XMLを対象とした制約の利用による情報視覚化方式

細部博史　本位田真一

国立情報学研究所

近年，次世代の文書フォーマット規格として，XMLが注目を集めている．今後，XMLは，インターネットやデータベース上の文書のための中核的技術へと成長することが期待され，将来的には，様々なデータがXML文書として蓄積されるようになることが予期されている．このように多様なデータを記述する標準フォーマットとしてのXMLの地位が高まるにつれて，XML文書をより直観的な形で視覚化する手法が必要となる．しかし，従来の手法では，グラフや木などの構造を持った図形的な視覚表現が困難である．このような状況に対応するために，本研究では，XMLを対象とした，新しい情報視覚化方式を提案する．この方式の特徴は，XSLTスタイルシートによる視覚化の枠組に対して，制約を利用した視覚化手法を統合した点にある．その実現のために，本研究では，新しい高水準な視覚化言語XML-VLを構築する．本研究では，この方式に基づき，情報視覚化システムをJava言語で実装した．

A Constraint-Based Approach to Information Visualization for XML

Hiroshi Hosobe and Shinichi Honiden

National Institute of Informatics

XML is attracting much attention as a next-generation document format standard. It is expected to be a key technology for Internet and database documents, and then various data will be stored as XML documents. Along with the spread of XML as a standard data format, it becomes increasingly necessary to visualize XML documents in intuitive form. However, existing visualization techniques can hardly treat structural visual representations such as graphs and trees. To cope with this situation, we propose a novel information visualization method for XML. The characteristic of this method is the integration of a constraint-based technique with the visualization framework using XSLT stylesheets. To realize this, we construct a new high-level visualization language called XML-VL. Using this method, we have developed an information visualization system in Java.

1. はじめに

近年，次世代の文書フォーマット規格として，**Extensible Markup Language (XML)**[7]が注目を集めている．XMLは，ネットワーク上での文書やデータの交換を目的として，World Wide Web Consortium (W3C)により勧告された規格である．XMLは，言語を定義する枠組であり，用途に応じて様々な形式の言語を定義することができる．また，その特徴として，Document Object Model (DOM)[24]など，XMLによって記述されたデータへアクセスするためのインターフェースが整備されている点が挙げられる．

今後，XMLは，インターネットやデータベース上の文書のための中核的技術へと成長することが期待されている．そして，将来的には，様々なデータがXML文書として蓄積されるようになることが予期されている[14),15)]．

多様なデータを記述する標準フォーマットとしてのXMLの地位が高まるにつれて，必然的に，XML文書をユーザーが閲覧する手段の重要性が増してくる．基本的に，XML文書はテキストデータとして表現されるため，そのままテキストエディター等で閲覧することが可能である．しかし，この方法では，データの構造を直観的に把握することが困難である．従って，XML文書をより直観的な形で**視覚化**する手法が必要となる．

最も単純なXML文書の視覚化方法は，ファイルシステムのブラウザーのように，XML文書を木構造として表示するものである．この方法は，単純とはいえ，極めて汎用性が高いため，XML文書専用のエディターなどで多く利用されている．

より高度なXML文書の視覚化手法としては，Extensible Stylesheet Language (XSL)[1] から派生した，**XSL Transformations (XSLT)**[10] を用いる枠組がある．この枠組では，対象となるXML文書を，他の視覚化用の言語に変換し，対応する処理系で表示することで，視覚化が実現される[23]．XSLTは，XML文書を別の形式に変換するためのスクリプト言語というべきものであり，このような変換を記述するスクリプトは，**スタイルシート**と呼ばれる．XSLT自体もXMLの1種であり，スタイルシートはXMLとして記述される．

XSLTを用いる枠組では，XML文書の形式に応じて処理系を作成する必要がないため，様々な形式が存在し得るというXMLの特異性に対して有効に対処できる．すなわち，対象となるXML文書の形式に応じてスタイルシートを用意するだけで，視覚化が容易に実現できるという利点がある．この枠組では，データの文書が修正された場合に，スタイルシートを再適用するだけで，新しいデータに応じた視覚化を得ることが可能である．また，同一のデータに対して，異なるスタイルシートを個別に適用することで，異なる視覚化を得ることも可能である．

