_1 \equiv \tau_2$ then $\tau_1 \subseteq \tau_2$ and $\tau_2 \subseteq \tau_1$. Thus from Lemma 6.3 it follows that $\tau_2 \cup \tau_1 \equiv_{\mathcal{I}_r} \tau_2$ and $\tau_1 \cup \tau_2 \equiv_{\mathcal{I}_r} \tau_1$. Then by the *union-commutativity* axiom and the inference rules, $\tau_1 \equiv_{\mathcal{I}_r} \tau_2$. Conversely, if $\tau_1 \equiv_{\mathcal{I}_r} \tau_2$, then by the *union-idempotent* axiom and the inference rules, $\tau_1 \cup \tau_2 \equiv_{\mathcal{I}_r} \tau_2$ and $\tau_2 \cup \tau_1 \equiv_{\mathcal{I}_r} \tau_1$. Again from Lemma 6.3 it follows that $\tau_1 \subseteq \tau_2$ and $\tau_2 \subseteq \tau_1$; that is, $\tau_1 \equiv \tau_2$.

It is worth mentioning that Lemma 6.3 is also interesting in its own right, since it is common for XML queries to check containment of XPath expressions.

The proof of the lemma is given in the appendix. It should be mentioned that the proof can be made effective, thus providing an algorithm for testing containment, and hence for testing equivalence.

Fragment $\mathcal{X}$. We now consider $\mathcal{X}$. Let $\mathcal{I}$ be $\mathcal{I}_r$ excluding the *descendants*, *descendants-union* and *child-descendants* axioms. For a finite alphabet Σ_0, assume that it can be enumerated as $l_1, \ldots, l_n$. A finite axiom system $\mathcal{I}^f(\Sigma_0)$ consists of rules in $\mathcal{I}$ except for the l-child rules and with the addition of the following rule:

$$l_1 \ \cup \cdots \cup \ l_n \ \equiv \ \downarrow \qquad \text{(finite-alphabet)}$$

Note that for each $l \in \Sigma_0$, $l \cup \downarrow \ \equiv \ \downarrow$ can be derived from the *finite-alphabet* and *union-idempotent* axioms of $\mathcal{I}^f(\Sigma_0)$.

One can verify that $\mathcal{I}$ is an axiomatization of $\mathcal{X}$ under general equivalence (the default notion), and that for each finite Σ_0, $\mathcal{I}^f(\Sigma_0)$ is a finite axiomatization of $\mathcal{X}$ for equivalence over Σ_0 (see the appendix for a proof).

Theorem 6.4: For any $\mathcal{X}$ expressions τ_1 and τ_2,

- $\tau_1 \equiv \tau_2$ iff $\tau_1 \equiv_{\mathcal{I}} \tau_2$, and $\tau_1 \subseteq \tau_2$ iff $\tau_2 \cup \tau_1 \equiv_{\mathcal{I}} \tau_2$;
- For each finite Σ_0, $\tau_1 \equiv_{\Sigma_0} \tau_2$ iff $\tau_1 \equiv_{\mathcal{I}^f(\Sigma_0)} \tau_2$, and $\tau_1 \subseteq_{\Sigma_0} \tau_2$ iff $\tau_2 \cup \tau_1 \equiv_{\mathcal{I}^f(\Sigma_0)} \tau_2$. □

6.3 Normal forms for $\mathcal{X}_r$ and $\mathcal{X}$

We study here normal forms for $\mathcal{X}_r$ and $\mathcal{X}$.

A fragment F of XPath has a *normal form* if there is a function λ from F to a subset F_{nl} of F such that for any expressions τ_1 and τ_2 of F, $\tau_1 \equiv \tau_2$ iff $\lambda(\tau_1) = \lambda(\tau_2)$, i.e., $\lambda(\tau_1)$ and $\lambda(\tau_2)$ are syntactically equal to each other. We shall say that F_{nl} is a *normal form* of F. Observe that λ is determined by F_{nl}. For an expression τ in F, we refer to $\lambda(\tau)$ as the *normal form* of τ w.r.t. F_{nl}, or simply the normal form if F_{nl} is clear from the context. Similarly, we can say that F has a normal form for equivalence over a finite alphabet Σ_0.

A normal form is useful for simplifying the analysis of equivalence of XPath expressions since it allows one to reduce semantic equivalence to syntactic equality.

Fragment $\mathcal{X}_r$. An $\mathcal{X}_r$ expression is said to be *union-free* if it does not contain union '$\cup$'. Below we give a normal form for union-free $\mathcal{X}_r$ expressions. Normal forms for $\mathcal{X}_r$ expressions with unions are still under investigation.

A union-free $\mathcal{X}_r$ expression α is in the *normal form* if (1) it does not contain ϵ (resp. $\emptyset$) unless $\alpha = \epsilon$ (resp. $\alpha = \emptyset$); (2) any contiguous sequence in α consisting of $\downarrow$'s and at least one $\downarrow^*$ is of the form $\downarrow^*/\downarrow/\downarrow^*/\cdots/\downarrow/\downarrow^*$ (with the same number of occurrences of $\downarrow$); and (3) α does not contain consecutive $\downarrow^*/\downarrow^*$. The next proposition shows that the set of union-free $\mathcal{X}_r$ expressions of this form is indeed a normal form for all union-free $\mathcal{X}_r$ expressions (see the appendix for a proof).

Proposition 6.5: For any union-free $\mathcal{X}_r$ expression α there exists a unique $\mathcal{X}_r$ expression α' such that $\alpha \equiv \alpha'$ and α' is in the normal form above. Furthermore, in the case of a finite alphabet Σ_0, for any union-free α there exists such a unique α' such that $\alpha \equiv_{\Sigma_0} \alpha'$. □

Fragment $\mathcal{X}$. We next define a normal form for general $\mathcal{X}$ expressions. An $\mathcal{X}$ expression τ is in the *union normal form* if it is of the form $\alpha_1 \cup \cdots \cup \alpha_k$, where for each α_i, referred to as a *disjunct* of τ, (1) α_i does not contain ϵ unless $\alpha_i = \epsilon$, and τ does not contain $\emptyset$ unless $\tau = \emptyset$; (2) α_i is a *union-free* expression; and (3) α_i is not contained in any other α_j, i.e., $\alpha_i \not\subseteq \alpha_j$ for all $j \neq i$.

When considering a fixed finite alphabet $\Sigma_0 = \{l_1, \ldots, l_n\}$, we add the condition (4) τ does not contain $\downarrow$, which is represented instead by $l_1 \cup \cdots \cup l_n$.

The following then holds (see the appendix for a proof):

Proposition 6.6: For any $\mathcal{X}$ expression τ there is an $\mathcal{X}$ expression τ' such that $\tau \equiv \tau'$ and τ' is in the union normal form satisfying conditions (1)–(3) given above. For every finite alphabet Σ_0 there is $\tau' \equiv_{\Sigma_0} \tau$ with τ' satisfying (1)–(4). Moreover, in both cases τ' is unique up to ordering of the disjuncts. □

7 Conclusion

We have investigated important structural properties of a variety of XPath fragments, parameterized with modalities that include qualifiers, upward traversal (the parent and ancestor axes), and recursion (descendant and ancestor). First, we have provided characterizations of the fragments with qualifiers in terms of both logics and tree patterns (Fig. 3). Second, we have given a complete picture of the closure properties of all these fragments (Fig. 4), under both general equivalence and root equivalence. Finally, we have established preliminary results on axiom systems and normal forms for some of these fragments. These results not only advance our understanding of different XPath modalities, but are also useful for simplification of XPath expressions and optimization of XML queries.

