完全な検証を行わずに XML 文書に単純型を付与する方式

Assigning simple types to XML documents without full validation

村田 真♠
Makoto MURATA

XML データベース検索言語 XQuery は，スキーマに基づいた型情報が XML 文書に付与されていることを前提とする．型情報を付与する処理は，XML スキーマ言語 W3C XML Schema に基づく検証の一部として行われる．しかし，この方法には W3C XML Schema 以外のスキーマ言語（とくに RELAX NG）に利用できないという問題点，検証という重い処理が必要になるという問題点がある．これらの問題点を解決するため，検証を行うことなしに単純型だけを付与する方式を提案する．本方式は，文書が妥当であることを保証せず，複合型も扱わないが，単純型を正しく付与することは保証する．逆に言えば，本方式は妥当でない文書に単純型を付与することができる．

In XQuery, it is assumed that XML documents are accompanied with type information obtained from schemas. Assigning type information is done as part of W3C XML Schema validation. However, this approach has two problems: (1) other schema languages including RELAX NG are not supported, and (2) W3C XML Schema validation, which is very expensive, becomes mandatory. To overcome these problems, this paper proposes a novel approach for type assignment that does not require full validation but is restricted to single types. Although this approach does not ensure validity of documents and cannot handle complex types, it can correctly assign simple types. In other words, this approach can assign simple types to invalid documents.

1. はじめに

XQuery[1]は，W3Cで制定中のXMLデータベース検索言語である．XQueryが対象とするXML文書は，XML 1.0[2]に規定されているXML文書とは厳密には異なる．すなわち，XML 1.0の構文解析によって得られる木はスキーマに基づく型情報を含まないが，XQueryの検索対象となる木はスキーマに基づく型情報を含む．型情報は，構文解析を行った後に，スキーマ言語W3C XML Schema[3]に基づく検証によって得られる（ただし他の方法が排除されているわけではない）．

型情報は，大きく二つに分かれる．一つは，xsd:intやxsd:booleanのような単純型(simple type)である．XML文書に現れる文字列が，実際にはどんなデータを表しているのかは単純型によって定まる．もう一つは，複合型(complex type)である．複合型は，要素がどんな子要素やテキストを持つかについての制約である．W3C XML Schemaでは，複合型宣言(complex type declaration)によって宣言される．

単純型を付与することには，XQuery検索式の信頼性や性能が向上するという利点がある．例えば，<a>3という要素の内容は単なる文字列に過ぎないが，単純型xsd:intを付与すれば3という整数として扱うことができる．これは型エラーの検出やインデックスファイルの作成に役立つ．

しかし，W3C XML Schemaに基づく検証によって型情報を得るという方法には，二つの問題がある．一つは，検証という重い処理が必須になることである．NicolaとJohn[4]は，大手ユーザにおけるXMLデータベースの性能について調査し，検証がXMLデータベース全体の性能を低下させることを報告している．もう一つの問題点は，W3C XML Schema以外のスキーマ言語，とくにRELAX NG[5]に適用できないことである．調査[6]によれば，2003年にWWW上にあるXML文書のうちW3C XML Schemaを利用しているものは0.09%に過ぎない．

これら二つの問題点を解決するため，本論文では，検証を行うことなしに単純型だけを付与する方式を提案する．この方式は，文書がスキーマに適合していることを保証せず，複合型も扱わないが，単純型を正しく付与することは保証する．逆に言えば，本方式はスキーマに適合しない文書に単純型を付与することができる．

単純型の付与は，パスオートマトンと部分的な検証を併用して行う．パスオートマトンは，スキーマから構築されるオートマトンであり，ルート要素からのパスを走査して要素・属性の型を選択する．パスオートマトンが単純型を選んだときには，要素・属性の内容である文字列がこの単純型に属することを部分的に検証した上で，この単純型を付与する．簡単のため要素のみについて説明するが，属性を扱えるように拡張することは容易である．

2. スキーマ

本研究では，スキーマを正規木文法によって表現する．DTD，W3C XML Schema，RELAX NGを含む多くのスキーマ言語が，木オートマトンによって表現可能なことが知られている[7]．

正規木文法は四つ組 $G = (N, \Sigma, S, P)$ である．ここで，

- N は非終端記号の有限集合，
- Σ は名前の有限集合，
- S は $\Sigma \times N$ の部分集合，
- P は生成規則 $x \rightarrow r$ の有限集合である（x は N の元，r は $\Sigma \times N$ の上の正規表現）．

