データベースシステム 123－10
（2001．1．23）

DTD作成ツールDSD (Document Structure Detector) の実装

田中秀和† 梅村恭司†

近年，各種文書，データの保存，配布において情報の電子化が行われている．この電子化を行う際にXMLを利用することが注目されている．XMLは汎用的なデータ記述言語であり，特徴としては，タグを自由に定義でき，一般的なデータの表現ができることである．しかし，これらのデータをシステムで処理するには，全てのXMLドキュメントを参照しなければならない．そこで，この構造を定義したDTDが必要とされるが，構造を定義するDTDの作成は，複雑であり手間がかかる．このため，DTDをもっと効率よく作成する方法が望まれている．そこで我々は，DTDのないXMLドキュメントから，それと整合性のあるDTDを生成するツールの作成を行った．本稿では，このツールについて作成した背景と実現方法などについて述べる．

Implementation of DTD preparation tool DSD(Document Structure Detector)

HIDEKAZU TANAKA† and KYOJI UMEMURA†

Recently, in the preservation and distribution of various document and data, information is computerized. In this computerization, it is noticed that XML is getting attention. As the features of XML, a tag can be freely defined, and it is why the expression of the general data is possible. However, it is not easy to make DTD that defines the structure of XML. Therefore, the method for efficiently making DTD is desired. We have developed a tool, which form a consistent DTD from the XML documents without DTD. In this paper, this tool is described with its motivation and implementation.

1. はじめに

近年，インターネットの普及に伴い各種文書，データが電子化され，その保存，配布においてXML (Extensible Markup Language) [1] の利用が注目されている．XMLはインターネット汎用的なデータ記述言語であり，特徴としては，タグを自由に定義することができることである．タグの固定的なHTML (Hypertext Markup Language) とは異なり，XMLは自由に定義することができるのでWebページだけではなく，一般的なデータも表現することが可能である．しかし，XMLはタグを自由に定義できることで，その構造を定義するDTD (Document Type Definition) を作成することは複雑で手間がかかってしまう．また，データをマージすることによって，XMLの構造が変化したときにDTDを作成し直さなければならなくなる．

このように，XMLの構造を定義するDTDを作成するのには，多大な時間と労力が必要である．このため，DTDを効率よく作成することが望まれている．そこで我々は，XMLドキュメントを作成する際に，構造を先にDTDで定義するのではなく，DTDのないXMLドキュメントが作成されていて，それから，それと整合性のあるDTDを作成するツールDSD (Document Structure Detector) の作成を行った．本稿では，このツールの実現方法，実現上の制限などについて述べる．

2. 実現方法

2.1 XML

XMLドキュメントは，XML宣言，DTD，およびXMLインスタンス（XMLドキュメントのXML宣言，DTDを除く文書データ）[2] から成り立っている．(XML宣言とDTDは必須ではない.)

我々の作成したツールは，XMLインスタンスから，

† 豊橋技術科学大学
Toyohashi University of Technology

```
<DOC>
  <TITLE>DTD作成ツールDSDの実装</TITLE>
  <AUTHOR>田中秀和</AUTHOR>
  <AUTHOR>梅村恭司</AUTHOR>
</DOC>
```

図1　XMLインスタンスの例
Fig. 1　Example of XML instance

出現回数を表現

必ず1回出現	<!ELEMENT X (A) >
0回もしくは1回出現	<!ELEMENT X (A?) >
0回以上出現	<!ELEMENT X (A*) >
1回以上出現	<!ELEMENT X (A+) >

出現順序を表現

Xの要素としてA,Bが順番に出現
<!ELEMENT X (A,B) >
Xの要素としてA,Bのいずれかが出現
<!ELEMENT X (A|B) >

その他の構造表現

Xの要素としてテキストが出現
<!ELEMENT X (#PCDATA) >
Xの要素が何も出現しない（空要素）
<!ELEMENT X EMPTY >
Xの要素として任意の要素が出現（任意要素）
<!ELEMENT X ANY >

