XPath Satisfiability with Parent Axes or Qualifiers
Is Tractable under Many of Real-World DTDs

Yasunori Ishihara
Osaka University
ishihara@lst.osaka-u.ac.jp

Nobutaka Suzuki
University of Tsukuba
nsuzuki@slis.tsukuba.ac.jp

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

Shogo Shimizu
Gakushuin Women’s College
shogo.shimizu@gakushuin.ac.jp

Toru Fujiwara
Osaka University
fujiwara@lst.osaka-u.ac.jp

Abstract
This paper aims at finding a subclass of DTDs that covers many of the real-world DTDs while offering a polynomial-time complexity for deciding the XPath satisfiability problem. In our previous work, we proposed RW-DTDs, which cover most of the real-world DTDs (26 out of 27 real-world DTDs and 1406 out of 1407 DTD rules). However, under RW-DTDs, XPath satisfiability with only child, descendant-or-self, and sibling axes is tractable.

In this paper, we propose MRW-DTDs, which are slightly smaller than RW-DTDs but have tractability on XPath satisfiability with parent axes or qualifiers. MRW-DTDs are a proper superclass of duplicate-free DTDs proposed by Montazerian et al., and cover 24 out of the 27 real-world DTDs and 1403 out of the 1407 DTD rules. Under MRW-DTDs, we show that XPath satisfiability problems with (1) child, parent, and sibling axes, and (2) child and sibling axes and qualifiers are both tractable, which are known to be intractable under RW-DTDs.

Categories and Subject Descriptors H.2.3 [Database Management]: Languages; F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems; H.2.4 [Database Management]: Systems

General Terms Algorithms, Languages, Theory

Keywords XPath, satisfiability, complexity

1. Introduction
XPath satisfiability 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 expression specifies a pattern of (possibly branching) paths from the root of a given XML document. The answer to an XPath expression 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 $v$. A given XPath expression $p$ is satisfiable under a given DTD (Document Type Definition) $D$ if there is an XML document $T$ conforming to $D$ such that the answer to $p$ for $T$ is a nonempty set.

One of the motivations for research on XPath satisfiability is query optimization. When (part of) an XPath expression is found unsatisfiable, we can always replace the expression with the empty set without evaluating it. Another motivation is to decide consistency and absolute consistency of XML schema mappings [1,13], which are desirable properties for realizing XML data exchange and integration. The decision problem of such properties can be reduced to XPath satisfiability problem.

Unfortunately, it is known that satisfiability under unrestricted DTDs is in P only for a very small subclass of XPath expressions, namely, XPath with only child axis, descendant-or-self axis, and path union [2,3]. To the best of our knowledge, two approaches have been adopted so far in order to resolve the intractability of XPath satisfiability. The approach adopted by Genevès and Layaida is to translate XPath expressions to formulas in modal second-order (MSO) logic [6] and in a variant of $\mu$-calculus [7,8]. Regular tree grammars [15], which are a general model of XML schemas and a proper superclass of DTDs, are also translated to such formulas. Then, satisfiability is verified by fast decision procedures for MSO and $\mu$-calculus formulas. The other approach is to find a tractable combination of XPath classes and DTD classes. For example, Lakshmanan et al. examined satisfiability under non-recursive DTDs [13], and Benedikt et al. investigated non-recursive and disjunction-free DTDs [2,3,5]. However, non-recursiveness does not broaden the tractable class of XPath. Disjunction-freeness definitively broadens the tractable class of XPath, but disjunction-free DTDs are too restricted from a practical point of view.

There are two successful results of the latter approach. The first one is duplicate-free DTDs [13], DF-DTDs for short, proposed by Montazerian et al. A DTD is duplicate-free if every tag name appears at most once in each content model (i.e., the body of each DTD rule). Table 1 shows an empirical survey of real-world DTDs. Many of the DTDs are selected according to the examination by Montazerian et al. [14], and several practical DTDs such as MathML and SVG are included in the examined DTDs. As shown in the table, 1386 out of 1407 real-world DTD rules are duplicate-free. Montazerian et al. also showed that satisfiability of XPath expressions with child axis and qualifiers is tractable [14]. Later, other several tractable XPath classes were presented in our previous work [16]. The tractability mainly stems from easiness of analyzing non-cooccurrence among tag names. More formally, a subexpression $e|e'$ of a content model specifies non-cooccurrence between the tag names in $e$ and those of $e'$. In DF-DTDs, each tag name can appear at most once in the content model, so complicated non-cooccurrence among tag names is not expressible.

The other successful result is disjunction-capsuled DTDs [9,11,12], DC-DTDs for short, and their extension DC*-DTDs [11]. A DTD is disjunction-capsuled if in each content model, every disjunction operator appears within a scope of a Kleene star operator. For example, $a(b|c)$ is DC but $(a|b)c$ is not. XPath expressions...
Table 1. The numbers of RW, MRW, DF, and DC+\# rules in real-world DTDs.

<table>
<thead>
<tr>
<th>DTD Name</th>
<th>Total</th>
<th>RW</th>
<th>MRW</th>
<th>DF</th>
<th>DC+#</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBLP</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>Ecoknowmics</td>
<td>224</td>
<td>224</td>
<td>223</td>
<td>223</td>
<td>222</td>
</tr>
<tr>
<td>LevelOne</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>26</td>
</tr>
<tr>
<td>MathML-2.0</td>
<td>181</td>
<td>181</td>
<td>181</td>
<td>181</td>
<td>181</td>
</tr>
<tr>
<td>Mondial</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>Music ML</td>
<td>12</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>News ML</td>
<td>118</td>
<td>118</td>
<td>118</td>
<td>118</td>
<td>118</td>
</tr>
<tr>
<td>Newspaper</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Opml</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>OSD</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>P3P-1.0</td>
<td>85</td>
<td>85</td>
<td>85</td>
<td>73</td>
<td>83</td>
</tr>
<tr>
<td>PSD</td>
<td>66</td>
<td>66</td>
<td>66</td>
<td>66</td>
<td>64</td>
</tr>
<tr>
<td>Reed</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>RSS</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>29</td>
</tr>
<tr>
<td>SigmodRecord</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>SimpleDoc</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td>SSMIL-1.0</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>SVG-1.1</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>78</td>
<td>77</td>
</tr>
<tr>
<td>TV-Schedule</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>VoiceXML-2.0</td>
<td>62</td>
<td>62</td>
<td>62</td>
<td>62</td>
<td>62</td>
</tr>
<tr>
<td>Xbel-1.0</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>XHTML1-strict</td>
<td>77</td>
<td>77</td>
<td>77</td>
<td>77</td>
<td>74</td>
</tr>
<tr>
<td>XMark DTD</td>
<td>77</td>
<td>77</td>
<td>77</td>
<td>77</td>
<td>76</td>
</tr>
<tr>
<td>XML Schema</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>25</td>
<td>20</td>
</tr>
<tr>
<td>XML Signature</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td>44</td>
<td>45</td>
</tr>
<tr>
<td>XMILTV</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>Yahoo</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>Total</td>
<td>1407</td>
<td>1406</td>
<td>1403</td>
<td>1386</td>
<td>1380</td>
</tr>
</tbody>
</table>

Figure 1. Relationship among DTD classes.

were supposed to consist of \( \downarrow \) (child axis), \( \downarrow^* \) (descendant-or-self axis), \( \uparrow \) (parent axis), \( \uparrow^* \) (ancestor-or-self axis), \( \rightarrow^+ \) (following-sibling axis), \( \leftarrow^+ \) (preceding-sibling axis), \( \cup \) (path union), and [ ] (qualifier). Then, it was shown that the satisfiability under DC-DTDs for XPath expressions without upward axes or qualifiers is tractable. The tractability is mainly from the fact that in DC-DTDs, any non-cooccurrence of tag names is a b ailed by the surrounding Kleene star operator. DC-DTDs were extended to \( DC^+\) by allowing operators \( \wedge \) (zero or one occurrence) and \( \vee \) (one or more occurrences) in a restricted manner, and then, to \( DC^+\). By allowing a new operator \# representing “either or both.” Precisely, \# is an \((m + l)\)-ary operator and \((a_1, \ldots, a_m)\#(b_1, \ldots, b_l)\) is equivalent to \(a_1 \cdot a_2 \cdot \ldots \cdot a_m \cdot b_1 \cdot \ldots \cdot b_l\) if \(a_1 \cdots a_m \cdot b_1 \cdots b_l\) is equivalent to \(a_1 \cdots a_m \cdot b_1 \cdots b_l\). Especially, \(a_1 \# a_2 \# \ldots \# a_m \# b\) means “either or both of a and b.” As shown in Table 1, 1380 out of 1407 real-world DTD rules are \( DC^+\). Amazingly, all the tractability of DC-DTDs is inherited by \( DC^+\). Although more than 98% of real-world DTD rules are DF or \( DC^+\), the ratio of DF-DTDs or \( DC^+\)-DTDs is not so high.

