XML 文書のアクセス制御における cascade の最適化

Optimizing Cascade for Access Control of XML Documents

岩井原 瑞穂♥　王 波♦
チャットウィチェンチャイ ソムチャイ♠
Mizuho IWAIHARA　Bo WANG
Somchai CHATVICHIENCHAI

近年，XML による情報流通が活発になるにつれ，XML 文書の情報を保護するためのアクセス制御が重要となってきている．アクセス制御ポリシーでは要素レベルのルールを設定し，細かな制御が可能であるが，情報の交換において XML 文書が変換・合成された後，ポリシーが冗長になる場合がある．本稿では，ポリシーのターゲット要素について，それを根とする部分木全体にポリシーを適用するか否かという cascade の設定を最適化し，各ノードに適用されるルール数を最小にするアルゴリズムを示す．

As documents containing sensitive information are exchanged over the internet, access control of XML documents is becoming important. Access control policies of XML involve node-level granularity as well as cascading option, which specifies whether an access control policy shall be applied to the whole subtree from the target node of the rule. In this paper, we consider the optimization problem such that minimizing the number of policies applied to nodes, by optimizing cascade options of given rule sets.

1. はじめに

近年，XMLによる情報流通が活発になるにつれ，XML文書の情報を保護するためのアクセス制御が重要となってきている．XML文書内のデータにおけるきめの細かいアクセス制御は，XML文書の構造・内容に直接的にアクセスの制約をかける．ゆえに，アクセス権限判定ポリシーはXML文書の構造・内容に深く関わる．

XML文書の構造は，アプリケーションの拡張や組織間のデータ交換など様々な理由で変わる傾向があるので，新しい構造に適合するように権限判定を修正しなければならないが，新しいポリシーは冗長な場合が多い．また，ユーザが個人情報に関するアクセス制御情報を自ら定義する場合，個々のユーザの定義したポリシーをまとめて簡略化する必要がある．

既存研究ではいくつかのアクセス制御モデルが提案されている[1][2][3][4]．XACML[5]はXML文書のアクセス制御に関するOASISの標準である．ポリシーの簡略化は重要な問題であり，文献[7]ではポリシー最小化アルゴリズムが示されている．文献[7]はdescendant-overrides(子孫優先)というポリシー衝突解消方式のもとでの簡略化を行なっているが，本稿では，Deny- overrides，Permit- overrides，およびFirst-applicableと呼ばれる衝突解消方式それぞれにおけるポリシー簡略化のアルゴリズムを提案し，計算機実験によるポリシー簡略化の定量的評価を検討する．

以下，本稿は第2節においてアクセス制御ポリシーの簡略化アルゴリズムについて述べ，第3節においてポリシー簡略化の定量的評価を述べる．第4節はまとめと今後の課題である．

2. アクセス制御ポリシーの簡略化

2.1 Cascade 最適化による簡略化

アクセス制御ルールに用いる XPath 式において，ターゲットとなる要素とそれを根とする部分木全体に同じ権限判定値を設定することは有用であり，多くのアクセス制御モデルで採用されており[1][3], cascade と呼ばれることが多い．本稿では cascade の設定を最適化することにより，ポリシーをできるだけ簡略化されたものにするアルゴリズムについて検討する．Cascade を付加するとは，XPath 式においてターゲットとなる要素に descendant-or-self を設定することに相当する．そして cascade を持たないポリシーを入力とし，ルール数が最小となるように cascade を設定することを目標とする．

表１　ノードに設定するルールの記号

Table.1 Symbols of Instance-level Rules

記号	意味	ルール数
n	ノード e にはルールを設定しない．	0
+	ノード e に権限判定値が a(e) で cascade を行うルールを設定する．	1
-	ノード e に権限判定値が a(e) で cascade を行わないルールを設定する．	1
±	ノード e に権限判定値が a(e) で cascade を行わないルール，次に権限判定値が a(e)の反対で cascade を行うルールの２つを設定する（e 自身には前者，e の子孫には後者のルールが適用される）．	2

表１は，ノードごとに設定するアクセス制御ルールを４つの場合に分けたものである．ノード e にはあらかじめ権限判定値 a(e) ∈{permit, deny}が与えられているものとする．n は e にルールを設定しない（祖先から cascade されるものを利用する），+は a(e)の値を e に設定し，しかもそれを子孫に cascade する，-は a(e)の値を e に設定するが子孫には cascade しない，±は a(e)の値を e に設定するが，子孫には a(e)とは反転させた権限判定値を cascade させるというものである．１つのノードに対するルールの設定方法は，冗長

