社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－DD－58（15）
2006－EIP－34（15）
2006／11／30

NXp 集合を用いた XML 文書アクセス制御処理の高速化

王　命玲　静岡大学大学院情報学研究科　cs1103@s.inf.shizuoka.ac.jp
白井 靖人　静岡大学情報学部情報科学科　shirai@inf.shizuoka.ac.jp

あらまし：XML 文書に対してその要素単位でアクセス制御を行う際には，アクセス要件を記述したアクセス制御ルールの集合を予め用意しておき，各要素をアクセス制御ルールの一つ一つと照合して，アクセス可否の判定を行う必要がある。アクセス制御処理にかかる時間は，この照合の回数に比例する。したがって，処理の高速化を図るには，要素とアクセス制御ルールとの照合回数を減らす必要がある。本論文では，アクセス制御ルールを記述するための構造として NXp 集合を提案し，要素とアクセス制御ルールとの照合回数の減少を図る。従来の方法と NXp 集合を用いた方法を比較する実験を行い，アクセス制御処理の高速化における NXp 集合の有効性を示す。

Accelerating the access control on XML documents using NXp sets

M. L. Wang, Graduate School of Informatics, Shizuoka University, cs1103@s.inf.shizuoka.ac.jp
Yasuto Shirai, Faculty of Informatics, Shizuoka University, shirai@inf.shizuoka.ac.jp

ABSTRCT: This paper proposes an NXp set as a means to accelerate the access control on XML documents. Combined with XPath, NXp allows us to specify the access control policy using fewer access control rules, thereby resulting in an access condition table with fewer entries. With fewer entries, the number of comparisons of the requested XML element against the access condition table becomes less, resulting in a shorter processing time.

1　はじめに

XML（eXtensible Markup Language）[1]は，分野や業態を越え，各種組織体間での情報流通の共通書式として，多くの注目を集めている。安全な情報流通を行うためには，XML 文書の適切なアクセス制御管理が必要なことは，いうまでもない。

XML 文書の文書単位でのアクセス制御については，従来のファイルのアクセス制御の問題としてとられることができる。一方，XML 文書の中身に目を転じると，事態は大きく異なっている。XML 文書中には，様々なアクセス要件をもつ要素が混在している。すなわち，一つの文書の中に，アクセス可能な部分と不可能な部分とが混在することになり，文書単位だけでなく，要素単位でのアクセス制御が必要となることが知られている[2,5,6]。

アクセス制御の要件に関する記述を，アクセス制御ポリシーという。XML 文書の要素単位でのアクセス制御においては，このアクセス制御ポリシーはいくつかのアク

セス制御ルールから構成される。実際のアクセス制御処理においては，アクセス制御ルールを表の形に変換しておき，アクセス対象のノードをその表の各エントリと照合して，そのノードに対するアクセスの可否を判定する。アクセス制御ルールの数が多いと，表のエントリ数も増え，その結果，照合の回数も増加する。アクセス制御処理にかかる時間は，この照合の回数に比例するため，処理の高速化にあたっては，表のエントリ数を小さく抑えることが重要になる。

アクセス制御ルールでは，その一部として，アクセス対象を記述する。従来の方法では，アクセス対象の指定には，Xpath[3]を用いていた。

本稿では，アクセス制御ルール中でのアクセス対象の記述にあたって，XPath の欠点を補うものとして NXp 集合を提案する。NXp 集合とは，アクセス制御ポリシーによってアクセス可能となる要素すべてを含む集合である。アクセス判定を行う際には，要素をこの集合と照合するだけでアクセス可否の判定が可能であり，照合回数の大幅な減少が期待される。

本稿の構成は，以下のとおりである。第 2 章では，関連研究[2]として，XPath を用いてアクセス対象をしてアクセス制御処理を行う方法の概要を紹介する。本稿の提案する NXp 集合については，第 3 章において，それを用いた場合のアクセス制御処理の概要と合わせて紹介する。処理全体の流れは，XPath と NXp 集合を併用する以外は，従来の方法とほぼ同じである。XPath を用いた場合と，NXp 集合を併用した場合で，処理時間にどの程度の差が出るか，実験を行った結果を第 4 章で紹介する。