One open problem is the finite axiomatizability of $\mathcal{X}_r$ for $\equiv_{\Sigma_0}$, and similarly for the other fragments. It is well known that regular expressions do not have a finite axiom system when the inference rules are restricted to the ones given in Section 6, even when the alphabet consists of a single symbol [21]. Although $\mathcal{X}_r$ is a large class of regular expressions, we speculate that $\equiv_{\Sigma_0}$ is finitely axiomatizable for $\mathcal{X}_r$. We are currently investigating this issue for $\mathcal{X}_r$ terms, which may contain free variables. Another topic for future work is to study the expressiveness and closure properties of other fragments, especially those allowing negations in qualifiers. The third topic is to strengthen our logical characterizations in two directions. The logical characterizations

characterization	$\mathcal{X}_{[\,]}$	$\mathcal{X}^{\uparrow}_{[\,]}$	$\mathcal{X}_{r,[\,]}$	$\mathcal{X}^{\uparrow}_{r,[\,]}$
logic	$\exists^{+}(child)[loc, down](c,s)$	$\exists^{+}(child)[loc](c,s)$	$\exists^{+}(child, desc)[down](c,s)$	$\exists^{+}(child, desc)(c,s)$
tree patterns	$\bigcup \mathrm{TP}(child)[down]$	$\bigcup \mathrm{TP}(child)$	$\bigcup \mathrm{TP}(child, desc)[down]$	$\bigcup \mathrm{TP}(child, desc)$

Figure 3: Characterizations of the expressiveness of the fragments

closure properties		$\mathcal{X}$	$\mathcal{X}_{[\,]}$	$\mathcal{X}^{\uparrow}$	$\mathcal{X}_r$	$\mathcal{X}^{\uparrow}_{[\,]}$	$\mathcal{X}_{r,[\,]}$	$\mathcal{X}^{\uparrow}_r$	$\mathcal{X}^{\uparrow}_{r,[\,]}$
intersection	under $\equiv$	Yes	Yes	No	Yes	Yes	Yes	No	Yes
	under $\equiv_r$	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
complementation	under $\equiv$	No	No	No	No	No	No	No	No
	under $\equiv_r$	No	No	No	No	No	No	No	No

Figure 4: The closure properties of the fragments

given here can be extended to handle the case where trees have *data values*. In addition, the characterizations can be made more precise by mapping into positive existential logic with two variables. The fourth topic is to reinvestigate these issues in the presence of DTDs/XML Schema and integrity constraints, which commonly coexist with XPath expressions in practice. Finally, we are also exploring the use of our results in developing rewrite systems and algorithms for simplifying XPath expressions.

References

[1] S. Amer-Yahia, S. Cho, V. Lakshmaman, and D. Srivistava. Minimization of tree pattern queries. In *SIGMOD*, 2001.

[2] A. Berglund et al. *XML Path Language (XPath) 2.0.* W3C Working Draft, Dec. 2001. `http://www.w3.org/TR/xpath20`.

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

[4] D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi. Reasoning in expressive description logics. In *Handbook of Automated Reasoning*, pages 1581–1634. Elsevier, 2001.

[5] D. Chamberlin et al. XQuery 1.0: An XML query language. W3C Working Draft, June 2001. `http://www.w3.org/TR/xquery`.

[6] C. Chan, P. Felber, M. Garofalakis, and R. Rastogu. Efficient filtering of XML documents with XPath expressions. In *ICDE*, 2002.

[7] J. Clark. *XSL Transformations (XSLT).* W3C Recommendation, Nov. 1999. `http://www.w3.org/TR/xslt`.

[8] J. Clark and S. DeRose. *XML Path Language (XPath).* W3C Recommendation, Nov. 1999. `http://www.w3.org/TR/xpath`.

[9] A. Deutsch and V. Tannen. Containment for classes of XPath expressions under integrity constraints. In *KRDB*, 2001.

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

[11] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic.* The MIT Press, 2000.

[12] C. Hoffmann and M. O'Donnell. Pattern matching in trees. *JACM*, 29(1):68–95, 1982.

[13] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison Wesley, 1979.

[14] P. Kilpelainen and H. Manilla. Ordered and unordered tree inclusion. *SIAM J. Comput.*, 24(2):340–356, 1995.

[15] G. Miklau and D. Suciu. Containment and equivalence of XPath expressions. In *PODS*, 2002.

[16] T. Milo, D. Suciu, and V. Vianu. Typechecking for XML transformers. In *PODS*, 2001.

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

[18] F. Neven and T. Schwentick. Expressive and efficient languages for tree-structured data. In *PODS*, 2000.

[19] D. Olteanu, H. Meuss, T. Furche, and F. Bry. XPath: Looking forward. In *Workshop on XML-Based Data Management (XMLDM)*, 2002.

[20] P. Ramanan. Efficient algorithms for minimizing tree pattern queries. In *SIGMOD*, 2002.

[21] V. Redko. On defining relations for the algebra of regular events. *Ukrainskii Matematicheskii Zhurnal*, 16:120–126, 1964. (Russian).

[22] J. Robie, J. Lapp, and D. Schach. *XML Query Language (XQL).* Workshop on XML Query Languages, Dec. 1998.

[23] H. Thompson et al. XML Schema. W3C Working Draft, May 2001. `http://www.w3.org/XML/Schema`.

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

[25] P. Wood. On the equivalence of XML patterns. In *Computational Logic 2000*, 2000.

[26] P. Wood. Minimising simple XPath expressions. In *WebDB*, 2001.

Appendix

Proof of Proposition 2.1

Obviously, if there is an edge from F_1 to F_2 in Fig. 1 then from the syntactic definition of these fragments it follows that $F_1 \subseteq F_2$. For the other direction, it suffices to show the following.

Claim 1: None of $\mathcal{X}_{[\,]}$, $\mathcal{X}^{\uparrow}$ and $\mathcal{X}_r$ is contained in any of the other two fragments.

Consider (1) $\epsilon[\text{label} = A]$ in $\mathcal{X}_{[\,]}$, (2) $\uparrow$ in $\mathcal{X}^{\uparrow}$, and (3) $\downarrow^*$ in $\mathcal{X}_r$. It is easy to see the following: expression (1) witnesses that $\mathcal{X}_{[\,]} \not\subseteq \mathcal{X}^{\uparrow}$ and $\mathcal{X}_{[\,]} \not\subseteq \mathcal{X}_r$, expression (2) witnesses $\mathcal{X}^{\uparrow} \not\subseteq \mathcal{X}_{[\,]}$ and $\mathcal{X}^{\uparrow} \not\subseteq \mathcal{X}_r$, and expression (3) witnesses $\mathcal{X}_r \not\subseteq \mathcal{X}_{[\,]}$ and $\mathcal{X}_r \not\subseteq \mathcal{X}^{\uparrow}$.