現状では，XSLTによる視覚化において，視覚化言語としてHTMLを用いることが多い．この場合は，WWWブラウザー上に，章構造を伴った文章や，表や箇条書きとして，XML文書を視覚化することが可能である．また，紙への印刷など，より高精細な出力を必要とする場合には，XMLによるTEXとでもいうべき，XSL Formatting Objects (XSL FO)[1] を利用する方法も可能である．さらに，より複雑な視覚化のためには，Scalable Vector Graphics (SVG)[11] を利用できる．SVGは，いわばXMLによるDisplay PostScriptであり，あらゆる種類の表示が可能で，アニメーションも実現できる．

しかし，これらの視覚化言語だけで，実用上十分であるとはいえない．一般に，グラフや木などの構造を持った図形的な視覚表現は有用であるが，このような用途に対して，HTMLやXSL FOは不向きであり，SVGは，その低水準さのために，XSLTスタイルシートの記述が困難になるからである．

現実に，計算機が扱い得るデータで，図による視覚表現が有用である例は多い．例えば，関係データベースの設計には，実体と属性の関係や，実体間の関連の表現のために，実体関連図が用いられる．オブジェクト指向開発には，クラス階層やオブジェクト構成の表現に，UML等による図が使用される．プロジェクトのスケジュール立案においては，個々の作業の順序関係を表す，PERT図が利用される．そして，今後，このような情報が，XMLとして蓄積されていく可能性は高い．

これらの視覚表現に対する主要な課題は，データが備えているグラフや木などの構造を図形的に表現することである．そのためには，具体的な図形の配置を計算することが必要となる．例えばSVGを用いる場合，XSLTスタイルシートの内部や，出力となるSVGデータに埋め込んだスクリプトで，図形配置処理を実現することは可能である．しかし，この方法はスタイルシートの記述に手間が掛かり，実現のコストが高くなってしまう．

このような状況に対応するために，本研究では，XMLを対象とした，新しい情報視覚化方式を提案する．その基本的なアイデアは，XSLTスタイルシートを用いた枠組において，**制約**による図形配置の機能を導入した高水準な視覚化言語を採用することで，図形的な視覚表現の実現を容易化するというものである．制約を用いる場合，図形間の幾何的関係を宣言的に記述しておくだけで，システムが図形配置を自動的に計算するため，図形配置を簡潔に実現することが可能である．このため，制約による図形配置は，これまでにも様々な視覚化システムにおいて採用されている[2)~4),6),8),9),12),16),17),20),22]．

提案する方式のさらなる特徴として，**制約階層**[5] の採用が挙げられる．これによって，制約が十分に与えられていない状況でも，デフォルトの図形配置を得ることができる．また反対に，制約が必要以上に指定されている状況では，矛盾し合う制約を緩和して，妥協的な図形配置を求めることも可能である．このような機能は，スタイルシートの記述をさらに容易化し，XMLの多様性に柔軟に対応していくための有力な手段となる．

以上の方式の実現のために，本研究では，制約を用いた新しい高水準な視覚化言語XML Visualization Language (XML-VL) を構築する．XML-VLでは，図形要素の間に幾何的な制約を与えることで，宣言的に図の構造が記述される．本研究では，XML-VLを採用した視覚化システムをJava言語で実装した．

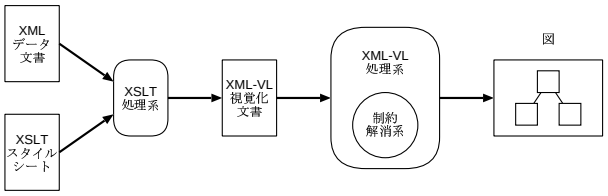

図 1 提案する視覚化方式

本論文は以下の構成からなる．まず，本研究が提案する視覚化方式の概要を説明する．次に，新たに構築した視覚化言語 XML-VL について述べる．その後，視覚化の実例を与え，視覚化システムの実装面を述べる．さらに，提案した方式の評価と関連研究との比較の後，結論と今後の課題を述べる．

