ソフトウェア工学 129－17
（２０００．１１．１７）

制約規則に基づく XML 変換言語に関する一考察

横関 大子郎, 村山 隆彦, 山本 修一郎
NTT 情報流通プラットフォーム研究所
〒180-8585 東京都武蔵野市緑町 3-9-11
TEL: 0422-59-4870
E-mail: yokozeki.daigoro@lab.ntt.co.jp

概要:

企業間情報交換に現在XMLが使われはじめ，企業毎に異なるXMLの形式を変換するXSLT等の技術の重要性は高まってきている。本稿では制約規則に基づくXML変換言語XTLについて考察する。XTLは汎用的なXML変換言語であり、具体的な処理は、操作系により与える。操作系として選択、生成、変換、同値分割、統合などの例を示す。またXTLがXSLTに対して記述性で優れていることならびに、その有効性と限界を明らかにする。

キーワード:XML, 変換, 問い合わせ言語, XSLT, 半構造, 帳票

A Consideration on a constraint based XML transformation Language

Daigoro YOKOZEKI, Takahiko MURAYAMA, Shuichiro YAMAMOTO
NTT Information Sharing Platform Laboratories.
3-9-11 Midori-cho Musashino-shi, Tokyo 180-8585
+81 422 59 4870
E-mail: yokozeki.daigoro@lab.ntt.co.jp

Abstract

XML has been adopted in the B2B integration. Since each company has their own XML data format, the importance of XML transformation languages like XSLT is growing. In this paper, we consider a constrained based XML Transformation Language (XTL) that aims to promote B2B information sharing. XTL is a multi-purpose XML transformation language, and the concrete semantics of the XTL is given by operational definitions. We show the examples of XTL's operation include selection, generation, transformation, division and integration). We also show the effectiveness and the limitation of the XTL. in comparison with XSLT.

keywords: XML, transformation, query language, XSLT, semi-structure, form

1. はじめに

企業間の業務プロセスを統合する上では、流通対象としての伝票や仕様書のような構造をもつ帳票をXMLで表現するだけではなく、帳票に関する処理も必要となる。このようなXML文書の処理にはある帳票の項目を他の帳票の一部に転記する、1つの帳票を複数に分割する、複数の帳票を1つの帳票に合成するなどがある. すなわち、異なる構造を持つ帳票に含まれる要素情報を他の帳票への転記、帳票の分割、帳票の合成などを実現する必要がある。このような帳票情報の転記や分割、合成などを帳票に対する操作と呼ぶ。

このような帳票をXMLデータベースで管理する場合、B2B取引のメンバごとにアクセス権限を与えておき、必要な部分だけを追加、削除、更新できる必要がある。このような処理は帳票の一部をタグ構造に基づいて選択したり変形する帳票操作であると考えられる。したがって、B2Bのための情報交換プラットフォームを効率的に実現するためにはXMLで記述された帳票を管理するだけでなく、上述した操作を含めて実現する必要がある。

この問題を解決するため、入力側と出力側のXML文書の要素を制約規則に基づいて対応付けておき、この制約規則に基づいてXML文書の操作を実行するXML変換方式を提案した[1]。本方式の利点は、帳票と帳票間の関連付けるだけの記述により帳票操作を容易に実現できるため、業務担当者だけでB2Bの情報交換サービスを構築でき企業間の業務プロセス統合を促進できることである。

以下では、2章でXML変換を記述するための言語XTL(XML Transformation Language)設計方針, 言語仕様とその具体例を説明する。3章でXTLの有効性に関する考察を述べ、5章でまとめる。