As a result, this also shows that none of $\mathcal{X}_{[\,]}$, $\mathcal{X}^{\uparrow}$ and $\mathcal{X}_r$ is contained in $\mathcal{X}$.

Claim 2: None of $\mathcal{X}_{r,[\,]}$, $\mathcal{X}^{\uparrow}_{[\,]}$ and $\mathcal{X}^{\uparrow}_r$ is contained in the any of the other two fragments.

Consider (1) $\mathcal{X}_{r,[\,]}$ expression $\downarrow^*/A[\downarrow^*/B]$, (2) $\mathcal{X}^{\uparrow}_{[\,]}$ expression $\uparrow[\text{label} = A]$, and (3) $\mathcal{X}^{\uparrow}_r$ expression $\downarrow^*/B/\uparrow^*$. It is not difficult to verify the following: expression (1) witnesses that $\mathcal{X}_{r,[\,]} \not\subseteq \mathcal{X}^{\uparrow}_{[\,]}$ and $\mathcal{X}_{r,[\,]} \not\subseteq \mathcal{X}^{\uparrow}_r$, expression (2) witnesses $\mathcal{X}^{\uparrow}_{[\,]} \not\subseteq \mathcal{X}_{r,[\,]}$ and $\mathcal{X}^{\uparrow}_{[\,]} \not\subseteq \mathcal{X}^{\uparrow}_r$, and expression (3) witnesses $\mathcal{X}^{\uparrow}_r \not\subseteq \mathcal{X}_{r,[\,]}$ and $\mathcal{X}^{\uparrow}_r \not\subseteq \mathcal{X}^{\uparrow}_{[\,]}$. For example, the proof of Theorem 5.1 gives a formal argument for that $\downarrow^*/A[\downarrow^*/B]$ is not equivalent to any expression in $\mathcal{X}^{\uparrow}_r$.

As a result, this also shows that none of $\mathcal{X}_{r,[\,]}$, $\mathcal{X}^{\uparrow}_{[\,]}$, and $\mathcal{X}^{\uparrow}_r$ is contained in any of $\mathcal{X}_{[\,]}$, $\mathcal{X}^{\uparrow}$, and $\mathcal{X}_r$.

Claim 3: $\mathcal{X}^{\uparrow}_{r,[\,]}$ is not contained in any of $\mathcal{X}_{r,[\,]}$, $\mathcal{X}^{\uparrow}_{[\,]}$, and $\mathcal{X}^{\uparrow}_r$.

It is not difficult to show that $\epsilon[\uparrow^*/A]/\downarrow^*$ is not equivalent to any expression in any of $\mathcal{X}_{r,[\,]}$, $\mathcal{X}^{\uparrow}_{[\,]}$ and $\mathcal{X}^{\uparrow}_r$.

Proof of Proposition 3.1

Obviously, every $\mathcal{X}^{\uparrow}_{r,[\,]}$ expression is in $\mathcal{X}^{\uparrow}_{r,[\wedge,\vee]}$. Thus it suffices to prove that every $\mathcal{X}^{\uparrow}_{r,[\wedge,\vee]}$ expression $p[q]$ is equivalent to an $\mathcal{X}^{\uparrow}_{r,[\,]}$ expression. This can done by rewriting $p[q]$ to an equivalent expression that contains neither '$\wedge$' nor '$\vee$', based on the rules $p\,[q_1 \wedge q_2] \Rightarrow p\,[q_1]\,/\epsilon\,[q_2]$ and $p\,[q_1 \vee q_2] \Rightarrow p\,[q_1] \cup p\,[q_2]$. Each such step rewrites an expression to an equivalent form, and one can then convert $p\,[q]$ to an equivalent $\mathcal{X}^{\uparrow}_{r,[\,]}$ expression in a finite number of steps. $\square$

Proof of Theorem 3.2

The containments $\bigcup\mathrm{TP}(\mathit{child}, \mathit{desc}) \subseteq \mathcal{X}^{\uparrow}_{r,[\,]} \subseteq \exists^+(\mathit{child}, \mathit{desc})(c, s)$ were given in the body of the paper. We now give the proof of $\exists^+(\mathit{child}, \mathit{desc})(c, s) \subseteq \bigcup\mathrm{TP}(\mathit{child}, \mathit{desc})$.

As mentioned in the body of the paper, we prove the more general fact that an arbitrary formula $\phi(c, s_1, \ldots, s_n)$ in $\exists^+(\mathit{child}, \mathit{desc})(c, s)$ is equivalent to a tree pattern query. Let ϕ be a formula, and write ϕ as $\exists x_1 \ldots x_n \gamma$ where γ is quantifier-free. Turn γ into disjunctive normal form, and move the disjunction outside of the existential quantification. Since the set of tree pattern queries is clearly closed under unions, it suffices to show that the result holds for ϕ of the form $\exists x_1 \ldots x_n \gamma(\vec{x}, c, \vec{s})$, where γ is a conjunction of statements of the form $\mathit{desc}(v_1, v_2)$, $\mathit{child}(v_1, v_2)$ and $P(v)$ (with '=' eliminated via variable substitution). Let V be the set of variables in γ and consider the structure $L(\gamma) = (V, E, l, c, s_1, \ldots, s_n)$, with constants for the variables c, $\vec{s}$ of γ, where $E(v_1, v_2)$ iff $\mathit{child}(v_1, v_2)$ or $\mathit{desc}(v_1, v_2)$ is a conjunct of γ, and l labels an edge with *child* if the first case holds and with *desc* if the second case holds and the first does not. By adding extra quantifiers, we can assume that any two variables v_1 and v_2 have a least upper bound in the relation E, and we can also assume that there are no edge relations $E(v_1, v_2)$ derivable from other edge relations by transitivity (by removing any such redundant clauses), and that l labels every node with exactly one proposition (or with '$*$', if no appropriate formula $P(v)$ is present).

It should be remarked that for an arbitrary structure $L = (V, E, l, c, s_1, \ldots, s_n)$ we can apply the semantics for tree patterns and talk about an equivalent formula $F(L)$ in $\exists^+(\mathit{child}, \mathit{desc})(c, s)$. Clearly, if (V, E) forms a tree, then $F(L)$ is equivalent to a tree pattern; in what follows we will modify the structure $L = L(\gamma)$ to get an equivalent structure in which (V, E) is a tree.

We first show that we can take (V, E) to be acyclic. First, if there is any cycle in (V, E) that contains an edge labeled *child*, then γ is equivalent to the tree pattern query *false*, and similarly if there are two nodes in the same strongly-connected component of (V, E) with different node labellings. Otherwise, we take the quotient of $L(\gamma)$ by the equivalence relation "being in the same strongly connected component", and obtain an acyclic graph. We can check that the new structure L' has $F(L')$ equivalent to $F(L(\gamma))$.