2　関連研究

2.1　XML

XML 文書は，要素の階層構造をマーク付けの規則によってテキスト形式に記述したものと考えることができる。要素には，属性を付加することもできる。本稿でのアクセス制御処理の説明で例として用いる XML 文書を図 1 に示す。ここでは，要素へのアクセスが可能かどうかの判定に興味があるので，どの要素も内容は空としている。

```
<a>
  <b>
    <i><m/><n/></i>
    <f><j/></f>
  </b>
  <c>
    <g><k/><p/></g>
    <h/>
  </c>
  <d/>
</a>
```

図 1　XML 文書のサンプル

2.2　アクセス制御ポリシー

アクセス制御ポリシー[4]は，任意の数のアクセス制御ルールから構成される。

アクセス制御ルールは，主体，アクセス制御モード，アクセス対象の三つの部分から構成され，以下の書式にしたがって，記述する。

(主体, アクセス制御モード, アクセス対象)

主体：具体的な利用者，ロール，計算機のプロセスなど，XML 文書の内容にアクセスしようとする主体を表す。

アクセス制御モード：アクセス制御の内容を表し，許可フラグと動作の組合せとして記述する。

許可フラグ: アクセス可能を示す“＋”と，アクセスを禁止する“―”がある。

動作：単一要素の読込みを示す“r”と，その要素と子孫にあたる全ての要

素の読込みを示す“R”がある。

アクセス対象：XML 文書内の要素を記述する。XPath を用いて記述する。

例1（role:doctor, +r, /record）：ロールが“doctor”ある主体には，文書のルート要素である“record”の読込みが許可される。

例2（role:doctor, +R, /record/name）：ロールが“doctor”である主体は，要素“/record/name”及びその子孫のすべての要素に対して，読み込みが許可される。

2.3 アクセス条件テーブル

図 1 の XML 文書に対して，次の五つのアクセス制御ルールから構成されるアクセス制御ポリシーが適用されているとする。

R1：（role:user, ＋r, /a）

R2：（role:user, ＋r, /a/c）

R3：（role:user, ＋r, /a/c/g）

R4：（role:user, ＋R, /a/b）

R5：（role:user, －R, /a/b/i）

複数のアクセス制御ルールが適用されると，それぞれのルールの効果の“重ね合わせ”を行う。各ノードへのアクセス可否の判定は，重ね合わせの結果に基づいて行われる。R1～R2 を，図 1 の XML 文書上に重ね合わせた結果を図 2 に示す。

図 2 中の○印の要素は，アクセスが許可されている要素である。△印の要素は，アクセスが禁止されていて，アクセス要求が拒否される要素である。□印の要素は，アクセス制御ポリシーには明示的な記載のない，該当するルールのない要素である。該当するルールがない場合，可否判断をどのように行うかは，アプリケーション等に依存する。本稿では，該当するルールがない場合も，アクセス要求は拒否するものとして扱う。ただし，アクセス拒否が，禁止によるものか，ルールなしによるものなのかは区別できるようにする。

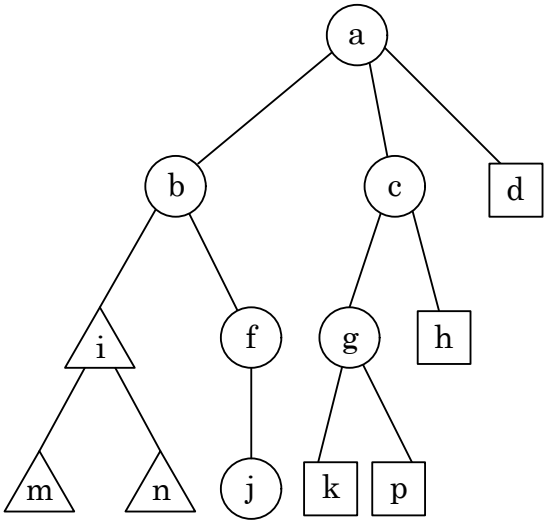


図 2　R1～R5 を重ね合わせた結果

アクセス制御ルールを重ね合わせた結果を表形式に表現したものを，アクセス条件テーブルという。表 1 は，図 2 の状況を表したアクセス条件テーブルである。

表 1　アクセス条件テーブル

対象パス	ローカルアクセス条件	子孫へのアクセス条件
/a	TRUE	FALSE
/a/c	TRUE	FALSE
/a/c/g	TRUE	FALSE
/a/b	TRUE	not(ancestor-or-self::i)

