D-001

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

An Algorithm for a Subclass of XPath Satisfiability Problem

福島 雄司† 鈴木 伸崇†
Yuuji Fukushima Nobutaka Suzuki

1. まえがき

XPath の充足可能性問題とは，DTD D と XPath 式 p が与えられた時に，D に関して妥当かつ p に対する問合せ結果が空でない XML データが存在するか否かを決定する問題である．ここで，DTD D に関して妥当な XML データが蓄積されており，それに対して XPath 式 p で問い合わせることを考える．p が D の下で充足不能な場合，問合せ結果は必ず空となるので，このような式を実行することは無意味である．したがって，充足不能な XPath 式は実行前(コンパイル時)に検出できることが望ましい．

XPath の充足可能性問題は一般に決定不能であり，XPath 式に種々の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，まず，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 完全であることを示す．次に，この制約および duplicate-free[3]という DTD に関する条件の下で，同問題を多項式時間で解くことのできる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．

XPath の充足可能性問題に関する主な研究としては，文献[1]のほか，文献[2,3]がある．文献[2]では，DTD の存在を仮定しない場合の充足可能性問題について考察している．文献[3]では，祖先方向の軸(parent や ancestor)を含まない場合において，DTD にある制約(duplicate-free および covering)を置いた場合， いくつかの条件の下で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}上の正規表現への写像，r ∈ Σ は文書要素である．以下，本稿における XPath 式とは，XPath の仕様[4]における絶対ロケーションパスを指すものとする．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．DTD D と XPath 式 p が与えられた時に，p が D の下で充足可能であるか否かを決定する問題を，XPath の充足可能性問題という．

3. NP 完全性

本節では，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を置いたとしても XPath の充足可能性問題が NP 完全であることを示す．

[定理 1] XPath の充足可能性問題は，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 NP 完全である．

†筑波大学大学院図書館情報メディア研究科，Graduate School of Library, Information and Media Studies, University of Tsukuba

[略証] まず，この問題が NP に属することは自明である．次に，この問題が NP 困難であることを，3SAT 問題からの帰着により示す．3SAT 問題のインスタンスを

$$\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$$

とする．ここで，節 C_i は 3 つ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．このインスタンスから，XPath の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．

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

ここで，T_i と F_i は要素名(節の名前)の系列を略記したもので，次のように定義される．

- まず，T_i は，ϕ において変数 x_i が正リテラルとして出現する節を列挙したものである．すなわち，x_i が節 $C_{i1},\cdots,C_{ik}$ に正リテラルとして出現するとき，$T_i = C_{i1},\cdots, C_{ik}$ となる．言い換えると，T_i は，x_i に真を代入した場合に真となる節を列挙したものである．
- 一方，F_i は，ϕ において変数 x_i が負リテラルとして出現する節を列挙したものである．すなわち，x_i が節 $C_{i1},\cdots,C_{ik}$ に負リテラルとして出現するとき，$F_i = C_{i1},\cdots, C_{ik}$ となる．言い換えると，T_i は，x_i に偽を代入した場合に真となる節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．

/child::r/child::C_1/parent::r/child::C_2/ ⋯ /parent::r/child::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 インスタンスに対してのみ検索結果が空でない．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． □

4. アルゴリズム

前節の結果から，XPath の軸は child と parent のみ，ノードテストは要素名のみ，かつ，述語を使用しないという条件の下でも 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．本節では，上記の条件に加えて DTD に duplicate-free[3]という条件を置いた場合に，同問題を多項式時間で解くアルゴリズムを示す．ここで，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．

DTD を D=(d ,r)とする．XPath 式 p を以下のように表す．

/axis[1]::nt[1]/axis[2]::nt[2]/ ⋯ /axis[n]::nt[n]

ここで，axis[i]は軸，nt[i]は要素名である．このアルゴリズムは, 次のように p の軸に従って木 t を構成しながら各ノードの検証を行う．

1. 初期状態では，木 t は空である．
2. 各 i=1,…,n に対して以下の処理を行う．
   - axis[i]が child の場合:
     tが空の場合：i=1 である．nt[i]がDの文書要素rと一致するかどうか調べ，一致しなければ”no”を返す．一致した場合，ラベルnt[i]を持つノードを作成してtに追加し，このノードをカレントノードとする．
     tが空でない場合：
     ・ カレントノードの子ノードにラベル nt[i]をもつものが存在しない場合：カレントノードのラベル l の内容モデル $d(l)$を参照し，カレントノードの全ての子ノードと nt[i]が同時に出現し得るかどうかを判定する．出現し得る場合，ラベル nt[i]を持つノード n を作成し，カレントノードの子として t に追加する（n をカレントノードとする）．出現し得ない場合，”no”を返す．
     ・カレントノードの子ノードにラベル nt[i]をもつものが存在する場合：カレントノードのラベル l の内容モデル $d(l)$ を参照し，nt[i]が（D 中の`*’によって）複数個出現し得るかどうか判定する．出現し得る場合，ラベル nt[i]を持つノード n を作成し，カレントノードの子として t に追加する（n をカレントノードとする）．出現し得ない場合，カレントノードの子でラベル nt[i]を持つものが存在するので，そのノードをカレントノードとする．
   - axis[i]が parent の場合:
     t が空であるか，カレントノードの親が存在しない場合，”no”を返す．そうでない場合，nt[i]がカレントノードの親ノードのラベルと一致しているかどうかを検証する．一致する場合，カレントノードをその親ノードに移し，一致しない場合，”no”を返す．
3. “yes”を返す．