♥ 正会員　京都大学大学院情報学研究科社会情報学専攻
iwaihara@i.kyoto-u.ac.jp
♦ 非会員　京都大学大学院情報学研究科社会情報学専攻
wang@db.soc.i.kyoto-u.ac.jp
♠ 正会員　県立長崎シーボルト大学国際情報学部
somchaic@sun.ac.jp

なものを除くとこの４通りのみである．

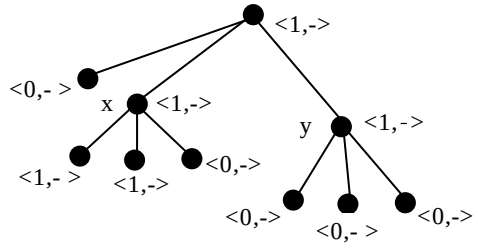
同じノードに複数のルールが適用される場合，衝突が起きる．衝突解消の方法として，以下の３つが考えられる．

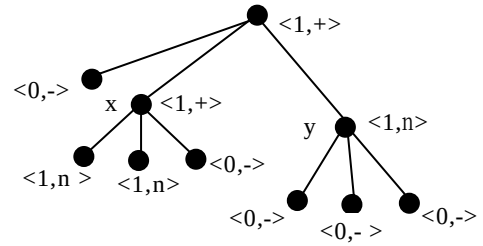
先行優先(First-applicable)： ポリシーのすべてのルールに線形的な順番をつけておく．ルールを最初から最後までの順番で評価する．ノードに適用可能な最初のルールのdecision（判定値）を当該ノードの判定結果とする．

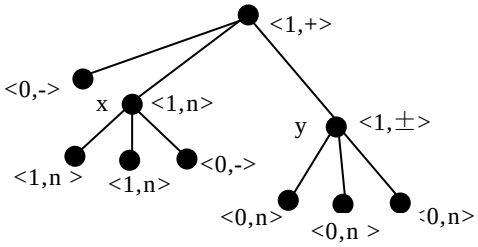
拒否優先(Deny-overrides)：ポリシー内のすべてのルールについて，どれか１つのルールの判定値がdenyであれば結果はdenyとする．すべてのルールが permit，あるいはいくつかのルールがpermitで残りのルールすべてがNotApplicable（判定できない）の場合のみ結果をpermitとする．

許可優先(Permit-overrides)： ポリシー内のすべてのルールについて，どれか１つのルールの判定値がpermitであれば結果はpermitとする．すべてのルールが denyあるいは幾つかのルールがdenyで残りのルールすべてがNotApplicableの場合のみ結果をdenyとする．

cascade を用いることにより，ルール数がどのように削減されるかを例により示す．図１には，与えられた文書木と各ノードの権限判定値について，cascade を全く使用しないでルールを設定したものであり，計 10 個のルールを必要としている．これに対し，図２は一部 cascade を用いたものであり，たとえばノード x で権限判定値 1 を+で cascade させることにより，２つの子供のルールを省略できている．まだノード y は根から cascade される値を使用できるため，y にルールを設定する必要はない．図３はさらに cascade の最適化を行って，ルール数を 5 まで削減したものである．ノード y では，±を設定することによって，y 自身では値 1 を，y の子には 0 を cascade させることにより，簡略化している．







2.2 First-applicable における cascade 最適化

与えられたアクセス制御ポリシーについて，First-applicable のもとで cascade を最適化し，ポリシーのルール数を最小化する，線形時間アルゴリズムを以下に示す．

アルゴリズム 1.
入力：各ノード e に 0(deny)または 1(permit)の権限判定値 a(e)が与えられた XML 文書 D
出力：Dに対するポリシーP_D