Table 1 shows that 8 out of the 27 DTDs are DF, 12 are not \( DC^+\), and 6 are neither DF nor \( DC^+\). To overcome this weakness, we proposed \( RW\)-DTDs\[11\], which are a proper superclass of both DF-DTDs and \( DC^+\)-DTDs. To be specific, RW-DTDs are not just the union of them, but a “hybrid” class of them. In each content model \( c = c_1 \cdots c_n \) of an RW-DTD, each subexpression \( c_i \) is either \( DC^+\) or DF in the whole content model. For example, \( a^* (b/c) a^* \) is neither DF nor \( DC^+\), but it is RW because the non-\( DC^+\) part \( b/c \) is DF in the whole content model. On the other hand, \( a^* (b/c) a^* \) is not RW because the non-\( DC^+\) part \( b/c \) contains \( b \), which appears twice in the whole content model. RW-DTDs cover 26 out of the 27 real-world DTDs, 1406 out of the 1407 DTD rules (see Table 1 again). However, RW-DTDs do not inherit all the tractability of the original DTD classes. Actually, XPath satisfiability with only child, descendant-or-self, and sibling axes is tractable under RW-DTDs.

This paper aims at finding a large subclass of RW-DTDs under which XPath satisfiability becomes tractable for a broader class of XPath expressions. The source of the intractability of XPath satisfiability under RW-DTDs seemed tag name occurrence of some fixed, plural number of times \[11\]. According to this observation, in this paper we propose \( MRW\)-DTDs\[1, 11\], which are RW-DTDs such that in each content model, each symbol appears in the scope of a repetitive operator (i.e., \( \ast \) or \( + \)) or DF in the whole content model. For example, \( a^\ast ba^\ast \) is MRW, but \( a^\ast ba \) is not MRW (although it is RW) because the rightmost \( a \) is not in the scope of any repetitive operator or DF in the whole content model. MRW-DTDs are still a proper superclass of DF-DTDs but incomparable to \( DC^+\)-DTDs (see Figure 1). MRW-DTDs cover 24 out of the 27 real-world DTDs, 1405 out of the 1407 DTD rules (see Table 1 again).

Next, this paper shows that under MRW-DTDs, XPath satisfiability problems with (1) child, parent, and sibling axes, and (2) child and sibling axes and qualifiers without disjunction (denoted [ ]), are both tractable. Table 2 summarizes the results of this paper and related works. Note that under RW-DTDs, satisfiability for child axes with either parent axes or qualifiers is known to be NP-complete \[11\]. Similarly to the case of RW-DTDs, the decision algorithm for XPath satisfiability under MRW-DTDs consists of the following two checks: (1) Check the satisfiability of a given XPath expression under the DTD obtained by replacing each disjunction with concatenation in a given MRW-DTD. In other words, satisfiability is analyzed as if the given MRW-DTD did not specify any non-cooccurrence of tag names; and (2) Check that the given XPath expression does not violate the non-cooccurrence specified by the original MRW-DTD. The first check can be done by the efficient algorithm for XPath satisfiability under DC-DTDs \[8, 10\]. To perform the second check, we have to keep track of sets of already-traversed sibling tag names and associate the sets with nodes of a tree structure. Since each tag name can appear at most once or unboundedly many times in MRW-DTDs, association of the sets to a tree structure is uniquely determined. That enables us an efficient satisfiability checking.

The rest of this paper is organized as follows. In Section 2 several preliminary definitions to formalize the XPath satisfiability
problem are provided. In Section 3, MRW-DTDs are proposed. The tractability results under MRW-DTDs are presented in Section 4. Section 5 summarizes the paper.

2. Preliminaries

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 \( \lambda \) to a function on sequences, i.e., for a sequence \( v_1 \cdots v_n \) of nodes, let \( \lambda(v_1) \cdots \lambda(v_n) = \lambda(v_1) \cdots \lambda(v_n) \). A tree is sometimes denoted by a term, e.g., \( a(b(c)) \) denotes a tree consisting of three nodes; the root has label \( a \), and its left and right children have labels \( b \) and \( c \), respectively. Attributes are not handled in this paper.

2.2 DTDs

A regular expression over an alphabet \( \Sigma \) consists of constants \( e \) (empty sequence) and the symbols in \( \Sigma \), and operators \( \cdot \) (concatenation), \( * \) (repetition), \( | \) (disjunction), \( ? \) (zero or one occurrence), \( + \) (one or more occurrences), and \# (either or both). Here, \# is an \((m + l)-ary\) operator and \( (a_1, \ldots, a_m) \cdot \#(b_1, \ldots, b_l) \) is equivalent to \( a_1 \cdots a_m b'_1 \cdots b'_l | a'_1 \cdots a'_m b_1 \cdots b_l \). We exclude \( \emptyset \) (empty set) because we are interested in only nonempty regular languages. The concatenation operator is often omitted as usual. The string language represented by a regular expression \( e \) is denoted by \( L(e) \).

A regular expression \( e \) is duplicate-free \( \text{DF} \) (DF for short) if every symbol in \( e \) appears only once. On the other hand, a regular expression \( e \) is \( \text{DC}^{+\#} \) \( \text{(13)} \) if \( e \) is in the form of \( e_1 e_2 \cdots e_n \) (\( n \geq 1 \)), where each \( e_i \) (\( 1 \leq i \leq n \)) is either

- a symbol in \( \Sigma \),
- in the form of \( (e'_i)^* \) for a regular expression \( e'_i \),
- in the form of \( (e'_i)^? \) for a \( \text{DC}^{+\#} \) regular expression \( e'_i \),
- in the form of \( (e'_i)^+ \) for a regular expression \( e'_i \), or
- in the form of \( (e''_1, \ldots, e''_m) \#(e''_1, \ldots, e''_n) \) for \( \text{DC}^{+\#} \) regular expressions \( e''_1, \ldots, e''_m \) and \( e''_1, \ldots, e''_n \).

\( \text{DC}^{+\#} \) regular expressions are intended to exclude any non-cooccurrence among symbols. The argument of operators \( * \) and \( + \) can be an arbitrary regular expression. Such operators can abolish any non-cooccurrence specified by their argument because the operators can repeat any subexpression of their argument arbitrary times. On the other hand, the argument of operators \( ? \) and \# must be a \( \text{DC}^{+\#} \) regular expression because the operators cannot repeat their argument. A \( \text{DC}^{+\#} \) regular expression \( e \) is disjunction-capsuled \( \text{(3)} \) (DC for short) if \( e \) does not contain \( ?, +, \text{ or } \# \).

The length of a DC regular expression \( e = e_1 e_2 \cdots e_n \) is defined as the number \( n \) of subexpressions of the top-level concatenation operator, and denoted by \( \text{len}(e) \). Moreover, \( i \) (\( 1 \leq i \leq \text{len}(e) \)) is called a position and each \( e_i \) is called the \( i \)-th subexpression of \( e \).

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

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

Regular expression \( P(a) \) is called the content model of label \( a \).