If (V, E) has the property that between the root of V and any other node there is exactly one directed E-path, then $L(\gamma)$ is a tree pattern and we are done. For an acyclic labeled partial order L let $p(L)$ be the number of paths from the root to a leaf. We will give a function DC from labeled acyclic partial orders such that $F(L)$ is equivalent to the union of $F(L_i)$ for $L_i \in DC(L)$ and such that whenever L is not a tree $p(L_i) < p(L)$ for every $L_i \in DC(L)$. Iterating the function DC until each newly obtained partial order is a tree gives us a collection of edge-and-node labeled trees L_i such the $F(L(\gamma))$ is equivalent to the disjunction $F(L)$.

We define DC on $L = (V, E, l)$ as follows: take any v_1, v_2 such that there are two disjoint (except for end-points) paths from v_1 to v_2, and choose any two such paths p_1 and p_2, where the p_i's do not include v_1 and v_2 (and hence are completely disjoint). Each path p_i can be coded by a string $\mathit{code}(p_i)$ consisting of nodes alternating with either *child* or *desc*. We call any string w consisting of alternating nodes of V and elements of $\{\mathit{child}, \mathit{desc}\}$ a *code*, and for any code let $\mathit{Nodes}(w)$ be the nodes appearing in w.

Choose an *interleaving* of p_1 and p_2; that is, a code w, an order-preserving f_1 mapping $\mathit{Nodes}(\mathit{code}(p_1))$ bijectively into $\mathit{Nodes}(w)$, and order-preserving f_2 mapping $\mathit{Nodes}(\mathit{code}(p_2))$ bijectively into w such that:

- $\mathit{Nodes}(w)$ are exactly the union of the ranges of f_1 and f_2,
- whenever a sequence (v, child, v') appears as a (contiguous) subword of p_i, then $(f(v), \mathit{child}, f(v'))$ appears as a subword of w.

Given any interleaving, there is a corresponding structure formed by replacing $\mathit{range}(p_1) \cup \mathit{range}(p_2)$ with w and connecting nodes that were connected to a node n in some p_i with the image $f_i(n)$ in w, and connecting nodes within w according to the edges coded in w.

One can check that for every such L' formed from this process, $F(L')$ implies $F(L)$. Furthermore, one can construct a surjection of the root-to-leaf paths in L into the root-to-leaf paths of L' that is not 1–1, hence showing $p(L') < p(L)$.

This completes the proof of the inclusion and hence the theorem. □

Proof of Theorem 3.3

The proof that $\mathcal{X}^{\uparrow}_{[\,]}$ is contained in $\exists^+(\mathit{child})[\mathit{loc}](c, s)$ is straightforward, and the proof that the set $\bigcup\mathrm{TP}(\mathit{child})$ of unary child tree patterns is contained in $\mathcal{X}^{\uparrow}_{[\,]}$ follows from the same construction as in Theorem 3.6. It remains to show that $\exists^+(\mathit{child})[\mathit{loc}](c, s)$ is contained in $\bigcup\mathrm{TP}(\mathit{child})$. We go through the same argument as in Theorem 3.6. The only difference is in the step justifying that for every two variables, there is some upper bound in the lattice $L(\gamma)$: in Theorem 3.6 this was made true by adding extra quantifications, but in this case it is guaranteed by the fact that all variables are required to be reachable from c in a fixed number of steps. □

Proof of Theorem 3.6

We can show $\mathcal{X}_{r,[\,]} \subseteq \exists^+(\mathit{child}, \mathit{desc})[\mathit{down}](c, s)$ by induction, using the inductive semantics for $\mathcal{X}_{r,[\,]}$ in terms of logic; for example, the inductive translation of $p/{\downarrow^*}$ is $\exists s'\, \phi_p(c, s') \wedge \mathit{desc}(s', s)$ where ϕ_p is the coding of the path p, and by applying the induction hypothesis to ϕ_p and the transitivity of desc, we see that the result is in $\exists^+(\mathit{child}, \mathit{desc})[\mathit{down}](c, s)$.

To see $\exists^+(\mathit{child}, \mathit{desc})[\mathit{down}](c, s) \subseteq \bigcup\mathrm{TP}(\mathit{child}, \mathit{desc})[\mathit{down}]$, inspect the proof of $\exists^+(\mathit{child}, \mathit{desc})(c, s) \subseteq \bigcup\mathrm{TP}(\mathit{child}, \mathit{desc})$ of Theorem 3.2 and note that in this case the variable c will be an upper bound on all variables, and hence the translation there will put c at the root of the tree.

Finally, to see $\bigcup\mathrm{TP}(\mathit{child}, \mathit{desc})[\mathit{down}] \subseteq \mathcal{X}_{r,[\,]}$, inspect the translation given for $\bigcup\mathrm{TP}(\mathit{child}, \mathit{desc}) \subseteq \mathcal{X}^{\uparrow}_{r,[\,]}$ of Theorem 3.2 and note that if c is at the root, it produces no occurrences of $\uparrow$ or $\uparrow^*$. $\square$

Proof of Theorem 3.7

Again the direction $\mathcal{X}_{[\,]} \subseteq \exists^+(\mathit{child})[\mathit{loc}, \mathit{down}](c, s)$ can be seen by inspecting the translation. The direction $\exists^+(\mathit{child})[\mathit{loc}, \mathit{down}](c, s) \subseteq \bigcup\mathrm{TP}(\mathit{child})[\mathit{down}]$ follows because the original translation from $\exists^+(\mathit{child}, \mathit{desc})(c, s)$ to $\bigcup\mathrm{TP}(\mathit{child}, \mathit{desc})$ has already been shown to produce both a $\bigcup\mathrm{TP}(\mathit{child})$ query in the case of a formula in $\exists^+(\mathit{child})[\mathit{loc}](c, s)$, and a $\bigcup\mathrm{TP}(\mathit{child}, \mathit{desc})[\mathit{down}]$ query in the case of a formula in $\exists^+(\mathit{child}, \mathit{desc})[\mathit{down}](c, s)$; as a result it must produce a query in the intersection of tree pattern queries $\bigcup\mathrm{TP}(\mathit{child}) \cap \bigcup\mathrm{TP}(\mathit{child}, \mathit{desc})[\mathit{down}]$ for a formula in $\exists^+(\mathit{child})[\mathit{loc}, \mathit{down}](c, s)$. For the last direction, the translation from $\bigcup\mathrm{TP}(\mathit{child})[\mathit{down}]$ to $\mathcal{X}_{[\,]}$ again uses the translation in the proof of $\bigcup\mathrm{TP}(\mathit{child}, \mathit{desc}) \subseteq \mathcal{X}^{\uparrow}_{r,[\,]}$, and for $\bigcup\mathrm{TP}(\mathit{child})[\mathit{down}]$ queries this will produce neither upward modalities nor descendant axes. $\square$

Proof of Theorem 5.1

We show the following claims. The proofs of the other parts were given in the body of the paper.

1. $\mathcal{X}_{r,[\,]} \not\subseteq \mathcal{X}^{\uparrow}_r$.