1. Dの根をeとする．eが子を持たないときは，rule (e) = “-“として終了する．eが子を持つ場合は，eの子の集合を$\{e_1,\ldots,e_k\}$とする．各子e_iについて，アルゴリズム 1 を再帰的に適用し，e_iを根とする部分木D_iに対するポリシーP_{Di}を求める．e_iに対するルールはrule(e_i)に格納される．
2. $cost^+ = 0, cost^- = 0, cost^{\pm} = 1$ とする．
3. for each e_i in $\{e_1,\ldots,e_k\}$ {
  if ($a(e_i) = a(e)$) {
   switch rule(e_i) {
    case “+” : $cost^-$ += 1; $cost^{\pm}$+= 1;
    case “-” : $cost^-$ += 1; $cost^{\pm}$+= 1;
    case “±” : $cost^+$ += 2; $cost^-$ += 2; $cost^{\pm}$+= 1;
   }
  }
  else {
   switch rule(e_i) {
    case “+” : $cost^+$ += 1; $cost^-$ += 1;
    case “-” : $cost^+$ += 1; $cost^-$ += 1;
    case “±” : $cost^+$ += 1; $cost^-$ += 2; $cost^{\pm}$+= 2;
   }
  }
}
4. if ($cost^+ = \min(cost^+, cost^-, cost^{\pm})$) {
  rule(e) = “+”;
  for each e_i in $\{e_1,\ldots,e_k\}$ {
   if ($a(e_i) = a(e)$) {
    if (rule(e_i) = “+”) OR (rule(e_i) = “-”) {
     rule(e_i) = “n”;
    }
   }
   else {
    if (rule(e_i) = “±”) {
     rule(e_i) = “-”;
    }
   }
  }
}
else if ($cost^{\pm} = \min(cost^+, cost^-, cost^{\pm})$) {
  rule(e) = “±”;
  for each e_i in $\{e_1,\ldots,e_k\}$ {

```
            if (a(e_i) ≠ a(e)) {
               if (rule(e_i) = "+") OR (rule(e_i) = "-") {
                  rule(e_i)   = "n";
               }
            }
         }
      }
      else {
         rule(e) = "-";
      }
   }
return;
```

[定理 1] アルゴリズム 1 は入力のXML文書Dのノード数|D|についてO(|D|)時間で停止し，First-applicableの衝突解消アルゴリズムを用いる場合のルール数最小のポリシーP_Dを求める．

[証明] アルゴリズム 1 において，各ノード e について高々１回のみ部分木の根として再起呼び出しが行われ，１回の呼び出しでは e とその子の数に比例した計算時間であるため，全体で O(|D|)時間であることがわかる．

ルールの合計数については，表１に示す各ノードのルールの数を合計したものである．以下，ルール数の最小性についてDのノード数に関する帰納法を用いる．

(1) D が１つのノードのみからなる場合は，１つのルール“-“がアルゴリズムにより得られ，また実際にこのルールが必要であるため，最小性は成り立つ．

(2) Dよりノード数が小さい任意の文書D’について，アルゴリズム 1 は最小のルール集合を求めると仮定する．このとき，Dの根eの子$e_1,\ldots,e_k$それぞれを根とする部分木を$D_1,\ldots,D_k$とする．アルゴリズム 1 は，各部分木のルール集合をステップ 1 で再帰的に求める．ステップ 3 では，eと$e_1,\ldots,e_k$の間で，eのルールを”+”，”-”，”±”のそれぞれのルールを設定したときに，省略できるルールを除いたルール数の合計を計算している．ステップ 4 では，ステップ 3 で求めたルールの合計数を最小とするルールをeに設定する．そして部分木の根$e_1,\ldots,e_k$それぞれについて省略できるルールを１つ除くかまたはそのままにする．ここで，もしアルゴリズム 1 よりもルール数が小さいポリシーP'_Dが存在したとする．もし，P'_Dの各部分木$D_1,\ldots,D_k$がP_D と一致するならば，根eのルール数がP_Dより小さいはずであるが，rule(e)はeと$e_1,\ldots,e_k$の間で最小に選んであるためそれはあり得ない．すると，部分木$D_1,\ldots,D_k$のいずれかでP'_DはP_Dより小さな解を求めているはずである．ところが帰納法の仮定によりP_Dは$D_1,\ldots,D_k$の部分で最小のルール数を求めて，これからさらに部分木の根$e_1,\ldots,e_k$のルールを削減しているか同じであるため，これもあり得ない．よってP_Dはルール数が最小である．□

2.3 Deny-overrides および Permit-overrides における cascade 最適化

Permit-overrides は Deny-overrides と同様な議論が可能であるため，一般性を失うことなく Deny-overrides について述べる．まず，Deny-overrides を First-applicable で模倣することについて検討する．First-applicable はルールの順序により，先行するルールの判定値が優先するというものであるから，Deny-overrides を模倣するには，Deny を行なうルールを先行させるように，ルールの順序付けを行なえばよいと考えられる．より一般的には，Deny-overrides と等しいか少ないルール数の First-applicable のもとでのポリシーが存在することを示せる．

