NVDL におけるストリーム処理による XML 複合文書の検証

Stream NVDL Validation for XML Compound Documents

宮下 尚[1] 村田 真[2]

Hisashi MIYASHITA Makoto MURATA

近年、名前空間を利用して、XML の多様な語彙を組み合わせて XML 複合文書が記述されることが多くなっている。このような XML 複合文書に対しての検証を行なう技術として、NVDL (Namespace-based Validation Dispatching Language)が注目されている。NVDL は、XML 複合文書を名前空間によって分割して、適切なスキーマによって検証することを可能とする。しかし、NVDL では、ストリーム処理を行なうことは簡単ではない。なぜなら、複数の分割結果を許している上に、各々の分割結果に対して個別に検証器を起動できるためである。本稿では、NVDL による検証を、ストリーム処理で行なうことができるアルゴリズムを提案する。

Recently, many XML compound documents are composed with multiple XML vocabularies by using XML namespace. NVDL (Namespace-based Validation Dispatching Language) is paid attention to validate such XML compound documents, which enables us to componentize schemas by namespaces by dividing XML documents and validating each fragment with an appropriate schema. However, streaming validation by NVDL is not easy because NVDL can divide documents in many ways and invoke a validator for each of them.

In this paper, we show a novel algorithm that can divide and validate incoming XML documents with stream processing.

1. はじめに

XML [4]は、一般の文書(XHTML[3])、メッセージ表現(SOAP[4])、グラフィックス表現(SVG[5])等、様々な分野において情報を記述するために利用されるようになってきている。XML では、名前空間 [3]という機構によって同一の文書中に、多様な語彙を組み合わせて記述することを可能としている。各々の XML による語彙に対して、一意性のある名前空間を割り当てることによって、名前の衝突が起こることなく一つの文書に複数の組み合わせることが可能となる。名前空間を用いて複数の語彙を組み合わせて記述された文書を XML 複合文書と呼ぶ。しかし、XML 複合文書の検証を従来までのスキーマ言語で行なうことは、難しいという問題点があった。NVDL(Namespace-based Validation Dispatching Language) [2]は、XML 複合文書を、名前空間を用いて分解・再構成をして一つまたは複数の検証器に送出するための技術である。NVDL によって、個々のスキーマは単一の名前空間に再構成された文書に対して検証をすることに集中することが出来る。

しかし、NVDL では、ストリーム処理によって検証を行なうことは簡単なことではないという問題がある。XML 文書の検証処理において、ストリーム処理ができることは重要である。ストリーム処理が出来ない場合には、始めに文書全体をメモリ中にロードする必要があるため、メモリ使用量及びレイテンシの点から不利になってしまうためである。

NVDL標準では、参照モデルによってNVDL検証のアルゴリズムが定義されている。しかし、この参照モデルでは、メモリ上に保持された木構造の操作としてアルゴリズムが定義されているため、そのままではストリーム処理が出来ない。

この参照モデルと等価なストリーム処理可能なアルゴリズムを実現することは、容易なことではない。なぜなら、名前空間によって分割された各々の文書の断片に対して、複数の操作を同時に可能としているためである。すなわち、文書の断片に対して、以下の複雑な動作を複数同時に処理しなくてはならない。

