情報処理学会論文誌 Vol.57 No.5 1477–1488 (May 2016)

推薦論文

XPath充足可能性を判定する多項式時間アルゴリズムの提案と評価

杉村 憲司[1,a)] 石原 靖哲[1,b)] 藤原 融[1,c)]

受付日 2015年6月3日, 採録日 2016年2月8日

概要：XPath 充足可能性判定問題は，問合せの最適化などに応用できる重要な問題であるが，一般には NP 困難であることが知られている．そのため，DTD や XPath 式のクラスを制限することで，多項式時間で充足可能性を判定できるアルゴリズムを提案する研究が行われている．本論文では，多項式時間充足可能性判定アルゴリズムを提案・実装し，その有用性を評価する．実装の結果，一般的なベンチマークに対して数十ミリ秒で判定でき，SAT ソルバを用いた手法に比べて十分に高速であることが分かった．

キーワード：問合せ解析，充足可能性，XPath，XML データベース

Proposal and Evaluation of Polynomial-time Algorithms for Deciding XPath Satisfiability

Kenji Sugimura[1,a)] Yasunori Ishihara[1,b)] Toru Fujiwara[1,c)]

Received: June 3, 2015, Accepted: February 8, 2016

Abstract: XPath satisfiability is an important problem that is useful for query optimization, but it is known to be NP-hard in general. Therefore, some researches propose polynomial-time algorithms for deciding XPath satisfiability for some restricted classes of DTDs and XPath expressions. In this paper, we propose and implement polynomial-time algorithms for XPath satisfiability, and discuss their usefulness. The satisfiability of a standard benchmark can be decided in several tens of milliseconds; it is much faster than the methods based on SAT solvers.

Keywords: query analysis, satisfiability, XPath, XML database

1. まえがき

近年，構造化データを記述可能なマークアップ言語として XML（Extensible Markup Language）がさかんに利用されており，XML 文書の特定の要素を指定する問合せ言語として XPath が広く用いられている．XPath は，XML 文書をラベルつき順序木に見立て，その根ノードからの経路を記述することで，XML 文書の特定の要素を指定する問合せ言語である．XPath はそれ自体が問合せ言語として用いられるだけでなく，XQuery や XSLT などの実用的な問合せ言語の一部としても利用されている．また，XML 文書のデータ構造を定義するスキーマ言語として，DTD（Document Type Definition）がしばしば用いられている．

与えられた XPath 式 p と DTD D に対し，p の評価結果が空とならないような，D に従う XML 文書 T が存在するとき，p は D のもとで充足可能であるという．XPath 充足可能性判定は問合せの最適化に有用であるが，一般には NP 困難，特に XPath 式が否定演算を含む場合は PSPACE 困難であることなどが知られている [3]．

XPath 充足可能性判定に関する既存研究は，2 通りの

[1] 大阪大学
Osaka University, Suita, Osaka 565–0871, Japan
[a)] s-kenji@ist.osaka-u.ac.jp
[b)] ishihara@ist.osaka-u.ac.jp
[c)] fujiwara@ist.osaka-u.ac.jp

本論文で提案するアルゴリズムは文献 [1], [2] にて著者らにより発表された成果の一部である．
本論文の内容は 2014 年 9 月の支部大会にて報告され，支部長により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

© 2016 Information Processing Society of Japan

表 1 XPath 充足可能性判定の計算量
Table 1 The complexity of XPath satisfiability.

XPath クラス								DTD クラス			
↓	$\downarrow^*$	↑	$\uparrow^*$	$\rightarrow^+$	$\leftarrow^+$	∪	[ ]	任意	被覆	$\mathbf{DC}^{?+}$	選言なし
+	+					+		P [3]	P	P	P
+	+					+	+	NP 完全 [3]	P [4]	P	P [3]
	+			+			+	NP 完全	**NP 完全**（定理 4）	**P**	**P**
+	+			+	+	+	+	NP 完全	**NP 完全**	**P**（定理 3）	**P**
+		+		+	+			NP	NP	**P**（定理 5）	**P**
+		+				+	+	NP 完全	NP 完全	NP 完全	NP 完全 [3]
+	+	+	+	+	+	+	+	NP 完全 [3]	NP 完全	NP 完全	NP 完全

太字は本論文の成果である．XPath クラスにおける+は，その演算子を含むことを表している．なお，·（self 軸）は計算量に影響を与えないため，表では省略している．また，/（式の連接）はすべてのクラスに含まれているため，やはり表では省略している．

アプローチに分類できる．1 つは，高速なソルバを用いて XPath の充足可能性判定を行う手法を提案する研究である．Genevès ら [5], [6], [7] の手法では，スキーマ言語として DTD より能力が高い有限木オートマトンを採用し，否定演算も含む幅広いクラスの XPath 式を対象として，スキーマや XPath 式を既存のソルバで判定可能な論理式に変換することによって，充足可能性判定を行う．この手法は，実験により，多くの場合に実用的な時間内で動作することが確認されているが，多項式時間で動作することは保証されていない．

もう 1 つは，充足可能性が多項式時間で判定できるような，XPath や DTD のクラスをみつけるというアプローチである．たとえば Benedikt ら [3] によると，「選言なし DTD」と呼ばれる DTD クラスでは一般の DTD クラスの場合に比べてより広い XPath クラスに対して充足可能性が多項式時間で判定できる．選言なし DTD とは，可能な子ラベル系列を指定する正規表現（内容モデルと呼ぶ）に選言演算子 | とたかだか 1 回の出現を表す演算子？を含まない DTD である．また，Montazerian ら [4] は「重複なし DTD」や「被覆 DTD」といった DTD クラスのもとでより広い XPath クラスに対して充足可能性が多項式時間で判定できることを示している．直観的に，重複なし DTD とは，内容モデル中に同じラベルが 2 度以上現れない DTD であり，被覆 DTD とは，内容モデルに現れるすべてのラベルが同時に出現するような子ラベル系列が許されている DTD である．さらに Montazerian らは，実世界の DTD のうちのどれくらいが重複なし DTD や被覆 DTD になっているかを調査している．しかし，彼らが検討の対象としている XPath クラスは XML 文書の木構造を上下（親子や先祖子孫）方向に探索する機能しかなく，XML 文書は順序木でモデル化されるという特徴をとらえきれていないという点で不十分である．

本論文の目的は以下のとおりである．

(1) 木構造を横（兄弟）方向に探索する機能を持つ XPath 式の充足可能性が多項式時間で判定できるような，実用上十分な大きさを持つ DTD クラスを提案すること．
(2) 上で得られた多項式時間判定アルゴリズムを実装し，以下の 2 つの観点から多項式時間判定アルゴリズムおよびその実装の有用性を評価すること．
  (a) 単体評価．判定に要する実行時間がどれくらいになるのか．また，比較的単純な実装上の工夫を施すことで，どの程度実行時間が高速化されるのか．
  (b) 比較評価．ソルバを用いた手法と比べて十分高速に判定できるのか．

本論文で対象とする XPath 式は，軸（XML 文書の木のノードからどの方向に探索を進めるかを表す演算子）として ·（self 軸），↓（child 軸），$\downarrow^*$（descendant-or-self 軸），↑（parent 軸），$\uparrow^*$（ancestor-or-self 軸），$\rightarrow^+$（following-sibling 軸），$\leftarrow^+$（preceding-sibling 軸）を持ち，式の構成子として /（式の連接），∪（式の和），[ ]（述語; 先行する式を条件づけする演算子）を持つものとする．さらに述語の内部では論理積 ∧ と論理和 ∨ を使用できるものとする．否定を表す演算は含まないが，$\rightarrow^+$ と $\leftarrow^+$ により横方向の探索が可能なクラスである．そして目的 (1) に関して，本論文では $\mathrm{DC}^{?+}$-DTD というクラスを提案する．直観的に，$\mathrm{DC}^{?+}$-DTD とは，内容モデルに現れる選言演算子 | が必ず閉包演算子 $*$ または $+$ で囲われている DTD である．$\mathrm{DC}^{?+}$-DTD は被覆 DTD の真の部分クラスであり，選言なし DTD のクラスを真に含む．そして，$\mathrm{DC}^{?+}$-DTD のもとで以下の 2 つの XPath クラスの充足可能性が多項式時間判定可能であることを示す．