図2　DTDでの基本的な構造の定義
Fig. 2　Definition of basic structures in DTD

構造を解析してDTDを生成するツールである．

XMLインスタンスは，実際の文書内容やデータ内容が記述されている部分であり，階層構造を持った要素の集合である．この要素を識別するためのものがタグである．XMLインスタンスの例を**図1**に示す．

DTDは，XMLの文書の型を定義する．DTDには，要素型宣言（element type declaration）と呼ばれる宣言によって，文書要素の名前（タグの名前），要素の出現順序，要素の出現回数について定義する．**図2**において，基本的なXMLの構造とDTDの定義を示す．

図2に示すように，要素の出現回数，出現順序を表現することができる．また，これらを組み合わせて複雑な構造も表現することが可能である．例として，図1のXMLインスタンスのDTDを**図3**に示す．

図3のDTDでは，DOCの要素としてTITLEとAUTHORが順番に出現し，さらに，AUTHORは1回以上出現することが定義されている．また，TITLEおよびAUTHORの要素がテキスト（#PCDATA）であると定義されている．

2.2　DTDの作成法

我々の作成したツールは，入力されたXMLインスタンスより，タグの抽出を行う（このとき，テキストデータは，#PCDATAとして扱われる.）．そして，そ

```
<DOCTYPE DOC [
<!ELEMENT DOC (TITLE,AUTHOR+) >
<!ELEMENT TITLE (#PCDATA) >
<!ELEMENT AUTHOR (#PCDATA) >
] >
```

図3　図1のXMLインスタンスのDTD
Fig. 3　DTD of XML instance in Fig. 1

XML instance

```
<X>
  <A>.....</A>
  <B>.....</B>
  <B>.....</B>
</X>
```

DTD

<!ELEMENT X (A,B+) >

Graphic representation

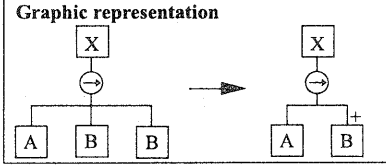

図4　DTDの作成（同一要素の連続出現）
Fig. 4　Preparation of DTD (Consecutive occurrence of same elements)

の抽出したタグのデータから階層構造のチェック（下位の要素が上位の要素の開始タグと終了タグの中にあるか）を行い，階層構造が正しくできていれば，DTDの生成を行う．もし階層構造にエラーがあればそのエラー箇所をユーザに返す．

DTDの生成では，図3に示したような構造の定義を基本に，以下に示すようなそれらを組み合わせた構造の解析を行って，DTDを生成する．

図4に示すXMLインスタンスのような，Xの要素としてA，B，Bと順序で要素が出現する構造では，同じ要素（B）が連続して出現していることを検出し，その要素をB+というように圧縮する．そして，A，Bというような順序で出現しているとして，図4に示すようなDTDを生成する．

図5に示すようなXMLインスタンスでは，Xの要素として，A，B，Aという順序で要素が出現する構造では，同じ要素（A）が出現していることを検出し，さらにそれは連続ではないので，その要素にはさまれている要素(B)を含めて，(A|B)+のように圧縮する．その結果，図5に示されるようなDTDを生成する．

また，**図6**，**図7**に示すような，同じ上位要素に含まれる下位要素の構造が出現する場合には，両方の構造に適合するDTDを生成する．図6では，一つ目のXの要素としてA，Bが出現するが，二つ目では，B

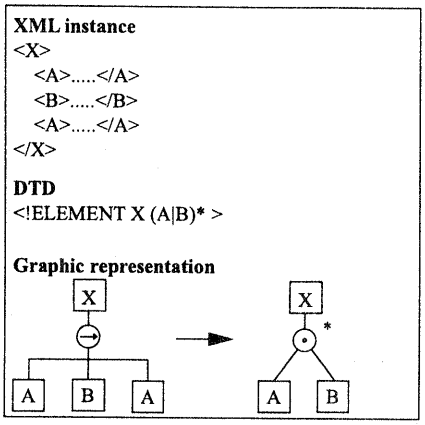

図5　DTDの作成（同一要素の再出現）
Fig. 5　Preparation of DTD
(Re-occurrence of same elements)