To see that the containment is proper, consider the expression ${\downarrow^*}/A[{\downarrow^*}/B]$. We claim that this $\mathcal{X}_{r,[\,]}$ expression is not root-equivalent to any $\mathcal{X}^{\uparrow}_r$ expression. To see this, assume that p is such an $\mathcal{X}^{\uparrow}_r$-expression. Let N be the size of p, and consider the following structure T: T has root rt, with child n_1, which in turn has two children n_2 and n_3, both labeled 'A'. The nodes n_2 and n_3 have beneath them a chain of length N of nodes, all labeled 'C', with the exception of the node n_4 at the end of the chain below n_2, which is labeled 'B'.

Clearly, $n_2 \in rt[\![{\downarrow^*}/A[{\downarrow^*}/B]]\!]$, and hence $n_2 \in rt[\![p]\!]$. As in the proof in Theorem 4.1, we can assume that p is a simple path. Consider the sequence of nodes in an accepting path $rt, v_1, \ldots, v_k, n_2$. If n_4 is not among these nodes, it is easy to see that $n_3 \in rt[\![p]\!]$, a contradiction. Let v_i be the last occurrence of n_4 on this path. We distinguish two cases:

1. n_1 is among $v_{i+1}, \ldots, v_k$. But then, by following the other path in the tree, and using the fact that the path does not reach a leaf again, we can see once more that $n_3 \in rt[\![p]\!]$, a contradiction.
2. n_1 is not among $v_{i+1}, \ldots, v_k$. Since the length of the chain of nodes above n_4 is greater than the number of symbols in p, it follows that one of the symbols in p is $\uparrow^*$ and this must correspond with at least one pair (v_j, v_{j+1}), $(j \geq i)$, where v_{j+1} is an ancestor, but not a direct parent, of v_j. In this case, we replace v_j, v_{j+1}, $\ldots$, v_k by their respective children, which shows that $n' \in r[\![p]\!]$, where n' is the child of n_2. Since n' is labeled 'C', this is a contradiction.

2. $\mathcal{X}^{\uparrow} = \mathcal{X}_{[\,]}$ in the absence of label tests.

We show that without label tests in qualifiers, each $\mathcal{X}_{[\,]}$ expression can be rewritten to a root-equivalent $\mathcal{X}_r^{\uparrow}$ expression. Note that $\eta[\epsilon] \equiv_r \eta$; thus we can assume that every $\mathcal{X}_{[\,]}$ expression is of the form $p_1 \cup \cdots \cup p_k$ with each p_i having the form $\eta_1[q_1]/\cdots/\eta_m[q_m]$, where η_j is one of $\emptyset$, ϵ, l, or $\downarrow$. Starting from the innermost qualifiers, we eliminate the qualifiers using the rewriting rules:

$$\begin{aligned} \eta[\epsilon] &\Rightarrow \eta\,; \\ \eta[\emptyset] &\Rightarrow \emptyset\,; \\ \eta[l/q] &\Rightarrow \eta/l[q]/\uparrow; \\ \eta[\downarrow/q] &\Rightarrow \eta/\downarrow[q]/\uparrow; \\ \eta[p_1 \cup p_2] &\Rightarrow \eta[p_1] \cup \eta[p_2]\,. \end{aligned}$$

Here η is any symbol and the rules can be applied at any nested depth of qualifiers. A straightforward induction on the number and the size of qualifiers shows that with these rules one can convert an $\mathcal{X}_{[\,]}$ expression into a root-equivalent $\mathcal{X}_r^{\uparrow}$ expression in a finite number of steps. Thus $\mathcal{X}^{\uparrow} = \mathcal{X}_{[\,]}$ in the absence of label tests in qualifiers.

Proof of Corollary 5.3

First observe that if a fragment is closed under intersection under general equivalence, it remains closed under root equivalence. Thus from Theorem 4.1 it follows that under root equivalence, all of the fragments except $\mathcal{X}^{\uparrow}$ and $\mathcal{X}_r^{\uparrow}$ are closed under intersection.

We now show that under root equivalence, $\mathcal{X}^{\uparrow}$ is closed under intersection. Consider arbitrary $\mathcal{X}^{\uparrow}$ expressions p_1 and p_2. Theorem 5.1 tells us that under root equivalence, there exist two $\mathcal{X}_{[\,]}$ expressions p_1' and p_2' equivalent to p_1 and p_2 respectively, such that p_i' has the form $p_{i,1} \cup \cdots \cup p_{i,k_i}$ and $p_{i,j}$ has the form $\eta_1^{i,j}[q_1^{i,j}]/\cdots/\eta_m^{i,j}[q_m^{i,j}]$, where $\eta_s^{i,j}$ is not ϵ unless $p_{i,j}$ is itself ϵ, and $q_s^{i,j}$ does not contain label tests. The intersection of p_1' and p_2' is a union of $\mathcal{X}_{[\,]}$ expressions of the form $\eta_1[q_1]/\cdots/\eta_m[q_m]$, one for each pair

$$\begin{aligned} p_{1,s} &= \eta_1^{1,s}[q_1^{1,s}]/\cdots/\eta_m^{1,s}[q_m^{1,s}] \\ p_{2,s'} &= \eta_1^{2,s'}[q_1^{2,s'}]/\cdots/\eta_m^{2,s'}[q_m^{2,s'}] \end{aligned}$$

such that for $j \in [1,m]$, $\eta_j^{1,s}$ (resp. $\eta_j^{2,s'}$) is not ϵ unless for all $j' > j$, $\eta_j^{1,s} = \epsilon$ (resp. $\eta_j^{2,s'} = \epsilon$; this is a reasonable assumption when conjunction $\wedge$ is allowed in qualifiers), $q_j = q_j^{1,s} \wedge q_j^{2,s'}$, and $\eta_j = \min(\eta^{1,s}, \eta^{2,s'})$. Here $\min(\eta, \eta')$ is defined as follows: (1) it is $\emptyset$ if either one of η or η' is $\emptyset$, or if η and η' are both different labels, or if one of them is ϵ whereas the other is not; (2) it is ϵ if both η and η' are ϵ; (3) it is l if one of η or η' is the label l and the other is $\downarrow$; and (4) it is η if $\eta = \eta'$. It is easy to verify that the union of all such expressions is indeed the intersection of p_1 and p_2. Observe that the union is an $\mathcal{X}_{[\,]}$ expression containing no label tests. Theorem 5.1 then implies that this expression can be expressed in $\mathcal{X}^{\uparrow}$ under root equivalence. In other words, the conjunction of p_1 and p_2 is expressible in $\mathcal{X}^{\uparrow}$.

However, $\mathcal{X}_r^{\uparrow}$ is not closed under intersection. To see this, consider the $\mathcal{X}_r^{\uparrow}$ expressions $\downarrow^*/A$ and $\downarrow^*/B/\uparrow^*$. The first expression selects those nodes in the tree labeled 'A', while the second selects those that have a descendant labeled 'B'. The intersection is therefore precisely $\downarrow^*/A[\downarrow^*/B]$, which has already been shown (Theorem 5.1, part 1) to be inexpressible in $\mathcal{X}_r^{\uparrow}$.