複数の生成規則が左辺の非終端記号を共有することは，本論文では許さない．

正規木文法 G による木の解釈とは，各要素に非終端記号を一つ付与したものであって，以下の条件を満たすものである．

- ルート要素のラベルが a であり，付与された非終端記号を x であるとすると，$a[x]$ は S に属する．
- どの要素 e とその子要素 $e_1, e_2, \ldots, e_n$ についても，以下の条件を満たす生成規則 $x \rightarrow r$ が存在する．
  - 要素 e に付与された非終端記号が x である
  - 子要素 $e_1, e_2, \ldots, e_n$ のラベルが $a_1, a_2, \ldots, a_n$ であり，付与された非終端記号が $x_1, x_2, \ldots, x_n$ であるとすると，$a_1[x_1]a_2[x_2]\ldots a_n[x_n]$ は正規表現 r にマッチする．

♠ 正会員　日本IBM(株)東京基礎研究所・国際大学研究所
mmurata@trl.ibm.com

ある木に対してスキーマG に基づく解釈が少なくとも一つ存在するとき，この木はG に照らして妥当であるという．

木の正規木文法による解釈は，型情報を付与したものと見なすことができる．ここでいう型とは，非終端記号である．非終端記号のうち，$x \rightarrow \varepsilon$の形の生成規則の左辺に出現するものを，本論文では単純型と見なす（ここでεは空列を表す）．すなわち，単純型とは，N の部分集合$\{x \in N \mid x \rightarrow \varepsilon \in P\}$である．逆に言えば，単純型$d$ を持つスキーマを正規木文法によって表現するには，非終端記号x_d及び生成規則$x_d \rightarrow \varepsilon$を導入する必要がある．

例として，正規木文法$G_1 = (N_1, \Sigma_1, S_1, P_1)$を考える．ここで，*xsd:int, xsd:string*は単純型を表す非終端記号である．

- $N_1 = \{$*従業員型, xsd:int, xsd:string*$\}$,
- $\Sigma_1 = \{$*従業員，番号，名前*$\}$,
- $S_1 = \{$*従業員[従業員型]*$\}$,
- $P_1 = \{$*従業員型* $\rightarrow$
  (*番号[xsd:int]，名前[xsd:string]*),
  xsd:int $\rightarrow \varepsilon$,
  xsd:string $\rightarrow \varepsilon$ $\}$

なお，この正規木文法は以下のRELAX NGスキーマ（短縮構文）と等価である．

```
start =
  element 従業員 {
    従業員型
  }

従業員型 =
  element 番号{xsd:int},
  element 名前{xsd:string}
```

次の文書D_1を考える．

```
<従業員>
  <番号>918</番号>
  <名前>村田</名前>
</従業員>
```

この文書はひとつの解釈が存在する．ルート要素「従業員」に，複合型「従業員型」を付与し，子要素「番号」と「名前」にそれぞれ単純型xsd:intとxsd:stringを付与したものである．したがって，この文書はG_1に照らして妥当である．

3. パスオートマトン

本章では，正規木文法からパスオートマトンを構築する方法を示す．

パスオートマトンは，正規木文法から構築される非決定的有限状態オートマトンである．正規木文法 $G = (N, \Sigma, S, P)$ からパスオートマトン $M[G]$ を次のように構築される．

- アルファベット: Σ をそのまま用いる．
- 状態: 各非終端記号を状態として用いる．また，開始状態 start を導入する．すなわち，$N \cup \{$start$\}$が状態集合である．
- 開始状態: start が唯一の開始状態である．
- 終了状態: すべての非終端記号を状態とする．すなわち，Nが終了状態集合である．
- 遷移関数: 生成規則に基づいて遷移関数 δを以下のように定義する (ここで a はΣに属する名前であり，x は非終端記号とする).
  - $\delta(x, a) = \{x' \mid$ ある生成規則 $x \rightarrow r \in P$ に対し $a[x']$ が r の中に出現する$\}$
  - $\delta($start, $a) = \{ x' \mid a[x'] \in S \}$

先のスキーマG_1 から構築されるパスオートマトン$M[G_1]$を次に示す．

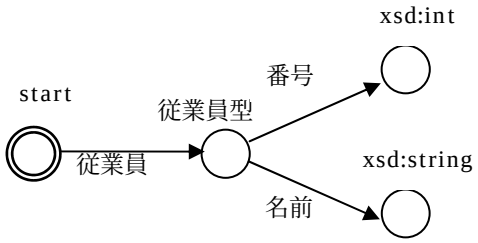

図1 パスオートマトン$M[G_1]$
Fig.1 Path automaton $M[G_1]$

- アルファベット: {従業員，番号，名前},
- 状態: 従業員型, xsd:int, xsd:string, start
- 開始状態: start
- 終了状態: 従業員型, xsd:int, xsd:string
- 遷移関数:
  - $\delta($start，従業員$) = \{$*従業員型*$\}$,
  - $\delta($ 従業員型，*番号*$) = \{$*xsd:int*$\}$,
  - $\delta($ 従業員型，*名前*$) = \{$*xsd:string*$\}$