2. XML変換言語XTL

2.1. 設計方針

前章で述べたXML変換を実現するため、XML変換言語XTLの設計では、業務担当者が容易に文書間の依存関係を定義できるように以下のような設計方針を策定した。

[方針 1] 制約言語とその解釈系を分離することにより、同一の制約に対して多様な変換を定義できること

[方針 2]制約言語では、入力文書構造と出力文書の構造をXMLで自然に対応付けることができること

まず[方針 1]を説明する。XML文書の変換をXML文書変換の目的としての操作とその操作の対象となるXML文書間の制約の対とし規定する。すなわち、

XML文書変換 ＝ 操作 ＋ 制約

であると考える。したがって、このアプローチでは同一の制約に対して複数の操作が存在する。各操作ごとに異なる制約言語表現を与えるのではなく、変換対象となるXML文書間の構造上の対応関係を記述する制約言語と、変換の意味付けを与える操作に分けることにより変換を明確に定義することができる。言い換えれば制約は文書の構成要素間の静的な関係を記述し、操作はその入力と出力との関係構造の上に成り立つ解釈のひとつを与えていると考えることができる。このようにXTLでは各変換機能毎に個別の言語、機能を用意するのではなく、汎用の変換記述用の制約言語と、その意味付けを与える操作系に分離することでXML文書の操作を業務担当者が理解すべき変換言語の語彙や構文を削減できるという特徴がある。以下では変換言語XTLに対してその操作系をXTE(XML transformation Engine)と呼ぶ。

たとえば以下のようなXTEの操作が考えられる。

(1) 選択　構造変換を行わず、特定の条件を満たす部分木の抽出を行う操作である。
(2) 生成　変換元文書がなく、新規にXML文書を生成する操作である。
(3) 構造変換　変換元の文書から値を収集し、値の依存関係や、要素名の置換等、XMLの構造情報の変換を行う操作である。
(4) 同値分割　(3)の構造変換と同時に、ある値の同値関係に基づき、文書を分割する操作である。
(5) 統合(更新)更新先文書と、更新元文書のある値(キー)の同値性に基づき、キーに関連する値を更新元文書から、更新先文書へ転記する処理である。

次に[方針 2]について説明する。XML やプログラミングに習熟していない業務担当者でも容易に帳票の変換を定義できるようにするためには可能な限り帳票の構造上の対応関係だけで帳票間の関係が定義できるだけでなく、制約言語には余分な構文を持ち込まないようにして言語を可能な限り単純化すべきである。このために、XTL では入力帳票と出力帳票の構成要素間を直接対応付けるようにし、帳票自身を制約言語の中に埋め込んでXMLで記述できることとした。

2.2. 制約言語 XTL

以下ではXMLを用いてXMLの変換を定義できる変換言語 XTL について述べる。まず変換言語の構文を以下に示す。ここで、<>で囲まれた英文字列で制約ルールの予約語タグを示す。また、a{,a}…で構文要素の繰り返しを表す。[a]で a があるかないかのいずれかであることを示す。a | b で構文要素 a と b のいずれかの選択を示す。英字列は非終端記号である。

XTL の制約ルールは複数のセクションから構成される。各セクションは、ソース・テンプレートとターゲット・テンプレートから構成され、一組の対応関係を表現している。

```
Rule ::= <rule> Section {, Section}…</rule>
Section ::= SourceTemplate  TargetTemplate
SourceTemplate ::=<source>Template</source>
TargetTemplate ::= <target> Template </target>
```

変換操作の場合、ソース・テンプレートが XML 文書の問い合わせにおける値抽出検索部分を表現し、ターゲット・テンプレートが XML 文書の再構築部分を表現している。ソース、ターゲット・テンプレートは、対称的な構造を持っており、逆方向にルールを適用する操作を与えることにより、逆変換への対応も可能となる。

テンプレートはテンプレート要素の入れ子からなる階層構造を持つ XML データである。

```
Template::=
  <TagName>TemplateBody</TagName>
TemplateBody::=
  <TagName>TemplateBody</TagName>
  {,<TagName>TemplateBody</TagName>}…
  | TemplateElement
```

テンプレート要素は、通常の XML データ要素 XMLelement、条件式 Expression、転記型タグ値変数 CopyTagValue、類別型タグ値変数 ClassTagValue、生成型タグ値変数 GenerateTagValue、マクロタグ値名 MacroTagName のいずれかである。

```
TemplateElement ::= XMLelement | Expression |
    CopyTagValue | ClassTagValue |
    GenerateTagValue | %MacroTagName%
CopyTagValue::= *TagValueName[Expression]
ClassTagValue::= ?TagValueName [ Expression ]
Expression::=
     Operator TagValueString |
     Operator [%MacroTagName% ]
Operator ::= == | != | >= | <= | > | <
GenerateTagValue ::=
     + TagValueString | + %MacroTagName%
```