- 文書の一断片に対して、新たに1つまたは複数の検証器を起動する。
- 文書の断片を、親要素や祖先要素の一部として、すでに起動された検証器によって検証する。

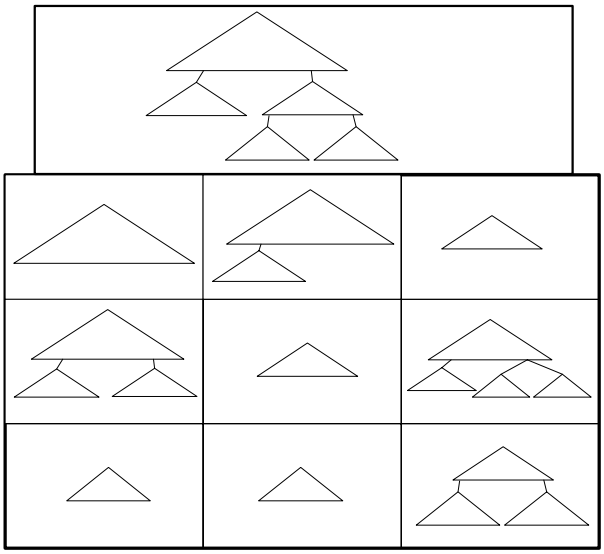

図 1 NVDL による検証処理の例

Fig.1 An Example of Validation Process by NVDL

図 1 に、検証処理の一つの可能性を挙げる。この処理では、図の一番上にある 5 つの三角形によって示されている XML 文書を、各々の三角形で表される XML 文書の断片に対して、新たに検証器を起動し、親要素もしくは祖先要素の一部として検証することの両方を同時に指示している。結果として、この例における NVDL の検証処理では、図中の下側に枠で囲んで表記した 9 種類の検証を同時に行なわなくてはならない。この図は、文書断片``1"だけからなる文書をスキーマ A に照らして検証する、文書断片``1"および``11"からなる文書をスキーマ B に照らして検証する、文書断片``11"だけからなる文書をスキーマ C に照らして検証する、文書断片``1"と``11"と``12"からなる文書をスキーマ D に照らして検証する

[1] 正会員 日本 IBM 東京基礎研究所, himi@jp.ibm.com

[2] 正会員 日本 IBM 東京基礎研究所, eb2-mrt@asahi-net.or.jp

[3] http://www.w3.org/TR /2001/REC-xhtml11-20010531

[4] http://www.w3.org/TR/soap12

[5] http://www.w3.org/TR/2003/REC-SVG11-20030114/

...ということを意味している。

このような検証をストリームで処理する場合には、例えば、``121"の頂点にある要素がストリームに出現した場合には、スキーマ G によって単独で検証を開始し、文書断片``1"にある要素の一部としてスキーマ F によって検証を行い、文書断片``12"にある要素の一部としてスキーマ I によって検証を行なう必要がある。特に、スキーマ F による検証では、``12"の文書断片を取り除き、``121"および``122"の文書断片を``1"の文書断片に接続して検証を行なわなくてはならない。このような複雑な検証を、メモリ中に木構造を保持せずに行なうことは単純なことではない。

NVDL では、このような複雑な分割・検証処理を許しているにもかかわらず、仕様では、ストリーム処理を阻害する致命的な条件は巧妙に排除されている。たとえば、分割や再構成を行なう処理は、木のルートからのパスだけで決定できるように設計されている。しかし、NVDL の検証をストリーム処理で行なう効率的なアルゴリズムについての研究は行なわれていなかった。

本稿では、このような複雑な NVDL の処理に対して、ストリーム実装ができる新しい検証アルゴリズムを提案する。このアルゴリズムでは、XML 文書の分割と、分割された各々の文書の検証を同時に、しかもメモリ中に文書の全体を配置することなく行うことが出来る。提案するアルゴリズムでは、ストリーム中のイベントに対して、プッシュダウン・オートマトン(PDA)を適用して、分割及び検証の規則を同時に複数割り当てることで処理を行なう。この方式によるメモリ使用量は、文書インスタンスについて木構造の高さにのみ依存し、幅の大きさには依存しない。

2. ストリーム実装可能な検証アルゴリズム

本節では、NVDLのストリーム実装可能な検証用アルゴリズムを提案する。まず、始めにNVDL標準の参照モデルについて概説し、次に、その参照モデルと同一の結果を得ることが出来るPDAによるアルゴリズムを提案する。最後に、分割された文書から検証器を選択して配送する機構について説明する。

2.1 NVDL 標準における検証の参照モデル

NVDL 標準の参照モデルにおける検証処理は、文書全体を同一の名前空間に属する文書の断片(セクションと呼ばれる)に分割することから始まる。次に、各々の要素セクションおよび属性セクションに対して動作を割り当てる。NVDL では、動作には *attach* 、 *unwrap* および *validate* の 3 種類が定義されている[6]。最後に、割り当てられた動作の解釈を行う。セクションに割り当てられた動作が *attach* であれば、そのセクションを文書中で祖先に位置するセクションに取り付ける。動作が *unwrap* であれば、そのセクションを消去する。動作が *validate* であれば、そのセクションをルートとして検証を行う。

2.1.1 要素セクション及び属性セクションへの分割

NVDL 標準の参照モデルにおける検証処理では、文書をセクションと呼ばれる単位に分割する。セクションは、要素セクション(element section)および属性セクション(attribute section)からなる。大まかに言うと、要素セクションとは、同じ名前空間に属する要素とその子孫からなる最大の文書断片である。そして、属性セクションとは、一つの要素中の属性の集合から構成され、同じ名前空間に属する属性からなる最大の集合である。

