限定されたXPath構文に関する充足可能性問題について

福島 雄司† 鈴木 伸崇†

筑波大学大学院図書館情報メディア研究科†

1. まえがき

DTD D とXPath式 p に対して，p に対する問合せ結果が空でなく，かつ D に関して妥当なXMLデータが存在し得るか否かを決定する問題をXPath充足可能性問題という．充足不能なXPath式を実行するのは無意味であり，このようなXPath式はコンパイル時に検出できることが望ましい．

XPath充足可能性問題は一般に決定不能であり，XPath式に様々な制約を加えても効率よく解けないことが知られている[1]．このため，効率よく解くことのできる，同問題の部分クラスを発見することは重要な課題である．本稿では，XPath式のノードテストは要素名のみ，かつ述語を用いないとの仮定の下で，以下の結果を示す．まず，(i)DTDは*を含まず，XPath式の軸をchild, following-sibling, preceding-siblingに限定する，(ii) DTDはduplicate-free[2]かつ*を含まず，XPath式の軸をchild, parent, descendantに限定する，いずれの場合も同問題がNP完全となることを示す．次に，XPath式の軸を上記(i)と(ii)の5種に限定し，DTDを*を含まないduplicate-freeかつ非再帰なものに限定した場合に，同問題を多項式時間で解くアルゴリズムを示す．最後に，このアルゴリズムに関する評価実験について述べる．

関連研究として，文献[1]のほか，文献[2,5]等がある．文献[2]では，祖先方向の軸(parentとancestor)を用いず，DTDに2種の制約(duplicate-freeとcovering)を課した場合に，同問題が多項式時間可解となる十分条件を示している．文献[5]では，XPath式の軸はchildとparentのみ，ノードテストは要素名のみ，かつ，述語を用いないという制約の下でも同問題がNP完全であり，更にDTDがduplicate-freeであると仮定すると同問題が多項式時間可解となることを示している．

2. 諸定義

Σを要素名の集合とする．DTDを組(d,r)と表す．ここで，dはΣからΣ∪{#PCDATA}上の正規表現集合への写像，$r \in$ Σは文書要素である．以下，XPath式として，XPathの仕様[3]における絶対ロケーションパスで，ノードテストは要素名のみ，かつ述語を用いないものを考える．D をDTD，p をXPath式とする．D に関して妥当，かつ，p の問合せ結果が空でないXMLデータが存在するとき，p は D の下で充足可能であるという．XPath充足可能性問題とは，p が D の下で充足可能であるか否かを決定する問題のことをいう．

3. NP完全性

本節では，**1.**で述べた(i)および(ii)いずれの条件下でも，XPath充足可能性問題がNP完全となることを示す．

[定理 1] XPath式の軸をchild, following-sibling, preceding-siblingのみに限定し，DTDは*を含まないと仮定しても，XPath充足可能性問題はNP完全である．

On XPath Satisfiability Problem under Restricted XPath Syntax
†Graduate School of Library, Information and Media Studies, University of Tsukuba

[証明] この問題がNPに属することは明らか．この問題がNP困難であることを，3SATからの帰着により示す．3SATのインスタンスを $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$ とする(一般性を失わず，n は偶数と仮定する)．ここで，節 C_i は3個のリテラルの論理和である．ϕ に出現する変数を $x_1,x_2,\cdots,x_m$ とする．ϕ からXPath充足可能性問題のインスタンスを構成する．まず，DTD $D=(d,r)$ を次のように定義する．

$$d(r) = b(T_1 \mid F_1)(T_2 \mid F_2) \cdots (T_m \mid F_m)b,$$
$$d(C_i) = d(b) = \#\mathrm{PCDATA} \quad (1 \leqq i \leqq n)$$