タグ値はタグ値変数と条件式から構成される。タグ値変数は、ソース、ターゲット・テンプレートの対応する部分を示す。条件式は"{}"の中に記述し、収集、生成する値を限定するセレクタ、パターン・マッチの役割を果たす。

タグ値変数は制約型タグ値変数、転記型タグ値変数があり、変数のプレフィックスにより区別される。転記型タグ値変数はプレフィックスが "*" で表現され、対応する項目を表現する。制約型タグ値変数は、同様に "?"で表現され、対応する構造を表現するが、それと同時に値の同値性を要求する。つまり、同一識別子を有する制約型タグ値変数には同一の値が入ることを要求する。制約型タグ値変数により、ジョインの実現が可能となる。

Operator はタグ値を Operator の右辺の TagValueString や%MacroTagName%で指定された値と比較するための演算子である。%MacroTagName%は実行時に文字列の値が決定される。

すでに述べたように XTL の具体的な意味付けは操作系により異なる。以下では変換操作を例として説明する。図 1 の変換を実現する XTL の例を図 2 に示す。

図 2 に示した変換ルールでは、*ID, *TYPE, *AMOUNT, *NAME がタグ値変数を表している。ソース・テンプレートとターゲット・テンプレートにおける同名のタグ値変数により、XML 文書間で

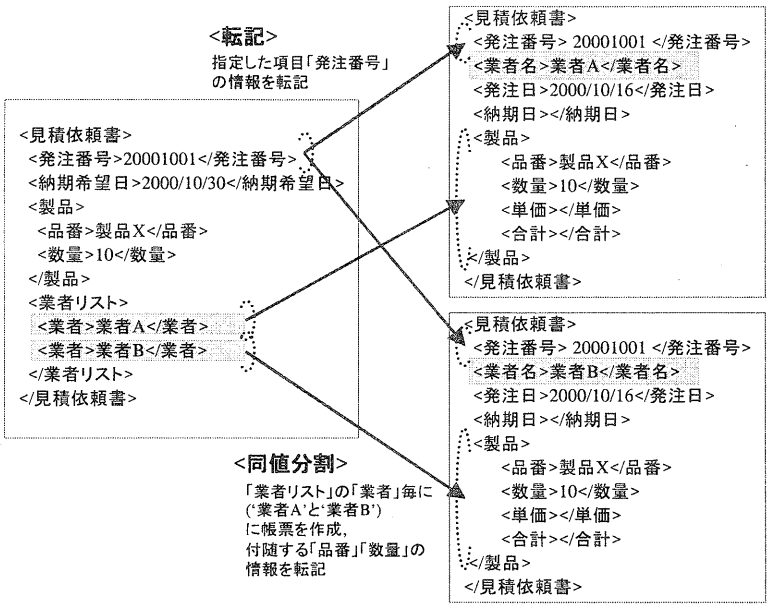

図 1: 帳票変換の例

```
<rule>
 <section>
  <source>
   <見積依頼書>
    <発注番号>*ID</発注番号>
    <製品>
     <品番>*TYPE</品番>
     <数量>*AMOUNT</数量>
    </製品>
    <業者リスト>
     <業者>*NAME{==%NAME%}</業者>
    </業者リスト>
   </見積依頼書>
  </source>
  <target>
   <見積依頼書>
    <発注番号>*ID</発注番号>
    <業者名>*NAME</業者名>
    <発注日>+%DATE%</発注日>
    <納期日>+</納期日>
    <製品>
     <品番>*TYPE</品番>
     <数量>*AMOUNT</数量>
     <単価>+</単価>
     <納期日>+</納期日>
    </製品>
   <見積依頼書>
  </target>
 </section>
</rule>
```

図 2: 図 1 の変換に対する XTL

これらのタグ値が互いに関連付けられることを示している。

ターゲット・テンプレートにおける 記号 "+" により TagValueString や%MacroTagName%で指定された文字列を追加することができる。図 2 の例では空のタグを追加している

また、%NAME%はマクロを表している。マクロは実行時に置換される文字列である。マクロは実行時に、操作系により値の定義が与えられる。マクロにより、一部の差異毎にルールを用意することなく、同一のルールを様々な目的に再利用することが可能となる。