- ·, ↓, $\downarrow^*$, $\rightarrow^+$, $\leftarrow^+$, /, ∪, [ ], ∧, ∨ からなる XPath 式．
- ·, ↓, ↑, $\rightarrow^+$, $\leftarrow^+$, / からなる XPath 式．

さらに，前者の XPath クラスについて，被覆 DTD のもとでの充足可能性判定は NP 完全であることを示す．目的 (1) に関する成果を**表 1** にまとめる．

次に，目的 (2) に関して本論文では，ベンチマークとして XMark [8] をベースにした DTD と，XPathMark [9] をベースとした XPath 式を用いた．これらは XML のベンチマークとして一般的に用いられているものである．実装し

© 2016 Information Processing Society of Japan

たプログラムの実行時間を計測した結果，一般的なPC上で数十ミリ秒で動作した．また，計算の途中結果をキャッシュしておくという工夫が判定高速化に有効であることを確認した．さらに，ソルバを用いた手法に同じベンチマークを与えたところ，早くても1分程度，遅い場合は2時間程度かかる場合があり，多項式時間判定アルゴリズムは十分に高速であることを確認した．

以降，2章で諸定義を行い，3章では多項式時間判定アルゴリズムと計算量について述べる．4章で，実装したシステムと，実装上の工夫について述べる．そして5章では単体評価，6章では比較評価について述べる．最後に7章で，まとめと今後の課題について述べる．

2. 諸定義

本章では，XML文書，DTD，およびXPathの定義について述べる．また，DTDのクラスについても述べる．

2.1 XML文書

XMLとはeXtensible Markup Languageの略で，独自の意味や構造を持ったマークアップ言語を作成するために，汎用的に用いられるマークアップ言語である．XMLはソフトウェア間の通信・情報交換に用いるデータ形式や，様々な種類のデータを保存するためのファイルフォーマットなどの定義に使われている．

本論文では，XML文書をラベルつき順序木と見なす．ここで，ラベルはXML文書中の要素名を表す．ノード v のラベルを $\lambda(v)$ で表す．関数 λ はノード列に対して自然に拡張される．すなわち，ノード列 $v_1v_2\cdots v_n$ について $\lambda(v_1v_2\cdots v_n)=\lambda(v_1)\lambda(v_2)\cdots\lambda(v_n)$ と定義する．

2.2 DTD

DTDとはDocument Type Definitionの略で，文書構造（文書型）を定義するためのスキーマ言語の1つである．DTDでは，XML文書内に記述することができる要素やその発生順序，発生回数，要素が持つ属性，属性の型などを記述することができる．ただし，本論文では，属性については扱わない．

DTD D は3項組 (Σ, r, P) で表現する．ここで Σ はラベルの有限集合，r は Σ の要素，P は Σ から Σ 上の正規表現集合への写像である．直観的には，r は木の根ノードのラベルを表しており，P は各ラベルが子として持ちうるラベル列の集合を表している．$P(l)$ をラベル l の内容モデル [10] と呼ぶ．

P における正規表現は，定数として ϵ（空語）と Σ に含まれるラベル，演算子として $\cdot$（連接，通常表記では省略），$|$（選言），$*$（0回以上の繰返し），$+$（1回以上の繰返し），$?$（たかだか1回の出現）で構成される．正規表現 e のサイズ $|e|$ を，e に現れる定数と演算子の個数の総和と定義する．DTD D のサイズ $|D|$ を $\sum_{l\in\Sigma}|P(l)|$ と定義する．

以下の2条件が満たされるとき，木 T はDTD $D=(\Sigma, r, P)$ に従うという．

- T の根ノード v_0 に対し，$\lambda(v_0)=r$ である．
- T の各ノード v とその子ノード $v_1, v_2, \ldots, v_n$ について，$P(\lambda(v))$ が表す正規言語は $\lambda(v_1v_2\cdots v_n)$ を含む．

D に従うすべての木の集合を $TL(D)$ と表す．一般性を失うことなく，$D=(\Sigma, r, P)$ は無駄なラベルを含まないと仮定する．すなわち，任意のラベル $a\in\Sigma$ について，a を含む木が $TL(D)$ に存在する．

定義1 正規表現 e が以下のいずれかを満たすとき，e は $\mathrm{DC}^{?+}$ であるという．

(1) $e\in\Sigma$.
(2) ある正規表現 e' に対し，$e=(e')^*$ または $e=(e')^+$.
(3) ある $\mathrm{DC}^{?+}$ な正規表現 e'，e'' に対し，$e=e'\cdot e''$ または $e=(e')^?$.

各 $a\in\Sigma$ について $P(a)$ が $\mathrm{DC}^{?+}$ であるDTD $D=(\Sigma, r, P)$ を，$\mathrm{DC}^{?+}$-DTDと呼ぶ． □

すなわち，$\mathrm{DC}^{?+}$ な正規表現とは，すべての選言演算子 $|$ が $*$ または $+$ によって囲われている正規表現である．

例1 $a^?(b|c)^+$ は $\mathrm{DC}^{?+}$ である．一方，$(a|b)^?c^+$ は $\mathrm{DC}^{?+}$ でない． □

正規表現 e が演算子 $|$ と $?$ を含まないとき，e を選言なしという．また，正規表現 e に現れるすべてのラベル集合を $\Sigma'\subseteq\Sigma$ とするとき，Σ' 中のすべてのラベルを含む語が e の表す言語に含まれるならば，e を被覆正規表現と呼ぶ [4]．たとえば，$\Sigma=\{a,b,c\}$ とすると，$a(a|b)$ や $a(a|b)c$ は被覆正規表現であるが，$a(b|c)$ は b と c を同時に含む語が $a(b|c)$ の表す言語に含まれないため被覆正規表現ではない．

表2 に，現実世界の27個のDTDについて，各内容モデルが被覆正規表現か，$\mathrm{DC}^{?+}$ か，選言なしかを調査した結果を示す．これらのDTDは，文献 [4] で調査対象となっていたもののうち，入手できなかったものやDTD名が異なるだけで内容が同一のものを除外し，Math MLやSVGなど知名度が高いと思われるものを追加したものである．$\mathrm{DC}^{?+}$-DTDは，既知のクラスである被覆DTDとほぼ同等の大きさを持ち，選言なしDTDよりはるかに大きいことが確認できる．

2.3 XPath

XPathは，マークアップ言語XMLに準拠した文書の特定の部分を指定する言語である．XPath式の構文はW3C XPath [11] をもとに，以下のように定義する．

$$p ::= \chi :: l \mid p/p \mid p\cup p \mid p[q],$$
$$\chi ::= \cdot \mid \downarrow \mid \uparrow \mid \downarrow^* \mid \uparrow^* \mid \rightarrow^+ \mid \leftarrow^+,$$
$$q ::= p \mid q\wedge q \mid q\vee q.$$

ただし，$l\in\Sigma$ とする．χ から導出される要素を軸と呼ぶ．

© 2016 Information Processing Society of Japan

表 2 現実世界の DTD における，被覆，DC$^{?+}$，選言なし内容モデルを持つラベルの数

Table 2 The number of labels with covering, DC$^{?+}$ and disjunction-free content models in real-world DTDs.

DTD 名	ラベル数			
	総数	被覆	DC$^{?+}$	選言なし
DBLP	36	**36**	**36**	22
Ecoknowmics	224	222	222	221
LevelOne	28	26	26	14
MathML-2.0	181	**181**	**181**	126
Mondial	40	**40**	**40**	30
Music ML	12	**12**	**12**	7
News ML	118	114	114	72
Newspaper	7	**7**	**7**	**7**
Opml	15	**15**	**15**	14
OSD	15	14	14	12
P3P-1.0	85	83	81	68
PSD	66	64	64	53
Reed	16	**16**	**16**	**16**
Rss	30	29	29	27
SigmodRecord	11	**11**	**11**	10
SimpleDoc	49	**49**	**49**	16
SSML-1.0	16	**16**	**16**	8
SVG-1.1	80	77	77	15
TV-Schedule	10	**10**	**10**	8
VoiceXML-2.0	62	**62**	**62**	30
Xbel-1.0	9	**9**	**9**	6
XHTML1-strict	77	75	74	21
XMark DTD	77	76	76	65
XML Schema	26	20	20	0
XML Signature	45	**45**	44	35
XMLTV	40	**40**	**40**	36
Yahoo	32	**32**	**32**	**32**
合計	1407	1381	1377	971