2. 提案する視覚化方式の概要

本研究で提案する視覚化方式は，図 1 のように，3 つの XML 文書を伴った，2 つの段階からなる．まず，第 1 段階で，視覚化対象の元データを含む任意形式の XML 文書に対して，XSLT 形式のスタイルシート文書を適用することで，XML-VL 形式の視覚化用の文書が生成される．第 2 段階では，第 1 段階で生成された視覚化用の文書に基づき，図が構成されて表示される．以下では，単に，視覚化対象の文書をデータ文書，スタイルシート文書をスタイルシート，視覚化用の文書を視覚化文書と呼ぶ．

第 1 段階におけるスタイルシートは，データ文書の形式に対応して記述する必要がある☆．スタイルシートを適切に設計することで，1 節でも述べたように，データ文書の修正に応じて新しい視覚化文書を容易に生成できる．また，複数のスタイルシートを個別に適用して，異なる視覚化文書を生成することも可能である．

第 2 段階における視覚化文書には，図を構成するための表現として，グラフィカルオブジェクトの指定や，それらの間の幾何的関係を表す制約の宣言などが含まれる．これらの表現を制約解消系で処理することで，実際のグラフィカルオブジェクトの位置や大きさが決定され，最終的に図が表示される．これらの点の詳細については，次節で述べる．

☆ データ文書の形式は，必ずしも文書型定義で定められたものである必要はない．しかし，スタイルシートの設計作業上，文書型定義が与えられていることが望ましい．

3. 視覚化言語 XML-VL

本節では，本研究において構築した，視覚化文書を記述するための言語 XML-VL について述べる．

3.1 モ デ ル

XML-VL は，XML の 1 種であり，視覚化のための要素を記述できる．XML-VL が提供する要素は，以下の 3 種類である．

図形： グラフィカルオブジェクトを表す．

結線： 2 つの図形を線で結ぶ．

制約： 図形間の幾何的関係を与える．

XML-VL において，図形の幾何的関係は全て制約として表現される．そして，制約の集合を解消した結果として，全体的な図形の配置が定められる．

制約解消は，**制約階層** [5] の理論に基づく．制約階層では，制約の矛盾が存在する場合にも適当な解が得られるよう，制約を緩和することができ，各制約には，**強さ**と呼ばれる優先度が与えられる．XML-VL における制約の強さには，優先度の高い順に，`required`, `strong`, `medium`, `weak` の 4 種類がある．`required` は，必ず満たされるべき必須制約を示し，それ以外の強さは，同じ強さ以上の制約に対して矛盾する場合に緩和され得る選好制約を示す．

制約要素として，明示的に記述する制約には，強さとして `required` または `strong` を与えることができる．一方，強さ `medium` と `weak` は，デフォルトの配置を表現するためにシステムが内部的に生成する，暗黙の制約に対して付与される．

3.2 構文とその基本的意味

本項では，各要素の構文とその基本的意味を述べる．

3.2.1 図 形 要 素

基本的な図形要素として，現在，テキスト，長方形，菱形，楕円形があり，それぞれ型 `text`, `rectangle`, `diamond`, `oval` で表される．また，特別な図形要素として，見えない枠を形成する `box` がある．各図形要素には，属性 `id` によって個別の名前を与えることができる．それ以外にも，図形要素の種類に応じた属性があり，例えば `text` 要素の場合は，属性 `label` によって，表示すべき文字列を指定できる．

`text` 以外の図形要素は，その内容として，子の図形要素を持つことが可能である．例えば，以下は，Rectangle というラベルのテキストオブジェク

トを子とする長方形オブジェクト r を表す.

```
<rectangle id="r">
 <text label="Rectangle"/>
</rectangle>
```

なお，text 要素が id 属性を持たない場合には，単に文字データを書くことで，同等の効果を得ることが可能である．例えば，以下は，上の例と同値である．

```
<rectangle id="r">Rectangle</rectangle>
```

3.2.2 結線要素

結線要素は，型 connect で表される．結線要素は，2 つの属性 from と to によって，接続すべきオブジェクトに対応する図形要素を参照する．また，属性 label によって，結線の中央付近にラベルを表示することも可能である．以下は，図形要素 r1 と r2 に対応する 2 つのオブジェクトを直線で接続する結線要素である．

```
<connect from="r1" to="r2"/>
```