2.3. 変換方式

(1) 再帰性と半構造性の処理

構造変換の操作系では、ソース・テンプレートに従って XML 文書を探索する。このときタグ値変数部分を値として収集する。タグ値変数に対応する値が部分木の場合、他のセクションのソース・テンプレートの適用を試みる。もしセクションに記述されたどのソース・テンプレートも部分木に適用できない場合、部分木をそのままタグ値とする。セクションを用いることで、再帰的な構造を有するXML文書に対応可能となる。子の構造が異なる同一名の要素に対してもセクションを分割することで、異なった構造には、異なったセクションが適用されるため、半構造性にも対応可能である。

セクションを利用した場合の XTL の動作概念図を図 3 に示す。

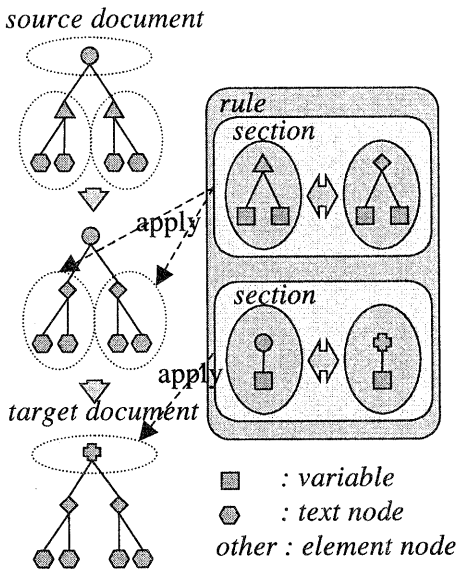

図 3: 再帰的なルール適用の動作概念図

(2) グルーピング、ソーティングによる階層構造の整形処理

XML では情報を階層的に表現するため、ネットワーク型や関係表のような情報構造に対して複数の XML 表現が対応する場合がある。変換結果としての XML を生成するためには異なる XML 表現のうちでどれを選択するかを決める必要がある。このため XTL では option 属性によって XML 情報をどのようにして整形するかを指定できるようにしている。整形にはグルーピングとソーティングがある。option 属性で指定されたタグを親とする部分木の集合をグルーピングするときに用いる要素タグを key 属性のタグ値変数名で指定する。また、option 属性で指定されたタグを親とする部分木の集合を要素タグの値としての文字列の値でソーティングするときに用いる要素タグを orderby 属性のタグ値変数名で指定する。

```
<TagName
    option="Sequence" key="TagValueList">
  TemplateBody
</TagName>
<TagName
    option="Sequence"
```

	業者 A	業者 B
製品 X	9,800	8,800
製品 Y	12,800	14,800

【構造1】

```
<見積>
  <製品>
    <業者>業者A</業者>
    <製品名>製品X</製品名>
    <価格>9,800</価格>
  </製品>
  <製品>
    <業者>業者A</業者>
    <製品名>製品Y</製品名>
    <価格>12,800</価格>
  </製品>
</見積>
<見積>
  <製品>
    <業者>業者B</業者>
    <製品名>製品X</製品名>
    <価格>8,800</価格>
  </製品>
  <製品>
    <業者>業者B</業者>
    <製品名>製品Y</製品名>
    <価格>14,800</価格>
  </製品>
</見積>
```

【構造2】

```
<見積>
  <製品>
    <業者>業者A</業者>
    <製品名>製品X</製品名>
    <価格>9,800</価格>
  </製品>
  <製品>
    <業者>業者B</業者>
    <製品名>製品X</製品名>
    <価格>8,800</価格>
  </製品>
</見積>
<見積>
  <製品>
    <業者>業者A</業者>
    <製品名>製品Y</製品名>
    <価格>12,800</価格>
  </製品>
  <製品>
    <業者>業者B</業者>
    <製品名>製品Y</製品名>
    <価格>14,800</価格>
  </製品>
</見積>
```

図 4: 表構造データの XML による表現例

【構造 1】

```
<target>
  <見積 option="ALL" key="*company">
    <製品 option="ALL">
      <業者>*company</業者>
      <製品名>*product</製品名>
      <価格>*price</価格>
    </製品>
  </見積>
</target>
```