A duplicate-free DTD (DF-DTD for short) is a DTD such that \( P(a) \) is DF for every \( a \in \Sigma \). A disjunction-capsuled DTD (DC-DTD for short), is a DTD such that \( P(a) \) is DC for every \( a \in \Sigma \). A \( \text{DC}^{+\#} \) DTD is a DTD such that \( P(a) \) is \( \text{DC}^{+\#} \) for every \( a \in \Sigma \).

**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 children 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 \( D \).

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 \( a \).

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.

2.3 XPath expressions

The syntax of an XPath expression \( p \) is defined as follows:

\[
p \ ::= \; \chi \; ::= \; l \mid p/p \mid p \cup p \mid p[q],
\]

\[
\chi \ ::= \; \downarrow | \uparrow | \downarrow^* | \uparrow^* | \cdot \oplus \mid \leftarrow^+,
\]

\[
q \ ::= \; p \amp q \mid q \lor q,
\]

where \( l \in \Sigma \). Each \( \chi \) is in \( \{l, \uparrow, \downarrow, \downarrow^*, \uparrow^*, \cdot \oplus, \leftarrow^+ \} \) and is called an axis. Also, a subexpression in the form of \( [q] \) is called a qualifier. An expression in the form of \( \chi :: l \) is said to be atomic. The size of an XPath expression \( p \) is defined as the number of atomic subexpressions in \( p \).
The semantics of an XPath expression over a tree $T$ is defined as follows, where $p$ and $q$ are regarded as binary and unary predicates on paths from the root node of $T$; respectively. In what follows, $v_0$ denotes the root of $T$, and $v$ and $v'$ denote nodes of $T$. Also, $w$, $w'$, and $w''$ are nonempty sequences of nodes of $T$ starting by $v_0$, unless otherwise stated.

- $T \models (\mathbin{\downarrow}: l)(w, w')$ if path $w'$ exists in $T$ and $\lambda(v') = l$.
- $T \models (\mathbin{\uparrow}: l)(w, w)$ if path $w$ exists in $T$ and the label of the last node of $w$ is $l$.
- $T \models (\mathbin{\downarrow}: l)(w, w', w'')$ if path $ww'$ exists in $T$ and the label of the last node of $w'w''$ is $l$, where $w''$ is a possibly empty sequence of nodes of $T$.
- $T \models (\mathbin{\uparrow}: l)(w, w', w'')$ if path $ww'$ exists in $T$ and the label of the last node of $w'w''$ is $l$, where $w'$ is a possibly empty sequence of nodes of $T$.
- $T \models (\mathbin{\downarrow}+: l)(w, w', w'')$ if paths $w$ and $w'$ exist in $T$, $v'$ is a following sibling of $v$, and $\lambda(v') = l$.
- $T \models (\mathbin{\downarrow}:+ l)(w, w', w'')$ if paths $w$ and $w'$ exist in $T$, $v'$ is a preceding sibling of $v$, and $\lambda(v') = l$.
- $T \models (p \lor q)(w, w')$ if there is $w''$ such that $T \models p(w, w'')$ and $T \models q(w')$.
- $T \models (p|q)(w, w')$ if $T \models p(w, w')$ and $T \models q(w')$.
- $T \models p(w)$ if there is $w''$ such that $T \models p(w, w'')$.
- $T \models (q \land q')(w)$ if $T \models q(w)$ and $T \models q'(w)$.
- $T \models (q \lor q')(w)$ if $T \models q(w)$ or $T \models q'(w)$.

A tree $T$ satisfies an XPath expression $p$ if there is a node $v$ such that $T \models p(v_0, v)$, where $v_0$ is the root node of $T$. An XPath expression $p$ is satisfiable under a DTD $D$ if some $T \in TL(D)$ satisfies $p$.

In this paper, we often consider qualifiers without disjunction. In this case the syntax of $p$ is simply redefined as

$$p \ ::= \ X : l \mid p/p \mid p \lor p \mid p[p].$$

Note that conjunction can be represented by a sequence of qualifiers (e.g., $p[p] \land p'$ can be represented by $p(p[p])$).

Following the notation of [2,3], a subclass of XPath is indexed by $X(\cdot)$. For example, the subclass with child axes and qualifiers without disjunction is denoted by $X(\downarrow, \downarrow] \land)$.

## 3. Modeling Many of Real-World DTDs

In this section, we introduce MRW-DTDs, which are a subclass of RW-DTDs [11].

RW-DTDs are defined as a hybrid class of DF and DC$^{+\#}$-DTDs. Formally, a regular expression $e$ is RW if $e$ is in the form of $e_1 e_2 \cdots e_n (n \geq 1)$, where each $e_i (1 \leq i \leq n)$ is either

- $\text{DC}^{+\#};$
- or
- a regular expression consisting of only symbols from $X$ appearing once in $e$.

A DTD $D$ is called an RW-DTD if each content model of $D$ is RW. Although RW-DTDs cover most of real-world DTDs, it is shown that satisfiabilities of $X(\downarrow, \downarrow]$ and $X(\downarrow, \downarrow] \land)$ under RW-DTDs are both NP-complete [11]. This intractability is caused by non-repetitive symbols (i.e., appearing outside the scope of any $*$ and $+$ operators) in a DC$^{+\#}$ part, which raise a combinatorial explosion. To handle this problem, we define a slightly restricted version of RW-DTDs, denoted MRW-DTDs, in which non-repetitive symbols must appear at most once in each context model.

### Table 3. The forms of the content models of the 4 rules that are not MRW.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(F1)</td>
<td>$a^*b^*c$ (1 rule in Ecoknowmics)</td>
</tr>
<tr>
<td>(F2)</td>
<td>$a</td>
</tr>
<tr>
<td>(F3)</td>
<td>$(a</td>
</tr>
</tbody>
</table>

**Definition 3.** An RW-DTD $D = (\Sigma, r, P)$ is called an MRW-DTD if for each content model $e$ and each symbol appearing in $e$, $a$ appears once in $e$ whenever $a$ is outside the scope of any $*$ and $+$.

**Example 1.** Let $D = (\{r, a, b, c\}, r, P)$ be a DTD where $P(r) = (a|b)^*ca^+$, $P(a) = P(b) = P(c) = e$.

Then $D$ is an MRW-DTD. On the other hand, consider a DTD $D' = (\{r, a, b, c\}, r, P')$, where $P'(r) = (a|b)^*ca^+$, $P'(a) = P'(b) = P'(c) = e$.

Then $D'$ is RW but not MRW since in $P'(r)$ symbol $a$ occurs twice but one of them appears outside the scope of $*$ and $+$.

We examined 27 real-world DTDs, 1407 rules (see Table 1). During the examination, we found 6 DTD rules which are not syntactically MRW but can be transformed into equivalent MRW rules. Specifically, the content models of the rules have the following forms:

- $ab^*c$ (1 rule in Music ML),
- $a^*(bc^g\ast d^a\ast a^\ast\ast\ast d\ast a)\ast$ (1 rule in P3P-1.0),
- $a^*b^*(cdef^\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\as
Next, we provide a necessary and sufficient condition for satisfiability of XPath expressions in $\mathcal{X}(\downarrow, \downarrow^*, \uparrow, \rightarrow^*, \leftarrow^*, \cup, \{, \})$ under MDF/DC-DTDs. Similarly to our previous work\cite{[A]}, we introduce a schema graph of a given MDF/DC-DTD, which represents parent-child relationship as well as the possible positions of the children specified by the MDF/DC-DTD. Then we define a satisfaction relation between schema graphs and XPath expressions.

After that, we propose efficient algorithms for deciding the satisfaction relation for two cases, namely, $p \in \mathcal{X}(\downarrow, \downarrow^*, \uparrow, \rightarrow^*, \leftarrow^*, \cup, \{, \})$. The decision algorithms consist of the following two checks: (1) Check the satisfiability of $p$ under the DC-DTD obtained by replacing disjunction with concatenation in a given MDF/DC-DTD; and (2) Check that $p$ does not violate the non-cooccurrence specified by the original MDF/DC-DTD. Actually, the satisfaction relation is defined so that both of the checks can be done simultaneously.