権限判定を行なうコストの観点からは，Deny-overrides では文書木を根から判定対象のノードに降下しながら各ルールの評価値を求めていった場合，途中でも deny の値を与えるルールが見つかれば，そこで木の探索を終了して，権限判定は deny であると結論することができる．このため祖先で deny を行なうルールが多い場合など， Deny-overrides が有利な局面が考えられる．一方 ，Deny-overrides では deny をあるノードから cascade させた場合，そのノードからの部分木全体が deny になる．そのため，もしその部分木中に permit のノードが１つでもあるならば, cascade は使用できないことになる. また，アルゴリズム 1 で用いた±については，a(e)=permit のノード e については, deny が優先されるため±では deny に判定されてしまい，用いることができない．以上を考慮すると，Deny-overrides のもとでの cascade 最適化あるアルゴリズムを構成することができるが，紙面の制約により省略する．アルゴリズムの基本的構成は，First-applicable と同じく，動的計画法に基づき，葉から根の方向へ最小ルール数を与える cascade の設定を求めながら上昇するものである．計算時間はアルゴリズム１と同じく O(|D|)となる．Permit-overrides については，Deny-overrides のアルゴリズムにおいて permit と deny を交換すれば得られる．

図１の XML 文書とアクセス制御ルールについて，Deny-overrides に基づいて cascade の最適化を行なった結果を図４に示す．この例では，ルール数は１０から６に削減されている．

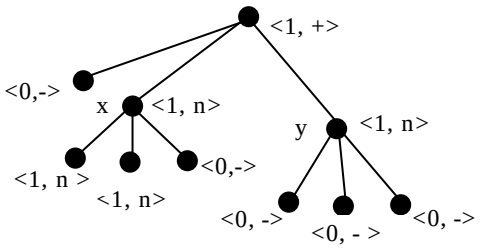


図４ Deny-overrides, cascade 最適化：ルール数 6
Fig. 4 Deny-overrides, cascade optimized. Rule count: 6

3. ポリシー簡略化の定量的評価

実用的な DTD として VISA 送り状[6]の DTD（アクセス制御情報なし）を用いて, 評価実験を行った. Pentium 4 CPU 3.20GHz，2GB RAM の Windows XP PC を使って， Java j2sdk_1.4.2_05 を用いてアルゴリズムを実装した．

VISA の DTD は, 要素がおよそ 350 種類で, 属性が約 610 種類で，文書木の深さが最大 5 層である．この DTD において 1,000 個のインスタンスをランダムに生成し，アクセス制御ルールを割り当てた．各要素に新しい属性としてルールの値（deny か permit）とルールのタイプ(‘n’, ‘+’, ‘-’, ‘±’)の２つを設定した．ルールが deny の確率は 5%の段階とし，5%～95%の範囲であった．

まず簡略化前後のルール数を計算し，ルール数をどれだけ減らしたかを計測した（図５）．First-applicable において，ルール数の削減効果がもっとも顕著である．ルールが deny

の確率が低いときは，permit の cascade が多く，一方 deny の確率が高いときは，deny の cascade が多いので，図の両端は簡略化によりルール数が大幅に減ったことを示している．Permit-overrides の折れ線は，Deny-overrides のそれと対称的になっている． また，ユーザ・ビューの生成時間を用いて，アクセス判定時間を評価した（図６）．