ここで，T_i と F_i は要素名(節の名前)の系列であり，次のように定義される．まず，T_i は正リテラル x_i を含む節の名前を列挙したものである．すなわち，正リテラル x_i を含む節が節 $C_{i1},\cdots,C_{ik}$ であるとき，$T_i = C_{i1}\cdots C_{ik}$ となる．同様に，F_i は負リテラル$\neg x_i$ を含む節の名前を列挙したものである．次に，XPath式 p を次のように定義する．

p = /child::r/child::b
/following-sibling::C_1/following-sibling::b
/preceding-sibling::C_2/preceding-sibling::b
…
/following-sibling::C_{n-1}/following-sibling::b
/preceding-sibling::C_n

p は，r の子要素として $C_1,C_2\cdots,C_n$ すべてをもつXMLデータに対してのみ問合せ結果が空でない．このことから，ϕ が充足可能であるときかつそのときのみ p が D の下で充足可能であることが容易に示せる． □

DTD D のどの内容モデルも「同じ要素を複数含むことが無い」ものである場合，D はduplicate-freeであるという．例えば，$a(a|b)^*$ という内容モデルをもつDTDは a を2個含むのでduplicate-freeでない．次の定理が成り立つ (紙数の都合で証明は省略する)．

[定理 2] XPath式の軸をchild, parent, descendantのみに限定し，DTDはduplicate-freeかつ*を含まないと仮定しても，XPath充足可能性問題はNP完全である． □

4. アルゴリズム

前節の結果から，(a)XPath式の軸をchild, parent, descendant, following-sibling, preceding-siblingのみに限定し，かつ(b)DTDがduplicate-freeかつ*を含まない場合でも，XPath式の充足可能性を効率よく判定することは困難である．しかし，DTDに非再帰という制約を加えると，同問題を多項式時間で解くことのできるアルゴリズムが構成できる．

このアルゴリズムは，XPath式 p のロケーションステップを前から順に処理し，木 t を構成していく．t において，次のロケーションステップの処理対象となるノードには印が置かれる．descendant軸は1つの軸から複数のノードに到達できるので，印は複数存在し得る．なお，上記(a)と(b)にDTD D が非再帰という条件を加えた場合，このアルゴリズムは D が*を含む場合でも正常に動作するが，多項式時間で動作

することが示されているのは D が*を含まない場合のみである．

入力：XPath 式 p = /axis[1]::nt[1]/axis[2]::nt[2]/ … /axis[n]::nt[n], DTD $D = (d, r)$
出力：p が D の下で充足可能ならば “yes”，そうでなければ “no”
1. ルートノードを作成し，そこに印を置く．ルートノードのラベルは root, d(root) = r と仮定する．
2. 各 i = 1 ... n において以下を行う．
3. t 中の各印 m に対して以下を行う．
4. axis[i] = child の場合：
5. process-child(m);
6. axis[i] = parent の場合：
7. process-parent(m);
8. axis[i] = following-sibling の場合：
9. process-following-sibling(m);
10. axis[i] = preceding-sibling の場合：
11. process-preceding-sibling(m);
12. axis[i] = descendant の場合：
13. process-descendant(m);
14. t に印が存在すれば“yes”，そうでなければ “no” を返す．

以下，process-child, process-parent, process-descendant を示す．process-following-sibling と process-preceding-sibling は紙数の都合で省略する．delete(m)は「印 m が置かれたことのあるノードで，他の印は置かれたことのないもの」を t から削除する．

process-child(m)
1. m の置かれたノード n_m のラベルを l とする．
$d(l)$ に nt[i]が含まれる場合：
2. t において，n_m がラベル nt[i]をもつ子ノード n' をもつ場合：
3. $d(l)$ 中の*の位置により，n_m がラベル nt[i]をもつ子ノードを複数もち得るか否かを調べる．
4. 複数もちうる場合：
5. ラベル nt[i]をもつノードを n_m の子ノードとして追加し，そのノードに m を移す．
6. 複数もち得ない場合：
7. m を n' に移す．
8. t において，n_m がラベル nt[i]をもつ子ノードをもたない場合：
9. ラベル nt[i]をもつノードを n_m の子ノードとして作成し，そのノードに m を移す．
10. $d(l)$ に nt[i]が含まれない場合：
11. delete(m);

process-descendant(m)
1. m の置かれたノード n_m のラベルを l とする．D を参照し，l から nt[i]へ至るラベル列の集合$\{L_1,\cdots,L_k\}$を求める．
2. $k \geq 1$ の場合：
3. 各 i ($1 \leq i \leq k$)に対して，n_m から L_i に沿って経路を辿り，その終端に新たな印 m_i を置く．ただし，L_i のうち t に出現しない部分は新たにノードと辺を作成して t に追加する．追加されたノードは，「m_i が置かれたことのあるノード」として扱う．
4. m を t から削除する．
5. k=0 の場合：
6. delete(m);

process-parent(m)
1. 親ノードのラベルが nt[i]である場合：
2. その親ノードに m を移す．
3. 親ノードのラベルが nt[i]でない場合：
4. delete(m);