例として、図2に示したXML複合文書を考える。この文書を要素セクションと属性セクションに分割すると、図3に示したように、3個の要素セクションと、1個の属性セクションが作られる。

```
<ex xmlns="ns" xmlns:ns1="ns1" xmlns:ns2="ns2">
  <e1 ns2:b="b" ns2:c="c">t1</e1>
  <ns1:e2><e/></ns1:e2>
</ex>
```

図 2 XML 複合文書の例

Fig.2 An Example of XML Compound Document

要素セクション 1:

```
<ex xmlns="ns" xmlns:ns1="ns1" xmlns:ns2="ns2">
<e1! asn1 >t1</e1>esn2 </ex>
```

要素セクション 2:

```
<ns1:e2 xmlns:ns1="ns1">esn3</ns1:e2>
```

要素セクション 3:

```
<e xmlns="ns"/>
```

属性セクション 1:

```
{ns2:b="b" ns2:c="c"}
```

図 3 要素セクション・属性セクションに分割された XML 複合文書

Fig.3 An XML Compound Document Divided into Element and Attribute Sections

要素セクション1の中にある *asn1* は、属性スロットノード (attribute slot node)とよび、もともとそこに属性セクション1があったことを指し示すための印である。同様に *esn2, esn3*は要素スロットノード (element slot node)と呼び、そこに要素セクション2, 3がそれぞれあったことを示す。

2.1.2 セクションに対する動作の割り当て

NVDLは、前節の様にして作られたセクションに対してMealy機械によって動作を割り当てることで検証方法を決定する。例として図4のNVDLを考える。このNVDLを、NVDL標準に基づいてMealy機械に変換すると図5のようになる。この図中の各エッジには入力シンボル(名前空間URI及び、属性セクション(as)か要素セクション(es)かを表すシンボルのペア)と、適用される動作が記述されている。NVDLでは、一つの遷移に複数の *validate* 動作を割り当てることが出来るため、Mealy機械は非決定的になり得る。つまり、一つのセクションに複数の *validate* 動作が割り当てられる可能性がある。このMealy機械を、図3で示したように分割された文書に対して適用すると、図6のように動作を割り当てることが出来る。

2.1.3 動作の解釈

属性セクション及び要素セクションに動作が割り当てられた後は、それらのセクションから文書を再構成し、検証を行う。要素セクションに対する動作と、属性セクションに対する動作は、以下に示すように少々異なる。

[6] NVDL では allow、deny、 attachPlaceHolder という動作も定義されている(本稿では割愛する)。

```
<rules
  xmlns="http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0">
 <namespace ns="ns">
  <validate schema="ns-schema.rng">
   <mode>
    <namespace ns="ns1">
     <unwrap><mode>
      <namespace ns="ns">
       <attach/>
      </namespace>
     </mode></unwrap>
    </namespace>
    <namespace ns="ns2" match="attributes">
     <validate schema="ns2-schema.rng"/>
    </namespace>
   </mode>
  </validate>
 </namespace>
```

図 4 NVDL スキーマの例
Fig. 4 An NVDL Schema

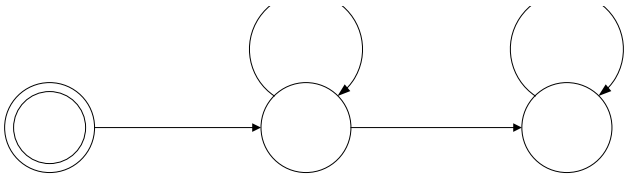

図 5 図 4 の NVDL スキーマに対応する Mealy 機械
Fig. 5 A Mealy machine of the NVDL schema in Fig. 4

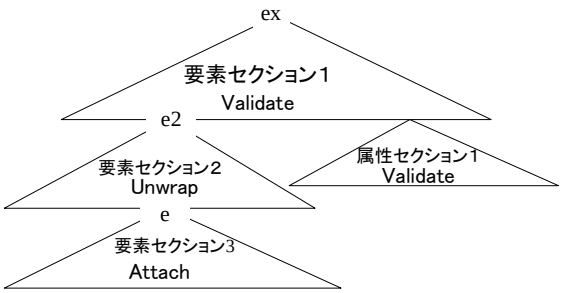

図 6 XML 複合文書に対する動作の割り当て
Fig. 6 Assigning Actions to the XML Compound Document

要素セクションに *validate* が割り当てられている場合には、その要素セクションを文書要素として、*validate* 動作に割り当てられている検証器で検証を行う。要素セクションに *attach* が割り当てられている場合は、親の要素セクションの対応する要素スロットノードに取り付ける。要素セクションに *unwrap* が割り当てられている場合には、その要素セクションが持つすべての要素スロットノードに(*attach* によって)取り付けられたすべての要素セクションを、親の要素セクションの対応する要素スロットノードに取り付ける。

