Validity of Positive XPath Queries with Wildcard in the Presence of DTDs

Kenji Hashimoto
Nara Institute of Science and Technology, Japan
k-hasimt@is.naist.jp

Yohei Kusunoki
Osaka University, Japan
y-kusunk@ist.osaka-u.ac.jp

Yasunori Ishihara
Osaka University, Japan
ishihara@ist.osaka-u.ac.jp

Toru Fujiwara
Osaka University, Japan
fujiwara@ist.osaka-u.ac.jp

ABSTRACT

This paper discusses the validity problem for positive XPath queries with wildcard in the presence of DTDs. A given XPath query p is valid under a DTD D if, for every XML document T conforming to D, the answer to p on T is nonempty. The validity problem is one of the basic static analyses of queries, together with the satisfiability and the containment problems. Although the validity problem is the dual of the satisfiability problem, the complexity of validity for positive XPath classes is not obvious because the XPath class does not contain the negation operator. In this paper, first, it is shown that the path union operator in XPath queries easily makes the validity problem intractable. Then, we focus on wildcard, which is a special case of path union and more popular than path union in the real world. Interestingly, wildcard together with child and descendant-or-self axes and qualifier causes intractability while the validity problem becomes tractable for XPath classes defined by any combination of three of child axis, descendant-or-self axis, qualifier, and wildcard.

1. INTRODUCTION

1.1 Overview

Static analysis of XPath queries is one of the major theoretical topics in the field of XML databases. XPath is a query language for XML documents, where an XML document is often regarded as an unranked labeled ordered tree. An XPath query specifies a pattern of paths from the root of a given XML document. The answer to an XPath query for an XML document T is a set of nodes v of T such that the specified path pattern matches the path from the root to v.

There are two major subtopics in the research on static analysis of XPath queries: XPath satisfiability and XPath containment in the presence of DTDs (Document Type Definitions). A given XPath query p is *satisfiable* under a given DTD D if there is an XML document T conforming to D such that the answer to p for T is a nonempty set. On the other hand, a given XPath query p *contains* a given XPath query p' under a given DTD D if for every XML document T conforming to D, the answer to p' for T is a subset of the answer to p for T. These subtopics are strongly motivated by query optimization.

On the other hand, as far as the authors know, there is little research on *XPath validity* in the presence of DTDs. A given query p is *valid* under a DTD D if every XML document conforming to D satisfies p. The validity checking of an XPath query is useful when the query is regarded as a Boolean query, i.e., the issuer of the query is interested in whether the query matches the XML document rather than the set of returned nodes.

The validity problem is the dual of the satisfiability problem. Specifically, it is the complement of the satisfiability if the class of XPath allows the negation operator, because p is valid under a DTD D if and only if $\neg p$ is unsatisfiable under the DTD D. Hence, the complexity of validity for many classes of XPath with negation can be derived immediately from the results on the satisfiability problem. However, if the class of XPath is positive (i.e., the negation operator is not allowed), the complexity of validity is not obvious. In summary, the validity problem of positive XPath queries is worth investigating from both practical and theoretical points of view.

As for conjunctive queries over trees, which is a proper superclass of the positive XPath queries without path union, Björklund et al. [2] investigated the complexity of not only satisfiability and containment problems but also validity problem with respect to an XML schema. They showed that conjunctive query validity is 2EXPTIME-complete with respect to a DTD or a finite tree automaton, which is an abstraction of a RELAX NG schema. Our goal in this paper is to identify the boundary between tractability and intractability of the validity problem for the positive XPath queries.

Our work is summarized as follows. We first show that validity of XPath queries with path union (denoted $\cup$) with one of child (denoted $\downarrow$) and descendant-or-self (denoted $\downarrow^*$) axes is coNP-hard. Since $\cup$ brings the semantics of logical disjunction into XPath, the result may not be so surprising. Next, we focus on wildcard (denoted $*$), which is a special

Table 1: The complexity of validity, satisfiability, and containment under DTDs.

↓	↓*	[ ]	∗	∪	validity	satisfiability	containment
+				+	**coNP-hard**	in PTIME[1]	coNP-hard[12]/in EXPTIME[12]
+	+	+	+		**coNP-hard**	NP-complete[1]	coNP-hard[12]/in EXPTIME[12]
+	+		+		**in PTIME**	in PTIME[1]	in EXPTIME[12]
+		+	+		**in PTIME**	NP-complete[1]	coNP-hard[12]/in EXPTIME[12]
+	+	+			**in PTIME**	NP-complete[1]	coNP-hard[12]/in EXPTIME[12]

case of ∪ and more popular than ∪ in the real world. However, it is shown that validity of XPath queries with ↓, ↓*, qualifier (denoted [ ]), and ∗ is coNP-hard. Finally, we show that the validity problem becomes tractable by excluding one of ↓, ↓*, [ ], and ∗ from the latter intractable subclass. Thus, these results show a tight boundary between tractability and intractability for positive XPath queries with wildcard.

Table 1 summarizes the results of this paper. A "+" in a multicolumn means "one of them". For example, the first row of the table represents the class with ∪ and one of ↓ and ↓*. Bold letters indicate the contributions of this paper. We note that in our paper, logical connectives ∧ and ∨ in qualifier are not allowed.

1.2 Related work

As stated above, XPath satisfiability and containment have been extensively studied so far. As for satisfiability, Benedikt et al. investigated how the combinations of XPath components such as downward/upward axes, qualifiers, and negations affect the complexity of the satisfiability problem [1]. They also derived a number of results on XPath containment from the results on the satisfiability of XPath queries with negation. Other components of XPath such as sibling axes [1, 6, 8, 9], following-sibling and preceding-sibling axes [14], and data equality [1, 5] were investigated.

As for XPath containment, Miklau and Suciu investigated how the combination of descendant axis, qualifier, and wildcard affect the complexity of XPath containment without DTDs [11]. XPath containment in the presence of DTDs were studied by many researchers such as Deutsch and Tannen [3], Wood [16], and Neven and Schwentick [12, 13]. Recently, ten Cate and Lutz showed the complexity of the containment of queries in XPath 2.0 [15].

On the other hand, some researches tackled the intractability of XPath static analysis problems in the presence of schema information by using appropriate logics. Libken and Sirangelo presented a single-exponential direct translation from an LTL-like logic for trees into an automata model, and applied the translation to static analysis problems of XPath [10]. Genevès and Layaïda presented a practically fast algorithm to solve XPath static analysis problems, based on a variant of μ-calculus [7]. In their approach, XPath queries are translated to formulas of the logic, and then, validity, as well as satisfiability and containment, is verified by fast decision procedures for μ-calculus formulas.