### 4.1 Tractability identicalness between MRW-DTDs and MDF/DC-DTDs

First, let us review satisfaction preservation relation $\sim$ discussed in\cite{[B]}. Let $e$ and $e'$ be regular expressions. We write $e \sim e'$ if they satisfy the following two conditions:

- every $w \in L(e)$ is a subsequence (i.e., can be obtained by deleting zero or more symbols) of some $w' \in L(e')$; and
- every $w' \in L(e')$ is a subsequence of some $w \in L(e)$.

Let $D = (\Sigma, r, P)$ and $D' = (\Sigma, r, P')$. We write $D \sim D'$ if $P(a) \sim P'(a)$ for each $a \in \Sigma$. Since DTDs are assumed to have no useless symbols, $D \sim D'$ implies that

- every $T \in TL(D)$ can be obtained by deleting zero or more subtrees of some $T' \in TL(D')$; and
- every $T' \in TL(D')$ can be obtained by deleting zero or more subtrees of some $T \in TL(D)$.

Let $p \in \mathcal{X}(\downarrow, \downarrow^*, \uparrow, \rightarrow^*, \leftarrow^*, \cup, \{, \})$ and suppose that $D \sim D'$. Then, $p$ is satisfiable under $D$ if and only if $p$ is satisfiable under $D'$, because our XPath class is passive (i.e., does not contain negation operator) and not sensitive to next sibling (i.e., cannot detect existence of nodes between two sibling nodes). Thus, we have the following theorem:

**Theorem 1** (\cite{[B]}). Suppose that classes $C$ and $C'$ of DTDs satisfy the following property: for each DTD $D' \in C'$, there exists $D \in C$ such that $D \sim D'$ and $D$ can be computed efficiently from $D'$. Then, for any subclass $X$ of $\mathcal{X}(\downarrow, \downarrow^*, \uparrow, \rightarrow^*, \leftarrow^*, \cup, \{, \})$, if the satisfiability problem for $X$ under $C$ is in $P$, the same problem under $C'$ is also in $P$.

To apply Theorem 1 to MRW-DTDs, we introduce the following mapping $\delta$:

- $\delta(\epsilon) = \epsilon$,
- $\delta(a) = a$ for each $a \in \Sigma$,
- $\delta(e_1 \cdot e_2) = \delta(e_1) \cdot \delta(e_2)$,
- $\delta(e^*) = (\delta(e))^*$,
- $\delta(e_1|e_2) = \delta(e_1)\delta(e_2)$,
- $\delta(e^+) = (\delta(e))^+$, and
- $\delta((e_1 \ldots e_m)\#(e_21, \ldots, e_{22})) = \delta(e_1) \cdot \delta(e_{1m}) \cdot \delta(e_{21}) \cdot \delta(e_{22})$.

Intuitively, $\delta$ removes all the $\#$ operators, and replaces all the $+$ and $\#$ operators with $*$ and $\cdot$ operators, respectively. For example, $\delta(a^*b(c#d))a^*$ = $a^*bda^*$. The next lemma is almost immediate:

**Lemma 1.** For any content model $e$ of an MRW-DTD, $\delta(e)$ is MDF/DC.

Moreover, $\delta$ preserves satisfiability:

**Lemma 2** (\cite{[B]}). $e \sim \delta(e)$ for any regular expression $e$.

For a DTD $D = (\Sigma, r, P)$, let $\delta(D)$ denote the DTD $(\Sigma, r, \delta(P))$, where $\delta(P(a)) = \delta(P'(a))$ for each $a \in \Sigma$. By the above lemmas, we have $D \sim \delta(D)$, and obviously $\delta(D)$ can be computed efficiently from $D$. Moreover, $\delta(D)$ is MDF/DC if $D$ is an MRW-DTD. Hence, from Theorem 1 and the fact that MDF/DC-DTDs are a subclass of MRW-DTDs, we have the following corollary:

**Corollary 1.** For any subclass $X$ of $\mathcal{X}(\downarrow, \downarrow^*, \uparrow, \rightarrow^*, \leftarrow^*$, $\cup, \{, \})$, the satisfiability problem for $X$ under MRW-DTDs is in $P$ if and only if the same problem under MDF/DC-DTDs is in $P$.

### 4.2 Schema graphs and sibling-constraint mappings

First, we introduce schema graphs. Let $D$ be an MDF/DC-DTD. Let $D_{DC}$ denote the DC-DTD obtained by replacing every disjunction operator appearing outside of any Kleene stars in a content model with the concatenation operator. For example, a content model $(a|b|d)^*e^*$ in $D$ is replaced with $(a^*b|d)\#e^*$ in $D_{DC}$. Then, the schema graph of $D$ is defined as that of $D_{DC}$.

**Definition 4.** The schema graph $G = (U, E)$ of a DC-DTD $D_{DC} = (\Sigma, r, P)$ is a directed graph defined as follows:

- A node $u \in U$ is either
  - $(\downarrow, 1, \ldots, r)$, where $\downarrow$ is a new symbol not in $\Sigma$, or
  - $(a, i, \omega, b)$, where $a, b \in \Sigma, 1 \leq i \leq \text{len}(P(a))$ such that $b$ appears in the $i$-th subexpression $e_i$ of $P(a)$, and $\omega = "\cdots"$ if $e_i$ is a single symbol in $\Sigma$ and $\omega = "\cdots"$ otherwise.
- An edge from $u$ to $u'$ exists in $E$ if and only if $\lambda(u) = \lambda_{par}(u')$.

The schema graph of an MDF/DC-DTD $D$ is that of the corresponding DC-DTD $D_{DC}$.

**Example 2.** Let $D = ((a, b, c), \lambda, \omega)$ be an MDF/DC-DTD, where

$P(r) = r = \lambda^*a^*b^*c^*$, $P(a) = \epsilon$, $P(b) = \lambda$, $P(c) = \epsilon$.

Then, the corresponding DC-DTD $D_{DC} = (a, b, c, \lambda, \omega)$ is as follows:

$P_{DC}(r) = \lambda^*a^*b^*c^*$, $P_{DC}(a) = \epsilon$, $P_{DC}(b) = \lambda$, $P_{DC}(c) = \epsilon$.

The schema graph $G$ of $D$ and $D_{DC}$ is shown in Figure 2.

Suppose that $T_{DC} \in TL(D_{DC})$ for a DC-DTD $D_{DC}$. As stated in\cite{[C]}, there exists a mapping $\theta$, called an $SG$ mapping of $T_{DC}$, from the set of nodes of $T_{DC}$ to the set of nodes of the schema graph of $D_{DC}$ with the following properties:

- $\theta$ maps the root node of $T_{DC}$ to $(\downarrow, 1, \ldots, r)$.
- Let $v$ be a node of $T_{DC}$ and $e_1 \cdots e_v$ be the children sequence of $v$. Then, $\theta(v) = (\lambda(v), i_j, \omega(v), \lambda(v))$, where $1 \leq i_j \leq \text{len}(P(\lambda(v)))$, $\omega(v) = "\cdots"$ if the $i_j$-th subexpression of $P(\lambda(v))$ is a single symbol in $\Sigma$ and $\omega_j = "\cdots"$ otherwise, and
such that $T$ is a sibling-constraint mapping. A pair $(T, \theta)$ is called a unique SG mapping.