3.2.3 制約要素

制約要素としては，大きさ調節，包含関係，整列，グラフ上の隣接があり，それぞれ型 size, inside, alignment, adjacent で表される．制約の強さは，属性 strength によって与えられ，その他にも，制約の種類に応じた属性がある．以下は，図形要素 r1 と r2 に対応する 2 つのオブジェクトを，中央で揃えて垂直に整列する，強さ strong の制約を表す.

```
<alignment shapes="r1 r2" type="center"
  strength="strong"/>
```

3.3 デフォルトの配置

XML-VL では，制約が十分に与えられていない状況で，図のデフォルトの配置を得ることが可能である．これは，暗黙的な制約を生成することで実現されるもので，以下の 2 つの場合がある．

3.3.1 階層的な図形

図形要素が階層的な構造を持つ場合に，暗黙的な制約が生成される．具体的には，親子関係を持つ要素間に，包含関係を表す必須制約が与えられ，兄弟関係にある要素間に，垂直方向に整列する medium の制約が与えられる．例えば，次のような階層的な要素群を考える．

```
<rectangle id="r">
 <oval id="c1"/>
 <oval id="c2"/>
</rectangle>
```

これは，内部的に，以下の要素列として処理される．

```
<rectangle id="r"/>
<oval id="c1"/>
<oval id="c2"/>
<inside parent="r" children="c1 c2"
  strength="required"/>
<alignment shapes="c1 c2" type="center"
  strength="medium"/>
```

3.3.2 図形の大きさ

全ての rectangle, diamond, oval, box 要素に対しては，その形状をデフォルトの大きさへ可能な限り近づける，暗黙的な weak の制約が与えられる．これによって，子を含む図形要素を適切な大きさに調節し，図全体の均整をとることができる．

4. 実　　例

本節では，本研究で提案した方式による視覚化の実例を 2 つ与える．

4.1 概　　要

最初の例では，データベースにおける実体関連スキーマを実体関連図として視覚化する．具体的には，グラフ構造を表すデータに対して単純なスタイルシートを適用することで，グラフ上の隣接制約を用いた図形配置の記述へと変換する．ここでは，入力と出力のデータのいずれもがほぼ平坦（非階層的）な構造であるため，変換の処理も簡単なものとなる．

第 2 の例では，組織のメンバー構成を表すデータを組織図として視覚化する．具体的には，階層的な要素からなるデータを元に，木構造の図形配置を求める．ここでは，要素の親子関係によってデータの階層構造が表現されているため，その適切な処理が必要となる．XML 文書では一般に要素の階層構造が多用されるため，これは，本研究の視覚化方式が扱い得る問題の典型例の 1 つであるといえる．

4.2 実体関連図

最初の例として，データベースにおける実体関連スキーマを実体関連図として視覚化する場合を与える．まず，実体関連スキーマが，以下のような XML データによって表現されているとする．ここでは，実体が要素 entity, その属性が要素 attribute, 実体間の関連が要素 relationship として表されている．

```
1 <?xml version="1.0"?>
2 <erschema>
3  <entity id="DEPARTMENT">
4   <attribute id="DName"/>
```

```
 5   <attribute id="DNum"/>
 6  </entity>
 7  <entity id="PROJECT">
 8   <attribute id="PName"/>
 9   <attribute id="PNum"/>
10  </entity>
11  <entity id="EMPLOYEE">
12   <attribute id="EName"/>
13   <attribute id="ENum"/>
14  </entity>
15  <relationship id="CONTROLS"
      entity1="DEPARTMENT" cardinality1="1"
      entity2="PROJECT" cardinality2="N"/>
16  <relationship id="MANAGES"
      entity1="EMPLOYEE" cardinality1="1"
      entity2="DEPARTMENT" cardinality2="1"/>
17  <relationship id="WORKS_FOR"
      entity1="EMPLOYEE" cardinality1="N"
      entity2="DEPARTMENT" cardinality2="1"/>
18  <relationship id="WORKS_ON"
      entity1="EMPLOYEE" cardinality1="M"
      entity2="PROJECT" cardinality2="N"/>
19 </erschema>
```