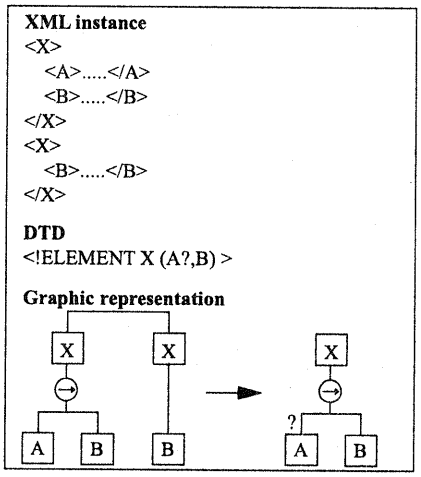

図6　DTDの作成（要素の非出現）
Fig. 6　Preparation of DTD
(Absense of element)

しか出現しないため，AはA?として変換され，DTDは図6に示すようなDTDを生成する．

図7に示すXMLインスタンスでは，一つ目のXの要素はA，Bと順に出現していて，二つ目ではB，Aの順で出現している．この場合，二つ目の要素ではBの前にAが出現しないため，AはA?として，さらに二つ目でBのあとにAが現れるため，Bのあとに A?が出現し，結果的にA?，B，A?という出現順序となり，Aが2度出現するので図5と同じようになる．

以上のように，DSDは，基本のDTDの定義を組み合わせたDTDの生成も行うが，さらに，これらよりも複雑な構造も，同様にして処理することにより，

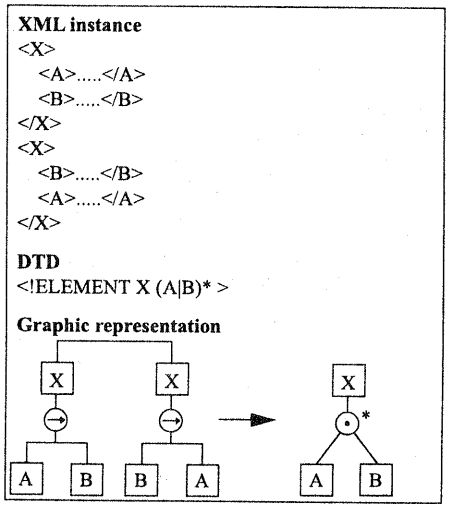

図7　DTDの作成（要素の出現順序の変化）
Fig. 7　Preparation of DTD
(Changed order of elements)

そのXMLインスタンス構造に整合するDTDの生成も行う．

2.3 実装環境

このツールの実装環境として，Java2の動作するMicrosoft Windows 2000，およびVine Linux 2.0 CR（Kernel 2.2.14-vl6）上で実装を行った．

3. サンプル

我々のツールによって生成されるDTD，および入力したXMLインスタンスを以下に示す．

まず，NACSISテストコレクション（NTCIR-1）[3]のデータでのDTDの生成結果について示す．NTCIR-1のデータは図8に示すような構造の文書を約33万件含んでいる．今回はこの文書のうち上位10000件（約9MB）について本ツールを用いてDTDの生成を行った（NTCIR-1のデータには要素RECの集合となっているが，その上位要素がなかったため，NACSISという上位要素をデータに追加した）．その結果を図9に示す．