また，$\chi :: l$ の形の式を原子式と呼ぶ．p のサイズ $|p|$ を，p に現れる原子式の個数と定義する．

木 T における XPath 式の意味を以下のように定義する．ここで p と q は，T の根ノードからの経路集合上の，それぞれ 2 引数と 1 引数の述語として定義される（経路集合ではなく単なるノード集合上の述語としても定義できるが，充足可能性判定アルゴリズムの正当性の証明を見通しよくするため，本論文ではこのように定義する）．以下では，v や v' は T のノードを表し，w，w'，w'' は T の根ノードからの空でない経路（根ノードから始まる，長さ 1 以上のノード系列）を表す．

- T 上の経路 wv が存在して $\lambda(v) = l$ のとき，$T \models (\cdot :: l)(wv, wv)$．
- T 上の経路 wv' が存在して $\lambda(v') = l$ のとき，$T \models (\downarrow :: l)(w, wv')$．
- T 上の経路 wv が存在して w の最終ノードのラベルが l のとき，$T \models (\uparrow :: l)(wv, w)$．
- T 上の経路 w，w' が存在して，w が w' の接頭辞でありかつ w' の最終ノードのラベルが l のとき，$T \models (\downarrow^* :: l)(w, w')$．
- T 上の経路 w，w' が存在して，w' が w の接頭辞でありかつ w' の最終ノードのラベルが l のとき，$T \models (\uparrow^* :: l)(w, w')$．
- T 上の経路 wv，wv' が存在して，v' が v の右の兄弟で $\lambda(v') = l$ のとき，$T \models (\rightarrow^+ :: l)(wv, wv')$．
- T 上の経路 wv，wv' が存在して，v' が v の左の兄弟で $\lambda(v') = l$ のとき，$T \models (\leftarrow^+ :: l)(wv, wv')$．
- $T \models p(w, w'')$ かつ $T \models p'(w'', w')$ であるような T 上の経路 w'' が存在するとき，$T \models (p/p')(w, w')$．
- $T \models p(w, w')$ または $T \models p'(w, w')$ のとき，$T \models (p \cup p')(w, w')$．
- $T \models p(w, w')$ かつ $T \models q(w')$ のとき，$T \models (p[q])(w, w')$．
- T 上の経路 w' が存在して $T \models p(w, w')$ のとき，$T \models p(w)$．
- $T \models q(w)$ かつ $T \models q'(w)$ のとき，$T \models (q \wedge q')(w)$．
- $T \models q(w)$ または $T \models q'(w)$ のとき，$T \models (q \vee q')(w)$．

木 T の根を v_0 とする．$T \models p(v_0, w)$ を満たす T の経路 w が存在するとき，木 T は XPath 式 p を充足するという．XPath 式 p と DTD D に対し，p を充足する木 $T \in TL(D)$ が存在するならば，p は D のもとで充足可能であるという．

3. XPath 充足可能性判定多項式時間アルゴリズム

本章では，本論文で提案する多項式時間判定アルゴリズムを説明する．このアルゴリズムは，スキーマグラフという概念に基づいている．DC$^{?+}$-DTD D のスキーマグラフを G とすると，XPath 式 p が D のもとで充足可能であるときかつそのときのみ，p は G によって充足（形式的な定義は後述）されるという性質を示す．したがって，G によって充足されるかを多項式時間で判定できるような XPath 式クラスに p が属するとき，p の D のもとでの充足可能性は多項式時間可解となる．

3.1 スキーマグラフ

定義 2 DC$^{?+}$-DTD $D = (\Sigma, r, P)$ のスキーマグラフ $G = (U, E)$ とは，以下のような有向グラフである．

(1) ノード $u \in U$ は，以下のどちらかである．

- $(\bot, 1, -, r)$．ただし，$\bot \notin \Sigma$ である．
- (a, i, ω, b)．ただし，$a, b \in \Sigma$ である．$P(a)$ 中のすべての演算子 ? を取り除き，演算子 + を $*$ で置き換えて得られる正規表現を e とすると，$e = e_1 \cdots e_n$ の形に書ける（ただし各 e_j はラベル単体か $(e'_j)^*$ の形である）．この n に対し i は $1 \le i \le n$ を満たす．b は e の i 番目の部分式 e_i に現れるラベルで

© 2016 Information Processing Society of Japan

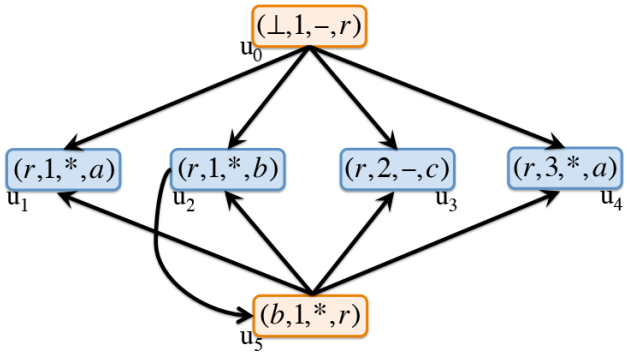

図 1 スキーマグラフの例
Fig. 1 An example of a schema graph.

ある．そして，e_i がラベル単体のとき ω は $-$ であり，そうでなければ ω は $*$ である．

ノード u の第 1 要素を $\lambda_{par}(u)$，第 2 要素を $pos(u)$，第 3 要素を $\omega(u)$，第 4 要素を $\lambda(u)$ と表す．$\lambda(u)$ を u のラベルと呼ぶ．

(2) $\lambda(u) = \lambda_{par}(u')$ のときかつそのときのみ，u から u' への有向辺が存在する． □

例 2 DC$^{?+}$-DTD $D = (\{r,a,b,c\}, r, P)$，$P(r) = (a|b)^*c^?a^+$，$P(a) = \epsilon$，$P(b) = r^*$，$P(c) = \epsilon$ を考える．D のスキーマグラフは**図 1** のようになる． □

D のスキーマグラフ $G = (U, E)$ は $O(|D|^2)$ 時間で生成可能である．また，U のサイズは $O(|D|)$，E のサイズは $O(|D|^2)$ である．

$T \in TL(D)$ ならば，T の各ノードを D のスキーマグラフの各ノードに対応づけることができる．形式的には，以下の条件を満たす，T のノード集合から D のスキーマグラフのノード集合 U への写像 θ が（一般には複数）存在する．

- θ は T の根ノードを $(\bot, 1, -, r)$ に写す．
- v を T のノードとし，$v_1 \cdots v_k$ を v の子系列とする．$P(\lambda(v))$ に対して定義 2 (1) で定まる n を考える．このとき $w_1 \cdots w_n = v_1 \cdots v_k$ を満たす n 個のノード系列 $w_1, \ldots, w_n$ が存在し，以下が成立する．
  - $\lambda(w_i)$ は $P(\lambda(v))$ の i 番目の部分式 e_i が表す正規言語に含まれる．
  - w_i に含まれる各ノード v_j について，$\theta(v_j) = (\lambda(v), i, \omega_i, \lambda(v_j)) \in U$．

本論文では，このような写像 θ を T の SG 写像と呼ぶ．

例 3 **図 2** に示す木 T は，例 2 の DTD D に従っている．以下の写像 θ は T の SG 写像の 1 つである．

$$\begin{aligned} \theta(v_0) &= u_0, \\ \theta(v_1) = \theta(v_3) = \theta(v_8) &= u_1, \\ \theta(v_2) &= u_2, \\ \theta(v_4) &= u_3, \\ \theta(v_5) = \theta(v_6) = \theta(v_9) &= u_4, \\ \theta(v_7) &= u_5. \end{aligned}$$

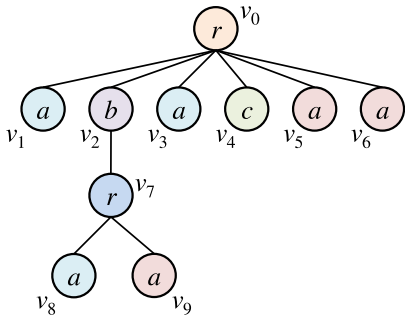

図 2 木の例
Fig. 2 An example of a tree.