We aim at investigating how the combinations of positive XPath components of positive XPath components affect the complexity of validity. This paper studies the validity problem of positive downward XPath queries with wildcard. As stated above, if the class of XPath is restricted to be positive, the problem is not the dual problem of the satisfiability, and therefore, the complexity is not obvious.

1.3 Organization of the paper

The rest of the paper is organized as follows. Preliminary definitions are given in Section 2. Then, the intractability and tractability results are presented in Sections 3 and 4 respectively. Section 5 summarizes the paper.

2. DEFINITION

2.1 XML Documents

An XML document is represented by an unranked labeled ordered tree. The label of a node v, denoted $\lambda(v)$, corresponds to a tag name. We extend λ to a function on sequences, i.e., for a sequence $v_1 \cdots v_n$ of nodes, let $\lambda(v_1 \cdots v_n) = \lambda(v_1) \cdots \lambda(v_n)$. For a tree T, let $r(T)$ denote the root of T. Attributes are not handled in this paper.

2.2 DTDs

A regular expression over an alphabet Σ consists of constants $\emptyset$ (empty set), ϵ (empty sequence), and the symbols in Σ, and operators $\cdot$ (concatenation), $*$ (zero or more occurrences), and $|$ (disjunction). The concatenation operator is often omitted. The string language represented by a regular expression e is denoted by $L(e)$.

DEFINITION 1. *A DTD is a triple $D = (\Sigma, r, P)$, where:*

- *Σ is a finite set of labels,*
- *$r \in \Sigma$ is the root label, and*
- *P is a mapping from Σ to the set of regular expressions over Σ. $P(a)$ is called the content model of label a.*

DEFINITION 2. *A tree T conforms to a DTD $D = (\Sigma, r, P)$ if*

- *the label of the root of T is r, and*
- *for each node v of T and its child sequence $v_1 \cdots v_n$, $L(P(\lambda(v)))$ contains $\lambda(v_1 \cdots v_n)$.*

Let $TL(D)$ denote the set of all the trees conforming to D.

The size of a regular expression is the number of constants and operators appearing in the regular expression. The size of a DTD is the sum of the sizes of all content models.

In this paper, we assume that every DTD $D = (\Sigma, r, P)$ contains no useless symbols. That is, for each $a \in \Sigma$, there is a tree T conforming to D such that the label of some node of T is a. Given a DTD with useless symbols, another equivalent DTD without useless symbols can be computed in linear time to the sizes of the given DTD.

Let $D = (\Sigma, r, P)$, $\sigma \in \Sigma$, and $S \subseteq \Sigma$. Let (D, σ) denote a DTD D' such that $TL(D') = TL((\Sigma, \sigma, P))$. Let P' be a mapping such that for each $\sigma \in \Sigma$, $P'(\sigma)$ is the regular expression obtained from $P(\sigma)$ by replacing a with $\emptyset$. Let $[D, a, \sigma]$ denote a DTD D' such that $TL(D') = TL((\Sigma, \sigma, P'))$. In other words, $T' \in TL([D, a, \sigma])$ if and only if the label of the root of T' is σ, those of all the other nodes are not a, and T' is a subtree of some $T \in TL(D)$. We assume that (D, σ) and $[D, a, \sigma]$ contain no useless symbols because such DTDs without useless symbols can be computed from (Σ, σ, P) and (Σ, σ, P') in linear time, respectively.

2.3 XPath Expressions

The syntax of an XPath query p is defined as follows:

$$\begin{aligned} p &::= \chi :: l \mid p/p \mid p \cup p \mid p[p], \\ \chi &::= \downarrow \mid \downarrow^* \end{aligned}$$

where l is in Σ or a *wildcard* $*$. Each $\chi \in \{\downarrow, \downarrow^*\}$ is called an *axis*. The operators / and $\cup$ are called a *path concatenation* and a *path union*, respectively. A subexpression in the form of $\chi :: l$ is called a *location step*. Also, a subexpression in the form of $[p]$ is called a *qualifier*. The size of an XPath query p, denoted by $|p|$, is defined as the number of location steps occurring in p.

Definition 3. *The semantics of an XPath query over a tree T is defined as the following binary predicate on nodes:*

- $T \models (\downarrow :: l)(v, v')$ *if v' is a child of v and $\lambda(v') = l$.*
- $T \models (\downarrow :: *)(v, v')$ *if v' is a child of v.*
- $T \models (\downarrow^* :: l)(v, v')$ *if v' is v or a descendant of v, and $\lambda(v') = l$.*
- $T \models (\downarrow^* :: *)(v, v')$ *if v' is v or a descendant of v.*
- $T \models (p_1/p_2)(v, v')$ *if there is v'' such that $T \models p_1(v, v'')$ and $T \models p_2(v'', v')$.*
- $T \models (p_1 \cup p_2)(v, v')$ *if $T \models p_1(v, v')$ or $T \models p_2(v, v')$.*
- $T \models (p_1[p_2])(v, v')$ *if $T \models p_1(v, v')$ and $T \models p_2(v', v'')$ for some v''.* □

A tree T *satisfies* an XPath query p relative to a node v_c if there is a node v such that $T \models p(v_c, v)$. In particular, if v_c is the root node of T, then we simply say that T satisfies p. An XPath query p is *valid* under a DTD D if every $T \in TL(D)$ satisfies p.

A subclass of XPath which allows to use axes and operators in a list C and path concatenation / is denoted by $\mathcal{X}(C)$. For example, the subclass with child axis, predicate, and path concatenation is denoted by $\mathcal{X}(\downarrow, [\ ])$. We note that we do not include path concatenation in C but every subclass of XPath we consider in this paper allows path concatenation. We write the XPath validity problem for a fragment $\mathcal{X}$ of XPath as $\mathsf{VLD}(\mathcal{X})$.

3. INTRACTABILITY RESULTS

We first show that subclasses of XPath with path union are intractable. More precisely, the validity problem of XPath queries with path union and one of child and descendant-or-self axes is coNP-hard. Moreover, we show that, even without path union, the validity problem of XPath queries with child and descendant-or-self axes, qualifier, and wildcard is coNP-hard.

Theorem 1. $\mathsf{VLD}(\mathcal{X}(\downarrow, \cup))$ *and* $\mathsf{VLD}(\mathcal{X}(\downarrow^*, \cup))$ *are coNP-hard.*

Proof. We show only that $\mathsf{VLD}(\mathcal{X}(\downarrow, \cup))$ is coNP-hard. It can be verified in a similar way that $\mathsf{VLD}(\mathcal{X}(\downarrow^*, \cup))$ is coNP-hard.