For complementation, the arguments used in Theorem 4.2 still apply under root equivalence. $\square$

Proof of Lemma 6.3

The soundness of $\mathcal{I}_r$ can be verified by induction on the length of $\mathcal{I}_r$-proofs. We next show the completeness of $\mathcal{I}_r$: for any $\mathcal{X}_r$ expressions τ and τ', if $\tau \subseteq \tau'$ then $\tau \cup \tau' \equiv_{\mathcal{I}_r} \tau'$.

We begin with the proof by introducing some notations. Observe that using the union-distributivity axioms of $\mathcal{I}_r$, one can convert an $\mathcal{X}_r$ expression τ to a *union form:* $\alpha_1 \cup \cdots \cup \alpha_k$, such that each α_i does not contain the union operator '$\cup$', referred to as a *union-free* expression. Furthermore, using $\mathcal{I}_r$ axioms one can rewrite each union-free expression α such that (1) it does not contain ϵ (resp. $\emptyset$) unless $\alpha = \epsilon$ (resp. $\alpha = \emptyset$), (2) any contiguous sequence in α consisting of $\downarrow$'s and at least one $\downarrow^*$ is of the form $\downarrow^*/\downarrow/\downarrow^*/\cdots/\downarrow/\downarrow^*$ (with the same number of $\downarrow$ occurrences); and (3) α does not contain consecutive $\downarrow^*/\downarrow^*$. Indeed, the empty-set and empty-path axioms assert condition (1), the *child-descendants* and *descendants* axioms assert condition (2), and the example given in Section 6 shows how to convert $\downarrow^*/\downarrow^*$ to $\downarrow^*$, so as to meet condition (3). Thus w.l.o.g. we assume that any union-free expression satisfies these conditions. For a union-free expression α, we define its *length*, $|\alpha|$, to be 0 if α is $\emptyset$ or ϵ, and $|\beta| + 1$ if it is of the form η/β, where η is one of label l, $\downarrow$, or $\downarrow^*$.

To prove the lemma, it suffices to show

Claim: For any union-free α and $\mathcal{X}_r$ expression τ, if $\alpha \subseteq \tau$ then $\alpha \cup \tau \equiv_{\mathcal{I}_r} \tau$.

For if this holds, then the *union-idempotent* axiom and the inference rules yield $\tau \cup \tau' \equiv_{\mathcal{I}_r} \tau'$.

We next verify this claim. Assume that τ is in the union form as $\alpha_1 \cup \cdots \cup \alpha_m$. We prove the claim by induction on the length of α.

Induction basis: If α is $\emptyset$ then the claim obviously holds by the *empty-set-union* axiom. If α is ϵ then by conditions (1) and (3) given above, it is easy to show by contradiction that there must be α' in the union-form of τ that is either ϵ or $\downarrow^*$; thus the *union-idempotent* and *union-descendants* axioms will prove this case. Hence the claim holds when $|\alpha| = 0$.

Inductive step: Assume the claim for $\alpha = \beta$. We need only show the claim for $\alpha = \eta/\beta$, where η is one of l, $\downarrow$, or $\downarrow^*$ because of condition (1) above. We consider the following cases of η.

Case 1. η is a label l.

Let S be the set of all union-free expressions in the union form of τ. Consider the following sets obtained from S:

$$\begin{aligned} S_{*,1} &= \{\beta' \mid \downarrow^*/\beta' \in S\}, \\ S_{*,2} &= \{\downarrow^*/\beta' \mid \downarrow^*/\beta' \in S\}, \\ S_d &= \{\beta' \mid \downarrow/\beta' \in (S \cup S_{*,1})\}, \\ S_l &= \{\beta' \mid l/\beta' \in (S \cup S_{*,1})\}, \end{aligned}$$

and define the $\mathcal{X}_r$ expression:

$$\widehat{\tau} = \bigcup S_l \ \cup \ \bigcup S_d \ \cup \ \bigcup S_{*,2}.$$

To show the claim for this case, it suffices to prove:

Subclaim 1: $\beta \subseteq \widehat{\tau}$.

Subclaim 2: $l/\widehat{\tau} \cup \tau \equiv_{\mathcal{I}_r} \tau$.

For if these hold, then by the induction hypotheses and Subclaim 1, $\beta \cup \widehat{\tau} \equiv_{\mathcal{I}_r} \widehat{\tau}$. Thus by the *left-distributivity* axiom we have $l/\beta \ \cup \ l/\widehat{\tau} \equiv_{\mathcal{I}_r} l/\widehat{\tau}$; that is, $\alpha \ \cup \ l/\widehat{\tau} \equiv_{\mathcal{I}_r} l/\widehat{\tau}$. From this and Subclaim 2 one obtains $\alpha \cup \tau \equiv_{\mathcal{I}_r} \tau$ using the *union-idempotent* axiom and the inference rules.

Subclaim 1 can be verified as follows. Suppose by contradiction $\beta \not\subseteq \widehat{\tau}$. Then there exists a *simple path* ρ, i.e., a sequence of labels, such that $\rho \subseteq \beta$ but $\rho \not\subseteq \widehat{\tau}$. We show that this leads to a contradiction. Since $\alpha \subseteq \tau$, we have $l/\rho \subseteq \tau$. Then there must be a union-free expression α' of τ such that $l/\rho \subseteq \alpha'$. Obviously if α' is $\emptyset$ or ϵ then $l/\rho \not\subseteq \alpha'$. Now assume that $\alpha' = \eta'/\beta'$ where η' is one of the following: a label, $\downarrow$ or $\downarrow^*$. If η' is a different label l' then $l/\rho \not\subseteq \alpha'$. If η' is the same label l, then β' is in S_l; if it is $\downarrow$, then β' is in S_d. If it is $\downarrow^*$, since $\downarrow^* \equiv \epsilon \cup \downarrow/\downarrow^*$, we must have either $\rho \subseteq \beta'$ or $\rho \subseteq \downarrow^*/\beta'$. For the former case, by the condition (3) β' cannot start with $\downarrow^*$; hence it must be in $S_{*,1}$ and thus in S_d and S_l; for the latter case, $\downarrow^*/\beta'$ is in $S_{*,2}$. These contradict the assumption that $\rho \not\subseteq \widehat{\tau}$. Thus Subclaim 1 holds.

We next show Subclaim 2. Observe the following: First, $\downarrow/\beta' \cup l/\beta' = \downarrow/\beta'$ by the *l-child* and *left-distributivity* axioms, and $\downarrow^*/\beta' \cup l/\beta' \equiv_{\mathcal{I}_r} \downarrow^*/\beta'$ by the *descendants, left-distributivity, descendants-union* and *empty-path* axioms; thus $l/(\bigcup S_l) \cup \tau \equiv_{\mathcal{I}_r} \tau$ and $l/(\bigcup S_d) \cup \tau \equiv_{\mathcal{I}_r} \tau$ by the inference rules. Second, $\downarrow^*/\beta' \equiv_{\mathcal{I}_r} \downarrow^*/\downarrow^*/\beta'$ as shown by the example given in Section 6; thus $l/(\bigcup S_{*,2}) \cup \tau \equiv_{\mathcal{I}_r} \tau$. Putting these together, we have that Subclaim 2 holds.