また，$\theta'(v_8) = u_4$，$\theta'(v) = \theta(v)$ $(v \neq v_8)$ で定義される写像 θ' も T の SG 写像である．

スキーマグラフ G と XPath 式 p についての充足関係を以下のように定義する．以下では，u，u' は G のノードであり，s，s'，s'' は $(\bot, 1, -, r)$ から始まる空でない G 上の経路である．

- G 上の経路 su が存在して $\lambda(u) = l$ のとき，$G \models (\cdot :: l)(su, su)$．
- G 上の経路 su' が存在して $\lambda(u') = l$ のとき，$G \models (\downarrow :: l)(s, su')$．
- G 上の経路 su が存在して s の最終ノードのラベルが l のとき，$G \models (\uparrow :: l)(su, s)$．
- G 上の経路 s, s' が存在して，s が s' の接頭辞でありかつ s' の最終ノードのラベルが l のとき，$G \models (\downarrow^* :: l)(s, s')$．
- G 上の経路 s, s' が存在して，s' が s の接頭辞でありかつ s' の最終ノードのラベルが l のとき，$G \models (\uparrow^* :: l)(s, s')$．
- G 上の経路 su，su' が存在して，$\lambda(u') = l$ であり，かつ「$\omega(u)$ が $-$ ならば $pos(u) < pos(u')$ で $\omega(u)$ が $*$ ならば $pos(u) \leq pos(u')$」のとき，$G \models (\rightarrow^+ :: l)(su, su')$．
- G 上の経路 su，su' が存在して，$\lambda(u') = l$ であり，かつ「$\omega(u)$ が $-$ ならば $pos(u) > pos(u')$ で $\omega(u)$ が $*$ ならば $pos(u) \geq pos(u')$」のとき，$G \models (\leftarrow^+ :: l)(su, su')$．
- $G \models p(s, s'')$ かつ $G \models p'(s'', s')$ であるような G 上の経路 s'' が存在するとき，$G \models (p/p')(s, s')$．
- $G \models p(s, s')$ または $G \models p'(s, s')$ のとき，$G \models (p \cup p')(s, s')$．
- $G \models p(s, s')$ かつ $G \models q(s')$ のとき，$G \models (p[q])(s, s')$．
- G 上の経路 s' が存在して $G \models p(s, s')$ のとき，$G \models p(s)$．
- $G \models q(s)$ かつ $G \models q'(s)$ のとき，$G \models (q \wedge q')(s)$．
- $G \models q(s)$ または $G \models q'(s)$ のとき，$G \models (q \vee q')(s)$．

このように定義した G による p の充足関係が，D における p の充足可能性と一致することを以下に示す．

補題 1 D を DC$^{?+}$-DTD，p を XPath 式とし，G を D のスキーマグラフとする．ある木 $T \in TL(D)$ について $T \models p(w, w')$ が成り立つならば，T の任意の SG 写像 θ について $G \models p(\theta(w), \theta(w'))$ が成り立つ．

© 2016 Information Processing Society of Japan

略証： G は $TL(D)$ に属するあらゆる木のトポロジを含んでいるため，T において v が v' の子ならば，G において $\theta(v')$ から $\theta(v)$ への有向辺が存在する．この事実を用いて，p の構造に関する帰納法により，容易に題意を示すことができる． □

補題 2 D を DC$^{?+}$-DTD，p を XPath 式とし，G を D のスキーマグラフとする．$G \models p(s, s')$ が成立するならば，ある木 $T \in TL(D)$，その SG 写像 θ，および T 上の根ノードからの経路 w，w' が存在して，$\theta(w) = s$，$\theta(w') = s'$，かつ $T \models p(w, w')$ が成立する．

略証： p の構造に関する帰納法により示す．p が原子式のとき，D が無駄なラベルを含まないことから，題意を満たす木 T と SG 写像 θ の存在を示せる．

$p = p_1/p_2$ とし，$G \models p(s, s')$ が成立すると仮定する．G による充足関係の定義より，G 上のある経路 s'' が存在して，$G \models p(s, s'')$ かつ $G \models p(s'', s')$ である．ここで帰納法の仮定を用いると，以下が得られる．

- ある木 $T_1 \in TL(D)$，その SG 写像 θ_1，および T_1 上の根ノードからの経路 w_1，w_1' が存在して，$\theta_1(w_1) = s$，$\theta_1(w_1') = s''$，かつ $T \models p_1(w_1, w_1')$．
- ある木 $T_2 \in TL(D)$，その SG 写像 θ_2，および T_2 上の根ノードからの経路 w_2，w_2' が存在して，$\theta_2(w_2) = s''$，$\theta_2(w_2') = s'$，かつ $T \models p_2(w_2, w_2')$．

このとき，以下を満たす $T \in TL(D)$ とその SG 写像 θ が存在する．

(1) T から部分木をいくつか削除すると T_1 になり，T_1 のノード集合を定義域とする θ は θ_1 と一致する．
(2) T から部分木をいくつか削除すると T_2 になり，T_2 のノード集合を定義域とする θ は θ_2 と一致する．
(3) 上で得られた T_1 における経路 w_1' と，上で得られた T_2 における経路 w_2 は，T における同一の経路である．

(1) と (2) を満たす $T \in TL(D)$ と θ の存在は，D が DC$^{?+}$-DTD であることから保証される．D において選言演算子 $|$ は $*$ や $+$ の内側にしか現れないため，D に違反しないように T_1 と T_2 に適当に部分木を追加することで，同一の木 $T \in TL(D)$ に変形することが可能である．さらに，$\theta_1(w_1') = \theta_2(w_2)$ $(= s'')$ より，(3) も満たす T と θ の存在が保証される．この木 $T \in TL(D)$ と SG 写像 θ について，$\theta(w_1) = s$，$\theta(w_2') = s'$，かつ $T \models p(w_1, w_2')$ が成立する．

p が他の形をしているときも同様の議論で証明できる． □

補題 1, 2 より，以下の定理が得られる．すなわち，DC$^{?+}$-DTD $D = (\Sigma, r, P)$ のもとでの p の充足可能性を判定したければ，D のスキーマグラフ G を構成し，$G \models p((\bot, 1, -, r), s)$ を満たす s が存在するかを調べればよい．

定理 1 DC$^{?+}$-DTD $D = (\Sigma, r, P)$ のスキーマグラフを G とする．D のもとで XPath 式 p が充足可能であるときかつそのときのみ，$(\bot, 1, -, r)$ から始まる空でない G 上の経路 s が存在して $G \models p((\bot, 1, -, r), s)$ が成り立つ． □

3.2 充足可能性判定問題の NP 完全性

文献 [3] の補題 4.5 は，$\downarrow$，$\uparrow$，$\downarrow^*$，$\uparrow^*$，$/$，$\cup$，$[\,]$ からなる XPath 式 p が DTD D のもとで充足可能な場合，その式を充足する高さ $(3|p| - 1)|\Sigma|$ 以下の木 T が $TL(D)$ に存在することを保証している．この補題に含まれていない軸 $(\cdot, \rightarrow^+, \leftarrow^+)$ は木における上下方向の移動をともなわないため，本論文で対象としている XPath クラスに対しても補題 4.5 は成立する．そして，文献 [3] の定理 6.9 では，選言なし DTD のもとで，$\downarrow$，$\uparrow$，$/$，$\cup$，$[\,]$ からなる XPath 式の充足可能性判定が NP 困難であることを示している．選言なし DTD は DC$^{?+}$-DTD の真の部分クラスであるため，本論文で対象とする XPath 式の充足可能性判定問題は DC$^{?+}$-DTD のもとで NP 完全である．

さらに，本論文の定理 1 と文献 [3] の補題 4.5 より，以下の定理が成り立つ．

定理 2 DC$^{?+}$-DTD $D = (\Sigma, r, P)$ のスキーマグラフを G とする．$(\bot, 1, -, r)$ から始まる空でない G 上の経路 s_1，s_2 が存在して $G \models p(s_1, s_2)$ が成り立つとする．このとき，経路長がたかだか $(3|p| - 1)|\Sigma|$ であるような，$(\bot, 1, -, r)$ から始まる空でない G 上の経路 s_1'，s_2' が存在して，$G \models p(s_1', s_2')$ が成り立つ． □

この定理は充足可能性判定の NP アルゴリズムを示唆している．すなわち，$G \models p((\bot, 1, -, r), s)$ を満たすような，長さがたかだか $(3|p| - 1)|\Sigma|$ の G 上の経路 s を推測すればよい．