The coNP-hardness of $\mathsf{VLD}(\mathcal{X}(\downarrow, \cup))$ is shown by reduction from the 3DNF tautology problem, which is coNP-complete. We now consider an instance $\phi = (L_{1,1} \wedge L_{1,2} \wedge L_{1,3}) \vee \cdots \vee (L_{n,1} \wedge L_{n,2} \wedge L_{n,3})$ of 3DNF tautology, where each $L_{i,k}$ is a member of $\{x_1, \ldots, x_m, \bar{x}_1, \ldots, \bar{x}_m\}$. Then, we define a DTD $D = (\Sigma, r, P)$ as follows:

$$\begin{aligned} &\Sigma = \{r, x_1, \ldots, x_m, \bar{x}_1, \ldots, \bar{x}_m\}, \\ &P(r) = x_1 \mid \bar{x}_1, \\ &P(x_j) = P(\bar{x}_j) = x_{j+1} \mid \bar{x}_{j+1} \text{ for each } j\ (1 \le j < m), \\ &P(x_m) = P(\bar{x}_m) = \epsilon. \end{aligned}$$

Also, an XPath query p is defined as $p_1 \cup \cdots \cup p_n$, where

$$\begin{aligned} p_i &= p_{i1}/\cdots/p_{im}, \text{ for each } i\ (1 \le i \le n), \\ p_{ij} &= \begin{cases} \downarrow :: x_j & \text{if } \exists k. L_{i,k} = x_j, \\ \downarrow :: \bar{x}_j & \text{if } \exists k. L_{i,k} = \bar{x}_j, \\ (\downarrow :: x_j \cup \downarrow :: \bar{x}_j) & \text{otherwise.} \end{cases} \end{aligned}$$

Each $T \in TL(D)$ is a tree without branch and represents a truth assignment. Each p_i corresponds to the term $L_{i,1} \wedge L_{i,2} \wedge L_{i,3}$, and each p_{ij} corresponds to the literal of the variable x_j. It is easily verified that ϕ is valid if and only if p is valid under D. □

Theorem 2. $\mathsf{VLD}(\mathcal{X}(\downarrow, \downarrow^*, [\ ], *))$ *is coNP-hard.*

Proof. The coNP-hardness of $\mathsf{VLD}(\mathcal{X}(\downarrow, \downarrow^*, [\ ], *))$ is shown by reduction from the 3DNF tautology problem.

We now consider an instance $\phi = (L_{1,1} \wedge L_{1,2} \wedge L_{1,3}) \vee \cdots \vee (L_{n,1} \wedge L_{n,2} \wedge L_{n,3})$ of 3DNF tautology, where each $L_{i,k}$ is a member of $\{x_1, \ldots, x_m, \bar{x}_1, \ldots, \bar{x}_m\}$. Then, we define a DTD $D = (\Sigma, r_1, P)$ (See Fig. 1) as follows:

$$\begin{aligned} \Sigma = {} & \{t, f, s, g\} \cup \{r^i \mid 1 \le i \le 2n\} \\ & \cup \{d^i_j \mid 1 \le i < 2n, i \ne n, 1 \le j \le m\} \\ & \cup \{x^n_j, \bar{x}^n_j \mid 1 \le j \le m\}, \\ & P(t) = P(f) = P(s) = P(g) = \epsilon, \\ & P(r^i) = s d^i_1\ (1 \le i < n),\ P(r^n) = s(x^n_1 \mid \bar{x}^n_1), \\ & P(r^i) = d^i_1\ (n < i < 2n),\ P(r^{2n}) = g, \\ & P(d^i_j) = t f d^i_{j+1}\ (1 \le i < 2n, i \ne n, 1 \le j < m), \\ & P(d^i_m) = t f r^{i+1}\ (1 \le i < 2n, i \ne n), \\ & P(x^n_j) = t(x^n_{j+1} \mid \bar{x}^n_{j+1})\ (1 \le j < m), \\ & P(\bar{x}^n_j) = f(x^n_{j+1} \mid \bar{x}^n_{j+1})\ (1 \le j < m), \\ & P(x^n_m) = t r^{n+1}, P(\bar{x}^n_m) = f r^{n+1}. \end{aligned}$$

We give the following XPath query $p \in \mathcal{X}(\downarrow, \downarrow^*, [\ ], *)$:

$$p = \downarrow^* :: *[\downarrow :: s]/p_1/p_2/\cdots/p_n/\downarrow :: *,$$

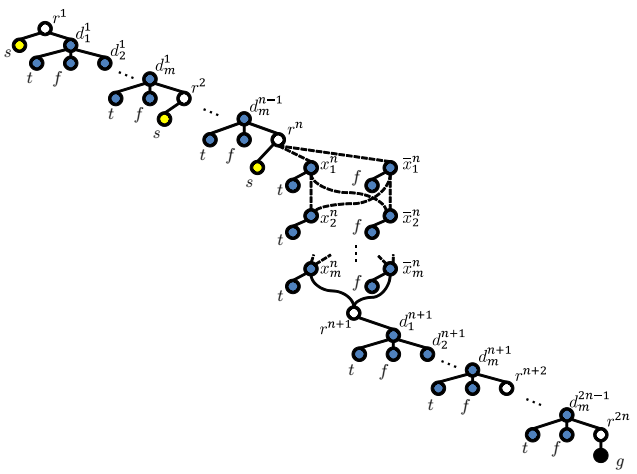

Figure 1: The DTD encoding a 3DNF tautology instance in the proof of Theorem 2

where

$$p_i = p_{i1}/p_{i2}/\cdots/p_{im}/\downarrow::* \quad (1 \le i \le n),$$

$$p_{ij} = \begin{cases} \downarrow::*[\downarrow:: t] & \text{if } \exists k.L_{i,k} = x_j, \\ \downarrow::*[\downarrow:: f] & \text{if } \exists k.L_{i,k} = \bar{x}_j, \\ \downarrow::* & \text{otherwise.} \end{cases}$$

The part of each tree $T \in TL(D)$ between the nodes with the label r^n and r^{n+1} represents a truth assignment. Each p_i corresponds to the i-th term $L_{i,1} \wedge L_{i,2} \wedge L_{i,3}$, and each p_{ij} corresponds to the literal of the variable x_j appearing in the i-th term.

Assume that ϕ is valid. Consider an arbitrary tree $T \in TL(D)$. Let v^i $(1 \le i < 2n)$ denote the node of T with $\lambda(v^i) = r^i$. Under the truth assignment represented by T, some term $L_{I,1} \wedge L_{I,2} \wedge L_{I,3}$ becomes true. By the definition of p, we have that $T \models p_I(v^n, v^{n+1})$. Moreover, we have that $T \models \downarrow^*::*[\downarrow:: s](r(T), v^i)$ for any i $(1 \le i < n)$, and that $T \models p_i(v^l, v^{l+1})$ for any $1 \le i \le n$ and $1 \le l < 2n$ such that $l \ne n$. Thus, $T \models \downarrow^*::*[\downarrow:: s]/p_1/\cdots/p_{I-1}(r(T), v^n)$ and $T \models p_{I+1}/\cdots/p_n(v^{n+1}, v^{2n-I+1})$. Therefore, T satisfies p.