一方で、属性セクションに *validate* が割り当てられている場合には、その属性セクションに対して *virtualElement* という名前の仮の要素にたいして属性セクションを取り付けて、指定された検証器で検証を行う。属性セクションに *attach* が割り当てられている場合には、親の要素セクションの対応する属性スロットノードに取り付ける。属性セクションに *unwrap* が割り当てられている場合には、何も行わない。

以上の動作の解釈を図6で示した例に対して適用すると、図7で示すように検証が行われる。すなわち、要素セクション2の部分が *unwrap* 動作によって削除され、かわりに要素セクション3が要素セクション1に *attach* 動作によって取り付けられ、これが検証単位1となる。そして、(図4での4行目の) *validate* 動作によって、*ns-schema.rng* を用いて検証されることになる。

また、属性セクション1は、(図4での13行目の) *validate* 動作によって、仮要素である *virtualElement* に取り付けられて、*ns2-schema.rng*を用いて検証されることになる。

結果として、この例においては、*ns-schema.rng* を用いて

```
<ex xmlns="ns"
    xmlns:ns1="ns1" xmlns:ns2="ns2">
  <e1>t1</e1>
  <e/>
</ex>
```

が検証されることになる。また、*ns2-schema.rng*を用いて

```
<virtualElement xmlns="..."
 xmlns:ns2="ns2" ns2:b="b" ns2:c="c"/>
```

が検証されることになる(なお、"..." には*virtualElement* のために予約されている名前空間URIである http://purl.oclc.org/dsdl/nvdl/ns/instance/1.0 が入る)。

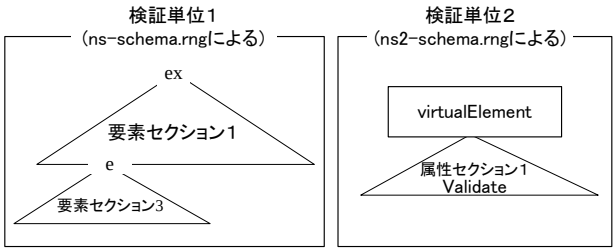

図 7 図 6 による動作の解釈によって行なわれる検証
Fig. 7 Validations by the Assigned Actions in Fig. 6

2.2 ストリーム処理可能なアルゴリズム

前節で説明したNVDLの参照モデルによるアルゴリズムによって、確かに文書を分割して検証することが可能である。しかし、以上のような動作は木構造の変形を伴う上、同時に複数の候補を検証する必要があるため、ストリーム処理をそのまま行なうことは出来ない。本節ではPDAを用いたストリーム処理可能なアルゴリズムを提案する。

2.2.1 PDA による動作の割り当て

2.1.2節でみたように、属性・要素セクションを入力列とみなせば、動作の割り当ては非決定有限オートマトン(NFA)で行うことが出来る。しかし、SAXストリームにおけるイベントでは、NFAでは入れ子の構造を認識することが出来ない。なぜなら入れ子の階層は、有限で押さえられないため、正規言語では表現できないためである。そのため、本アルゴリズムでは、スタックを導入して入れ子の構造を認識しつつ、2.1.2節のMealy機械を元にして以下の様にPDAを作成する。

1. すべての状態において、同一要素セクション中では、スタックに同一の名前空間を積み込むだけで遷移を行わない様にする。
2. 開始タグが現れた場合には、スタックに以前の状態をプッシュし、Mealy機械の指定に従って新しい状態に移行する[7]。
3. 終了タグが現れた場合には、スタックをポップして以前の状態に戻る。

このようにして、図5のMealy機械をPDAに変換すると、図8の様になる。なお、この図においては、単純化のために1, 2によって生成される遷移については記述していない。また、属性に関しては、同一要素上にある同一名前空間の属性をまとめて処理することで、要素と同様の扱いが可能となる。