次に，上記の形式で表された実体関連スキーマのデータを，実体関連図を表すXML-VLの文書へと変換する，以下のようなスタイルシートを用意する．ここでは，entity, attribute, relationship要素を図形要素に変換し，それらの間にグラフ上の隣接を表す制約要素adjacentを与えることで，これを実現する．

```
 1 <?xml version="1.0"?>
 2 <xsl:stylesheet xmlns:xsl=...omitted...>
 3  <xsl:output method="xml" ...omitted.../>
 4  <xsl:template match="erschema">
 5   <xmlvl>
 6   <xsl:apply-templates select="entity"/>
 7   <xsl:apply-templates select="relationship"/>
 8   </xmlvl>
 9  </xsl:template>
10  <xsl:template match="entity">
11   <xsl:element name="rectangle">
12    <xsl:attribute name="id">
13     <xsl:value-of select="@id"/>
14    </xsl:attribute>
15    <xsl:value-of select="@id"/>
16   </xsl:element>
17   <xsl:apply-templates select="attribute"/>
18  </xsl:template>
19  <xsl:template match="attribute">
    ...omitted...
45  </xsl:template>
46  <xsl:template match="relationship">
    ...omitted...
97  </xsl:template>
98 </xsl:stylesheet>
```

このスタイルシートにおける典型的な処理の1つは，第10〜18行で，entity要素に対して，長方形の図形要素を生成し，そのid属性と内容として，entity要素のid属性を設定する部分である．

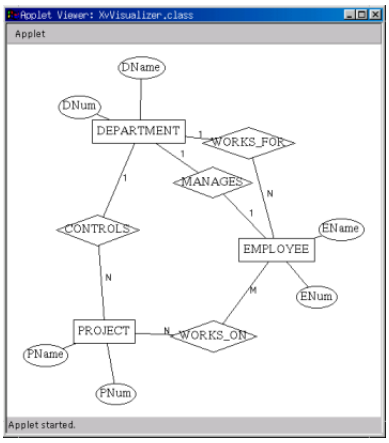

図 2　実体関連図

実際に，以上のデータ文書とスタイルシートから，以下のようなXML-VLによる視覚化文書が生成され，図2のような視覚化が得られる．

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <!DOCTYPE xmlvl SYSTEM "xmlvl.dtd">
 3 <xmlvl>
 4  <rectangle id="DEPARTMENT">
     DEPARTMENT</rectangle>
 5  <oval id="DName">DName</oval>
 6  <adjacent shape1="DName"
     shape2="DEPARTMENT" distance="80"/>
 7  <connect from="DName" to="DEPARTMENT"/>
 8  <oval id="DNum">DNum</oval>
 9  <adjacent shape1="DNum"
     shape2="DEPARTMENT" distance="80"/>
10  <connect from="DNum" to="DEPARTMENT"/>
    ...omitted...
45 </xmlvl>
```

4.3 組　織　図

第2の実例として，組織のメンバー構成を表すデータを組織図として視覚化する例を与える．ここでは，組織は要素sectionによって階層的に分けられ，各section要素には，1つのhead要素と，複数のsectionおよびmember要素が含まれている．これらの要素は，4.2節の場合と異なり，id属性を持たない．

```
 1 <?xml version="1.0"?>
 2 <organization>
 3  <section>
 4   <head>President A</head>
 5   <section>
 6    <head>Director B</head>
 7    <section>
 8     <head>Chief C</head>
 9     <member>D</member>
10     <member>E</member>
```

```
11    </section>
12    <member>F</member>
13    <section>
14     <head>Chief G</head>
15     <member>H</member>
16    </section>
17   </section>
18   <section>
19    <head>Director I</head>
20    <member>J</member>
21    <section>
22     <head>Chief K</head>
23     <member>L</member>
24     <member>M</member>
25    </section>
26    <member>N</member>
27   </section>
28  </section>
29 </organization>
```

このような形式のデータを，組織図を表す視覚化文書へと変換するため，以下のスタイルシートでは，再帰的に section 要素を処理し，また，必要に応じて，XSLT の generate-id 関数により，id 属性の自動生成を行っている．制約の生成には，box 要素による整列配置を利用しており，第 12〜14 行で，同じレベルの section と member を上側揃えで水平に整列し，第 10〜15 行で，これらとその直接の head を中央揃えで垂直に整列している．