Conversely, assume that p is valid under D. Consider an arbitrary truth assignment, and let $T \in TL(D)$ be a tree representing the truth assignment. Again, let v^i $(1 \le i < 2n)$ denote the node of T with $\lambda(v^i) = r^i$. Since T satisfies p, there must be a node $v^{I'}$ $(1 \le I' \le n)$ such that

- $T \models \downarrow^*::*[\downarrow:: s](r(T), v^{I'})$, and
- $T \models p_1/\cdots/p_n(v^{I'}, v^{n+I'})$.

Thus, by the definition of p and D, we must have $T \models p_I(v^n, v^{n+1})$ where $I = n - I' + 1$. That is, the truth assignment corresponding to the part between v^n and v^{n+1} satisfies the term $L_{I,1} \wedge L_{I,2} \wedge L_{I,3}$, and hence, ϕ. □

As for upper bounds, the satisfiability for $\mathcal{X}(\downarrow, \uparrow, [\,], *, \cup, \neg)$ and $\mathcal{X}(\downarrow, \downarrow^*, \uparrow, \uparrow^*, [\,], *, \cup, \neg)$ are known to be PSPACE-complete and EXPTIME-complete respectively [1] where $\uparrow$ and $\uparrow^*$ are parent and ancestor axes respectively, and $\neg$ is the negation operator. The validity problems for these classes are also PSPACE-complete and EXPTIME-complete respectively because validity and satisfiability are dual for the classes, which allows negation operators. Thus, $\mathsf{VLD}(\mathcal{X}(\downarrow, \cup))$ is in PSPACE, and $\mathsf{VLD}(\mathcal{X}(\downarrow^*, \cup))$ and $\mathsf{VLD}(\mathcal{X}(\downarrow, \downarrow^*, [\,], *))$ are in EXPTIME. We have not found tighter bounds of them. We conjecture that they are not in coNP because it is hard to represent a witness for unsatisfiability in polynomial size.

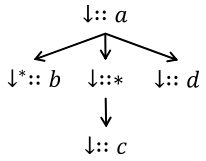

Figure 2: The tree rep. of $\downarrow:: a[\downarrow^*:: b][\downarrow:: */\downarrow:: c]/\downarrow:: d$

4. TRACTABILITY RESULTS

In this section, we show that the validity problem becomes tractable by excluding one of $\downarrow$, $\downarrow^*$, $[\,]$, and $*$ from the intractable class $\mathcal{X}(\downarrow, \downarrow^*, [\,], *)$. To that end, we give a subclass of $\mathcal{X}(\downarrow, \downarrow^*, [\,], *)$ which (almost) contains the subclasses with any three of $\downarrow$, $\downarrow^*$, $[\,]$, and $*$, and then provide a polynomial-time algorithm to check the validity of a query in the subclass.

4.1 Definition of a tractable class

We define a subclass of $\mathcal{X}(\downarrow, \downarrow^*, [\,], *)$, where we shall show that validity can be decided in polynomial time. For this, we first introduce tree representations of XPath queries.

Tree representation of XPath. We define the (unranked labeled unordered) tree representation of $p \in \mathcal{X}(\downarrow, \downarrow^*, [\,], *)$. The tree representation $t(p)$ of p and the "exit" node $d(p)$ of $t(p)$ are defined inductively as follows:

- $t(\chi :: l)$ is a tree which has only a node with label $\chi :: l$, and $d(p)$ is the node.
- $t(p_1/p_2)$ is a concatenation of $t(p_1)$ and $t(p_2)$ where the root of $t(p_2)$ is a child of $d(p_1)$, and $d(p_1/p_2) = d(p_2)$.
- $t(p_1[p_2])$ is a concatenation of $t(p_1)$ and $t(p_2)$ where the root of $t(p_2)$ is a child of $d(p_1)$, and $d(p_1[p_2]) = d(p_1)$.

Henceforth, we represent $p \in \mathcal{X}(\downarrow, \downarrow^*, [\,], *)$ as a term which means $t(p)$. For $p = \chi :: l(p_1, \ldots, p_n)$, let $rt(p)$ denote the root $\chi :: l$ of p, $s(p)$ denote $\{p_1, \ldots, p_n\}$, and $\lambda(\chi :: l) = l$. For example, $p = \downarrow:: a[\downarrow^*:: b][\downarrow:: */\downarrow:: c]/\downarrow:: d$ is represented by $\downarrow:: a(\downarrow^*:: b, \downarrow:: *(\downarrow:: c), \downarrow:: d)$, as depicted in Fig. 2.

Definition of $\mathcal{X}(\downarrow, \downarrow^*, [\,], *)^\dagger$. Now we define $\mathcal{X}(\downarrow, \downarrow^*, [\,], *)^\dagger$ as the following subclass of $\mathcal{X}(\downarrow, \downarrow^*, [\,], *)$: $q \in \mathcal{X}(\downarrow, \downarrow^*, [\,], *)^\dagger$ if for each subtree q' of q such that $rt(q') = \downarrow^*:: *$, there is no subtree $q'' \in s(q')$ such that

- $rt(q'') = \downarrow:: a$ and the height of q'' is at least two, or
- $rt(q'') = \downarrow:: *$.

For example, the subclass $\mathcal{X}(\downarrow, \downarrow^*, [\,], *)^\dagger$ includes $\downarrow^*:: *(\downarrow:: a, \downarrow^*:: b(\downarrow:: *))$ but neither $\downarrow^*:: *(\downarrow:: a, \downarrow:: b(\downarrow:: *))$ nor $\downarrow^*:: *(\downarrow:: a, \downarrow:: *)$.

We shall show that the validity problem for $\mathcal{X}(\downarrow, \downarrow^*, [\,], *)^\dagger$ under any DTD is in PTIME. Since this subclass includes $\mathcal{X}(\downarrow, [\,], *)$, $\mathcal{X}(\downarrow^*, [\,], *)$, and $\mathcal{X}(\downarrow, \downarrow^*, [\,])$, the tractability of

$\mathsf{VLD}(\mathcal{X}(\downarrow,\downarrow^*,[\,],*)^\dagger)$ implies the tractability of the validity problem for each of these three classes. As for $\mathcal{X}(\downarrow,\downarrow^*,*)$, by the definition of $\mathcal{X}(\downarrow,\downarrow^*,[\,],*)^\dagger$, any $q \in \mathcal{X}(\downarrow,\downarrow^*,*)$ that has the following subtree p is not in $\mathcal{X}(\downarrow,\downarrow^*,[\,],*)^\dagger$:

- $p = \downarrow^* :: *(p')$ such that $rt(p') = \downarrow :: a$ and the height of p' is at least 2, or
- $p = \downarrow^* :: *(p')$ such that $rt(p') = \downarrow :: *$.