$|\text{sequences of nodes, i.e., } \theta|\text{ and } e|\text{ symbol is }\beta|\text{ and }\eta|\text{ is a unique SG mapping }\\beta|\text{ consists of only DFS nodes of }T|\text{ and a tree }\eta|\text{ in }\beta|\text{ is a prefix of the "current path" of the analysis. The last node of the path }\eta|\text{ satisfies }\beta|\text{ and }\eta|\text{ is DC. Moreover, a DF symbol is DFS if it is outside the scope of }\eta|\text{. For example, consider }\eta|\text{ as DFS nodes of schema graphs in a similar way. A path }\eta|\text{ on a schema graph }G|\text{ is DFS if }\eta|\text{ consists of only DFS nodes of }G|\text{ (we regard }\eta|\text{, }\eta|\text{, and }\eta|\text{ as DFS).}

Example 3. Consider the MDF/DC-DTD $D$ defined in Example 2 and a tree $T \in TL(D)$ shown in Figure 3. In this case, there is a unique SG mapping $\theta$ of $T$, which is also shown in Figure 3.

Definition 5. A sibling-constraint mapping $\beta$ is a partial mapping from non-empty paths from $(\bot, 1, -, r)$ on $G = (U, E)$ to the powerset of $U$ such that

1. $\beta(s)$ is defined only for a finite number of $s$; and
2. if defined, $\beta(s)$ is a set of DF children of the last node of $s$.

We write $\beta \supseteq \beta'$ if $\beta(s) \supseteq \beta'(s)$ whenever $\beta'(s)$ is defined. Let $\beta \cup \beta'$ denote the least upper bound of $\beta$ and $\beta'$ with respect to $\supseteq$.

$|T|\text{ be a tree in }TL(D)|\text{ and }\theta|\text{ be its SG mapping. Let }\beta|\text{ be a sibling-constraint mapping. A pair }\eta|\text{ (}T|, \theta)|\text{ satisfies }\beta|\text{ if for each }s|\text{ such that }\beta(s)|\text{ is defined, there is a path }\eta|\text{ on }T|\text{ such that }\theta(\eta) = s|\text{ and }\beta(s) \subseteq \theta(SibDF_T(\eta))|\text{, where}

$SibDF_T(\eta) = \{v' | v' \text{ is a child of } v \text{ in } T\}$

such that $\lambda(v')$ is DF in $P(\lambda(v))|\text{.}

Definition 6. A sibling-constraint mapping $\beta$ is consistent if, for each path $s \cdot u$ on $G$ such that $\beta(s \cdot u)$ is defined, there exists a string in $L(P(\lambda(u)))$ that contains all $\lambda(u')'s$ where $u' \in \beta(s \cdot u)$.

It is not difficult to see that $\beta$ is consistent if and only if there are a tree $T \in TL(D)$ and its SG mapping $\theta$ such that $(T, \theta)$ satisfies $\beta$.

Example 4. Consider the schema graph $G$ in Figure 2 and let $\beta$ be the following sibling-constraint mapping: $\beta(\eta_0) = \{\eta_2, \eta_3\}$, $\beta(\eta_0, \eta_2) = \emptyset$, and $\beta(\eta_0, \eta_3) = \{\eta_6\}$. Then, $\beta$ is consistent. Actually, $(T, \theta)$ shown in Figure 3 satisfies $\beta$.

Next, consider $\beta' = \beta \cup \{\eta_0 \rightarrow \{\eta_1\}\}$. In this case, $\beta'(\eta_0) = \{\eta_2, \eta_3, \eta_4\}$ and $\beta'$ is not consistent because there is no string in $L(P(\lambda(\eta_0)))$ (i.e., $L(\{\eta' | \eta' \text{ is }\text{a DC-DTD}\})$) which contains all of $\lambda(\eta_2)|\text{, }\lambda(\eta_3)$, and $\lambda(\eta_4)$ (i.e., $a$, $b$, and $c$).

4.3 A necessary and sufficient condition for XPath satisfiability

We define a satisfaction relation $|=_{MDF/DC}$ between schema graphs and XPath expressions. Then, we show that $|=_{MDF/DC}$ coincides with XPath satisfiability under MDF/DC-DTDs.

In our previous work [10], we provided a satisfaction relation $|=_{DC}$ between schema graphs and XPath expressions, and showed that $|=_{DC}$ coincides with XPath satisfiability under DC-DTDs. More precisely, we showed that for any XPath expression $p \in X(\bot, \bot, \bot^*, \rightarrow^*, \leftarrow^*, \cup, \{\}), T |\text{ satisfies }\beta$|\text{ if and only if }G |\text{ satisfies }\eta|\text{ and }\eta|\text{ is an SG mapping of }T|\text{.}

Now, our target is MDF/DC-DTDs, so we have to analyze non-coocurrence specified by MDF/DC-DTDs. To do so, we augment the parameters of $p$ by sibling-constraint mappings introduced in the previous section. That is, we will define $|=_{MDF/DC}$ so that, roughly speaking, $G |\text{ satisfies }\eta|\text{, where }\theta|\text{ is an SG mapping of }T|\text{.}

Actually, it is not necessary to keep all sibling-constraint information. Only the following cases must be handled by $\beta(s)$:

- The case where $s$ is DFS. Then, for any $T \in TL(D)$, $\lambda(s)$ is a unique label path on $T$ if exists. Hence, the last node of the path can be visited many times. So, sibling-constraint information $\beta(s)$ at $s$ must be maintained.
- The case where $s$ is a prefix of the “current path” of the analysis. The last node of the “current path” can be considered as the context node. By using upward axes from the context node, any ancestor node may be revisited. So, sibling-constraint information $\beta(s)$ at $s$ must be maintained.

On the other hand, if $s$ does not meet the two cases above, $s$ contains a node inside the scope of some $\eta$. There is no way to always revisit the last node of $s$ in our XPath class, sibling-constraint information $\beta(s)$ at such $s$ does not have to be maintained.

We provide the formal definition of $|=_{MDF/DC}$. In what follows, let $u, u', \ldots, \text{ be nodes of }G$ and let $s, s', \ldots, \text{ be nonempty sequences of nodes of }G$ starting by $(\bot, 1, -, r)$, unless otherwise
stated. We introduce the following notations for readability:

\[
\psi(u) = \begin{cases} 
\{u\} & \text{if } u \text{ is DF,} \\
\emptyset & \text{otherwise,} 
\end{cases} 
\]

\[
\beta|_{DFS,s}(s') = \begin{cases} 
\beta(s') & \text{if } s' \text{ is DFS or} \\
\text{a proper prefix of } s, \\
\text{undefined otherwise.} 
\end{cases} 
\]

**Definition 7.** A satisfaction relation \( \models_{MF/DC} \) between a schema graph \( G \) and an XPath expression \( p \in X(\{\downarrow, \uparrow, \downarrow^*, \rightarrow^+, \leftarrow^+, \text{ and } \underbar{\downarrow}, \underbar{\uparrow}, \underbar{\downarrow^*}, \rightarrow^-, \leftarrow^-, \ulcorner, \urcorner \}) \) is defined as follows:

- \( G \models_{MF/DC} (l::l)((s, \beta), (s', \beta')) \) if
  - path \( su \) exists in \( G \),
  - \( \lambda(u') = l \),
  - \( \beta = \beta|_{DFS,s} \),
  - \( \beta' = \beta \cup \{s \rightarrow \psi(u')\} \), and
  - both \( \beta \) and \( \beta' \) are consistent.

- \( G \models_{MF/DC} (t::l)((s, \beta), (s', \beta')) \) if
  - path \( ss' \) exists in \( G \), where \( s' \) is a possibly empty sequence of nodes of \( G \),
  - the label of the last node of \( ss' \) is \( l \),
  - \( \beta = \beta|_{DFS,s} \),
  - \( \beta' = \beta|_{DFS,s} \), and
  - both \( \beta \) and \( \beta' \) are consistent.

- \( G \models_{MF/DC} (l::l)((s, \beta), (s', \beta')) \) if
  - path \( ss' \) exists in \( G \), where \( s' \) is a possibly empty sequence of nodes of \( G \),
  - the label of the last node of \( ss' \) is \( l \),
  - \( \beta = \beta|_{DFS,s} \),
  - \( \beta' = \beta|_{DFS,s} \),
  - both \( \beta \) and \( \beta' \) are consistent.