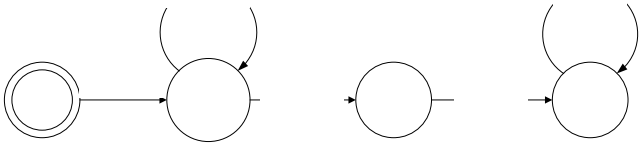

図8　図5のMealy機械に対応するPDA
Fig.8 A PDA Corresponding to the Mealy Machine in Fig.5

このようにして構成されたPDAを用いれば、ストリーム処理によって、各イベントに対して図6と同様に動作を割り当てることが出来る様になる。

2.2.2 ストリーム処理における動作の解釈

この節では、各イベントに割り当てられた動作から検証器を選び、適切に配送を行なう方法の説明を行なう。

対象のSAXイベントが *startElement* である場合(開始タグに対応する)、まず属性から処理を行う。この場合、2.2.1節で説明したとおり、同一名前空間の属性はまとめて処理する必要がある。属性に割り当てられた動作が *validate* の場合には、仮の要素である *virtualElement* 要素に同一の名前空間の属性を取り付けて、指定された検証器に配送を行う。割り当てられた動作が *attach* の場合には、もともとの要素にその属性を残す。そして、後述する要素に関する処理にしたがって検証器に配送される。これによって、もともとの属性スロットノードに取り付けることと同じ効果を得ることができる。割り当てられた動作が *unwrap* の場合には、その属性については、もともとの要素から取り除いて、何も行わない。

引き続き、要素に関する処理を行う。割り当てられた動作が *validate* である場合で、その同一要素セクションについて、まだ検証器が起動されていない(まだ新しい検証単位の文書が開始されていない)のであれば、*validate* 動作に対応した検証器に対して新しい検証単位を開始して、続いて、対象のSAXイベントをその検証器に送る。

割り当てられた動作が *attach* の場合には、PDA中のスタックをさかのぼり、動作が *validate* であるところに対応した検証器に対して、対象のSAXイベントを送出する(対象のSAXイベントは、 *validate* 動作が割り当てられた要素の子要素なので、すでに検証単位は開始されているはずである)。これによって、 *validate* 対象の要素セクションに該当する要素スロットノードに取り付けたことと、同じ効果を得ることが出来る。割り当てられた動作が *unwrap* である場合には、そのSAXイベントについては何もしない。

以上の様に処理を行なうことによって、図2の例では、図7の左側で示したように、要素セクションについての検証が行なわれることになる。

なお、2.2.1節によって生成されるPDAは非決定的であるため、一つのイベントに複数の *validate* 動作が割り当てられることが起こりうる。この場合、すべての解釈について検証を行う必要がある。この時、各々の解釈についてスタックの状態が異なることに注意する必要がある。

3. おわりに

本稿では、XML複合文書を検証する手段としてのNVDLについて説明し、ストリーム実装可能なアルゴリズムを示した。

NVDLは、XML複合文書を適切に分割することによって、すべてのXML語彙についての知識がなくとも、個々の名前空間の処理を統合することによって、文書全体の検証処理が可能であるように設計されている。XML複合文書は、今後利用の拡大が見込まれる分野であり[5]、今後は、検証処理以外にも利用可能な、より汎用性のある配送器(Dispatcher)としてNVDLを活用できるようにしたい。そのためには、XML複合文書の処理自身の研究、特にソフトウェアコンポーネントに分割された文書をどのように配信すればよいのかを、考慮する必要があると考えている。

[文献]

[1] XML 正規言語記述 RELAX 名前空間 (2003).標準情報 (TR) TR X 0044:2001

[2] Information technology---Document Schema Definition Languages (DSDL) - Part 4: Namespace-based Validation Dispatching Language (NVDL) (2006). ISO/IEC 19757-4.

[3] Bray, T., Hollander, D. and Layman, A.:Namespaces in XML. W3C Recommendation, January 14, 1999. http://www.w3.org/TR/1999/REC-xml-names-19990114.

[4] Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E. and Yergeau, F.: Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation February 04 2004, http://www.w3.org/TR/REC-xml-20040204.

[5] Mehrvarz, T., Appelquist, D., Pajanen, L. and Quint, V.: Compound Document by Reference Framework 1.0. Editor's Draft, June 15, 2006, http://www.w3.org/2004/CDF/specs/CDR/wp-1/cdf.html.

宮下 尚　Hisashi MIYASHITA

2001年Free Standards Group、OpenI18N WGでInput Method Subgroup Leaderとして入力メソッドの開発および標準化に従事. 2003年日本IBM東京基礎研究所に入社. 以来、XML、高信頼性ミドルウェア、アクセシビリティの研究開発に従事. ACM、情報処理学会会員.

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. インターネットコンファレンス'98論文賞. 情報処理学会業績賞.

[7] スタックに動作を記憶するためには、状態に動作を含める必要がある。このためには、Mealy機械をMoore機械に変換してPDAを構成すればよい。