国際経営・文化研究　Vol.7 No.1　January　2003

（論　文）

Web上における数式表現の意味形式記述への半自動変換システム開発

大武信之・秋山富貴子

キーワード

Web記述　　HTML　　XML　　MathML　　視覚障害者

1　はじめに

社会活動や日常生活において、インターネットは情報の収集と発信の道具として、欠かせない存在になっている。視覚障害者も健常者同様に、メールの利用およびホームページの閲覧を、スクリーン・リーダの読み上げを聞くことでインターネットの利用が可能である。全盲がメールの利用や、ホームページ閲覧を行っていることは余り知られていないが、画像や写真などの視覚に訴える部分以外の文字情報は、スクリーン・リーダに代表される感覚補償ソフトウェアや、点字ディスプレイを用いることで、情報アクセスが可能である。

次世代文書記述言語として、ISOによりSGML（Standard Generalized Markup Language）の言語仕様（ISO-8879）が1986年に定められた。現在インターネット上で、ホームページの記述にはHTML（Hyper Text Markup Languahe）が使用されているが、今後のWeb用マークアップ言語としては、HTMLに代わりXML（eXtented Markup Langage）[1] が主流となる。HTMLとXMLは、共にSGMLを母体として作成されたマークアップ言語である。現在使用されているHTMLでは、固定のタグしか使えずユーザ定義が行えない。一方、XMLはユーザが独自のタグを定義できるため、拡張性が高い。またHTMLでは数式記述用タグがないため、数式は画像として表示しているが、XMLでは数式記述がMathMLで定められている。これまで画像データとして表現されていた情報が、スクリーン・リーダが扱えるテキスト情報として表現できれば、視覚障害者が利用可能な情報源となる。現状での数式情報は、HTMLでは画像情報として表示されているが、次世代Web記述言語であるXMLでは、MathMLを用いたタグで表現される。ここでは画像情報を理解しにくい視覚障害者のための数式理解に関し、MathMLの利用について述べる。

2　Web記述言語

2.1　HTML

現在ホームページの記述として、HTMLが主流の言語として使用されている。HTMLは文

おおたけ　のぶゆき：筑波技術短期大学　教育方法開発センター　助教授
あきやま　ふきこ：筑波技術短期大学　保健管理センター　職員

書記述言語であるが、市販のワープロ・ソフト（例： Word, 一太郎）を用いることで、HTMLの文法を知らなくても、Web上のホームページを作成できる。また、ホームページ作成のための専用ソフトウェアも市販されており、写真や絵なども容易に貼り込むことができ、カラフルな表現が可能である。ただし、HTMLでは固定のタグしか使用できないため、ユーザ独自の記述法が指定ができない。HTMLにおいて、タグの種類が決まっている点は、欠点の一つではあるが、タグが固定されている故に、ワープロやホームページ作成ソフトで、HTMLファイルの作成が容易に行えるようになっている。

2.2 XML

XMLは、1998年にW3C（World Wide Web Consortium）[2] により勧告されたデータ記述言語の標準で、次世代文書記述言語として位置付けられている。W3Cは米国国防総省と欧州委員会の資金援助のもと、1994年10月マサチューセッツ工科大学計算機科学研究所（MIT/LCS）に、産学官共同コンソーシアムとして設立された。アジア地区では慶応義塾大学（SFC）がホストとして運営に加わっており、国際的会員制産学官共同コンソーシアムとして、500以上の組織が参加している。総務省も国が管理する文書の電子化にあたり、XMLを主要言語として検討を始めている。XMLは電子商取引やナレッジマネジメント、データストアなどITのあらゆる場面で注目を集め始め、ビジネスだけでなく行政などでも利用されようとしている。

次世代Web用記述言語として使用されるXMLがHTMLと異なるのは、ユーザ定義のタグが使え、自由な記述が可能な点である。固定のタグしか持たないHTMLと比較し、XMLはユーザ独自のタグが定義できるため、文書記述の方法が広がり、利便性も高くなる。さらにHTMLでは数式を記述するタグがないため、数式を表示するには画像データを表示する方法がとられているのに対し、XMLにおける数式記述はMathMLによるタグで記述され、MathMLの仕様はW3Cにより定められている。MathMLはXMLに含まれるマークアップ言語で、その他XMLには化学式記述のためのChemicalML、楽譜記述のためのMusicMLがある。XMLで使用可能な専門分野別マークアップ言語の中では、MathMLの詳細な仕様がW3Cにより公表 [3] されている。

2.3 SGML

HTMLとXMLは、SGML文書フォーマットのオープンスタンダードの流れを汲んでいる。XML登場以前に、Webで文書表示をするため、SGMLからHTMLが開発された。その後登場したXMLは、HTMLの成果を生かしつつ、HTMLとは別のものとしてSGMLから派生している。XMLは、複雑過ぎると言われるSGMLから機能を抜き出して洗練したもので、なおかつHTMLにあったWebの機能を生かすための機能を含んでいる。

2 2003年の電子政府実現に向け、電子署名・認証法（2001年4月1日施行）への対応が重要になる。従来の紙による申請に加え、国民（住民）は、インターネットを利用して電子署名を付加した電子申請を自宅から送ることも可能になり、SGMLの利用が期待できる。さらにSGMLはデータ管理や検索ができるのが特徴で、膨大なデータの中から必要な項目を簡単に抜き出すことが容易に行える。データのやり取りや更新が行えるので、経営情報システムや電子商取引等にも採用されることが予想される。

SGMLはテキストファイルではあるが、その仕様は複雑であるため、SGMLを直接記述する

には難がある。そのため、SGML記述を支援するエディタやソフトウェアが必要となるが、現段階では周辺をサポートする道具が不足しているのが現状である。従ってSGMLよりは簡便で、HTMLより拡張性の高いXMLの利用が促進される。