However, every query in $\mathcal{X}(\downarrow,\downarrow^*,*) - \mathcal{X}(\downarrow,\downarrow^*,[\,],*)^\dagger$ can be transformed in linear time into a query in $\mathcal{X}(\downarrow,\downarrow^*,[\,],*)^\dagger$ preserving validity by using the following facts:

- for any $a, b \in \Sigma$ where $a \neq b$, $\downarrow^* :: *(p')$ such that $rt(p') = \downarrow :: a$ is valid under (D, b) if and only if p'' such that $rt(p'') = \downarrow^* :: a$ and $s(p'') = s(p')$ is valid under (D, b). For example, $\downarrow :: b(\downarrow^* :: *(\downarrow :: a))$ is equivalent to $\downarrow :: b(\downarrow^* :: a)$ in terms of validity.
- $\downarrow^* :: *(p')$ such that $rt(p') = \downarrow :: *$ is valid under D if and only if $\downarrow :: *(p'')$ such that $rt(p'') = \downarrow^* :: *$ and $s(p'') = s(p')$ is valid under D. For example, $\downarrow^* :: *(\downarrow :: *(\downarrow :: a))$ is equivalent to $\downarrow :: *(\downarrow^* :: *(\downarrow :: a))$ in terms of validity.

Thus, by showing the tractability of $\mathsf{VLD}(\mathcal{X}(\downarrow,\downarrow^*,[\,],*)^\dagger)$, we can say that any subclass obtained by excluding one of $\downarrow$, $\downarrow^*$, $[\,]$, and $*$ from $\mathcal{X}(\downarrow,\downarrow^*,[\,],*)$, is in PTIME.

4.2 Decision algorithm

Given $D = (\Sigma, r, P)$ and $p \in \mathcal{X}(\downarrow,\downarrow^*,[\,],*)^\dagger$, our algorithm works as follows:

1. Compute the set $N(D,p)$ of labels such that p is valid under (D,σ) if and only if $\sigma \in N(D,p)$.
2. Check if r is in $N(D,p)$. If so, p is valid, and otherwise, p is invalid.

The set $N(D,p)$ is computed inductively as follows. Here, we define Σ as the identity of the intersection of subsets of Σ, i.e., for $\mathcal{N} \subseteq 2^\Sigma$, $\bigcap_{N\in\mathcal{N}} N = \Sigma$ if $\mathcal{N} = \emptyset$.

- $N(D, \downarrow :: a(p_1,\ldots,p_n)) =$
$$\begin{cases} \{\sigma \in \Sigma \mid \text{every } w \in L(P(\sigma)) \text{ has } a\} \\ \qquad\qquad \text{if } a \in \bigcap_{i=1}^n N(D,p_i), \\ \emptyset \qquad \text{otherwise.} \end{cases}$$
- $N(D, \downarrow :: *(p_1,\ldots,p_n)) =$
$$\{\sigma \in \Sigma \mid \text{every } w \in L(P(\sigma)) \text{ has some label in } \textstyle\bigcap_{i=1}^n N(D,p_i)\}$$
- $N(D, \downarrow^* :: a(p_1,\ldots,p_n)) =$
$$\begin{cases} \{\sigma \in \Sigma \mid TL([D,a,\sigma]) = \emptyset\} \\ \qquad\qquad \text{if } a \in \bigcap_{i=1}^n N([D,a,a],p_i), \\ \emptyset \qquad \text{otherwise.} \end{cases}$$
- Let $p = \downarrow^* :: *(p_1,\ldots,p_n)$ where for any $p_i \in s(p)$, p_i has no children if $rt(p_i) = \downarrow :: a$ for some $a \in \Sigma$, and $rt(p_i) \neq \downarrow :: *$. Let $sc(p) = \{p_i \in s(p) \mid a \in \Sigma, rt(p_i) = \downarrow :: a\}$.
$$L_c = \{\lambda(rt(p_i)) \mid p_i \in sc(p)\},$$
$$N_d = \bigcap_{p_i \in s(p) - sc(p)} N(D,p_i)$$

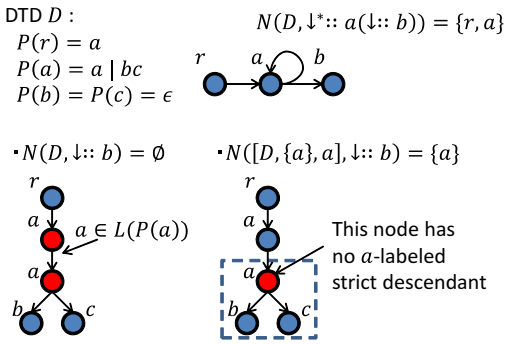

Figure 3: Computing $N(D, \downarrow^* :: a(\downarrow :: b))$

Then,

$$N(D, \downarrow^* :: *(p_1,\ldots,p_n)) = S$$

where S is the maximum set such that

$$S = \{\sigma \in N_d \mid \text{for each } w \in L(P(\sigma)),\ w \text{ has every label in } L_c \text{ or some label in } S\}.$$

If $p = \downarrow :: a(p_1,\ldots,p_n)$, $N(D,p)$ must satisfy the following two conditions: (1) for any $\sigma \in N(D,p)$, in any tree conforming to D, every node labeled with σ has an a-labeled child, and (2) each p_i is satisfied relative to any a-labeled node in any tree conforming to D. To check (1), for each $\sigma \in \Sigma$, we just check if every word conforming to the content model of σ has a. As for (2), for each p_i, we check whether $a \in N(D,p_i)$.

In the case that $p = \downarrow :: *(p_1,...,p_n)$, the label of node relative to which every p_i must be satisfied is not specified. Thus, for any $\sigma \in N(D,p)$, every node labeled with σ must have a child labeled with some label σ' such that each p_i is satisfied relative to any σ'-labeled node. In almost the same way as the case that $p = \downarrow :: a(p_1,\ldots,p_n)$, we check, for each $\sigma \in \Sigma$, if every word conforming to the content model of σ has some label σ' such that $\sigma' \in N(D,p_i)$ for every p_i.