- \( G \models_{MF/DC} (t::l)((s, \beta), (s', \beta')) \) if
  - path \( ss' \) exists in \( G \), where \( s' \) is a possibly empty sequence of nodes of \( G \),
  - the label of the last node of \( ss' \) is \( l \),
  - \( \beta = \beta|_{DFS,s} \),
  - \( \beta' = \beta|_{DFS,s} \), and
  - both \( \beta \) and \( \beta' \) are consistent.

- \( G \models_{MF/DC} (l::l)((s, \beta), (s', \beta')) \) if
  - path \( ss' \) exists in \( G \), where \( s' \) is a possibly empty sequence of nodes of \( G \),
  - the label of the last node of \( ss' \) is \( l \),
  - \( \beta = \beta|_{DFS,s} \),
  - \( \beta' = \beta|_{DFS,s} \), and
  - both \( \beta \) and \( \beta' \) are consistent.

The following lemmas can be shown immediately from the definition of \( \models_{MF/DC} \):

**Lemma 3.** If \( G \models_{MF/DC} p(s, \beta), (s', \beta') \), then \( \beta|_{DFS,s} = \beta \) and \( \beta|_{DFS,s'} = \beta' \). If \( G \models_{MF/DC} q(s, \beta) \), then \( \beta|_{DFS,s} = \beta \).

**Lemma 4.** Suppose that \( G \models_{MF/DC} p((s, \beta), (s', \beta')) \). If \( \beta(s') \) is defined for a DFS path \( s' \), then \( \beta(s') \subseteq \beta(s) \).

Now, we show that XPath expression \( p \in X(\{\downarrow, \uparrow, \downarrow^*, \rightarrow^+, \leftarrow^+, \text{ and } \underbar{\downarrow}, \underbar{\uparrow}, \underbar{\downarrow^*}, \rightarrow^-, \leftarrow^-, \ulcorner, \urcorner \}) \) is satisfiable under \( D \) if and only if \( G \models_{MF/DC} p((\downarrow, 1, -r), \beta|_\downarrow)) \) for some \( s' \) and \( \beta' \), where \( \beta_\downarrow \) is a mapping undefined everywhere. The following theorem corresponds to the only part if:

**Theorem 2.** Let \( p \in X(\{\downarrow, \uparrow, \downarrow^*, \rightarrow^+, \leftarrow^+, \underbar{\downarrow}, \underbar{\uparrow}, \underbar{\downarrow^*}, \rightarrow^-, \underbar{\rightarrow^-}, \ulcorner, \urcorner \}) \). Let \( D \) be an MF/DC-DTD and \( G \) be the schema graph of \( D \).

1. Suppose that \( T \models p(w, w') \) for some \( T \in TL(D) \) with an SG mapping \( 0 \). Let \( \beta \) be an arbitrary mapping satisfied by \( (T, \theta) \) such that \( \beta = \beta|_{DFS,s(w)} \). Then, there is a mapping \( \beta |_{DFS,s(w)} \) such that \( G \models_{MF/DC} p((\theta(w), \beta), (\theta(w'), \beta')) \).

2. Suppose that \( T \models q(w) \) for some \( T \in TL(D) \) with an SG mapping \( 0 \). Then, there is a mapping \( \beta |_{DFS,s(w)} \) such that \( G \models_{MF/DC} q((\theta(w), \beta), (\theta(w'), \beta')) \).

**Proof Sketch.** The theorem is proved by induction on the structure of \( p \).

**Basis.** Suppose that \( T \models (l::l)(w, w') \). Let \( \beta = \beta|_{DFS,s(w)} \) be any mapping satisfying \( \beta \). Then \( (T, \theta) \) satisfies \( T \) such that \( \beta|_{DFS,s(w)} \) is undefined since \( \beta = \beta|_{DFS,s(w)} \). If \( (T, \theta) \) is DFS, then \( \beta|_{DFS,s(w)} \) does not violate the non-cooccurrence because \( (T, \theta) \) satisfies \( \beta \) and path \( w' \) exists in \( T \). Hence, \( (T, \theta) \) also satisfies \( \beta = \beta|_{DFS,s(w)} \).

**Induction.** Suppose that \( T \models p([q])([w, w']) \) and \( (T, \theta) \) satisfies \( \beta \). By the definition of qualifiers, \( T \models [q(w', w)] \) and \( T \models [q(w)] \). Let \( \beta \) be an arbitrary mapping satisfying \( (T, \theta) \) such that \( \beta = \beta|_{DFS,s(w)} \). By inductive hypothesis, there are mappings \( \beta' \) and \( \beta'' \) satisfied by \( (T, \theta) \) such that \( G \models_{MF/DC} p((\theta(w), \beta), (\theta(w'), \beta')) \), and \( G \models_{MF/DC} q((\theta(w'), \beta''), \beta') \).

Moreover, by Lemma 3 we have \( \beta' = \beta|_{DFS,s(w)} \) and \( \beta'' = \beta|_{DFS,s(w')} \), and hence \( \beta' \cup \beta'' \) is satisfied by \( (T, \theta) \). This means that \( \beta' \cup \beta'' \) is consistent, and therefore, \( G \models_{MF/DC} p((\theta(w), \beta), (\theta(w'), \beta'')) \).

**The other cases are similarly proved.**

The if part is shown below:

**Theorem 3.** Let \( p \in X(\{\downarrow, \uparrow, \downarrow^*, \rightarrow^+, \leftarrow^+, \text{ and } \underbar{\downarrow}, \underbar{\uparrow}, \underbar{\downarrow^*}, \rightarrow^-, \leftarrow^-, \ulcorner, \urcorner \}) \). Let \( D \) be an MF/DC-DTD.

1. Suppose that \( G \models_{MF/DC} p((s, \beta), (s', \beta')) \). Then, there are \( T \in TL(D) \), its SG mapping \( 0 \), and paths \( w \) and \( w' \) on \( T \) such that \( \theta(w) = s, \theta(w') = s', \beta \) and \( \beta' \) are satisfied by \( (T, \theta) \), and \( T \models p(w, w') \).
In this section, we show that the necessary and sufficient condition is decidable in polynomial time if \( p \in X(\downarrow, \uparrow, \rightarrow^+, \leftarrow^+) \) or \( p \in X(\downarrow, \uparrow, \rightarrow^+, \leftarrow^+ \setminus \{\lambda\}) \).

4.4 Tractability

In this section, we show that the necessary and sufficient condition is decidable in polynomial time if \( p \in X(\downarrow, \uparrow, \rightarrow^+, \leftarrow^+) \) or \( p \in X(\downarrow, \uparrow, \rightarrow^+, \leftarrow^+ \setminus \{\lambda\}) \).

4.4.1 \( X(\downarrow, \uparrow, \rightarrow^+, \leftarrow^+) \)

Let \( p \in X(\downarrow, \uparrow, \rightarrow^+, \leftarrow^+) \). We show an efficient algorithm for deciding whether \( G \models \text{MDF/DC} p((s, \beta), (s', \beta')) \) or \( G \models \text{MDF/DC} p(s, \beta) \) for some \( s' \) and \( \beta' \).

Essentially, our algorithm \( \text{eval}_1 \) runs in a top-down manner with respect to the parse tree of \( p \), and computes the set of the second parameters \((s', \beta')\) of \( p \) for a given set of first parameters \((s, \beta)\). Let \( B \) denote a set of pairs of a path on \( G \) and a sibling-constraint mapping. Formally, \( \text{eval}_1 \) is defined as follows:

\[
\text{eval}_1(p, B) = \begin{cases} 
\{(s', \beta') : G \models \text{MDF/DC} p((s, \beta), (s', \beta')) \} & \text{if } p \text{ is atomic,} \\
\text{eval}_1(p_2, \text{eval}_1(p_1, B)) & \text{if } p = p_1 \cdot p_2.
\end{cases}
\]

In what follows, we show that \( \text{eval}_1(p, \{(\downarrow, 1, -r), (\beta, \lambda)\}) \) runs in a polynomial time.