【構造 2】

```
<target>
  <見積 option="ALL" key="*product">
    <製品 option="ALL">
      <業者>*company</業者>
      <製品名>*product</製品名>
      <価格>*price</価格>
    </製品>
  </見積>
</target>
```

図 5: グルーピングの XTL に表現例

```
    orderby="TagValueList">
  TemplateBody
</TagName>

Sequence::= ALL | FIRST | LAST |
    Number [ .. Number ]
TagValueList = TagValue {, TagValue } ...
```

option 属性の値である Sequence では、すべての要素を選択する ALL、最初の要素だけを選択する FIRST、最後の要素だけを選択する LAST、必要な要素の範囲だけを選択することを指定できる。

図 4 に表形式の情報を XML で表した場合の例を，図 5 に図 4 の構造 1 と 2 を表現する場合の変換ルールの一部を示す。

key 属性により、タグのグルーピング方法を指定する。図 5 の構造 1 の「見積」要素の key 属性は、「見積要素は*company の値を単位として、*company の値が異なるものは、別の「見積」要素になるように繰り返す」ということを表現している。つまり、「業者」タグの値毎に「見積」要素を作成している。

同様に構造 2 の場合、「見積要素は*product の値の異なるものを，別の「見積」要素になるように繰り返す」ということを表現している。つまり、製品名タグの値毎に「見積」要素を作成している。

2.4. XTL の変換例

(1) 同値分割

図 1 (a) は品名の値に応じて帳票を分割する例だが、それを実現する XTL を図 6 に示す。図 6 の例では、制約型タグ値変数(?NAME)にマッチする値の同値性に基づき、文書の分割を行っている。つまり、"品名"要素の子要素の値に応じて文書の分割を行っている。

(2) 統合(更新)

統合操作は、制約型タグ値変数を、キーとみなし、ソース、ターゲット文書間でキーが一致する部分を更新する操作である。

図 7 に統合処理の例を、図 8 にそれを実現する XTL の例を示す。

統合操作の場合、ソース・テンプレートが更新元文書の構造を、ターゲット・テンプレートが更新先文書の構造を表現している。図 8 の例では、更新元、更新先文書の?PROJECT 部分の値の同値性に基づき、*MEMBER 等、他のタグ値変数の値を転記している。

```
<rule>
  <section>
    <source>
      <見積依頼書>
      <発注番号>*ID</発注番号>
      <品番>*TYPE</品番>
      <数量>*AMOUNT</数量>
      <業者リスト>
       <業者>?NAME</業者>
      </業者リスト>
     </見積依頼書>
    </source>
    <target>
     <見積依頼書>
      <発注番号>*ID</発注番号>
      <業者名>?NAME</業者名>
      <発注日>+%DATE%</発注日>
      <納期日>+</納期日>
      <製品>
       <品番>*TYPE</品番>
       <数量>*AMOUNT</数量>
       <単価>+</単価>
       <納期日>+</納期日>
      </製品>
      <見積依頼書>
    </target>
  </section>
</rule>
```