In the case that $p = \downarrow^* :: a(p_1,\ldots,p_n)$, at first, each $\sigma \in N(D,p)$ must satisfy that in any tree conforming to D, any σ-labeled node has an a-labeled descendant. In other words, for each $\sigma \in N(D,p)$, the root of any tree conforming to (D,σ) has an a-labeled descendant. This is checked by deciding $TL([D,a,\sigma]) \neq \emptyset$ because $TL([D,a,\sigma])$ contains only and all trees in $TL((D,\sigma))$ such that the label a does not appear except in the root. In addition, we have to check whether each p_i is satisfied relative to some a-labeled node in any tree conforming to D. We note that checking whether $a \in N(D,p_i)$ for each p_i, as in the case that $p = \downarrow :: a(p_1,\ldots,p_n)$, does not always work well in this case. For example, consider $p = \downarrow^* :: a(\downarrow :: b)$ and the DTD D shown in Fig 3. Note that $N(D, \downarrow :: b) = \emptyset$ because b does not appear in $a \in L(P(a))$. However, in any tree conforming to D, the a-labeled node closest to leaves has a b-labeled child. Thus, $N(D,p)$ must be $\{r,a\}$. To realize it, we check whether $a \in N([D,a,a], \downarrow :: b)$, that is, whether every subtree rooted by the a-labeled node closest to leaves in any tree conforming to D, satisfies $\downarrow :: b$.

The case that $p = \downarrow^* :: *(p_1,\ldots,p_n)$ is complicated compared to the other cases. At first, we give an example to get insight on this case. Consider $\downarrow^* :: *(\downarrow :: a, \downarrow^* :: b)$ and

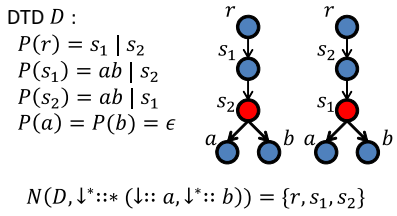

Figure 4: Computing $N(D, \downarrow^*::*(\downarrow::a, \downarrow^*::b))$

the DTD D shown in Fig 4. Then, by observing the meaning of D, we want $N(D,p)$ to be $\{r, s_1, s_2\}$. Note that for any label $\sigma \in \Sigma$, in any tree conforming to D, any node labeled with σ does not always have an a-labeled child, i.e., $N(D, \downarrow::a) = \emptyset$. Due to this, computing $N(D,p)$ in a similar fashion as the case that $p = \downarrow::*(p_1, \ldots, p_n)$ fails. That is, $N(D,p)$ is not the set of labels σ such that for any tree conforming to (D,σ), there is a node labeled with a label in $N(D, \downarrow::a) \cap N(D, \downarrow^*::b)$, which is the empty set. In this example, some node v labeled with s_1 or s_2 has both an a-labeled child and a b-labeled child in any tree, and v is reachable from any node labeled with r, s_1, or s_2. We observe that $N(D,p)$ is a subset of $N(D, \downarrow^*::b)$ and that for any $\sigma \in N(D,p)$, in any tree conforming to D, any node labeled with σ has some descendant which has a child labeled with a. Our algorithm first computes the set L_c of labels of location steps with child axis in $s(p)$, and the set N_d of labels σ such that each tree conforming to D satisfies any $p_i \in s(p) - sc(p)$, which has the root location step with descendant axis, relative to any σ-labeled node. Then, it computes maximum subset S of N_d such that in any tree conforming to D, any node labeled with any $\sigma \in S$ has some descendant which has a node labeled with each label in L_c.

4.3 Complexity

Let us consider the complexity of computing $N(D,p)$. To compute $N(D, \downarrow::a(p_1, \ldots, p_n))$, it is required to check whether every $w \in L(P(\sigma))$ has a. The check can be done in linear time to $|P(\sigma)|$ by using the syntax tree of $P(\sigma)$:

1. For each leaf of the syntax tree, if its label is a, then assign "T" to it; otherwise, assign "F".

2. In a bottom-up manner, assign a truth value to each internal node depending on the truth values of its children: if it is a "$\cdot$"-node, assign the disjunction of the truth values of its children; if it is a "$|$", assign the conjuction of the truth values of them; otherwise assign "F".

3. Return the truth value of the root.

For example, let $\sigma = ab|ca$, then we can see that a appears in any word in $L(\sigma)$ by assigning truth values to each node of the syntax tree of σ in a bottom-up manner as shown in Fig. 5(a). It requires linear time to the size of D to run this check for all $\sigma \in \Sigma$.

To compute $N(D, \downarrow::*(p_1, \ldots, p_n))$, we need to check whether every $w \in L(P(\sigma))$ has some label in N of size at most $|\Sigma|$. This check can be done in linear time in the same way as the above. Instead that only a-labeled leaves are assigned "T" in the first step, we just assign "T" to

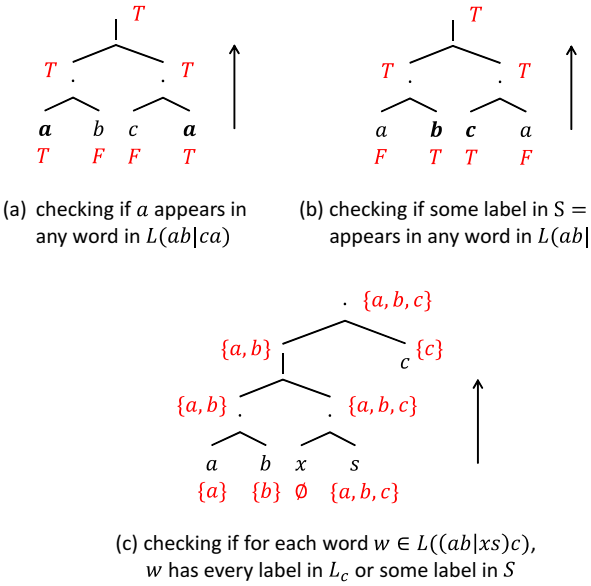

Figure 5: Checking on occurrences of labels in a content model

leaves labeled with a label in N. For example, let $\sigma = ab|ca$ and $N = \{b, c\}$, then we can see that some label in N appears in any word in $L(\sigma)$, as shown in Fig. 5(b). It also requires linear time to run this check for all $\sigma \in \Sigma$.

In computing $N(D, \downarrow^*::a(p_1, \ldots, p_n))$, it is necessary to decide whether the set of trees conforming to a given DTD is empty. The emptiness on a DTD can be determined in linear time to the size of the DTD [4]. In the emptiness check, the set of labels σ such that $TL((D,\sigma)) = \emptyset$ is also computed at the same time. Thus, $\{\sigma \in \Sigma \mid TL([D,a,\sigma]) = \emptyset\}$ can be computed in linear time.

In $N(D, \downarrow^*::*(p_1, \ldots, p_n))$, we need to check whether for each $w \in L(P(\sigma))$, w has every label in L_c or some label in S. This can be done as follows:

1. For each leaf of the syntax tree, if its label is in S, then assign L_c to it; if its label is in L_c, then assign the singleton including its label; otherwise, assign $\emptyset$.

2. In a bottom-up manner, assign a subset of L_c to each internal node depending on the sets of its children: if it is a "$\cdot$"-node, assign the union of the sets of its children; if it is a "$|$", assign the intersection of the sets; otherwise assign $\emptyset$.