生成されたDTDを見ると，要素RECが繰り返し出現し，RECの要素としてACCN，TITLなどの要素が順に現れる．この中でAUPKおよびKYWDは0回または1回の出現となっている．これは，データの中にそれらのデータがないものが含まれていたためである．また，同様にACCN，TITLの要素として，ABST以外はテキストデータ（#PCDATA）が出現する．ABSTはABST.P（概要の段落）の要素が1回以上出現し，ABST.Pの要素はテキストデータ

```
<REC>
<ACCN>gakkai-0000000001</ACCN>
<TITL TYPE="kanji">電気回路演習用CAIとその改良</TITL>
<AUPK TYPE="kanji">小野 敏夫 / 大川原 宣夫 / 栗原 秀行 / 小林 一郎</AUPK>
<CONF TYPE="kanji">昭和63年電気学会全国大会一般講演</CONF>
<CNFD>1988. 03. 29 - 1988. 03. 31</CNFD>
<ABST TYPE="kanji"><ABST.P>大学等での基礎的な電気回路演習を支援するCAIソフトウェアとその改良について述べている。本CAIはコンピュータが出題される回路を学習者各人のレベルに応じて自動的に作成すること、解答を数式で入力することが大きな特長である。また、誤った解答に対しては、原因の検討を容易にするメッセージが表示されるなど、効果的な個別学習が限られた設備・要員で実施可能であるよう配慮した。昨年度の学生による本CAIの使用結果のアンケート、および発表の場における質疑等を参考に、操作を容易とし、効果を上げるための改良を行った。</ABST.P></ABST>
<KYWD TYPE="kanji">電気回路 // 演習</KYWD>
<SOCN TYPE="kanji">電気学会</SOCN>
</REC>
```

図 8　NTCIR-1 データの例
Fig. 8　Example of NTCIR-1 data

```
<!DOCTYPE NACSIS [
<!ELEMENT NACSIS (REC+) >
<!ELEMENT REC (ACCN,TITL,AUPK?,CONF,CNFD,
ABST,KYWD?,SOCN) >
<!ELEMENT ACCN (#PCDATA) >
<!ELEMENT TITL (#PCDATA) >
<!ELEMENT AUPK (#PCDATA) >
<!ELEMENT CONF (#PCDATA) >
<!ELEMENT CNFD (#PCDATA) >
<!ELEMENT ABST (ABST.P+) >
<!ELEMENT ABST.P (#PCDATA) >
<!ELEMENT KYWD (#PCDATA) >
<!ELEMENT SOCN (#PCDATA) >
]>
```

図 9　NTCIR-1 のデータから生成した DTD
Fig. 9　Generated DTD from NTCIR-1 data

(#PCDATA) が出現する構造を定義する DTD が生成される．この DTD は NTCIR-1 のデータから見ても正しい DTD であると考えられる．

次に，図 10 に示すような DTD で構造を定義された図 11 に示す少し構造の複雑な XML 文書からの DTD の生成結果について述べる．この DTD を見ると，定義されている XML 文書の構造は，VEHICLE の要素として INVENTORY_NUMBER，MAKE，MODEL などが順に出現し，その中で OPTIONS，OWNER はさらに下位の要素を含んでいて，それらの要素はテキストデータを要素としている．

ここで，本ツールで生成された DTD を図 12 に示す．図 10 の定義した DTD と比較すると，VEHICLE の要素は，すべて同じように検出できている．しかし，OPTIONS の要素は図 10 ではすべての要素が 0 回または 1 回の出現と定義されているのに対して，本ツールで生成された DTD は Power_Locks，Power_Windows は 0 回または 1 回出現，Stereo は必ず 1 回出現，それ以外の要素は，0 回以上任意の順序で出現と定義されている．これは，Stereo ついては，入力された XML 文書の全てに出現したためであり，その他の要素は入力した XML 文書（図 11）を見てもわ

```
<!DOCTYPE VEHICLE [
<!ELEMENT VEHICLE (INVENTORY_NUMBER, MAKE,
                   MODEL,            YEAR,
                   PICTURE,          STYLE,
                   DOORS,            PRICE,
                   MILEAGE,          OPTIONS,
                   OWNER) >
<!ELEMENT INVENTORY_NUMBER (#PCDATA) >
<!ELEMENT MAKE (#PCDATA) >
<!ELEMENT MODEL (#PCDATA) >
<!ELEMENT YEAR (#PCDATA) >
<!ELEMENT PICTURE (#PCDATA) >
<!ELEMENT STYLE (#PCDATA) >
<!ELEMENT DOORS (#PCDATA) >
<!ELEMENT PRICE (#PCDATA) >
<!ELEMENT MILEAGE (#PCDATA) >
<!ELEMENT OPTIONS (Power_Locks?,  Power_Window?,
                   Stereo?,       Air_Conditioning?,
                   Automatic?,    Four_Wheel_Drive?,
                   Interiors?,    AlloyWheel?,
                   Note?) >
<!ELEMENT OWNER (NAME,EMAIL,PHONE) >
<!ELEMENT Power_Locks (#PCDATA) >
<!ELEMENT Power_Window (#PCDATA) >
<!ELEMENT Stereo (#PCDATA) >
<!ELEMENT Air_Conditioning (#PCDATA) >
<!ELEMENT Automatic (#PCDATA) >
<!ELEMENT Four_Wheel_Drive (#PCDATA) >
<!ELEMENT Interiors (#PCDATA) >
<!ELEMENT AlloyWheel (#PCDATA) >
<!ELEMENT NAME (#PCDATA) >
<!ELEMENT EMAIL (#PCDATA) >
<!ELEMENT PHONE (#PCDATA) >
]>
```