アクセス条件テーブルは，対象パス，ローカルアクセス条件，子孫へのアクセス条件の三つの項目から構成される。ローカルアクセス条件は，対象パスの末端の要素に対するアクセスの可否を示す。子孫へのアクセス条件は，対象パスの末端要素の子孫にあたる要素へのアクセスの可否を示す。ここで，ancestor-or-self::i とは，判定対象の要素が i かその子孫であるかを示す述語である。したがって，要素 b の子孫については，i かその要素であれば，アクセスが禁止されることになる。表 1 の第 1 行は R1，第 2 行は R2，第 3 行は R3，第 4 行は R4 及び R5 から，それぞれ導かれている。

2.4 アクセス可否の判定

アクセス条件テーブルを使ったアクセス可否の判定にあたっては，どの要素に対するアクセス要求もルート要素からのパスに基づくものと考える。

要求されたパスがテーブル中の対象パスに含まれていれば，該当するのローカルアクセスアクセス条件が適用される。

もし対象パスに含まれていない場合は，要求パスの各プリフィックスについて，テーブル中の対象パスに含まれていないかを調べる。もし含まれていれば，該当する子孫へのアクセス条件を適用する。

どのプリフィックスもテーブルに含まれていない場合は，アクセスを拒否する。

表 1 を使って，幾つかの要求パスに対してアクセス可否の判定を行った結果を表 2 に示す。

要求パス/a/b/f の場合について，判定手順の概要を説明する。この要求パスは，表 1 の対象パスには含まれていない。そこで，プリフィックスの/a/b について調べると，これは含まれている。そこで該当する子孫へのアクセス条件を適用する。対象パスの末端要素は f であり，i ではないので，アクセス条件を適用した結果は TRUE となり，アクセスは許可される。

表 2　アクセス可否の判定

要求パス	対象パス	アクセス条件	判定結果
/a	/a	TRUE	許可
/a/c	/a/c	TRUE	許可
/a/c/h	/a/c	FALSE	拒否
/a/b/f	/a/b	TRUE	許可
/a/b/i/n	/a/b	not(ancestor-or-self::i)	拒否

ここで，照合回数について考える。アクセス条件テーブルを調べる際は，常に第 1 行から順に調べるとして，先の/a/b/f の例について考える。要求パス/a/b/f がテーブルに含まれないと知るには，四つの行全てと照合する必要がある。次に，プリフィックス/a/b について同様に照合を行い，その 4 回目に照合が成功する。したがって，ここでは 8（＝4＋4）回の照合が必要だったことが分かる。

その他の幾つかの要求パスについて，アクセス可否の判定を行うのに必要な照合の回数を，表 3 に示す。ここで，/a/d については，アクセス条件テーブル中に該当するアクセス条件を見つけることができない場合であり，結果としてアクセス拒否と判定している。

表 3　必要な照合の回数

要求パス	判定結果	回数
/a/b	許可	4
/a/b/i/n	拒否	12
/a/d	拒否	6

一つの要求パスについて，そのプリフィックスをテーブルの各行と繰り返し照合する作業が必要となる場合，照合の回数が増加する傾向にあることが分かる。アクセス条件テーブルに格納される件数を減らすことができれば，XML 文書の構造に依らず，照合の回数を減らすことができる。

XML 文書に対するアクセス制御処理に要する時間は，ここでの照合の回数に大きく依存する。したがって，アクセス条件テーブルの件数を減らすことによって，アクセス制御処理の高速化を図る方法を考える。

3　NXp 集合を利用したアクセス制御

3.1　NXp 集合

アクセス条件テーブルの件数を減らすために，XML 文書の各階層でアクセス可能な要素をあらかじめ判定し，その結果を一まとめにして表すことを考える。ここに含まれる要素については，アクセスは許可さ