3. Return the result whether the set of the root is L_c.

For example, Let $\sigma = (ab|xs)c$, $L_c = \{a, b, c\}$, and $S = \{s\}$. Then we can see that for each $w \in L(P(\sigma))$, w has every label in L_c or some label in S, by assigning a subset of L_c to each node of the syntax tree of σ in a bottom-up manner as shown in Fig. 5(c). This check for $\sigma \in \Sigma$ takes $O(|\Sigma||P(\sigma)|)$ time, and thus it needs $O(|\Sigma||D|)$ time to do the check for all $\sigma \in \Sigma$. To obtain the greatest fixed point of S, we just repeat the above procedure until S does not change. Thus, S can be computed in $O(|\Sigma|^2|D|)$ time in addition to time to compute L_c and N_d.

The number of inductive step is the size of $|p|$. Thus, $N(D,p)$ can be computed in $O(|\Sigma|^2|p||D|)$ time.

4.4 Correctness

Next, we show the correctness of our algorithm by giving the following lemma.

LEMMA 1. *Let* $D = (\Sigma, r, P)$ *and* $p \in \mathcal{X}(\downarrow, \downarrow^*, [\ ], *)^\dagger$. $r \in N(D,p)$ *if and only if* p *is valid under* D.

PROOF. We show that $\sigma \in N(D,p)$ if and only if p is valid under (D,σ) by induction on the structure of p.

- Let $p = \downarrow :: a(p_1, \ldots, p_n)$. Assume that $\sigma \in N(D,p)$. Then, $a \in \bigcap_{i=1}^n N(D,p_i)$ and every $w \in L(P(\sigma))$ has a. Consider any tree $T \in TL((D,\sigma))$. The root of T has some child v labeled with a, and hence, $T \models \downarrow :: a(r(T), v)$. By the inductive hypothesis, each $p_i \in s(p)$ is valid under (D,a). Since the subtree of T rooted by v conforms to (D,a), for each $p_i \in s(p)$ there is some v_i such that $T \models p_i(v, v_i)$. Thus, T satisfies p.

  On the other hand, assume that p is valid under (D,σ). Then, for each $T \in TL((D,\sigma))$, there is some node v such that $T \models \downarrow :: a(r(T), v)$, and for each $p_i \in s(p)$, there is some node v_i such that $T \models p_i(v, v_i)$. The former implies that the root of every $T \in TL((D,\sigma))$ has some child labeled with a, and therefore, every $w \in L(P(\sigma))$ must have a. Moreover, the latter implies that each $p_i \in s(p)$ is valid under (D,a), because for every $T' \in TL((D,a))$, there is a tree $T \in TL((D,\sigma))$ such that any subtree rooted by any node labeled with a is identical to T' and even such T satisfies p. By the inductive hypothesis, $a \in N(D,p_i)$ for each $p_i \in s(p)$. Thus, $\sigma \in N(D,p)$.

- Let $p = \downarrow :: *(p_1, \ldots, p_n)$. Assume that $\sigma \in N(D,p)$. Then, every $w \in L(P(\sigma))$ has some label in $\bigcap_{i=1}^n N(D,p_i)$. The root of every tree $T \in TL((D,\sigma))$ has some child v such that $\lambda(v) \in \bigcap_{i=1}^n N(D,p_i)$. From the inductive hypothesis, for any label σ' in $\bigcap_{i=1}^n N(D,p_i)$ and any $p_i \in s(p)$, p_i is valid under (D,σ'). Thus, p is valid under (D,σ).

  Assume that p is valid under (D,σ). Consider any tree $T \in TL((D,\sigma))$. We now show by contradiction that there is some v such that $T \models \downarrow :: *(r(T), v)$ and $\lambda(v) \in N(D,p_i)$ for each $p_i \in s(p)$. We assume that for each child v_j^c of $r(T)$, there is some $p_i \in s(p)$ such that $\lambda(v_j^c) \notin N(D,p_i)$. By the inductive hypothesis, for each child v_j^c of $r(T)$, there is some $p_i \in s(p)$ such that p_i is not valid under $(D, \lambda(v_j^c))$. Thus, for each child v_j^c of $r(T)$, there is some tree $T_j \in TL((D, \lambda(v_j^c)))$ and some $p_i \in sc(p)$ such that $T_j \not\models p_i(r(T_j), v_j')$ for any node v_j'. Consider a tree T' obtained from T by replacing the subtree rooted by each child v_j^c of $r(T)$ with T_j respectively. T' conforms to (D,σ) and T' does not satisfy p, which is a contradiction. Thus, we can say that there is some child v of $r(T)$ such that $\lambda(v) \in N(D,p_i)$ for each $p_i \in s(p)$. Therefore, $\sigma \in N(D,p)$.

- Let $p = \downarrow^* :: a(p_1, \ldots, p_n)$. Assume that $\sigma \in N(D,p)$. Then, $a \in \bigcap_{i=1}^n N([D,a,a], p_i)$ and $TL([D,a,\sigma]) = \emptyset$. Consider any tree $T \in TL((D,\sigma))$. Since $TL([D,a,\sigma]) = \emptyset$, T has at least one node labeled with a. Then, T has some descendant v labeled with a such that v has no proper descendant labeled with a. The subtree of T rooted by v conforms to $TL([D,a,a])$. By the inductive hypothesis, p_i is valid under $[D,a,a]$. Thus, T satisfies p.

  Assume that p is valid under (D,σ). Consider any $T \in TL((D,\sigma))$. There is some node v such that $T \models \downarrow^* :: a(r(T), v)$, and for each $p_i \in s(p)$, there is some node v_i such that $T \models p_i(v, v_i)$. Thus, $TL([D,a,\sigma]) = \emptyset$. Let v' be a descendant of v such that $\lambda(v') = a$ and v' has no proper descendant labeled with a. We show by contradiction that for each $p_i \in s(p)$, there is some node v_i' such that $T \models p_i(v', v_i')$. We assume that there is some $p_i \in s(p)$ such that $T \not\models p_i(v', v'')$ for any v''. Let T' be the subtree rooted by v', and let T'' be a tree obtained from T by replacing all the subtrees rooted by the nodes labeled with a with T'. Then, T'' conforms to (D,σ) but T'' does not satisfy p, which is a contradiction. Thus, for each $p_i \in s(p)$, there is some node v_i' such that $T \models p_i(v', v_i')$. The subtree rooted by v' conforms to $[D,a,a]$, and for each $T' \in TL([D,a,a])$, there is a tree $T'' \in TL((D,\sigma))$ such that any subtree rooted by any node labeled with a is T'. Therefore, each $p_i \in s(p)$ must be valid under $[D,a,a]$. By the inductive hypothesis, $a \in N([D,a,a], p_i)$ for each p_i. Thus, $\sigma \in N(D,p)$.