3.3 上向き軸を含まない XPath 式に対する多項式時間アルゴリズム

軸や演算子として $\cdot$，$\downarrow$，$\downarrow^*$，$\rightarrow^+$，$\leftarrow^+$，$/$，$\cup$，$[\,]$，$\wedge$，$\vee$ のみを用いた XPath 式を，本論文では「上向き軸を含まない XPath 式」と呼ぶ．

p が上向き軸を含まない XPath 式のとき，$G \models p(s, s')$ の計算は s や s' の最終ノードのみに着目すればよいことが，以下の補題により保証される．

補題 3 $G \models p(su, s'u')$ が成り立つと仮定する．$(\bot, 1, -, r)$ から u への任意の G 上の経路 s_1u に対し，$(\bot, 1, -, r)$ から u' への G 上の経路 $s_1'u'$ が存在して $G \models p(s_1u, s_1'u')$ が成り立つ．同様に，$G \models q(su)$ が成り立つと仮定すると，$(\bot, 1, -, r)$ から u への任意の G 上の経路 s_1u に対し $G \models q(s_1u)$ が成り立つ．

略証： G による充足関係の定義より，上向き軸を含まない XPath 式においては，第 1 引数の経路の最終ノードのみに依存して充足関係が定まる．このことを用いて，p や q の構造に関する帰納法で題意を証明できる． □

したがって，G 上のすべての経路の集合を，最終ノードの等価性に基づく $|U|$ 個の同値類に分割し，同値類に対し

© 2016 Information Processing Society of Japan

て G が p を満たすかを計算すればよい．最終ノードが u である経路からなる同値類を単に u と書くと，上向き軸を含まない XPath 式 p の充足可能性は，以下で定義する関数 $eval_1(p)$ の結果に $((\bot, 1, -, r), u')$ なる u' が存在するかどうかで判定できる．

- p が原子式のとき，$eval_1(p) = \{(u, u') \mid \exists s, s', G \models p(su, s'u')\}$.
- $p = p_1/p_2$ のとき，$eval_1(p) = \{(u, u') \mid (u, u'') \in eval_1(p_1), (u'', u') \in eval_1(p_2)\}$.
- $p = p_1 \cup p_2$ のとき，$eval_1(p) = eval_1(p_1) \cup eval_1(p_2)$.
- $p = p_1[q]$ のとき，$eval_1(p) = \{(u, u') \mid (u, u') \in eval_1(p_1), u' \in eval'_1(q)\}$.
- $eval'_1(p) = \{u \mid (u, u') \in eval_1(p)\}$.
- $q = q_1 \wedge q_2$ のとき，$eval'_1(q) = eval'_1(q_1) \cap eval'_1(q_2)$.
- $q = q_1 \vee q_2$ のとき，$eval'_1(q) = eval'_1(q_1) \cup eval'_1(q_2)$.

p が原子式のとき，$eval_1(p)$ の計算は $O(|U|^2)$ 時間で行え，結果のサイズ $|eval_1(p)|$ は $O(|U|^2)$ である．$p = p_1/p_2$ のとき，$eval_1(p_1)$ と $eval_1(p_2)$ の結合には $O(|U|^4)$ 時間あれば十分であり，結果のサイズは $O(|U|^2)$ である．p が他の形をしているときも同様である．

以上の議論より，次の定理が得られる．

定理 3 p を「上向き軸を含まない XPath 式」とし，D を DC$^{?+}$-DTD とする．p が D のもとで充足可能であるときかつそのときのみ $((\bot, 1, -, r), u') \in eval_1(p)$ なる u' が存在する．これは，スキーマグラフの生成も含めて $O(|p||D|^4)$ 時間で判定できる．

例 4 例 2 の DC$^{?+}$-DTD D のもとでの XPath 式 $p = \downarrow :: b[\downarrow^* :: r]/ \rightarrow^+ :: a$ の充足可能性を考える．D のスキーマグラフを G とする．まず，p の各原子式 p_0 について，$G \models p_0(u, u')$ を満たすスキーマグラフのノードの組 (u, u') を**表 3** のように列挙する．次にこの表から $G \models (\downarrow :: b[\downarrow^* :: r])(u, u')$ を満たすノードの組 (u, u') をすべて求める．G による充足関係の [ ] の場合の定義より，(u_0, u_2)，(u_5, u_2) が求まる．最後に，$G \models p(u, u')$ を満たすノードの組 (u, u') をすべて求めると，(u_0, u_4)，(u_5, u_4) が求まる．$G \models p(u_0, u')$ を満たす u' が存在することが分かったため，p は D のもとで充足可能であると判定する． □

一方，被覆 DTD のもとでは，上向き軸を含まない XPath 式の充足可能性判定は NP 完全である．

表 3 各原子式を充足するノードの組
Table 3 Pairs of nodes satisfying each atomic XPath expression.

原子式	原子式を満たすノードの組
$\downarrow :: b$	$(u_0, u_2), (u_5, u_2)$
$\downarrow^* :: r$	$(u_0, u_0), (u_0, u_5), (u_2, u_5), (u_5, u_5)$
$\rightarrow^+ :: a$	$(u_1, u_1), (u_1, u_4), (u_2, u_4), (u_3, u_4), (u_4, u_4)$

定理 4 p を，$\downarrow^*$，$\rightarrow^+$，/，[ ] のみからなる XPath 式とする．被覆 DTD のもとでの p の充足可能性判定は NP 完全である．

証明： 3.2 節と同様の議論により，NP に属することを証明できる．以下，NP 困難性を示す．

以下のような 3SAT 問題のインスタンス ϕ を考える．

$$\phi = (L_{1,1} \vee L_{1,2} \vee L_{1,3}) \wedge \cdots \wedge (L_{n,1} \vee L_{n,2} \vee L_{n,3})$$

ここで各 $L_{i,j}$ は $x_1, \ldots, x_m, \bar{x}_1, \ldots, \bar{x}_m$ のうちのいずれかである．DC$^{?+}$-DTD $D = (\Sigma, r, P)$ を次のように定義する．

- $\Sigma = \{r, x_0, x_1, \ldots, x_m, t, f\}$,
- $P(r) = x_0$,
  $P(x_k) = (tx_{k+1}f \mid fx_{k+1}t)$ $(0 \le k \le m-1)$,
  $P(x_m) = \epsilon$.

各木 $T \in TL(D)$ と各変数 x_k $(1 \le k \le m)$ について，T は x_k とラベルづけされたノード v_k をちょうど 1 つ持つ．T は以下のような真理値割当てを表していると考える：v_k の右の兄弟がラベル t であるとき，x_k は真であり，そうでないとき，x_k は偽である．

XPath 式 p を次のように定義する．

$$p = \downarrow^* :: x_0[(q_{1,1} \vee q_{1,2} \vee q_{1,3}) \wedge \cdots \wedge (q_{n,1} \vee q_{n,2} \vee q_{n,3})].$$

ここで

$$q_{i,j} = \begin{cases} \downarrow^* :: x_k \,/\, \rightarrow^+ :: t & L_{i,j} = x_k \text{ のとき}, \\ \downarrow^* :: x_k \,/\, \rightarrow^+ :: f & L_{i,j} = \bar{x}_k \text{ のとき}. \end{cases}$$

x_i とラベルづけされたノードはちょうど 1 つしかないことと，その右の兄弟が x_i に割り当てられる真理値を表していることから，この帰着が正しいことは容易に確認できる． □

なお，$\rightarrow^+$ の代わりに $\leftarrow^+$ を用いても NP 完全性を証明できる．さらに，$\downarrow^*$ の代わりに $\downarrow$ を用いても NP 完全性を証明できる．

3.4 述語と先祖子孫軸を含まない XPath 式に対する多項式時間アルゴリズム

軸や演算子として $\cdot$，$\downarrow$，$\uparrow$，$\rightarrow^+$，$\leftarrow^+$，/ のみを用いた XPath 式を，本論文では「述語と先祖子孫軸を含まない XPath 式」と呼ぶ．

p が述語と先祖子孫軸を含まない XPath 式であるとし，S をスキーマグラフ $G = (U, E)$ において $(\bot, 1, -, r)$ から始まる空でない経路の有限集合とする．以下で定義される関数 $eval_2(p, S)$ は，ある $s \in S$ について $G \models p(s, s')$ を満たすようなすべての s' からなる集合を返す．したがって，$eval_2(p, \{(\bot, 1, -, r)\})$ が空集合でないときかつそのときのみ p は充足可能である．