図 10　サンプル XML データの DTD（Vehicle.dtd）
Fig. 10　DTD of sample XML data (Vehicle.dtd)

かるように順序が誤っている部分があるためである．

以上のように，本ツールは大量なデータも処理して，DTD を生成することができ，また，構造が複雑なデータの場合でも，そのデータを許容する DTD を生成することができる．また，生成された DTD から入力したデータの誤りも検出することができる．

4. 実現上の制限

我々の作成したツールの現在までの実現上の制限と

Vehicle_1.xml

```
<VEHICLE>
 <INVENTORY_NUMBER>1</INVENTORY_NUMBER>
 <MAKE>Dodge</MAKE>
 <MODEL>Durango</MODEL>
 <YEAR>1998</YEAR>
 <PICTURE>DodgeDurango.jpg</PICTURE>
 <STYLE>Sport Utility</STYLE>
 <DOORS>4</DOORS>
 <PRICE>18000</PRICE>
 <MILEAGE>32000</MILEAGE>
 <OPTIONS>
  <Power_Locks>Yes</Power_Locks>
  <Power_Window>Yes</Power_Window>
  <Stereo>Radio/Cassette/CD</Stereo>
  <Air_Conditioning>Yes</Air_Conditioning>
  <Automatic>Yes</Automatic>
  <Four_Wheel_Drive>Full/Partial</Four_Wheel_Drive>
  <Note>Very clean</Note>
 </OPTIONS>
 <OWNER>
  <NAME>Douglas Briggs </NAME>
  <EMAIL>DBriggs@aDomain.Com</EMAIL>
  <PHONE>781 781 7811</PHONE>
 </OWNER>
</VEHICLE>
```

Vehicle_2.xml

```
<VEHICLE>
 <INVENTORY_NUMBER>10</INVENTORY_NUMBER>
 <MAKE>Ford</MAKE>
 <MODEL>Explorer</MODEL>
 <YEAR>1996</YEAR>
 <PICTURE>FordExplorer.jpg</PICTURE>
 <STYLE>Sport Utility</STYLE>
 <DOORS>4</DOORS>
 <PRICE>18000</PRICE>
 <MILEAGE>105000</MILEAGE>
 <OPTIONS>
  <Power_Locks>Yes</Power_Locks>
  <Stereo>Radio/Cassette</Stereo>
  <Automatic>Yes</Automatic>
  <Interiors>Fabric</Interiors>
  <Air_Conditioning>Yes</Air_Conditioning>
  <Note>Pre-owned, very clean</Note>
 </OPTIONS>
 <OWNER>
  <NAME>Shawn Denoncour</NAME>
  <EMAIL>SDenoncour@aDomain.Com</EMAIL>
  <PHONE>781 781 785</PHONE>
 </OWNER>
</VEHICLE>
```