- Let $p = \downarrow^* :: *(p_1, \ldots, p_n)$ where for any $p_i \in s(p)$, p_i has no children if $rt(p_i) = \downarrow :: a$ for some $a \in \Sigma$, and $rt(p_i) \neq \downarrow :: *$. Assume that $\sigma \in N(D,p)$. Then, $\sigma \in S$. Consider any tree $T \in TL((D,\sigma))$. Since $\lambda(r(T)) = \sigma$ and $\sigma \in S$, the tree T has a node v such that $\lambda(v) \in S$ and no proper descendant of v has a label in S. Thus, we have that $T \models \downarrow^* :: *(r(T), v)$. Note that the subtree of T rooted by v conforms to $[D,S,\lambda(v)]$. Since $S \subseteq N_d$ and the inductive hypothesis, for each $p_j \in s(p) - sc(p)$, there is some node v_j such that $T \models p_j(v, v_j)$. Moreover, since v has no child labeled with any label in S, by the definition of S, every label in L_c appears as a label of a child of v. Thus, T satisfies p.

  We show that if $\sigma \notin S$, then p is not valid under (D,σ). Assume that $\sigma \notin S$. Then, $\sigma \notin N_d$ or $\sigma \in N_d - S$. If $\sigma \notin N_d$, then there is some $p_i \in s(p) - sc(p)$ such that $\sigma \notin N(D,p_i)$. By the inductive hypothesis, there is some tree $T' \in TL((D,\sigma))$ such that T' does not satisfy p_i. Since $rt(p_i)$ is in the form of $\downarrow^* :: \ell$, $T' \not\models p_i(v, v')$ for any two nodes v and v' of T'. Therefore, T' does not satisfy p and thus p is not valid under (D,σ). On the other hand, we assume that $\sigma \in N_d - S$. Then, we can construct a tree $T \in TL((D,\sigma))$ such that no node in T is labeled in S, and for any node v labeled with $\sigma' \in N_d - S$, some label in L_c does not appear as the label of any child of v. If the construction was impossible, then there would be some nonempty subset S' of $N_d - S$ such that for any $\sigma' \in S'$ and $w \in L(P(\sigma'))$, w has some label in $S \cup S'$ or every label in L_c. However, it violates the maximality of S that S does not include S', and thus T exists. Next, for any node v of T labeled with any $\sigma_o \notin N_d$, replace the subtree rooted by v with $T' \in TL((D,\sigma_o))$ such that T' does not satisfy some $p_i \in s(p) - sc(p)$. In addition, for any node v labeled with any label σ_c in $L_c - N_d$, replace the subtree rooted by v with $T_c \in TL([D,\{\sigma_c\},\sigma_c])$. The obtained tree T_f conforms to (D,σ) and T_f does not satisfy p. Thus, p is not valid under (D,σ). □

Finally, we obtain the following theorem.

THEOREM 3. $\mathsf{VLD}(\mathcal{X}(C))$ *is in PTIME where* C *consists of any three of* $\{\downarrow, \downarrow^*, *, [\,]\}$.

5. CONCLUSIONS

In this paper, we have discussed the validity problem of positive XPath queries with wildcard in the presence of DTDs. First, we have shown that the validity problem for the XPath class with path union, either of downward axes, and path concatenation is coNP-hard, and that even without path union, the validity problem of XPath queries with child and descendant-or-self axes, concatenation, qualifier, and wildcard is coNP-hard. Moreover, we have provided an XPath subclass which is a superclass of any class with three of child and descendant-or-self axes, concatenation qualifier, and wildcard, and shown that the validity problem for the subclass is in PTIME.

The tractability result in this paper is the XPath subclass with only downward axes, path concatenation, qualifier, and wildcard. However, we conjecture that the validity problem is still tractable even if sibling axes are incorporated to the known tractable class.

6. ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their insightful comments and suggestions. This research is supported in part by Grant-in-Aid for Scientific Research (C) 23500120 and Grant-in-Aid for Young Scientists (B) 22700099 from Japan Society for the Promotion of Science.

7. REFERENCES

[1] M. Benedikt, W. Fan, and F. Geerts. XPath satisfiability in the presence of DTDs. *Journal of the ACM*, 55(2), 2008.

[2] H. Björklund, W. Martens, and T. Schwentick. Optimizing conjunctive queries over trees using schema information. In *Proceedings of the 33rd international symposium on Mathematical Foundations of Computer Science*, pages 132–143, 2008.

[3] A. Deutsch and V. Tannen. Containment and integrity constraints for XPath. In *Proceedings of the 8th International Workshop on Knowledge Representation meets Databases*, 2001.

[4] W. Fan and L. Libkin. On XML integrity constraints in the presence of DTDs. *Journal of the ACM*, 49:368–406, 2002.

[5] D. Figueira. Satisfiability of downward XPath with data equality tests. In *Proceedings of the 28th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 197–206, 2009.

[6] F. Geerts and W. Fan. Satisfiability of XPath queries with sibling axes. In *Proceedings of the 10th International Symposium on Database Programming Languages*, pages 122–137, 2005.

[7] P. Genevès, N. Layaïda, and A. Schmitt. Efficient static analysis of XML paths and types. In *Proceedings of the ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation*, pages 342–351, 2007.

[8] Y. Ishihara, T. Morimoto, S. Shimizu, K. Hashimoto, and T. Fujiwara. A tractable subclass of DTDs for XPath satisfiability with sibling axes. In *Proceedings of the 12th International Symposium on Database Programming Languages*, pages 68–83, 2009.

[9] Y. Ishihara, S. Shimizu, and T. Fujiwara. Extending the tractability results on XPath satisfiability with sibling axes. In *Proceedings of the 7th International XML Database Symposium*, pages 33–47, 2010.

[10] L. Libkin and C. Sirangelo. Reasoning about XML with temporal logics and automata. *Journal of Applied Logic*, 8(2):210–232, 2010.

[11] G. Miklau and D. Suciu. Containment and equivalence for a fragment of XPath. *Journal of the ACM*, 51(1):2–45, 2004.

[12] F. Neven and T. Schwentick. On the complexity of XPath containment in the presence of disjunction, DTDs, and variables. *Logical Methods in Computer Science*, 2(3), 2006.

[13] T. Schwentick. XPath query containment. *SIGMOD Record*, 33(1):101–109, 2004.

[14] N. Suzuki and Y. Fukushima. Satisfiability of simple XPath fragments in the presence of DTD. In *Proceedings of the 11th International Workshop on Web Information and Data Management*, pages 15–22, 2009.

[15] B. ten Cate and C. Lutz. The complexity of query containment in expressive fragments of XPath 2.0. *Journal of the ACM*, 56(6), 2009.

[16] P. T. Wood. Containment for XPath fragments under DTD constraints. In *Proceedings of the 9th International Conference on Database Theory*, pages 297–311, 2003.