図 6: 同値分割を実現する XTL の例

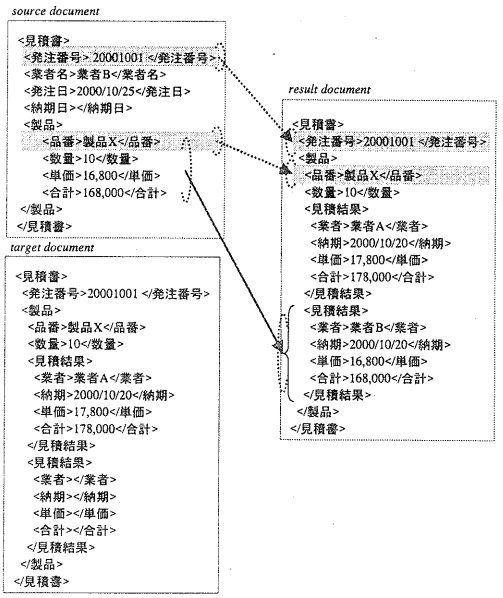

図 7: 統合処理の例

```
<rule>
  <section>
    <source>
      <見積書>
        <発注番号>?ID</発注番号>
        <業者名>*COMPANY</業者名>
        <納期日>*DATE</納期日>
        <製品>
          <品番>?PRODUCT</品番>
          <数量>?AMOUNT</数量>
          <単価>*PRICE</単価>
          <合計>*TOTAL</合計>
        </製品>
      </見積書>
    </source>
    <target>
      <見積書>
        <発注番号>?ID</発注番号>
        <製品>
          <品番>?PRODUCT</品番>
          <数量>?AMOUNT</数量>
          <見積結果>
            <業者>*COMPANY{!=null}</業者>
            <納期>*DATE</納期>
            <単価>*PRICE</単価>
            <合計>*TOTAL</合計>
          </見積結果>
        </製品>
      </見積書>
    </target>
  </section>
</rule>
```

図 8: 図 7 の統合処理を実現する XTL

3. 考察

3.1. XTL の表現範囲

XTL のテンプレートは XML 文書の構造に関する制約条件を記述しており XML 文書の集合に対応している。しかし複数の要素がある場合，各要素ごとの出現順序や、要素の出現回数に対する完全な情報を XTL のテンプレートで指定することは考えていない。この理由は XML 文書の完全な構造情報を与えるためには DTD や XML Schema [2]等の外部スキーマ定義情報を利用できるからである。

XTL テンプレート τ が表す XML 文書の集合 $X(\tau)$ はたとえば次のようにして与えられる。テンプレート τ に対する構文生成規則の集合を $G(\tau) = (V, \Sigma, P, \sigma)$ とする。ここで V は非終端記号の集合、Σ は終端記号の集合でタグとタグ値からなる。P は τ を生成するためのテンプレート構文に基づく構文規則の集合、σ は開始記号を示す非終端記号である。このとき、テンプレート τ が表現可能な XML 文書の集合 $X(\tau)$ は $G(\tau)$ を拡張した $GX(\tau)$ で与えられる。ここで、$x \in V$ のとき、x と x' を VX の要素とする。P の構文規則の右辺はひとつまたは複数のタグを持つかのどちらかである。したがって $x \rightarrow$ <a> y $\in P$ のとき、$x \rightarrow x'\{, x'\}\cdots$ と $x' \rightarrow$ <a> y を PX の要素とする。また $x \rightarrow < a_1 >$y1$</ a_1 > \cdots < a_n >$ yn$</ a_n > \in P$ のとき、$x \rightarrow x'\{, x'\}\cdots$ と $x' \rightarrow < a_1 >$ y1 $</ a_1 > | \cdots | < a_n >$ yn $</ a_n >$を PX の要素とする。このようにして構成された $GX(\tau) = (VX, \Sigma, PX, \sigma)$から生成される集合が $x(\tau)$ である。

$X(\tau) = \{w \in \Sigma \mid w$は$PX$により$\sigma$から生成される$\}$

XTL テンプレートとそれにマッチする文書の例を図 9 に示す。

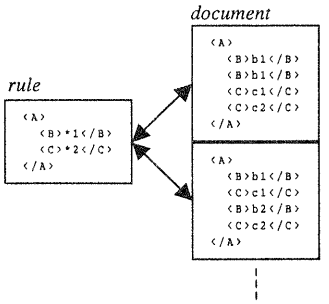

図 9: 複数の文書構造にマッチする XTL の例

ここで示した $X(\tau)$ の構成は一つの例であるが広い範囲の操作系に適用できる。もちろん操作系によってはより狭い範囲の XML 文書に限定するように $X(\tau)$ を構成することもできる。たとえば $PX = P$ とすれば XTL テンプレートと 1 対 1 で対応する。しかしこの場合 XML 文書の小さな差異を許容できなくなるので XML 文書ごとに類似した変換ルールを作成する必要がある。

この $X(\tau)$ の構成では異なるテンプレート τ_1 と τ_2 に対して $X(\tau_1) = X(\tau_2)$ となるものがある。この理由は親のタグが子として複数のタグを含む場合、子のタグの出現順序を区別しないように PX を構成しているからである。これはできるだけ単純なテンプレートで広い範囲の帳票を扱うことができるようにしていることとのトレードオフである。

3.2. XSLT との比較

XML に関する操作を実現するための方法として XSLT が提案されている[3]。XSLT では次の例のように変換のための手続きを記述する.