```
 1 <?xml version="1.0"?>
 2 <xsl:stylesheet xmlns:xsl=...omitted...>
 3  <xsl:output method="xml" ...omitted.../>
 4  <xsl:template match="organization">
 5   <xmlvl>
 6    <xsl:apply-templates select="section"/>
 7   </xmlvl>
 8  </xsl:template>
 9  <xsl:template match="section">
10   <box alignment="center">
11    <xsl:apply-templates select="head"/>
12    <box alignment="top">
13     <xsl:apply-templates
         select="section|member"/>
14    </box>
15   </box>
16  </xsl:template>
17  <xsl:template match="head">
18   <xsl:call-template name="person"/>
19  </xsl:template>
20  <xsl:template match="member">
21   <xsl:call-template name="person"/>
22  </xsl:template>
23  <xsl:template name="person">
24   <xsl:element name="rectangle">
25    <xsl:attribute name="id">
26     <xsl:value-of select="generate-id(.)"/>
27    </xsl:attribute>
```

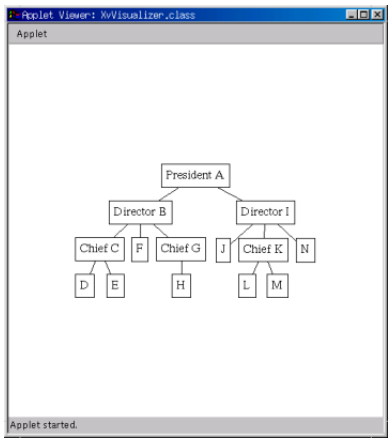

図 3　組織図

```
28    <xsl:value-of select="."/>
29   </xsl:element>
      ...omitted...
47  </xsl:template>
48 </xsl:stylesheet>
```

上記のデータ文書とスタイルシートによって，以下の視覚化文書と，図 3 のような視覚化が得られる．

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <!DOCTYPE xmlvl SYSTEM "xmlvl.dtd">
 3 <xmlvl>
 4  <box alignment="center">
 5   <rectangle id="N5">President A</rectangle>
 6   <box alignment="top">
 7    <box alignment="center">
 8     <rectangle id="Na">Director B</rectangle>
 9     <connect from="N5" to="Na"/>
10     <box alignment="top">
11      <box alignment="center">
12       <rectangle id="Nf">Chief C</rectangle>
13       <connect from="Na" to="Nf"/>
14       <box alignment="top">
15        <rectangle id="N12">D</rectangle>
16        <connect from="Nf" to="N12"/>
17        <rectangle id="N15">E</rectangle>
18        <connect from="Nf" to="N15"/>
19       </box>
20      </box>
        ...omitted...
31     </box>
32    </box>
      ...omitted...
53   </box>
54  </box>
55 </xmlvl>
```

5. 実　　装

本論文で述べた方式に基づき，XML のための情報視覚化システムを Java 言語で実装した．システムの実装において，XML 文書の構文解析に

はApache Xerces 1.1.3を用い，XSLT処理系としてApache Xalan 1.2.D02を使用した．また，制約解消系としては，著者が開発したChorus制約解消系[13]を採用した．プログラミング環境にはJava Development Kit 1.1.8相当のIBM製の処理系を利用し，Linux 2.2.14上で開発を行った．

実装したシステムは，単体のアプリケーションまたはアプレットとして実行できる．その起動の際に，データ文書のファイル名を引数として与えると，視覚化文書への変換を経て，その視覚化結果がウィンドウに表示される．スタイルシートの指定は，データ文書内に記述する場合と，システム起動の際に引数として与える場合の2通りが可能である．また，拡張子が`xmlvl`のファイルを与えた場合は，変換のプロセスは省略され，そのままXML-VL形式として解釈されて，図が出力される．

ユーザーとシステムの間のインタラクション機能として，システムが求めた視覚化結果に対して，グラフィカルオブジェクトのドラッグにより，ユーザーが修正を加えることが可能である．これは，視覚化を計算した後に，デフォルト配置のための選好制約を削除することで実現している．

6. 議　　論

本節では，本研究で提案した視覚化方式に関する評価の後，関連する研究との比較を行う．

6.1 提案した視覚化方式の評価

本研究では，XML文書の視覚化における主要な課題として，多様な形式のXML文書への対応と，図形的な視覚表現の実現の2点に焦点を合わせた．本研究で提案した視覚化方式では，これらの課題に応えるために，以下の方法を採用している．