Vehicle_3.xml

```
<VEHICLE>
 <INVENTORY_NUMBER>14</INVENTORY_NUMBER>
 <MAKE>GMC</MAKE>
 <MODEL>Suburban</MODEL>
 <YEAR>1997</YEAR>
 <PICTURE>GMCSuburban.jpg</PICTURE>
 <STYLE>Sport Utility</STYLE>
 <DOORS>4</DOORS>
 <PRICE>28000</PRICE>
 <MILEAGE>45000</MILEAGE>
 <OPTIONS>
          <Power_Locks>Yes</Power_Locks>
  <Stereo>Radio/Cassette/CD</Stereo>
  <Automatic>Yes</Automatic>
  <Interiors>Fabric</Interiors>
  <Air_Conditioning>Yes</Air_Conditioning>
 </OPTIONS>
 <OWNER>
  <NAME>Dealer</NAME>
 </OWNER>
</VEHICLE>
```

Vehicle_4.xml

```
<VEHICLE>
 <INVENTORY_NUMBER>19</INVENTORY_NUMBER>
 <MAKE>Jeep</MAKE>
 <MODEL>Gran Cherokee</MODEL>
 <YEAR>1997</YEAR>
 <PICTURE>JeepGranCherokee.jpg</PICTURE>
 <STYLE>Sport Utility</STYLE>
 <DOORS>4</DOORS>
 <PRICE>24000</PRICE>
 <MILEAGE>97000</MILEAGE>
 <OPTIONS>
  <Power_Locks>Yes</Power_Locks>
  <Power_Window>Yes</Power_Window>
  <Stereo>Radio/Cassette/CD</Stereo>
  <Automatic>Yes</Automatic>
  <Interiors>Leather</Interiors>
  <Air_Conditioning>Yes</Air_Conditioning>
  <Note>Most popular sport utility</Note>
 </OPTIONS>
 <OWNER>
  <NAME>Dealer</NAME>
 </OWNER>
</VEHICLE>
```

図 11　サンプル XML データ（Vehicle_?.xml）
Fig. 11　sample XML data(Vehicle_?.xml))

しては，XML の構造を解析して，DTD の要素型宣言を定義することのみが実現されている．他の DTD の宣言（属性リスト宣言，エンティティ宣言，記法宣言）に対するツールの開発を行っていないため，これらの宣言を定義することはできない．

また，DTD の本来の目的である構造のチェック機能はツールの性質上失われてしまう．

また，このツールによって作成された DTD の定義が，いささか甘くなることが挙げられる．複数の XML インスタンスから，それらと整合性のある DTD を生成し，入力された XML インスタンスを全て許容するため，定義が甘くなる部分が出てきてしまう．つまり，人手で作成した DTD に比べて単純な DTD しか生成しない．

以上が，このツールを実現するにあたって生じた制限である．

5. DSD の利用法

「DTD があって，それにしたがって処理対象のデータが整理されている」という状況は常には成り立たな

```
<!DOCTYPE VEHICLE [
<!ELEMENT VEHICLE (INVENTORY_NUMBER,MAKE,
MODEL,YEAR,PICTURE,STYLE,DOORS,PRICE,MILEAGE,
OPTIONS,OWNER) >
<!ELEMENT INVENTORY_NUMBER (#PCDATA) >
<!ELEMENT MAKE (#PCDATA) >
<!ELEMENT MODEL (#PCDATA) >
<!ELEMENT YEAR (#PCDATA) >
<!ELEMENT PICTURE (#PCDATA) >
<!ELEMENT STYLE (#PCDATA) >
<!ELEMENT DOORS (#PCDATA) >
<!ELEMENT PRICE (#PCDATA) >
<!ELEMENT MILEAGE (#PCDATA) >
<!ELEMENT OPTIONS (Power_Locks,Power_Window?,
Stereo,(Air_Conditioning|Automatic|Four_Wheel_Drive|Note
|Interiors|AlloyWheel)*) >
<!ELEMENT Power_Locks (#PCDATA) >
<!ELEMENT Power_Window (#PCDATA) >
<!ELEMENT Stereo (#PCDATA) >
<!ELEMENT Air_Conditioning (#PCDATA) >
<!ELEMENT Automatic (#PCDATA) >
<!ELEMENT Four_Wheel_Drive (#PCDATA) >
<!ELEMENT Note (#PCDATA) >
<!ELEMENT OWNER (NAME,EMAIL?,PHONE?) >
<!ELEMENT NAME (#PCDATA) >
<!ELEMENT EMAIL (#PCDATA) >
<!ELEMENT PHONE (#PCDATA) >
<!ELEMENT Interiors (#PCDATA) >
<!ELEMENT AlloyWheel (#PCDATA) >
]>
```