© 2016 Information Processing Society of Japan

$$eval_2(p,S)=\begin{cases}\{s' \mid \text{各 } s\in S \text{ について } G\models p(s,s')\} \\ \qquad (p \text{ が原子式のとき}),\\ eval_2(p_2, eval_2(p_1,S)) \\ \qquad (p=p_1/p_2 \text{ のとき}).\end{cases}$$

p が含む $\downarrow$ 軸，$\rightarrow^+$ 軸，$\leftarrow^+$ 軸では，行き先のノードが複数存在しうるため，一般に $|S|$ は $|p|$ に対して指数的となる．したがって，$eval_2$ を多項式時間で計算するためには，S の表現方法に工夫が必要である．まず，原子式の評価において経路の最後のノードだけが重要であることを示す．

補題 4 $p=\chi :: l$（ただし $\chi\in\{\cdot,\downarrow,\uparrow,\rightarrow^+,\leftarrow^+\}$）とし，$G\models p(su,ss')$ が成立するとする．このとき，$(\perp,1,-,r)$ から u までの G 上の任意の経路 $s''u$ について，$G\models p(s''u,s''s')$ が成り立つ．

証明： $p=\uparrow :: l$ とする．$G\models p(su,ss')$ を満たす s' は空系列のみである．したがって，s の最終ノードのラベルは l であり，$\lambda_{par}(u)=l$ である．よって，経路 $s''u$ が G に存在するならば，s'' の最終ノードのラベルは l でなければならず，$G\models p(s''u,s'')$ が成立する．他の場合は G による充足関係の定義より明らか． □

さらに，p を「述語と先祖子孫軸を含まない XPath 式」とすると，S 中の経路がすべて同じ長さであれば，$eval_2(p,S)$ 中の経路もすべて同じ長さになる．よって，$eval_2(p,\{(\perp,1,-,r)\})$ の計算においては，経路集合をノード集合の系列 $U_0U_1\cdots U_m$ で表すことができる．ここで $U_0=\{(\perp,1,-,r)\}$ であり，$s=u_0u_1\cdots u_m$ が $U_0U_1\cdots U_m$ の表す経路集合に属するのは各 $0\le i\le m$ について $u_i\in U_i$ のときかつそのときのみである．以上をふまえて，原子式に対する関数 $eval_2$ の詳細仕様を示すと，以下のようになる（$\leftarrow^+$ は $\rightarrow^+$ と同様なので省略する）．

- $eval_2((\cdot :: l),U_0\cdots U_{m-1}U_m)=U_0\cdots U_{m-1}U'_m$，ただし $U'_m=\{u'\in U_m \mid \lambda(u')=l\}$．
- $eval_2((\downarrow :: l),U_0\cdots U_m)=U_0\cdots U_mU_{m+1}$，ただし，$U_{m+1}=\{u' \mid \exists u\in U_m,(u,u')\in E,\lambda(u')=l\}$．
- $eval_2((\uparrow :: l),U_0\cdots U_{m-1}U_mU_{m+1})=U_0\cdots U_{m-1}U'_m$，ただし $U'_m=\{u'\in U_m \mid \lambda(u')=l\}$．
- $eval_2((\rightarrow^+ :: l),U_0\cdots U_{m-1}U_m)=U_0\cdots U_{m-1}U'_m$，ただし，$U'_m=\{u' \mid \exists u\in U_m,\ (\omega(u)\text{ が }-\text{ ならば } pos(u)<pos(u')),\ (\omega(u)\text{ が }*\text{ ならば } pos(u)\le pos(u')),\ \lambda(u')=l\}$．

原子式に対する $eval_2$ の計算は $O(|U|^2)$ 時間で行える．

以上の議論より，次の定理が得られる．

定理 5 p を「述語と先祖子孫軸を含まない XPath 式」とし，D を DC$^{?+}$-DTD とする．p が D のもとで充足可能であるときかつそのときのみ $eval_2(p,\{(\perp,1,-,r)\})\neq\emptyset$ が成立する．これは，スキーマグラフの生成も含めて $O(|p||D|^2)$ 時間で判定できる．

表 4 開発環境
Table 4 Development environment.

言語	Python 2.7.5
OS	Darwin Kernel Version 13.0.0

4. 実装

前章で述べた 2 種類の充足可能性判定多項式時間アルゴリズムを実装した．それぞれ，$eval_1$ と $eval_2$ の経過を出力するようにした．開発環境は**表 4** のとおりである．

上向き軸を含まない XPath 式に対する実装において，次の 2 通りの工夫を施した．

4.1 原子式の処理の並列化

上向き軸を含まない XPath 式 p に対する多項式時間アルゴリズムでは，まず p の各原子式が充足するスキーマグラフのノードの組を求める．この処理は原子式ごとに独立に行うことが可能であるので，プロセスベースで並列に処理することによって判定の効率化を図った．まず，PC の CPU のコア数に合わせてプロセスを生成する．そして生成された各プロセスに対して，p の原子式とスキーマグラフを引数として与え，これらを充足するようなノードの組の集合を得る．p のすべての原子式を処理し終えるまで，プロセスが処理を完了するたびに，原子式とスキーマグラフを与え，並列に処理を行うようにした．実装した並列処理を擬似コードで示すと次のようになる．

```
 1 G;            // スキーマグラフ
 2 p;            // XPath 式
 3 tuple_array; // 原子式を満たすノードの組の配列
 4
 5 process = Pool(); // プロセスの生成
 6 atomic_expression_array = split(p);
 7 parm_array;  // プロセスに与える引数
 8 for (atomic_exp in atomic_expression_array){
 9     parm_array.append((G, atomic_exp));
10 }
11 tuple_array = process.map(parm_array);
```

4.2 スキーマグラフの探索結果のキャッシュ

$G=(U,E)$ をスキーマグラフとする．上向き軸を含まない XPath 式に対する多項式時間アルゴリズムでは，子孫軸 $\downarrow^*$ を含む原子式に対し，スキーマグラフ上で到達可能なノードの組を求めなければならない．この計算には最悪で $O(|U|^2)$ の時間を要する．そこで，初めて子孫軸を含む原子式を処理したときに，その探索結果をメモリ上にキャッシュしておき，処理中に再び子孫軸を含む原子式の処理が発生した場合，メモリ上の探索結果を参照するようにした．

© 2016 Information Processing Society of Japan

表 5 実行環境
Table 5 Execution environment.

CPU	2.3 GHz Intel Core i7
OS	Darwin Kernel Version 13.0.0
RAM	8 GB
言語	Python 2.7.5

5. 単体評価

実装したプログラムを実行し，評価を行った．評価は**表 5** のような実験環境で行った．ベンチマークとして，XMark [8] をベースにした DTD と，XPathMark [9] をベースとした XPath 式を用いた．XMark は内容モデルが $DC^{?+}$ ではないラベルを 1 つ持つため，その部分だけを $DC^{?+}$ となるように変更した DTD を用いた．具体的には，ラベル description の内容モデル (text|parlist) を (text|parlist)* に変更した．実行結果は，5 回実行した実行時間の平均の値をとった．

以降では，見やすさのため，↓:: を省略して記す．たとえば，site/closed_auctions は ↓::site/↓::closed_auctions を意味する．

5.1 実行結果

5.1.1 上向き軸を含まない XPath 式に対する実行結果

以下に記した，XPathMark に含まれる XPath 式 A1–A8 に対する実行結果を**表 6** に示す．いずれの場合も 20 ミリ秒前後で充足可能性を判定できた．また，XPath 式 A2，A3 のみが 20 ミリ秒以上かかっているが，子孫軸を含む XPath 式の処理に時間を要していると考えられる．

A1: site/closed_auctions/closed_auction/annotation/description/text/keyword
A2: ↓*::closed_auction/↓*::keyword
A3: site/closed_auctions/closed_auction/↓*::keyword
A4: site/closed_auctions/closed_auction[annotation/description/text/keyword]/date
A5: site/closed_auctions/closed_auction[↓*::keyword]/date
A6: site/people/person[profile/gender ∧ profile/age]/name
A7: site/people/person[phone ∨ homepage]/name
A8: site/people/person[address ∧ (phone ∨ homepage) ∧ (creditcard ∨ profile)]/name

5.1.2 述語と先祖子孫軸を含まない XPath 式に対する実行結果

述語と先祖子孫軸を含まない XPath 式が XPathMark には存在しなかったため，XPathMark で与えられている式を少し改変して以下のような XPath 式 B1–B4 を用意した．実行結果を**表 7** に示す．いずれの場合も十数ミリ秒で充足可能性を判定できた．

B1: site/regions/samerica/item/↑::samerica/←+::europe
B2: site/categories/category/description/text/→+::parlist/↑::description
B3: site/open_auctions/open_auction/bidder/↑::open_auction/↑::open_auctions/→+::closed_auctions
B4: site/open_auctions/open_auction/↑::open_auctions/↑::site/closed_auctions/closed_auction

表 6 上向き軸を含まない XPath 式についての実行結果
Table 6 The execution result of XPath expressions without upward axis.