- スタイルシートの採用．本方式では，XSLTスタイルシートを利用することで，様々な形式のXMLデータを扱うことが可能である．
- 制約階層を用いた高水準な視覚化言語の構築．図形的な視覚化には，図形要素の配置を制約として宣言的に記述する方法が有効である．このため，本方式では，視覚化言語において，図形要素に関する幾何的な制約の利用を可能にしている．また，その枠組として制約階層を採用することで，制約が不足している場合に，デフォルトの配置が得られるようにしている．このことは，XMLの多様性に柔軟に対応していく上でも，有力な手段となり得る．

6.2 関連研究との比較

XMLの枠組の中で，本研究で提案したXML-VLに類似した言語として，DrawML[18]がある．DrawMLは図形作成のための言語で，オブジェクトの整列機能などを持っており，木やグラフの表示も可能である．従って，XSLTとDrawMLを利用することで，本研究に類似した視覚化を実現することも可能である．しかし，DrawMLは整列以外の制約処理機能を持たない上，構想の段階に留まったままで，実装されていない．

HTMLにおけるオブジェクトの配置のために，制約階層を用いる研究がなされている[4),6),21)]．これらによって，より柔軟なオブジェクト配置が実現できるだけでなく，仕様定義を簡明化し厳密化することが可能となる．ただし，これらはHTMLにおけるオブジェクト配置を扱ったものであり，本研究が対象とするような，様々なデータの図形的な視覚化を実現するものではない．

XMLやHTMLに限らない一般的な観点で，本研究に最も関係の深い視覚化方式として，視覚化システムTRIP[17),22)]に採用された変換モデルが挙げられる．TRIPでは，アプリケーションのデータ表現から，絵のデータ表現への変換を，抽象構造表現と絵構造表現を経由することで実現する．ここで，抽象構造表現と絵構造表現はPrologの述語として表されており，その変換はPrologにより行われる．また，絵構造表現から絵のデータ表現への変換は，制約解消系により行われる．このため，TRIPにおける抽象構造表現と絵構造表現は，本研究におけるデータ文書と視覚化文書にそれぞれ対応し，TRIPにおけるPrologによる変換は，本研究におけるXSLTによる変換に対応していると見なすことが可能である．TRIPに対する，本研究で提案した方式の優位点として，本方式では，制約階層によってデフォルトの図形配置を実現している点が挙げられる．TRIPにおける制約階層は，2つの優先度のみに制限されており，デフォルトの図形配置には適さない．

制約を用いた視覚的インターフェースには，図形言語に関する文法を利用したものが多い．特に近年は，Penguins[8),9)]，恵比寿[2),3)]，VIC[12)]，Rainbow[16)]など，constraint multiset grammar[19)]を採用したインターフェースが提案されている．このような文

法を用いた視覚的インターフェースでは，図の生成よりもむしろ，図の構造を抽出する空間的な構文解析の実現に重点が置かれている．従って，様々な応用データの視覚化を目的とする本研究とは，焦点の相違がある．

7. おわりに

本研究では，制約を利用することで，データを記述したXML文書を効果的に視覚化する方式を提案した．この方式では，XSLTスタイルシートにより，対象となるXML文書を視覚化言語XML-VLへ変換することで視覚化を実現する．また，本研究では，この方式に基づき，実際に情報視覚化システムをJava言語により実装した．

今後の課題としては，以下の研究を計画している．

- XML-VLの強化．図形や制約の種類を充実することで，より多様な視覚表現を可能にする．
- インタラクション機能の拡充．本研究の視覚化方式では，基本的にユーザーはXMLデータを閲覧するだけで，変更することはできない．そこで，例えばTRIP2[20),22)]の双方向変換モデルのような手法を導入して，図の修正によるデータの更新を可能にする．また，図の閲覧についても，ナビゲーションなど，機能の拡充を行う．
- XSLTスタイルシート作成の容易化．現時点において，スタイルシートの記述は基本的にユーザーに任されており，システムによる支援は，実際に出力された視覚化文書に関するXML-VLの文書型定義による検証のみである．しかし，元のデータ文書の文書型定義と，スタイルシートの間に，密接な関連があることが考えられるため，このことを積極的に利用することで，スタイルシート作成の容易化を試みる．