図12　サンプル XML データより生成した DTD
Fig. 12　Generated DTD from sample XML data

いところに DSD の利用価値がある．例えば以下のような状況である．

(1)　いろいろなところで作られて，類似しているが表現が異なる XML インスタンスのみからなる情報があるとき，タグを変換して，ひとつの大きな XML インスタンスを作成したい．

(2)　時間の経過から XML のタグが次第に変化してきている状況で，新しいデータと古いデータを同時に処理したい．

例えば，いろいろなところで作成される新聞記事の情報から，それぞれをまとめた新聞記事の XML を作りたいケースがあった．我々の経験では，紙に出力する元の情報から変形して XML インスタンスを生成することは自然であり，また，データの汎用性のために DTD を厳密に定義していない状態であることは理解できる．しかし，それを処理するための DTD を定義するためには，データの全体を検査しなければならないが，全ての構造が共通というわけでない．サンプル調査で DTD を作成してみると，本番の処理をしたときに次々と例外的なタグの使い方に遭遇することを経験した．このような場合に DSD により DTD を生成することによって，処理が可能になる．

6. 他の研究との比較

我々の作成した DTD を生成するツールと同様のアプローチで作成された DTD 生成ツール [4] が報告されている．本稿では，実現方法，特質，制限について詳しく述べているが，この報告ではそれらがされていない．

また，この報告のツールでは，類似した構造の XML インスタンスが存在するときには，ユーザに XML の構造を変化させて DTD を作成するかの選択を要求する．しかし，これでは大量のデータを扱う際には，ユーザに選択を要求する回数が増え，DTD を生成するのに多くの判断を要するため，逆に手間がかかる．それに対して我々のツールは，入力した XML インスタンスの構造を解析して，ユーザの判断を要求せずに DTD を生成している．DTD を生成する際のユーザの手間が減る．しかし，類似している XML インスタンスからは，それらを許容する DTD が生成され，DTD が大きなものとなってしまうことも考えられる．

7. ま と め

本稿では，我々が作成した DTD のない XML ドキュメントからそれと整合性のある DTD を生成するツールについて，その作成した背景，実現方法，実現上の制限について述べてきた．

本ツールは，ある程度大量なデータも処理することができる．そして，複雑な構造も解析して効率よく DTD を生成することのできるツールである．DTD を作成する上で，XML インスタンスから容易にでき，生成された DTD から XML の構造のミスも見つけることができ，有用なツールであると考えられる．

参 考 文 献

1) Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler: "Extensible Markup Language (XML) 1.0 (Second Edition)" , W3C Recommendation 6 October 2000, W3C, http://www.w3c.org/TR/REC-xml (2000).

2) XML/SGML サロン: 標準 XML 完全解説, 技術評論社 (1998).

3) Noriko Kando, Kazuko Kuriyama, Toshihiko Nozue, Koji Eguchi, Hiroyuki Kato and Souichiro Hidaka.: Overview of IR Tasks at the First NTCIR Workshop, NTCIR Workshop, vol. 1, pp. 11-44 (1999).

4) 鬼沢友和, 安井浩之, 松山実: "XML インスタンスからの統一的 DTD の作成", 情報処理学会第 61 回全国大会, 4J-02 (2000).