Case 2. η is $\downarrow$.

As in the proof for Case 1, we define the sets S, $S_{*,1}$, $S_{*,2}$, S_d and the following $\mathcal{X}_r$ expression:

$$\widehat{\tau} = \bigcup S_d \ \cup \ \bigcup S_{*,2}.$$

We show that Subclaim 1 given above and Subclaim 3 below still hold in this case. From these follows Claim.

Subclaim 3: $\downarrow/\widehat{\tau} \cup \tau \equiv_{\mathcal{I}_r} \tau$.

The proof of Subclaim 3 is similar to the proof of Subclaim 2 in Case 1. We next show Subclaim 1. Suppose by contradiction $\beta \not\subseteq \widehat{\tau}$. Then there exists a simple path ρ such that $\rho \subseteq \beta$ but $\rho \not\subseteq \widehat{\tau}$. Since $\alpha \subseteq \tau$, we have $\downarrow/\rho \subseteq \tau$. But for any label l in the alphabet Σ, $l/\rho \not\subseteq \downarrow/\widehat{\tau}$ since otherwise by the definition of $\widehat{\tau}$, we would have had $\rho \subseteq \widehat{\tau}$. Let S' consist of all the union-free expressions of the form l'/β' in S plus ϵ if $\epsilon \subseteq \tau$, where l' is a label. Then from the argument above and the definition of S_d and $S_{*,2}$ it follows that $\downarrow/\rho \subseteq \bigcup S'$, since $\downarrow/\rho \subseteq \tau$. Choose a label a such that no expression in S' is of the form a/β'. Hence $a/\rho \not\subseteq \bigcup S'$. This contradicts the assumption. Thus $\beta \subseteq \widehat{\tau}$, i.e., Subclaim 1 holds in this case.

Case 3. η is $\downarrow^*$.

Again as in the proof for Case 1, we define the sets S, $S_{*,1}$, $S_{*,2}$ and let

$$\widehat{\tau} = \bigcup S_{*,1} \ \cup \ \bigcup S_{*,2}.$$

We show Subclaim 1 and Subclaim 4 below for this case.

Subclaim 4: $\downarrow^*/\widehat{\tau} \cup \tau \equiv_{\mathcal{I}_r} \tau$.

Again with mild change the proof of Subclaim 2 in Case 1 proves Subclaim 4. We next show Subclaim 1 for this case. Suppose by contradiction $\beta \not\subseteq \widehat{\tau}$. Then there exists a simple path ρ such that $\rho \subseteq \beta$ but $\rho \not\subseteq \widehat{\tau}$. Since $\alpha \subseteq \tau$, we have $\downarrow^*/\rho \subseteq \tau$. But for any simple path ρ_*, $\rho_*/\rho \not\subseteq \downarrow/\widehat{\tau}$ since otherwise we would have had $\rho \subseteq \widehat{\tau}$ by the definition of $\widehat{\tau}$. Let S' consist of union-free expressions of either the form l/β' or the form $\downarrow/\beta'$ in S plus ϵ if $\epsilon \subseteq \tau$, where l is a label. Then from the argument above and the definition of $S_{*,2}$ it follows that $\downarrow^*/\rho \subseteq \bigcup S'$ since $\downarrow^*/\rho \subseteq \tau$. Note that each α' in S' can be written as $\eta_1/\cdots/\eta_s$, where for each $i \in [2, s]$, η_i is either a label l, or $\downarrow$ or $\downarrow^*$. Let us refer to i as the *position* of η_i, and η_i as the *symbol* at position i in α'. We define a function f such that $f(\alpha')$ is either the position j of the first occurrence $\downarrow^*$ in α' if there is one, i.e., $\eta_j = \downarrow^*$ but $\eta_i \neq \downarrow^*$ for any $i < j$, or the length of α' if it does not contain $\downarrow^*$. Observe that condition (2) given above excludes the possibility of $\downarrow/\beta'$ in S' such that β' starts with a contiguous sequence of $\downarrow$'s followed by $\downarrow^*$, since otherwise β' would have been put in $S_{*,1}$. Thus for any α' in S', either there is $i \in [1, f(\alpha')]$ such that the symbol at position i is a label, or α' consists of $\downarrow$'s only and its length equals $f(\alpha')$; in the latter case we say that the symbol at position $f(\alpha')+1$ is ϵ. Let n be the largest $f(\alpha')$ for all expressions α' in S'. From the discussion above it follows that we can find a simple path $\rho_* = a_1/\cdots/a_{n+1}$

with the property: for any α' in S', there is $j \in [1, n+1]$ such that a_j is not contained in η_j, which is at position j in α'. Thus $\rho_*/\rho \not\subseteq \bigcup S'$. This contradicts the assumption. Thus $\beta \subseteq \widehat{\tau}$. Hence Subclaim 1 also holds in this case.

Therefore, the claim holds for all the cases. This completes the proof of Lemma 6.3. □

Proof of Theorem 6.4

The first part of the theorem can be verified along the same lines as the proof of Lemma 6.3. The second part is done similarly except that the claim introduced in that proof needs to be shown here in a slightly different way. The claim can be stated for $\mathcal{X}$ as follows: for any union-free $\mathcal{X}$ expression α and $\mathcal{X}$ expression τ, if $\alpha \subseteq_{\Sigma_0} \tau$ then $\alpha \cup \tau \equiv_{\mathcal{I}^f(\Sigma_0)} \tau$. The claim can be proved by an induction on the length of α, which is the same as the one given in the proof of Lemma 6.3 except for the case when α is of the form $\downarrow/\beta$ (Case 2). Here it suffices to show $l/\beta \cup \tau \equiv_{\mathcal{I}^f(\Sigma_0)} \tau$ for each l in the alphabet Σ_0. Then the *finite-alphabet* axiom of $\mathcal{I}^f$ gives us $l/\alpha \cup \tau \equiv_{\mathcal{I}^f(\Sigma_0)} \tau$. □

Proof of Proposition 6.5

We show the first part of the proposition; the proof for the second part (for finite alphabet Σ_0) is analogous.

The existence of a normal form α' equivalent to a given union-free α can be shown by a straightforward structural induction on α. To show the uniqueness of α', the following claim suffices:

Claim: For any two union-free $\mathcal{X}_r$ terms α and α' in the normal form, if $\alpha \equiv \alpha'$ then $\alpha = \alpha'$, i.e., they are syntactically equal to each other.

We prove the claim by induction on the length of α.

Induction basis: When α is ϵ or $\emptyset$, by condition (1) of the definition of the normal form, it is easy to see that the claim holds.

Inductive step: Assume the claim for $\alpha = \beta$. We show the claim for $\alpha = \eta/\beta$, where η is one of l, $\downarrow$ or $\downarrow^*$. Since $\alpha \equiv \alpha'$, α' cannot be ϵ or $\emptyset$. Thus by condition (2) of the definition of the normal form, α' can be written as η'/β', where η' is also one of the symbols given above. We consider the following cases of η.