参考文献

1) Adler, S. et al.: Extensible Stylesheet Language (XSL) Version 1.0, Working Draft 27-Mar-00, W3C (2000).
2) 馬場昭宏, 田中二郎: Spatial Parser Generatorを持ったビジュアルシステム, 情処学論, Vol. 39, No. 5, pp. 1385–1394 (1998).
3) 馬場昭宏, 田中二郎: 「恵比寿」を用いたビジュアルシステムの作成, 情処学論, Vol. 40, No. 2, pp. 497–506 (1999).
4) Badros, G., Borning, A., Marriott, K. and Stuckey, P.: Constraint Cascading Style Sheets for the Web, *Proc. ACM UIST*, pp. 73–82 (1999).
5) Borning, A., Freeman-Benson, B. and Wilson, M.: Constraint Hierarchies, *Lisp and Symbolic Comput.*, Vol. 5, No. 3, pp. 223–270 (1992).
6) Borning, A., Lin, R. and Marriott, K.: Constraints and the Web, *Proc. ACM Multimedia*, pp. 173–182 (1997).
7) Bray, T., Paoli, J. and Sperberg-McQueen, C. M.: Extensible Markup Language (XML) 1.0, Recommendation 10-Feb-98, W3C (1998).
8) Chok, S. S. and Marriott, K.: Automatic Construction of User Interfaces from Constraint Multiset Grammars, *Proc. IEEE VL*, pp. 242–249 (1995).
9) Chok, S. S. and Marriott, K.: Automatic Construction of Intelligent Diagram Editors, *Proc. ACM UIST*, pp. 185–194 (1998).
10) Clark, J.: XSL Transformations (XSLT) Version 1.0, Recommendation 16-Nov-99, W3C (1999).
11) Ferraiolo, J.: Scalable Vector Graphics (SVG) 1.0 Specification, Working Draft 03-Dec-99, W3C (1999).
12) Fujiyama, K., Iizuka, K. and Tanaka, J.: VIC: CMG Input System Using Example Figures, *Proc. Intl. Symp. on Future Software Technology*, pp. 67–72 (1999).
13) 細部博史: Chorus: モジュール機構を備えた幾何制約解消系, インタラクティブシステムとソフトウェア VIII—日本ソフトウェア科学会 WISS2000, レクチャーノート/ソフトウェア学, Vol. 24, 近代科学社, pp. 91–100 (2000).
14) 池田実: XMLの概要と応用, 情報処理, Vol. 39, No. 6, pp. 515–520 (1998).
15) 石川博: XMLとデータベース—交換から格納・収納へ—, 情報処理, Vol. 41, No. 1, pp. 68–73 (2000).
16) 丁錫泰, 田中二郎: Rainbow: ビジュアルシステム生成系におけるレイアウト制約の実現, 情処学論, Vol. 41, No. 5, pp. 1246–1256 (2000).
17) Kamada, T. and Kawai, S.: A General Framework for Visualizing Abstract Objects and Relations, *ACM TOG*, Vol. 10, No. 1, pp. 1–39 (1991).
18) Lothigius, H.: DrawML Specification, Note 03-Dec-98, W3C (1998).
19) Marriott, K.: Constraint Multiset Grammars, *Proc. IEEE VL*, pp. 118–125 (1994).
20) Matsuoka, S., Takahashi, S., Kamada, T. and Yonezawa, A.: A General Framework for Bidirectional Translation between Abstract and Pictorial Data, *ACM Trans. Inf. Syst.*, Vol. 10, No. 4, pp. 408–437 (1992).
21) Michalowski, B.: A Constraint-Based Specification for Box Layout in CSS2, Tech. Rep. 98-06-03, Dept. of Computer Science and Engineering, University of Washington (1998).
22) Takahashi, S., Matsuoka, S., Miyashita, K., Hosobe, H. and Kamada, T.: A Constraint-Based Approach for Visualization and Animation, *Constraints J.*, Vol. 3, No. 1, pp. 61–86 (1998).
23) Tidwell, D.: Tutorial: Transforming XML documents (1999). http://www-4.ibm.com/software/developer/education/transforming-xml/index.html .
24) Wood, L. et al.: Document Object Model (DOM) Level 1 Specification (Second Edition), Working Draft 29-Sep-00, W3C (2000).