例 1: ルート要素"/"があれば要素 "X"を作る

例 2: 要素 A の下に要素 B が存在して要素 C

の子が c1 ならば要素"Z"、"W"、"U"を作る

したがって XSLT は入力としての XML 文書を段階的に構文解析しながら出力としての XML 文書への変換を記述するため、変換の定義が手順的で生成される XML 文書の構造が XSLT の記述から容易に判読できないだけでなく、業務担当者では XSLT の記述は困難であり、アプリケーション開発者による変換処理の開発が必要になる.

3.3. 適用上の限界

(1) 操作系の追加に対する XTL の適用性の検討

XML データベースへの適用を考えた場合、つまり、XTL をデータベースへの問い合わせ言語として見た場合、XML に対する部分木の追加、削除を行う機能が必要である。新たな操作形の追加に対する XTL の適用性、記述性の検討が必要である。

代表的な XML 問い合わせ言語 XML-QL[4]、XML Query[5]等との比較が必要である。

(2) タグ値変換

変換の観点から見た場合、現状の XTL は構造変換のみを対象としており、値変換は考慮していない。値変換を行うためには、タグ値変数に対する演算や、関数を定義、利用する機能を実装すれば良い。現在、この機能の実装を進めている。

(3) 意味的な一貫性保証

DB-STREAM[6]ではデータ流通システムを効率的に実現するために流通先データベースとのデータの整合性を保証するために流通ユニットと呼ぶ流通データの単位とその変換機能により異種データベース間で一貫性を保証する方式を明らかにしている。今後企業情報システムへの XML の適用が進展すると、異なる XML データベース間でこのような一貫性を保証する必要がある。流通単位が XML で記述されていれば、XTL により流通元から流通先の XML への変換を容易に定義することができると思われる。XTL データベースごとに一貫性を保証するための処理を作成しておき、変換結果を XML データベースに登録した時点でこの一貫性保証手順を実行するような制御方式を実現する必要がある。

(4) ルールの適用順序

セクションにより複数の変換ルールを扱うことができるので XTL では再帰性や半構造性を持つ XML 文書を変換できる。しかし、セクションの適用順序については制約ルールで制御する構文を持っていない。したがってセクションで記述された変換ルールの適用順序が操作系によって異なれば、同じ XML 文書と制約ルールの組に対しても変換結果が異なることがある。この場合、複数の制約ルール間でタグ値変数の一致性の評価順序を制御する必要がある。現状では制約ルールの記述順序に基づく適用制御を行っているため、記述順序を変更することにより制約ルールの適用順序を制御できる。しかし、より柔軟な制約ルールの適用を制御するための優先順位指定などの XTL 構文を導入する必要がある。代表的な制約ルールの適用法としては、内側優先、外側優先などが考えられる。このようなセクションの適用順序を操作系に対して指定する機能は理論的には操作系の変換能力を決定するので重要であり、今後の課題として挙げられる。

4. むすび

本論文では、B2B 分野における情報流通で期待されている XML ベースの情報流通における変換言語に対する機能要件を整理し、制約ルール指向の XML 変換言語である XTL について考察した.

今後の課題としては XTL の表現能力を明らかにすること, 変換ルールの自動生成などがある.

参考文献

[1] 横関大子郎,村山隆彦,山本修一郎,制約規則に基づく XML 情報変換方式の提案, 信学技報, Vol.100, No.206, pp.83-90, 2000.

[2] XML Schema, http://www.w3.org/TR/xmlschema-0/

[3] XSLT, http://www.w3.org/TR/xslt

[4] XML-QL: A Query Language for XML, http://www.w3.org/TR/NOTE-xml-ql/

[5] XML-Query, http://www.w3.org/XML/Query.html

[6]池田,伊土,石垣,村田: データ流通プラットフォームシステム: DB-STREAM, 情報処理学会論文誌, Vol. 38, No. 12, pp.2252-2565, 1997.