れる。含まれないものについては，拒否される。

ここで，アクセス可能な要素をあらかじめ判定しておいた結果を表現するための構造として，NXp 集合を次のように定義する。

【定義：NXp 集合】

XML 文書のルート要素から，一つ以上の要素にいたる経路上の全ての要素の集合を，NXp 集合という。

本稿で例として用いている XML 文書の場合で考えると，各階層でアクセス可能な要素は，

第 1 階層：a　　第 2 階層：b，c

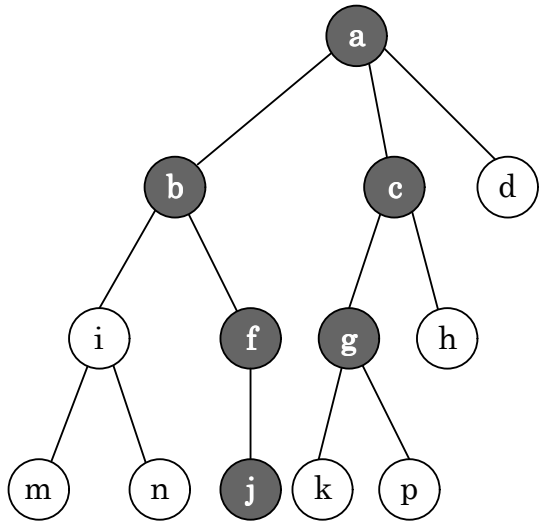
第 3 階層：f，g　　第 4 階層：j

である（図 3）。これらの要素は，ルート要素（a）から要素 j に到る経路と，要素 g に到る経路に含まれる全要素であり，一つの NXp 集合を構成している。

NXp 集合は，各要素の階層が分かるように，階層間を“|”で，同一階層内の要素は“,”で区切る記法を用いる。この例では，

|a|b, c|f, g|j

と表す（図 3）。



（グレイの部分が NXp 集合|a|b, c|f, g|j）

図 3　NXp 集合

NXp 集合は，XML 文書を一つの木構造と考えた場合，文書自体と根を共有する部分木と考えることができる。NXp 集合に含まれる要素は，木構造上で全て互いに連結されていて，NXp 集合中の任意の二つの要素について，それらを結ぶ経路が NXp 集合内に存在する。また，NXp 集合には，もととなった XML 文書のルート要素が必ず含まれている。

アクセス制御ルール中のアクセス対象として，XPath による記述に加えて，NXp 集合による記述を許すことによって，アクセス制御ポリシーを構成するルールの数を減り，その結果，アクセス条件テーブルの件数も減る。

3.2　NXp 集合を用いたアクセス制御処理

第 2 章で用いた例を，今度は XPath と NXp 集合を併用して処理する場合を考える。XPath と NXp 集合を併用することによって，R1～R5 のルールを，次の二つのルールで書き換えることができる。

XR1：(role:user, ＋r, |a|b,c|f,g|j)

XR2：(role:user, ＋R, /a/b)

アクセス可能な要素は，XR1 だけで全て表すことができる。しかし，R5 によって i 以下の要素が読込み禁止になっていることを明確にするために，XR2 を加えてある。

二つのルールを重ね合わせる際には，次のように考える。アクセス可能な要素は，全て XR1 で明らかになっている。XR2 以降のルールでアクセス可能とされている要素のうち，XR1 に含まれていないものは，実際にはアクセス禁止となっている。

R1～R5 でも触れていなかった d, h, k, p の要素については，ここでも触れることはない。こうして，R1～R5 と全く同様の内容を記述することができる。

XR1，XR2 をアクセス条件テーブルで表したものが，表 4 である。ここで，“対象”とは，通常のアクセス条件テーブルで“要

求パス”と呼んでいたものに相当する。対象が NXp 集合の場合，その子孫は定義されないので，子孫へのアクセス条件は“－”としてある。

表 4　アクセス条件テーブル（NXp 集合併用）

対象	ローカル アクセス条件	子孫への アクセス条件
|a|b, c|f, g|j	TRUE	－
/a/b	TRUE	TRUE

表 4 のアクセス条件テーブルを用いて要素のアクセス可能判定を行う手順と，その際必要となる照合の回数について考える。表 3 で取り上げた三つのパスについて判定を行った場合の照合回数を，表 5 に示す。ここでも，表 3 の場合と同様，要求パスとアクセス条件テーブルの各行との照合は，テーブルに記載された順に行うものとする。