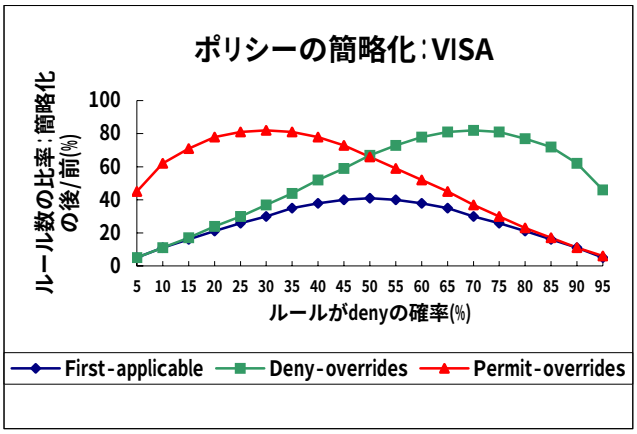


図５ ポリシーの簡略化結果

Fig.5 Result of Simplification of Policies

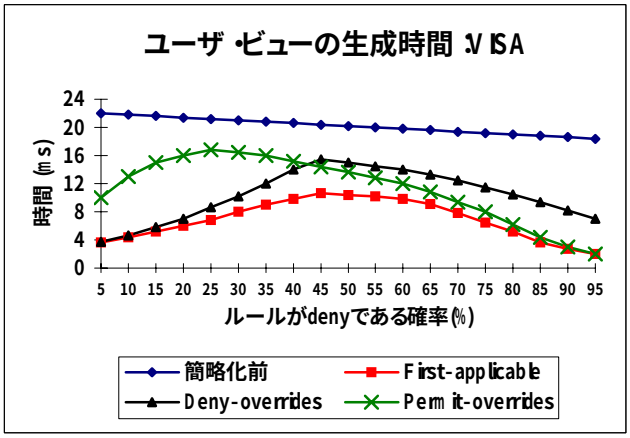


図６ ユーザ・ビューの生成時間

Fig.6 Generation Time of User View

ユーザ・ビューを生成するときに，インスタンスを１個ずつメモリに読み込んで DOM ツリーを築き，アクセス判定を行って新たにユーザ・ビューのツリーを作り，その生成時間を計る．DOM ツリーを２回築くが，元のツリーの生成時間はユーザ・ビューの生成時間に含めない．

ユーザ・ビューを作るのにかかる時間は，権限判定のための時間と，permit のノードを新しいツリーにコピーするための時間との合計である．したがって，簡略化前の生成時間だけでなく，いずれの生成時間も，deny の確率が増えるにつれ，一定のペースで下がる傾向にあると考えられる．

4. おわりに

インスタンスレベルでのアクセス制御ポリシー簡略化問題を検討し，cascade 最適化のアルゴリズムを提案し，計算機実験による定量的評価を行った．最適化によりルール数が大幅に減り（First-applicable の場合，最大 94%，平均 71% のルールが削減できる），アクセス判定時間も（First-applicable の場合，最大７分の１，平均５分の１まで）短縮されることが分かった．今後は，スキーマレベルでの簡略化問題について考察する予定である．

[文献]

[1] Bertino, E., Castano, S., Ferrari, S., and Mesiti, M. Specifying and enforcing access control policies for XML document sources. World Wide Web, 2000.

[2] Damiani, E., S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, “A Fine-Grained Access Control System for XML Documents,” ACM TISSEC, vol. 5, no. 2, 2002.

[3] Hitchens, M. and Varadharajan, V. RBAC for XML document stores. In Information and Communications Security, 3rd Int. Conference, LNCS2229, pp.121–135, Nov. 2001.

[4] Kudo, M., S. Hada, S. Paraboschi, and P. Samarati, “XML document security based on provisional authorization,” 7th ACM CCS, pp.87–96, Nov. 2000.

[5] http://www.oasis-open.org/committees/xacml, OASIS eXtensible Access Control Markup Language Technical Committee. XACML 1.0 Committee Specification Set. Nov. 2002.

[6] VISA: http://international.visa.com/fb/downloads/commprod/visaxmlinvoice/

[7] T. Yu, D. Srivastava, Laks V. S. Lakshmanan, H. V. Jagadish, “A compressed accessibility map for XML,” ACM TODS, Vol. 29 , No.2, pp. 363 – 402, June 2004.

岩井原 瑞穂 Mizuho IWAIHARA
1993年3月九大工学研究科情報学専攻博士後期課程修了．
2001年4月より京大情報学研究科社会情報学専攻助教授．日本データベース学会正会員，情報処理学会，電子情報通信学会，ACM, IEEE各会員．

王 波 Bo WANG
2002 年 3 月京都大学大学院経済学研究科修士課程修了．
2005 年 3 月京都大学大学院情報学研究科社会情報学専攻修士課程修了．

チャットウィチェンチャイ ソムチャイ Somchai CHATVICHIENCHAI
2004 年 3 月京都大学大学院情報学研究科社会情報学専攻博士後期課程修了．
2004 年 4 月より県立長崎シーボルト大学国際情報学部助教授．