First, given \( s, s' \), and \( \beta, \beta' \), there is at most one \( \beta' \) such that \( G \models \text{MDF/DC} p((s, \beta), (s', \beta')) \). That is, the combination of \( s \) and \( \beta \) does not cause combinatorial explosion. This property is formally stated by the following lemma:

**Lemma 5.** Let \( p \in X(\downarrow, \uparrow, \rightarrow^+, \leftarrow^+) \). Suppose that \( G \models \text{MDF/DC} p((s, \beta), (s', \beta')) \) and \( G \models \text{MDF/DC} p((s, \beta'), (s, \beta')) \). Then, \( \beta = \beta' \).

**Proof.** Immediate from the definition of \( \text{MDF/DC} \) since \( p \) contains none of \( \downarrow^*, \uparrow^*, \cdot \), and \( \cdot \).

Next, consider the explosion of the number of \( s \). Because of the nondeterminism of \( \downarrow^* \), \( \rightarrow^+ \), and \( \leftarrow^+ \), the number of \( s \) can be exponential in the size of \( p \). However, recall that we are interested in \((s', \beta')\) such that \( G \models \text{MDF/DC} p((\downarrow, 1, -r), (\beta, \lambda)) \). The following lemma implies that such \( s' \) is unique up to the labeling function \( \lambda \). Moreover, \( \beta' \) is also unique up to \( \lambda \):

**Lemma 6.** Let \( p \in X(\downarrow, \uparrow, \rightarrow^+, \leftarrow^+) \). Suppose that \( G \models \text{MDF/DC} p((s_1, \beta_1), (s_2, \beta_2)) \) and \( G \models \text{MDF/DC} p((s_1, \beta_2), (s_2, \beta_1)) \), where \( \lambda(s_1) = \lambda(s_2) \) and \( \beta_1(s_1') = \beta_2(s_2') \) for all \( s_1' \) and \( s_2' \). Then, \( \lambda(s_1') = \lambda(s_2') \). The lemma can be shown by induction on the structure of \( p \).

Let \( \beta'/\lambda \) denote the mapping such that \( \beta'/\lambda(s'' \lambda) = \beta'(s'') \) for any \( s'' \). Operators \( \cup \) and \( \text{DFS,} \) and consistency can be naturally redefined on \( \beta'/\lambda \) as long as \( \beta' = \beta'_{\text{DFS,}} \). We have to maintain only one \( \beta'/\lambda \) even if the number of \( s' \) explodes.

Finally, we have to introduce a concise representation of exponentially many \( s' \). To accomplish this, the following observation is useful:

**Lemma 7.** Let \( p \in X(\downarrow, \uparrow, \rightarrow^+, \leftarrow^+) \) be an atomic XPath expression. Suppose that \( G \models \text{MDF/DC} p((s, \beta)/\lambda, (s', \beta'/\lambda)) \). Then, for any path \( s'' \) on \( G \) such that \( \lambda(s'') = \lambda(s) \), we have \( G \models \text{MDF/DC} p((s'', \beta)/\lambda, (s'', \beta'/\lambda)) \).

**Proof.** Immediate from the definition of \( \text{MDF/DC} \) since \( p \) contains neither \( \downarrow^* \) nor \( \uparrow^* \).

In other words, for atomic \( p \), only the last node of \( s \) is meaningful. Hence, we use a sequence \( U_0 \cdots U_n \) of sets of nodes of \( G \) for representing the set of \( s \) or \( s' \), where \( U_0 = \{(\downarrow, 1, -r)\} \). As usual, \( s = u_0 u_1 \cdots u_n \) in \( U_0 U_1 \cdots U_n \) if \( u_i \in U_i \) for each \( i \).

The following is a refined version of our algorithm \( \text{eval}_1 \):

\[
\text{eval}_1(p, (U_0 \cdots U_n, \beta)/\lambda) = \begin{cases} 
\text{eval}_1(p, (U_0 \cdots U_n, \beta)/\lambda) & \text{if } p \in \downarrow^* \cup \uparrow^* \\
\text{eval}_1(p, (U_0 \cdots U_n, \beta)/\lambda) & \text{if } p \in \downarrow^* \cup \uparrow^*.
\end{cases}
\]

where \( s' \) is an arbitrary path in \( U_0 \cdots U_n \) and \( u' \) is an arbitrary node such that \( su' \) is a path on \( G \) and the label of \( u' \) is \( l \), and \( U_{n+1} \) is the set of such nodes \( u' \). If \( \beta/\lambda \cup \{s \mapsto \psi(u')/\lambda\} \) is not consistent, then the execution of \( \text{eval}_1 \) fails (i.e., \( p \) is unsatisfiable).

**If** \( p = \downarrow^* : \), **then** return \( (U_0 \cdots U_n, \beta)/\lambda \).

\[
\text{eval}_1(p, (U_0 \cdots U_n, \beta)/\lambda) = \begin{cases} 
\text{eval}_1(p, (U_0 \cdots U_n, \beta)/\lambda) & \text{if } p \in \downarrow^* \cup \uparrow^*.
\end{cases}
\]

where \( s' \) is an arbitrary path in \( U_0 \cdots U_{n-1} \) such that the label of the last node of \( s \) is \( l \).

**If** \( p = \rightarrow^+ : \), **then** return \( (U_0 \cdots U_{n-1}, \beta)/\lambda \).

where \( s' \) is an arbitrary path in \( U_0 \cdots U_{n-1} \) such that the label of the last node of \( s \) is \( l \).
where $s$ is an arbitrary path in $U_0 \cdots U_{n-1}$, $u'$ is an arbitrary node such that $s u'$ is a path on $G$, the label of $u'$ is $l$, and there is $u \in U_n$ such that $\text{pos}(u) < \text{pos}(u')$ if $\omega(u) = "-"$ and $\text{pos}(u) \leq \text{pos}(u')$ if $\omega(u) = "+"$, and $U_n'$ is the set of such nodes $u'$. If $\beta \land \{(s \mapsto \psi(u'))/\lambda\}$ is not consistent, then the execution of $\text{eval}_1$ fails. The case of $p = \leftarrow l$ is similar.

- If $p = p_1/p_2$, then return $\text{eval}_1(p_2), \text{eval}_1(p_1, (U_0 \cdots U_n, \beta/\lambda))$.

Let $G = (U, E)$. It takes $O(|U|)$ time to process an atomic XPath expression. Totally, it takes $O(|p||U|)$ time to run $\text{eval}_1(p, (\{1, \ldots, n\}, \beta/\lambda))$.

**Theorem 4.** XPath satisfiability for $\mathcal{X}(\uparrow, \rightarrow, [\lambda])$ under MRW-DTDs is decidable in polynomial time.

**Example 5.** Let $D$ be the MDF/DC-DTD given in Example 2. Consider the satisfiability of $p = \downarrow U/\rightarrow^+ : b/(\downarrow a/\rightarrow^+: b)$ under $D$. The execution of $\text{eval}_1$ is as follows. Recall that the schema graph of $D$ is given in Figure 2.

\[
\begin{align*}
\text{eval}_1(p, (\{u_0\}, \beta/\lambda)) &= \text{eval}_1(\downarrow a/\rightarrow^+: b, \text{eval}_1(\downarrow a/\rightarrow^+: b, (\{u_0\}, \beta/\lambda))) \\
&= \text{eval}_1(\downarrow a/\rightarrow^+: b, \text{eval}_1(\downarrow r/\rightarrow^+: b, (\{u_0\}, \beta/\lambda))) \\
&= \text{eval}_1(\downarrow a/\rightarrow^+: b, \text{eval}_1(\downarrow r/\rightarrow^+: b, \{(u_0, u_1, u_2, r \mapsto \emptyset)\})) \\
&= \text{eval}_1(\downarrow a/\rightarrow^+: b, \{(u_0, u_1, u_2, r \mapsto \emptyset)\}) \\
&= \text{eval}_1(\uparrow b, \text{eval}_1(\downarrow a, (\{u_0\}, u_3, r \mapsto b))) \\
&= \text{eval}_1(\uparrow b, \{(u_0, u_3, r \mapsto b)\}) \\
&= \{(u_0, u_3, r \mapsto b, c, r \mapsto \{a\})\}.
\end{align*}
\]

Hence $p$ is determined to be satisfiable. Actually, the tree $T$ in Figure 2 satisfies $p$.