表 5　必要な照合の回数（NXp 集合併用）

要求パス	判定結果	回数
/a/b	許可	1
/a/b/i/n	拒否	4
/a/d	拒否	2

要求パス/a/b については，テーブルの第 1 行との照合で，末端要素の b が NXp に含まれていることから，読込み可能であることが分かる。第 2 行との照合は必要ない。

要求パス/a/b/i/n については，テーブルの第 1 行との照合で，末端要素 n が NXp 集合に含まれないことが分かる。この時点で読込みが許可されないことは分かるが，読込みが禁止されているのか，ルール設定がないのかの違いは分からない。そこで，テーブルの第 2 行との照合に進むことになる。まず，/a/b/i/n と/a/b を比較し，異なることが分かる。次に，プリフィックスの/a/b/i と/a/b の比較，さらに/a/b と/a/b の比較と進んで，第 2 行の子孫へのアクセス条件が適用されることが分かり，TRUE の結果を得る。こうして，この要求パスは，アクセス可能な要素を集めた NXp 集合には含まれないが，別のルールでは読込みが許されていることが分かり，結果として読込みは拒否されることになる。必要な照合の回数は，第 1 行と 1 回，第 2 行と 3 回の，計 4 回となる。

要求パス/a/d については，第 1 行，第 2 のどちらにも該当しない。この場合は，該当するルールがないことを意味し，結果として読込みを拒否する。照合の回数は，各行と 1 回ずつの，計 2 回である。

表 3 と表 5 を比べると，照合回数が確実に減っていることが分かる。また，判定結果についても，全く違いはない。表では単に“拒否”となっているが，禁止による拒否と，該当ルールなしによる拒否との区別も，きちんと付くようになっている。

4　実験

アクセス制御ルールの記述及びアクセス条件テーブルの作成において，XPath だけを用いる従来の方法と，XPath と NXp 集合を併用する方法と比較するために，計算機実験を行った。

4.1　実験方法

同一内容のアクセス制御ポリシーに二つの方法を適用し，それぞれの方法に基づくアクセス条件テーブルを得る。それぞれのアクセス条件テーブルを使って，XML 文書の全要素についてアクセス可否の判定を行い，所要時間を計測する。

二つの方法の違いを調べるために，XML 文書は 2 種類用意した。一方は，木構造としてみたときに，縦よりも横に広がっているもの，もう一方は，横よりも縦に深く延びているものである。これら二つの文書に対して，アクセスが許可される要素の割合を様々に変えて実験を繰り返す。

4.2 実験環境

実験に用いた実験環境は以下のとおりである。

CPU： Intel Celeron 700MHz
メモリ： 256MB
OS： Microsoft Windows 2000

実行プログラムは Java JDK1.4 を使用して実装した。

実験に用いた XML の概要は以下のとおりある。

(1) “広い” XML 文書

総要素数： 895
階層の深さ： 6
同一階層の要素数： 108（最大）
ファイルサイズ： 14.23 k バイト

(2) “深い” XML 文書

総要素数： 968
階層の深さ： 35
同一階層の要素数： 17（最大）
ファイルサイズ： 16.38 k バイト

4.3 実験結果

計測を行った結果を，図 4 及び図 5 に示す。横軸は，アクセスを許可される要素の文書全体に占める割合，縦軸は，文書全体を処理するのに要した時間である。“XPath” のプロットは，XPath だけを用いた場合の所要時間，“NXp” のプロットは，XPath と NXp を併用した場合の所要時間を表している。

図 4 は，横の広がりの広い XML 文書に対して実験した結果，図 5 は，縦の深さの深い XML 文書に対して実験した結果を示している。アクセスが許可される要素の割合を，0.1 から 1 まで 0.1 おきに変化させて，所要時間の計算を繰り返した。