本節冒頭の条件(a)と(b)に D が非再帰という条件を加えた場合，このアルゴリズムの時間計算量は $O(|\Sigma|^2 \cdot |D|^2 \cdot |p|)$ である．証明は紙数の都合で省略する．

例として，次の XPath 式 p と DTD D = (d, list)を考える．

p = /child::list/child::item/parent::list/child::item
/descendant::a/parent::item

d(list) = item*
d(item) = a|b
d(a) = d(b) = #PCDATA

上記のアルゴリズムにより，図 1 の木 t が作成される．t に印が置かれているので，アルゴリズムは”yes”を返す．

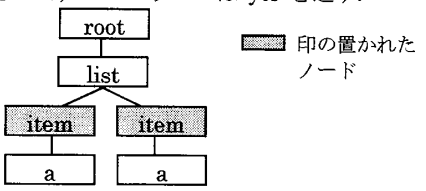

図 1：木 t

5. 評価実験

XPath 式の実行前に充足可能性判定を行い，充足不能な XPath 式の実行を回避することによって，どの程度実行時間が節約できるかを評価した．この指標として Saving Ratio[4]を用いる．XPath 式による問合せ時間を e，XPath 式の充足可能性の検証に要する時間を c とすると，Saving Ratio は $\frac{e-c}{e}$ と定義される．実験環境は，CPU：Core2 Duo 1.6GHz，メモリ：2GB，OS：Windows Vista Business，アルゴリズム実装言語：Java 2 SDK 1.6.0，XPath 実行環境：eXist 1.2.2 および SAXON-B.9.1，である．XPath 式に関しては，XPath 式を自動生成するプログラムを作成し，生成された 15 個の充足不能な XPath 式を使用した．それら式の Saving Ratio の平均値を図 2 に示す．問合せに時間を要する状況下においては，充足不能な XPath 式の実行を回避することによる時間節約効果は認め得ると考えられる．

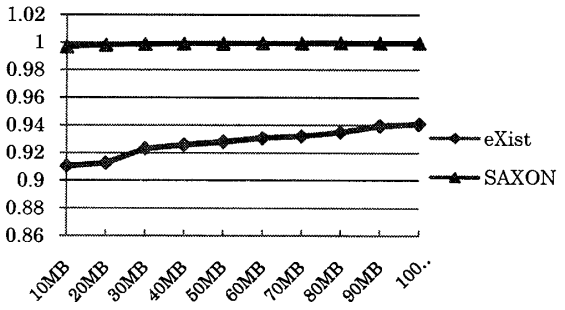

図 2：Saving Ratio

6. むすび

今後の課題として，(i)より一般的な XPath 式について考察する，(ii)評価実験における XPath 式の生成方法などについて更に検討する，などが挙げられる．

参考文献

[1] M. Benedikt, W. Fan, and F. Geerts, “XPath Satisfiability in the Presence of DTDs,” Journal of the ACM, Vol.55, Issue 2, Article 8, 79 pages, May 2008.
[2] M. Montazerian, P. T. Wood, and S. R. Mousavi, "XPath Query Satisfiability is in PTIME for Real-World DTDs," Proc. Xsym 2007, pp.17-30, 2007.
[3] J. Clark and S. DeRose, eds., XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath
[4] L.V.S. Lakshmanan, G. Ramesh, H. Wang, and Z. Zhao, “On Testing Satisfiability of Tree Pattern Queries”, Proc. VLDB, pp.120-131, 2004.
[5] 福島雄司，鈴木伸崇，” XPath 式の部分クラスに対する充足可能性判定アルゴリズム”, 第 7 回情報科学技術フォーラム講演論文集(第 2 分冊)，D001，pp.41～42，2008