このアルゴリズムのオーダーは O(|p|・|D|)である．次の定理が成り立つ（証明は省略）．

[定理 2] DTD D と XPath 式 p に対して，p が D の下で充足可能であるときかつそのときのみ上記アルゴリズムは”yes”を返す． □

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

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

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

上記のアルゴリズムにより，図 1 の木 t が作成される．p の最後のロケーションステップ child::b について，既に item は a を子に持っており，子ノードとして a と b を同時に持つことはできない．よってアルゴリズムは”no”を返す．

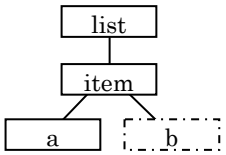

図 1：木 t

5. 評価実験

XPath 式の実行前に充足可能性判定を行って充足不能な XPath 式の実行を回避することにより，どれだけ時間短縮効果があるかを評価した．この指標として，Saving Ratio[7]を用いる．XPath 式による検索時間を 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，1 に近いほど望ましい．

アルゴリズムを Java で実装し，以下の手順で評価実験を行った．まず，Xmark[5]を用いて異なるサイズの XML データを生成し eXist[6]に格納した．次に，XPath 式の実行時間，充足可能性の判定時間を測定し，Saving Ratio を求めた．

実験に用いた(充足不能な)XPath 式を以下に示す．

・/site/text
・/site/categories/description
・/site/categories/category/description/text/keywords
・/site/categories/category/description/parlist/listitem/text/keywords
・/site/regions/europe/item/incategory/category
・/site/closed_auctions/parent::site/text
・/site/people/person/address/city/parent::homepage
・/site/catgraph/parent::site/regions/parent::site/people/parent::site/name
・/site/people/person/name/parent::person/parent::people/person/address/city/parent::address/parent::name

実行環境は CPU:Intel Core2 Duo 1.60GHz，メモリ:2GB，OS:Windows Vista Business，使用言語:Java 2 SDK 1.6.0，データベース:eXist 1.2.2 である．上記 XPath 式の Saving Ratio の平均値を図 2 に示す．データサイズが 5MB の場合で既に Saving Ratio は 0.93 を超えており，充足不能な XPath 式の実行の回避による時間短縮効果は認め得ると考えられる．

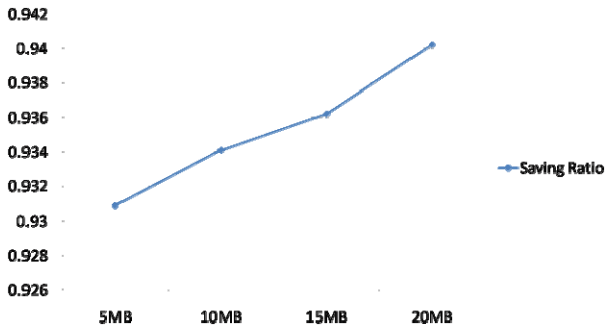

図 2：Saving Ratio

6. むすび

本稿では，まず 3 節で示した条件の下でも XPath の充足可能性問題が NP 完全であることを示した．次に，この条件に加えて DTD が duplicate-free である場合に，同問題を多項式時間で解くアルゴリズムを開発・実装し評価を行った．今後の課題として，descendant 等より多くの軸に対応したアルゴリズムの開発が挙げられる．

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] J. Hidders, "Satisfiability of XPath Expressions," Proc. DBPL 2003, pp.21-36, 2003.
[3] 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.
[4] J. Clark and S. DeRose, eds., XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath
[5] An XML Benchmark Project. xmlgen ver0.92.http://www.xml-benchmark.org/.
[6] Open Source Native XML Database. eXist ver1.2.2, http://exist.sourceforge.net/.
[7] L.V.S. Lakshmanan, G. Ramesh, H. Wang, and Z. Zhao, “On Testing Satisfiability of Tree Pattern Queries”, Proc. VLDB, pp.120-131, 2004.