XPath 式	実行時間 [ms]
A1	16.278
A2	25.959
A3	22.681
A4	15.876
A5	18.961
A6	16.290
A7	15.734
A8	16.450

表 7 述語と先祖子孫軸を含まない XPath 式についての実行結果
Table 7 The execution result of XPath expressions without qualifier, ancestor-or-self axis and descendant-or-self axis.

XPath 式	実行時間 [ms]
B1	10.802
B2	11.650
B3	11.376
B4	11.282

5.1.3 実装したアルゴリズムについての評価

今回は多項式時間アルゴリズムを Python で実装したが，一般的な PC でも数十ミリ秒で充足可能性を判定できることが判明した．したがって，これらのアルゴリズムは，問合せ最適化など高速性を要求される場面にも利用可能であると考えられる．

5.2 実装上の工夫による効果

4 章で述べたとおり，本実装では 2 通りの工夫を行った．その工夫による効果を評価する．

5.2.1 原子式の処理の並列化による効果

表 8 に示したとおり，並列化を行った方が遅くなるという結果となった．その理由としては，スキーマグラフの探索処理に要する時間に比べ，並列化のためにプロセスを生成するオーバヘッドの方が大きいということが考えられる．生成されるプロセスの個数はつねに一定であり，入力に依存しないため，プロセスの生成に要するオーバヘッドはつねに等しい．実際，プロセスの生成に要する時間を調

© 2016 Information Processing Society of Japan

表 8 並列化の有無による実行時間の比較
Table 8 Comparison of execution time with and without parallelization.

XPath 式	並列化あり [ms]	並列化なし [ms]
A1	39.864	16.278
A2	40.713	25.959
A3	38.751	22.681
A4	36.766	15.876
A5	36.499	18.961

表 9 プロセスの生成に要する時間
Table 9 The time required for process generation.

XPath 式	実行時間 [ms]
A1	16.278
A2	15.055
A3	16.518
A4	15.876
A5	15.659

表 10 探索結果のキャッシュの有無による実行時間の比較
Table 10 Comparison of execution time with and without search result cache.

XPath 式	キャッシュあり [ms]	キャッシュなし [ms]
A2	25.646	32.012
A3	22.131	22.547
C1	19.742	73.132

べたところ，**表 9** に示すようにつねに 16 [ms] 程度要しており，入力に依存していないことが分かる．したがって，スキーマグラフの探索処理に時間を要するようなデータサイズの大きい DTD では，並列化の効果が期待できると考えられる．

5.2.2 スキーマグラフの探索結果のキャッシュによる効果

スキーマグラフの探索結果のキャッシュの有無による実行時間の違いを**表 10** に示す．キャッシュを利用するほうが最大 3 倍程度高速に判定できた．また，再利用できる回数が多いほど，高速化が顕著になることが読み取れる．

A2: ↓*::closed_auction/↓*::keyword
A3: site/closed_auctions/closed_auction/↓*::keyword
C1: site/↓*::regions/↓*::samerica/↓*::item/
↓*::description/↓*::parlist/↓*::listitem

6. 比較評価

本章では，多項式時間判定アルゴリズムを実装したプログラムとソルバを用いた判定手法との比較を述べる．本比較では，ベンチマークとして 5 章と同様に，XMark を 1 カ所変更した $DC^{?+}$-DTD と，XPathMark やアルゴリズムの性能の評価に適当だと考えられる XPath 式を用いた．

6.1 XML Reasoning Solver との比較

1 章で述べた Genevès らの手法 [5], [6], [7] を実装したものとして，XML Reasoning Solver Project [12] がある．XML Reasoning Solver はオンライン環境でのみでしか実行できず，多項式時間アルゴリズムと同一の実験環境での比較が行えなかった．そこで本比較では，XPath 式サイズの変化による CPU 時間の比の比較を行った．その結果を**図 3** に示す．図 3 では XPath 式サイズを横軸にとり，縦軸に XPath 式サイズが 1 のときの実行時間をもとにした CPU 時間との比を比較した．また，比較に用いた XPath は以下の問合せを根から指定の XPath 式サイズに合わせたものを用いた．

- ↓*::site/↓*::regions/↓*::asia/↓*::item/↓*::description/↓*::parlist/↓*::listitem/↓*::text/↓*::keyword/↓*::emph

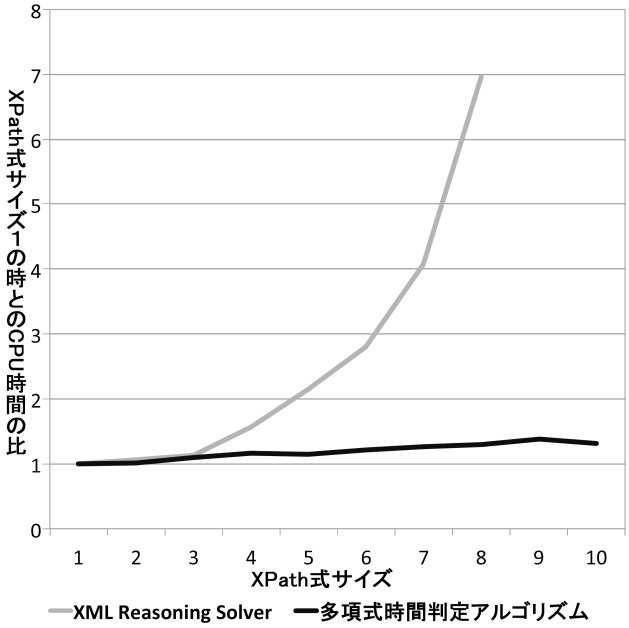

図 3 XML Resoning Solver と実装したプログラムとの比較
Fig. 3 Comparison between XML Reasoning Solver and our implementation.

本比較における XPath 式サイズ 1 のときの CPU 時間は，XML Reasoning Solver では 1739 [ms]，多項式時間判定アルゴリズムでは 12.986 [ms] である．また，XPath 式サイズが 9，10 のときは，XML Reasoning Solver ではタイムアウトになり判定できなかった．本比較結果から，多項式時間アルゴリズムでは XPath 式サイズが大きくなっても実行時間の変化が小さいのに対し，XML Reasoning Solver では XPath サイズが大きくなるに従って実行時間が大きく変化していることが分かる．

6.2 Sugar: SAT-based Constraint Solver との比較

定理 2 で示唆されている NP アルゴリズムを制約ソルバの Sugar [13] を用いて実装した．Sugar とは与えられた制約充足問題を命題論理の充足可能性判定問題に変換し，SAT ソルバを用いて解を求める SAT 型の制約ソルバである．Sugar では標準で，高速な SAT ソルバとして知られて

© 2016 Information Processing Society of Japan

表 11 Sugar での実行結果
Table 11 The execution result by Sugar.

XPath 式	サイズ	実行時間 [s]	Encoding CPU [s]	Solving CPU [s]
A1	7	429.67	377.37	52.24
A2	2	99.82	41.44	58.35
A3	4	347.25	166.31	180.92
A4	8	1202.52	1112.41	90.04
A5	5	399.1	204.97	194.04
A6	8	1022.01	934.19	87.78
A7	6	2918.28	333.45	2584.79
A8	9	8243.19	844.67	7398.43
B1	6	492.23	392.22	99.95
B2	7	779.72	662.02	117.62
B3	7	855.86	596.76	259.02
B4	7	762.99	673.89	89.07

表 12 多項式時間アルゴリズムと Sugar におけるメモリ使用量の比較結果
Table 12 Comparison of memory usage between our polynomial-time algorithms and Sugar.