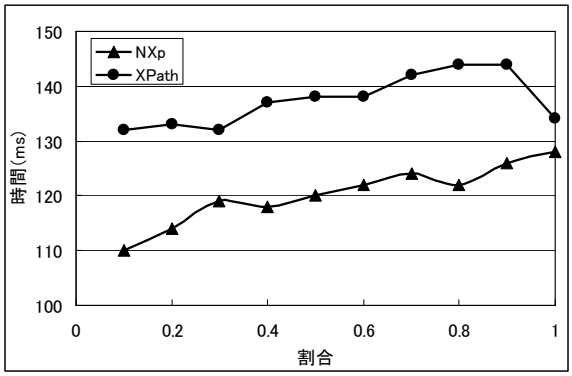


図 4 所要時間（“広い”XML 文書）

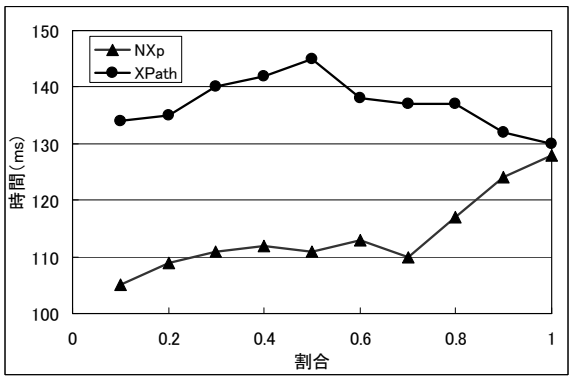


図 5 所要時間（“深い”XML 文書）

いずれの場合においても，XPath だけを用いるよりも，NXp 集合を併用する方が，処理時間で短くなっている。

アクセスが許可される要素の割合との関係を見てみると，XPath だけの場合は，割合が増えると，所要時間は一旦増加し，再び減少する。両者を併用する場合は，割合が増えると，総じて所要時間が増える傾向がある。その結果，割合が高くなると，両者の差は小さなものにとどまっている。

以上の結果，アクセスが許可される要素の割合によって程度の差はあるものの，NXp 集合を用いることが，アクセス制御処理の高速化につながることが確認できた。

5 まとめと今後の課題

XML 文書の要素単位のアクセス可否の判定を高速に行うための方法として，アク

セス制御ルール及びアクセス条件テーブルに NXp 集合を導入する方法を提案した。

従来の XPath だけを使う方法に比べると，NXp 集合を併用することによってアクセス制御ルールの数が減り，アクセス条件テーブルも小さなもので済むようになる。その結果，アクセスしようとしている要素とテーブルの各行とを比較照合する回数が減り，短い処理時間で済むようになる。

今後の課題として，アクセス制御ポリシーを，NXp を使ったアクセス制御ルールを使って効率的に記述する方法を開発する必要がある。具体的な方法としては，XPath を用いてルールを記述する方法が従来から用いられているので，XPath だけを使って記述したルールを，NXp を併用したものに自動的に変換する方法などが考えられる。

謝辞

本研究に関するご助言を頂きました静岡大学情報学部情報科学科の白井靖人助教授に深謝致します。

参考文献

[1] T.Bray,J.Paoli,C.M.Sperberg-McQueen, “Exensible Markup Language (XML) 1.0.W3C Reconmmendation,” available at http://www.w3g.org/TR/REC-XML

[2] M.kudo N.Seki, “XML Access Control Using Access-condition-table,” in IPSJ SIG Technical Report 2004-DBS-134(I)

[3] J.Clarkand, S.DeRose, “XML Path Language(Xpath) version 1.0 W3C Recommendation,” available at http://www.w3g.org/TR/xpath.

[4] M.Murata, A.Tozawa, M.kudo and S.Hada, “XML Access Control Using Static Analysis,” in ACM CCS, 2003.

[5] M.kudo and S.Hada,”XML Document Security based on Provisional Authorization,” in ACM CCS, 2000.

[6] S.Hada, M.Kudo,“XML Access Control Language:Provisional Authorizatiion for XML Document”, http://www.trl.ibm.com/projiects/xml/xacl/xacl-spec,html,2000.