Next, consider the satisfiability $p' = p/\rightarrow^+: c$ under $D$. Then, the execution of $\text{eval}_2$ would be as follows:

\[
\begin{align*}
\text{eval}_2(p, (\{u_0\}, \beta/\lambda)) &= \text{eval}_2(\uparrow b, \text{eval}_1(p, (\{u_0\}, \beta/\lambda))) \\
&= \text{eval}_2(\uparrow b, \{(u_0, u_3, r \mapsto b)\}) \\
&= \{(u_0, u_3, r \mapsto b, c, r \mapsto \{a\})\}.
\end{align*}
\]

However, $\{r \mapsto \{b, c\}, r \mapsto \{a\}\}$ is not consistent, and hence $p'$ is determined to be unsatisfiable.

### 4.4.2 $\mathcal{X}(\downarrow, \rightarrow, [\lambda])$

Let $p \in \mathcal{X}(\downarrow, \rightarrow, [\lambda])$. We show an efficient algorithm for deciding whether $G \models_{\text{MDF/DC}} p(\{1, \ldots, n\}, \beta/\lambda)$ for some $s'$ and $\beta'$.

For this case, our algorithm $\text{eval}_2$ runs in a bottom-up manner with respect to the parse tree of $p$, and essentially computes the set of all the paths $\{(s, \beta), (s', \beta')\}$ such that $G \models_{\text{MDF/DC}} p(s, \beta), (s', \beta')$. However, a naive implementation causes exponential runtime. Since the same properties as Lemmas 4 and 5 hold for this XPath class, we can use the ideas again in the previous section. Moreover, since this XPath class contains no upward axes, it suffices to maintain just the last nodes of $s$ and $s'$. However, to handle path concatenations and qualifiers, we need information which parameter of $\beta'$ is the “current node,” which is originally represented by $s'$. Here, we use $\lambda(s')$ instead of $s'$ itself to avoid explosion. To summarize, let us allow arbitrary (possibly empty) paths on $G$ as parameters of sibling-constraint mappings $\beta$, and let $\beta \land s'$ denote a mapping such that $(\beta \land s')(s'\land s) = \beta(s)$.

Now, $\text{eval}_2$ computes all the tuples $((u, \beta/\lambda), (u', \beta'/\lambda), \lambda(s'))$ such that $G \models_{\text{MDF/DC}} p(s, \beta, s', \beta', (s'\land s')\land \lambda)$ for any $s'$, where $\beta$ is the minimum mapping with respect to $\land$. The following is a formal description of our algorithm $\text{eval}_2$:

\[
\begin{align*}
\text{eval}_2(p, (\{u_0\}, \beta/\lambda)) &= \text{eval}_2(\downarrow r, \text{eval}_1(p, (\{u_0\}, \beta/\lambda))) \\
&= \text{eval}_2(\downarrow r, \{(u_0, u_1, r \mapsto \emptyset)\}) \\
&= \{(u_0, u_1, r \mapsto \emptyset, c, r \mapsto \{a\})\}.
\end{align*}
\]

- If $p = \downarrow r/\rightarrow^+: c$, return the set of $\{(u_0, \beta/\lambda), (u_1, r \mapsto c)\}$.

- If $p = \rightarrow^+: c$, return the set of $\{(u_0, \beta/\lambda), (u_1, r \mapsto c)\}$.
eval$_2$($\rightarrow^+ : b[\downarrow : a]$)

\[
= \{ ((u_1, \{ e \mapsto 0 \}), (u_4, \{ e \mapsto \{ b \}, b \mapsto \{ a \} \}), e),
\]

\[
((u_2, \{ e \mapsto \{ a \} \}), (u_3, \{ e \mapsto \{ a, b \}, b \mapsto \{ a \} \}), e),
\]

\[
eval_2(\downarrow : r \rightarrow^+ : b[\downarrow : a])
\]

\[
= \{ ((u_0, \beta_1 / \lambda), (u_3, \{ r \mapsto \{ b \}, r b \mapsto \{ a \} \}), r),
\]

\[
((u_1, \beta_2 / \lambda), (u_4, \{ r \mapsto \{ b \}, r b \mapsto \{ a \} \}), r),
\]

\[
((u_5, \beta_3 / \lambda), (u_4, \{ r \mapsto \{ b \}, r b \mapsto \{ a \} \}), r)\}.
\]

Since we have found $(s', \beta')$ such that $G \models_{MDF/DC} p((u_0, \beta_1 / \lambda), (s', \beta'))$, $p$ is determined to be satisfiable. Actually, the tree $T$ in Figure 3 satisfies $p$.

Next, consider the satisfiability $p' = p/\rightarrow^+: c$ under $D$. Then, the execution of eval$_2(\rightarrow^+ : c)$ is:

\[
eval_2(\rightarrow^+ : c)
\]

\[
= \{ ((u_0, \beta_1 / \lambda), (u_4, \{ e \mapsto \{ c \} \}), e),
\]

\[
((u_2, \{ e \mapsto \{ a \} \}), (u_4, \{ e \mapsto \{ a, c \} \}), e),
\]

\[
((u_3, \{ e \mapsto \{ b \} \}), (u_4, \{ e \mapsto \{ b, c \} \}), e)\}.
\]

Note that $\{ e \mapsto \{ a, c \} \}$ and $\{ e \mapsto \{ b, c \} \}$ are consistent because they are undefined at non-empty paths. Finally, the execution of eval$_2(p/\rightarrow^+: c)$ would be:

\[
\{ ((u_0, \beta_1 / \lambda), (u_4, \{ r \mapsto \{ b, c \}, r b \mapsto \{ a \} \}), r),
\]

\[
((u_1, \beta_2 / \lambda), (u_4, \{ r \mapsto \{ b, c \}, r b \mapsto \{ a \} \}), r),
\]

\[
((u_5, \beta_3 / \lambda), (u_4, \{ r \mapsto \{ b, c \}, r b \mapsto \{ a \} \}), r)\}.
\]

However, since $\{ r \mapsto \{ b, c \}, r b \mapsto \{ a \} \}$ is not consistent, eval$_2(p')$ returns the empty set. Hence $p'$ is determined to be unsatisfiable.

5. Conclusions

This paper has proposed a class of DTDs, called MRW-DTDs, which cover many of the real-world DTDs and have non-trivial tractability of XPath satisfiability. To be specific, MRW-DTDs cover 24 out of the 27 real-world DTDs, 1403 out of the 1407 DTD rules. Under MRW-DTDs, we have shown that satisfiability problems for $\lambda(\downarrow, \rightarrow^+, \downarrow^+, [\downarrow])$, $\lambda(\downarrow, \rightarrow^+, \downarrow^+, [\downarrow])$, and $\lambda(\downarrow, \rightarrow^+, \downarrow^+, [\downarrow])$ are all tractable.

Actually, we tried to show the intractability of the union $\lambda(\downarrow, \rightarrow^+, \downarrow^+, [\downarrow])$ of the tractable classes. However, we have finally found that reduction from 3SAT to the class is very difficult. One of our future work is to develop an efficient algorithm for determining the satisfiability of the union class under MRW-DTDs. As stated in Section 3, there have been two approaches to resolving the intractability of XPath satisfiability. The approach using fast decision procedures for MSO and $\mu$-calculus is fairly powerful from the practical point of view. It is reported that satisfiability (and other static analysis problems such as containment and coverage) was decided within one second for many XPath expressions taken from XPathMark. Another important direction of the future work is empirical evaluation of the proposed polynomial-time algorithms.

Acknowledgment

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

References