XPath 式	サイズ	多項式時間アルゴリズム [MB]	Sugar [MB]
A1	7	6.3164	1582.75
A2	2	6.6289	315.06
A3	4	6.5273	823.75
A4	8	6.2929	2657.62
A5	5	6.4687	883.07
A6	8	6.3710	2695.72
A7	6	6.3281	1289.69
A8	9	6.3789	2608.21
B1	6	6.4062	1418.42
B2	7	6.3476	2070.54
B3	7	6.3320	1827.90
B4	7	6.3164	2067.72

いる miniSAT [14] が用いられている．

Sugar を用いて 5 章で与えた A1–A8 と B1–B4 について充足可能性を判定した実行結果を**表 11** に示す．ここで，Encoding CPU とは XPath 式充足可能性判定問題を miniSAT で解くことのできる論理式に変換するのに要する CPU 時間のことである．また，Solving CPU とは変換した論理式を解くのに要する CPU 時間のことである．この表から，A7 を除いて XPath 式サイズが大きいほど CPU 時間が大きくなっていることが分かり，最大 2 時間 17 分ほど要していることが分かる．この Sugar の実行結果と表 6，7 で示した多項式時間アルゴリズムの実行結果を比較すると，CPU 時間に大きく差があることが分かる．さらに，Solving CPU のみと比較しても，多項式時間アルゴリズムと CPU 時間に大きく差があることが分かる．

次にメモリ使用量についての比較を**表 12** に示す．Sugar では，使用量の最も大きいもので 2 GB 以上メモリを消費しており，ほぼ XPath 式のサイズに従ってメモリ使用量が増加していることが分かる．一方，多項式時間アルゴリズムではどの XPath 式に対しても 6 MB 程度の消費量で，XPath 式サイズにかかわらず，ほぼ一定であることが分かる．

なお，Sugar での CPU 時間とメモリ使用量が大きく XPath 式サイズに依存しているのは，定理 2 で示唆されているアルゴリズムにおいて推測すべき経路の長さはたかだか $(3|p|-1)|\Sigma|$ とされており，XPath 式サイズ p が増加するに従って推測すべき経路の長さが増加するためと考えられる．従って，推測すべき経路長を小さく見積もることが可能ならば，高速化することが可能であると考えられる．

6.3 多項式時間アルゴリズムの有用性

ソルバを用いた手法は，DTD や XPath 式のより広いクラスを対象として，それらを統一的に扱うことができるという長所がある．しかし，本章での比較より，多項式時間アルゴリズムはソルバを用いた手法に比べて十分高速に動作し，スケーラビリティも高いことが分かった．多項式時間アルゴリズムは，理論的観点だけでなく実用上の観点からも十分に意義があるといえる．

7. あとがき

本論文では，$DC^{?+}$-DTD と呼ばれる DTD クラスのもとで，上向き軸を含まない XPath 式クラスと，述語と先祖子孫軸を含まない XPath 式クラスとに対する充足可能性判定問題が多項式時間可解であることを示した．さらに，判定アルゴリズムを実装し，その評価を行った．実装したプログラムは，一般的に用いられる XML のベンチマークに対して一般的な PC 上で数十ミリ秒で動作し，ソルバを用いた手法と比べて十分に高速に判定できることを確認した．これにより，提案された多項式時間判定アルゴリズムは，理論的観点だけでなく実用上の観点からも十分に意義があるといえる．

今後は，さらに広い DTD や XPath 式のクラスに対して充足可能性判定多項式時間アルゴリズムを提案・実装し，実世界における DTD と XPath 式に対応していく予定である．

参考文献

[1] Ishihara, Y., Morimoto, T., Shimizu, S., Hashimoto, K. and Fujiwara, T.: A Tractable Subclass of DTDs for XPath Satisfiability with Sibling Axes, *Proc. 12th International Symposium on Database Programming Languages*, Lecture Notes in Computer Science, Vol.5708, pp.68–83 (2009).

[2] Ishihara, Y., Shimizu, S. and Fujiwara, T.: Extending the Tractability Results on XPath Satisfiability with Sibling Axes, *Proc. 7th International XML Database Symposium*, Lecture Notes in Computer Science, Vol.6309,

© 2016 Information Processing Society of Japan

pp.33–47 (2010).
[3] Benedikt, M., Fan, W. and Geerts, F.: XPath Satisfiability in the Presence of DTDs, *J. ACM*, Vol.55, No.2, pp.8:1–8:79 (2008).
[4] Montazerian, M., Wood, P.T. and Mousavi, S.R.: XPath Query Satisfiability is in PTIME for Real-World DTDs, *Proc. 5th International XML Database Symposium*, pp.17–30 (2007).
[5] Genevès, P. and Layaïda, N.: A System for the Static Analysis of XPath, *ACM Trans. Inf. Syst.*, Vol.24, No.4, pp.475–502 (2006).
[6] Genevès, P. and Layaïda, N.: Deciding XPath Containment with MSO, *Data Knowl. Eng.*, Vol.63, No.1, pp.108–136 (2007).
[7] Genevès, P., Layaïda, N. and Schmitt, A.: Efficient Static Analysis of XML Paths and Types, *SIGPLAN Not.*, Vol.42, No.6, pp.342–351 (2007).
[8] Schmidt, A., Kersten, M., Florescu, D., Carey, Michael, M.J., Manolescu, I. and Waas, F.: The XML StoreBenchmark Project (2000), available from ⟨http://www.xml-benchmark.org⟩.
[9] Franceschet, M.: XPathMark: An XPath Benchmark for the XMark Generated Data, *Proc. 3rd International XML Database Symposium*, Lecture Notes in Computer Science, Vol.3671, pp.129–143 (2005).
[10] W3C: Extensible Markup Language (XML) 1.1 (2nd Edition), available from ⟨http://www.w3.org/TR/xml11/⟩.
[11] W3C: XML Path Language (XPath), available from ⟨http://www.w3.org/TR/xpath/⟩.
[12] Genevès, P., Layaïda, N., Schmitt, A., Gesbert, N. and Knyttl, V.: XML Static Analysis and Type Checking: Online Web Solver, available from ⟨http://tyrex.inria.fr/websolver/⟩.
[13] Tamura, N., Taga, A., Kitagawa, S. and Banbara, M.: Compiling finite linear CSP into SAT, *Constraints*, Vol.14, No.2, pp.254–272 (2009).
[14] Eén, N. and Sörensson, N.: An extensible SAT-solver, *Proc. 6th International Conference on Theory and Applications of Satisfiability Testing*, pp.502–518 (2003).

推薦文

関西支部では，推薦論文の検討対象として支部大会を利用することとした．そこで支部大会で口頭発表された論文のうち，6 ページに満たないものを除く 23 件を対象とし，各セッションの座長，実行委員から広く推薦を集めて候補を 4 件に絞り，各論文に対し事後評価者 2 名の評価を加え，実行委員会による審議を経て 3 件の推薦論文候補を選定し，上位 2 件をイベント推薦枠から，残り 1 件を支部年間推薦枠から推薦することとした．

本論文は，XML 文書の特定の要素を指定する問合せ言語として広く用いられた XPath の充足可能性判定のための多項式時間判定アルゴリズムを実装し，その実行時間を計測することによって効率性を検証したものであり，その有用性は高く評価できるものであり，推薦論文にふさわしいと実行委員会で判断された．

（関西支部支部長　藤原　融）

杉村 憲司　（学生会員）

2014 年大阪大学基礎工学部情報科学科卒業．2016 年大阪大学大学院情報科学研究科博士前期課程在学中．データベース理論や情報セキュリティに関心を持つ．

石原 靖哲　（正会員）

1990 年大阪大学基礎工学部情報工学科卒業．1994 年大阪大学大学院基礎工学研究科博士後期課程退学．奈良先端科学技術大学院大学助手等を経て，2007 年より大阪大学大学院情報科学研究科准教授．博士（工学）．データベース理論や情報セキュリティに関心を持つ．ACM，IEEE，電子情報通信学会各会員．

藤原 融　（正会員）

1981 年大阪大学基礎工学部情報工学科卒業，1986 年大阪大学大学院基礎工学研究科物理系専攻情報工学分野博士後期課程修了，工学博士．同年大阪大学基礎工学部助手．講師，助教授を経て，1997 年大阪大学大学院基礎工学研究科教授．1998～2000 年奈良先端科学技術大学院大学情報科学研究科教授（併任），2004 年大阪大学大学院情報科学研究科教授．符号理論，情報セキュリティに関する研究に従事．2013～2015 年関西支部支部長．電子情報通信学会フェロー．IEEE，ACM 各会員．

© 2016 Information Processing Society of Japan