Case 1. If η is a label l, then by contradiction it is easy to show that η' must be the same label l. Then by the induction hypothesis, $\beta = \beta'$. Thus $\alpha = \alpha'$.

Case 2. If η is $\downarrow$, there are two cases to consider. If it is not followed by $\downarrow^*$, then by condition (2) of the definition of the normal form, it cannot be followed by a contiguous sequence of $\downarrow$'s and then a $\downarrow^*$. In this case, it is easy to argue that it contradicts $\alpha \equiv \alpha'$ if η' is not $\downarrow$. If η is followed by $\downarrow^*$, then again by condition (2) α must be of the form $\downarrow^*/\downarrow/\downarrow^*/\cdots/\downarrow/\downarrow^*/\gamma$ where γ does not start with $\downarrow$ or $\downarrow^*$; this case will be handled by the proof for Case 3 below.

Case 3. If η is $\downarrow^*$, there are two cases to consider. If η is followed by $\downarrow$, then again by condition (2) α must be of the form $\downarrow^*/\downarrow/\downarrow^*/\cdots/\downarrow/\downarrow^*/\gamma$, where γ does not start with $\downarrow$ or $\downarrow^*$. Let n be the number of $\downarrow$ occurrences in the leading sequence of α before γ. Then a simple induction on n can show that α' must be of the form $\downarrow^*/\downarrow/\downarrow^*/\cdots/\downarrow/\downarrow^*/\gamma'$ with exactly n occurrences of $\downarrow$ in the leading sequence before γ', where γ' does not start with $\downarrow$ or $\downarrow^*$. Then by the induction hypothesis, $\gamma = \gamma'$. Thus $\alpha = \alpha'$. If η is not followed by $\downarrow$, by condition (3) of the definition of the normal form, it must be followed by a label l. In this case η' must be $\downarrow^*$, since otherwise it is easy to show that $\alpha \not\equiv \alpha'$. Again the induction hypothesis, $\beta = \beta'$. Thus $\alpha = \alpha'$. □

Proof of Proposition 6.6

One can rewrite τ into a union normal form $\alpha_1 \cup \cdots \cup \alpha_k$ by using $\mathcal{I}$ axioms (resp. $\mathcal{I}^f(\Sigma_0)$). Formally this can be verified by structural induction on τ. To prove the uniqueness of the union normal form, it suffices to show the following claims:

Claim 1: Let α, α' be union-free $\mathcal{X}$ expressions such that $\alpha \equiv \alpha'$ and they satisfy condition (1) of the definition of the union normal form. Then $\alpha = \alpha'$. Similarly, for a fixed finite Σ_0 containing all labels in α, α', we have that if $\alpha \equiv_{\Sigma_0} \alpha'$ and they satisfy (1), then $\alpha = \alpha'$.

Claim 2: Let α be a union-free $\mathcal{X}$ expression, and $\alpha'_1 \cup \cdots \cup \alpha'_m$ be the union normal form of an $\mathcal{X}$ expression τ such that $\alpha \subseteq \tau$. Then there is $j \in [1, m]$ such that $\alpha \subseteq \alpha'_j$. The above also holds for the union normal form for the finite alphabet case, satisfying conditions (1)–(4) above the statement of the theorem, where $\subseteq$ is replaced in both cases by $\subseteq_{\Sigma_0}$.

For if these hold, then Proposition 6.6 can be verified as follows (we outline only the first part of the proposition; the modification for the second part is immediate). Assume that an $\mathcal{X}$ expression τ has two union normal forms $\alpha_1 \cup \cdots \cup \alpha_k$ and $\alpha'_1 \cup \cdots \cup \alpha'_m$. Then from Claim 2 it follows that for any $i \in [1, k]$ there is $j \in [1, m]$ such that $\alpha_i \subseteq \alpha'_j$. Again by Claim 2 there must be some $s \in [1, k]$ such that $\alpha'_j \subseteq \alpha_s$. Then α_s must be α_i since otherwise it would be the case that $\alpha_i \subseteq \alpha_s$, which contradicts the definition of the union normal form. That is, $\alpha_i \equiv \alpha'_j$. By Claim 1, α_i and α'_j are syntactically equal to each other. Putting these together, for any $i \in [1, k]$ there is $j \in [1, m]$ such that α_i and α'_j are syntactically equal, and vice versa. Thus τ and τ' are syntactically equal up to different ordering of their disjuncts. That is, the union normal form of τ is unique.

The proof of Proposition 6.5 also proves Claim 1. We show Claim 2 by induction on the length of α:

Induction basis: If α is $\emptyset$ or ϵ, then by condition (1) of the definition of the union normal form and $\alpha \subseteq \tau$ there must be a disjunct α' of τ such that $\alpha \subseteq \alpha'$, and similarly for $\subseteq_{\Sigma_0}$. Thus Claim 2 holds when $|\alpha| = 0$.

Inductive step: Assume Claim 2 for $\alpha = \beta$. We show that it also holds for $\alpha = \eta/\beta$, where η is a label or $\downarrow$ (the latter is only considered in the case of unrestricted equivalence). We give separate arguments for the case of finite Σ_0 and $\subseteq_{\Sigma_0}$ and the case of equivalence over arbitrary labeled trees.

For the case of fixed alphabet Σ_0, by condition (4) of the definition of the union normal form, each symbol of α and τ is a label. Let S be the set of all disjuncts in the union normal form of τ, and S' be $\{\beta' \mid l/\beta' \in S\}$, i.e., it consists of all the expressions in S of the form l/β'. It is easy to verify that $\beta \subseteq_{\Sigma_0} \bigcup S'$. Thus by the induction hypotheses there is $\beta' \in S'$ such that $\beta \subseteq_{\Sigma_0} \beta'$. That is, $\alpha \subseteq_{\Sigma_0} l/\beta'$. Note that l/β' is a union-free expression of τ. Thus Claim 2 holds for the case of finite alphabet Σ_0.

In the case of arbitrary trees over an infinite label set Σ, each symbol of α and τ is either a label or $\downarrow$. If $\eta = \downarrow$, let S_d be $\{\beta' \mid \downarrow/\beta' \in S\}$, i.e., it consists of all the expressions in S of the form $\downarrow/\beta'$. It is easy to prove by contradiction that $\beta \subseteq \bigcup S_d$. Thus by the induction hypotheses there is $\beta' \in S_d$ such that $\beta \subseteq \beta'$. That is, $\alpha \subseteq \downarrow/\beta'$. If $\eta = l$, let S_l be $\{\beta' \mid \downarrow/\beta' \in S \text{ or } l/\beta' \in S\}$, i.e., it consists of all the expressions in S of the form $\downarrow/\beta'$ or l/β'. Again it is easy to prove by contradiction that $\beta \subseteq \bigcup S_l$. By the induction hypotheses there is $\beta' \in S_l$ such that $\beta \subseteq \beta'$. If $\downarrow/\beta'$ is in S then obviously $\alpha \subseteq \downarrow/\beta'$; otherwise l/β' must be in S and $\alpha \subseteq l/\beta'$. Thus Claim 2 also holds for infinite alphabet.

This completes the proof of Proposition 6.6. □