4. 型情報の付与

本章では，各要素までのパスに対してパスオートマトンを実行したのち，部分的な検証を行うことによって単純型を付与する方法を示す．

文書中のある要素eを考える．ルート要素から要素eまでのパスに対してパスオートマトン$M[G]$を実行すると，いくつかの状態からなる集合が得られる．この状態集合をe^Gで表すことにする．

e^Gに含まれるすべての単純型$t_1, t_2, \ldots, t_n$を考える．これらに対してeの内容を部分的に検証する．この結果には次の 4 つの場合がある．

場合1. n =0 すなわちe^Gが単純型を含まない
この場合には，部分的な検証を行う必要がない．XQueryは，すべてを許容する型としてxsd:anyTypeを提供しているので，これをeの型とする．

場合2. どのt_iに対しても部分的な検証が成功しない
どのt_iを用いることもできないので，xsd:anyTypeをeの型とする．

場合3. 複数のt_iに対して部分的な検証が成功する
一つのt_iに決めることができないので，xsd:anyTypeをeの型とする．

場合4. ただ一つのt_iに対して部分的な検証が成功する
成功したt_iをeの型とする．

先に示した文書D_1「番号」要素に対して，型情報を付与することを考える．パスオートマトン$M[G_1]$ の実行によって得られる状態は，単純型xsd:intである．次に，この要素の内容

である文字列"918"を, xsd:intに照らして部分的に検証する. この検証に成功するので, xsd:intをこの要素の型として付与することができる. 同様にして,「名前」要素に対して, xsd:string型を付与することができる. 一方, ルート要素である「従業員」に対しては, 型情報としてxsd:anyTypeを付与する. $M[G_1]$の実行によって得られる「従業員型」は, 単純型ではないからである.

次に, 妥当でない文書に対して型情報を付与する例を示す. 以下のXML文書D_2は, 余分な「備考」要素があるので, G_1に照らして妥当ではない.

```
<従業員>
  <番号>918</番号>
  <名前>村田</名前>
  <備考>なし</備考>
</従業員>
```

この文書に対して型情報を付与すると, 先の文書とほぼ同様の結果が得られる. すなわち, 要素「従業員」,「番号」,「名前」には, それぞれxsd:anyType, xsd:int, xsd:stringが付与される. 唯一の違いは, 要素「備考」にxsd:anyTypeが付与される点である（e^Gが空集合となるため）.

妥当でない場合には, そもそも解釈が存在しない. 妥当でない場合に型情報を付与するのは不適切であり, 検証によって妥当性を確認すべきであるという考え方もある. しかし, 実用的には, たとえ妥当でなくても型情報をできるだけ付与したいという要求がある. 文書の一部が誤っている又は現時点で完成できない場合でも, 単純型を利用した XQuery 検索が可能なのは, むしろ本研究の利点であると筆者は考える.

5. まとめと今後の課題

本論文では, 型情報の付与を検証から分離する方式を提案した. この方式は, スキーマからパスオートマトンを構築する処理, パスオートマトンを実行して型の候補を得る処理, 文字列を単純型と照合する処理の 3 つからなる.

本方式は文書の妥当性を保証せず, 複合型も扱わないが, 単純型を正しく付与することは保証する. 一方, 妥当でない文書についても, 単純型を利用した XQuery 検索が可能であるという特徴がある.

今後の課題として, 文書の妥当性が何らかの方法（例えば静的な型検査を行う XML 変換言語）によって保証されている場合への対処がある. この場合には, 複合型を付与することが可能になる. また, スキーマ以外の手段によってパスオートマトンを構築することも検討していく必要がある.

[謝辞]

DBWeb2003 の匿名の査読者のコメントに感謝する.

[文献]

[1] W3C: "XQuery 1.0: An XML Query Language", W3C Working Draft (2003).
[2] W3C: "Extensible Markup Language (XML) 1.0 (2nd Edition)", W3C Recommendation (2000).
[3] W3C: "XML Schema", W3C Recommendation (2001).
[4] Nicol, M. and John, J.: "XML Parsing: a threat to database performance", ACM CIKM (2003).
[5] OASIS: "RELAX NG Specification", OASIS Committee Specification (2001).
[6] Mignet, L., Barbosa, D. and Veltri, P.: "The XML Web: a First Study", International World Wide Web Conference (2003).
[7] Murata, M., Lee, D. and Mani, M.: "Taxonomy of XML Schema Languages using Formal Language Theory", Extreme Markup Languages (2001).

村田 真 Makoto MURATA
日本 IBM(株)東京基礎研特別研究員・国際大学研究所特任研究員. 1982 京都大学理学部卒業. W3C, IETF, OASIS, ISO/IEC, 国内委員会において XML の仕様制定・研究に従事. 情報処理学会会員. 日本データベース学会会員. インターネットコンファレンス'98 論文賞.