2.4　MathML

数式を記述するマークアップ言語として、TeXとLaTeX［4］はよく知られた言語で、TeXのマクロであるLaTeXで記述された科学技術文書は多く存在する。とくに理工系の学会論文では、LaTeXで記述することを奨励している。論文誌の投稿において、手書き原稿に対してワープロ原稿を提出した場合、掲載料を割り引く学会があり、科学技術文書を扱う学会では、LaTeX原稿をさらに割り引く特典を与えるところもある。TeXがアメリカ数学会の登録商標であることからも、TeXとLaTeXの価値の高さが伺い知れる。LaTeXは記述方法も容易で、サンプルを参照しながらであれば、コマンドの詳細を覚えることなく初心者にでも記述することができる。一方、XMLを記述するのは容易ではないため、何らかのアプリケーションソフトウェアを利用しファイルを得るか、特定のデータからXMLに変換を行うプログラムを用いてファイルを作成する。XMLはLaTeXほど容易に記述できるものではなく、
XMLに変換する道具を必要とする。XMLの数式部分はMathMLで記述されるため、XML同様エディタ等を用いて直接MathMLを記述することは容易ではない。したがって、MathMLファイルを作成するには、XML同様に何らかの道具を必要とする。本論では、XMLにおける数式記述言語であるMathMLによる数式意味記述の自動変換を行うプログラムについて4節で述べる。

3　数式記述

現在公開されているMathMLの定義書［3］には、MathMLを使用する上での利便性と共に、視覚障害者が使用する上での有効性が説かれている。視覚障害者に関する配慮が記述されている定義書や仕様書の類は稀有であるが、これは連邦政府の著作権改正とリハビリテーション法改正と、アクセシビリティ［5］の考え方が浸透してきた結果である。仕様書の中に書かれているMathMLを使用する上での利便性とは、Mathmatica（Wolfram Research Inc.）に代表される数式ソフトウェアや表計算ソフトウェア等への応用を指す。さらに、これまで数式の計算を手作業でプログラムに組んでいたものを、その一部をMathMLから自動生成することも可能となる。また、記述方法の自由度が高いLaTeXに対してMathMLは厳密な数式記述が可能で、文脈依存や曖昧性を除去した記述が行えるため、数式の読み上げを行うための音声化に有効である。W3Cのワーキンググループでは、全盲であるT.V. Raman［6］の経験と助言を、MathMLの開発に役立てている。

3.1　MathML記述形式

XMLにおける数式記述言語であるMathMLは、LaTeX同様の記号配列を定める記述に従った表示（presentation）形式による書き方と、数式の意味を含めた意味（content）形式による書き方の2種類の定義法がある。

1. 表示形式定義 …… presentaion
2. 意味形式定義 …… content

presentaion形式はLaTeX同様の表示的な構文を記述する方法で、content形式は数式の意味を記述する方法である。MathMLには3つの記述方法があり、それらはpresentaionによる表示形式記述か、contentによる意味形式記述か、両者の混在形式によるものである。本論では、表示形式と意味形式のみを扱い、表示形式と意味形式の混在形式による方法は除外する。

1. 表示形式記述 …… presentaion
2. 意味形式記述 …… content
3. 混在形式記述 …… presentaion + content

2次方程式 $a\chi^2 + b\chi + c = 0$ の解の公式をLaTeXで記述すると、次の2通りが標準的な書き方である。

```
x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}
x={-b\pm\sqrt{b^2-4ac}}\over{2a}
```

$$\longrightarrow \quad \chi = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$$

2次方程式の解の公式をLaTeXでは、上記2通りのわずか1行で記述できるが、MathMLで記述にすると図1のようになる。LaTeXは見かけ上の表示が正しければ書き方は自由であるので、LaTeXの数式記述をMathMLにおける表示（presentation）形式記述に変換すれば、数式を含むXMLファイルができる。現在LaTeXの数式部分を、MathMLに変換するプログラムがいくつか存在（4.1節）するが、その殆どが表示形式に変換するもので意味形式に変換するものは稀有である。本論では、既存ツールによりMathMLの表示形式に出力されたものを、さらに意味形式に自動変換し、MathMLの利用価値を高める。

3.2 表示形式・意味形式

数式をLaTeXで記述する場合、LaTeXの書き方は自由度が高い。例えば、定積分の範囲は、積分記号に下付き添え字と上付き添え字で積分の範囲を指定するが、上付きと下付きを逆にしても、表示は以下のように同じである。

```
\int_0^1 f(x) dx
\int^1_0 f(x) dx
```

$$\longrightarrow \quad \int_0^1 f(\chi)\,d\chi$$

上記積分の式において、積分範囲が［0, 1］であれば、これに従った順序（下限～上限）で記述するのが自然であるが、あえて逆（上限～下限）に記述してもLaTeXでは問題ない。また、同一の表示に対し、異なる命令で表示することも可能である。

4

```
\frac{a}{b} , {a} \over {b}
```

$$\longrightarrow \quad \frac{a}{b}$$

LaTeXによる数式記述は、記号の配置位置を記述するもので、意味記述は含まれていない。例えば、次の2つの数式記号は、LaTeXでは共に分数として記述する。

```
\frac{ax}{by}
```

$$\longrightarrow \quad \frac{a\chi}{by}$$

```
\frac{dx}{dy}
```

$$\longrightarrow \quad \frac{d\chi}{dy}$$

上記2つの数式は、一般的に前者を分数として読み、後者を微分記号とみなして読む。LaTeXが表示形式のみの記述で、微分記号を分数でしか記述できないのに対して、MathMLでは微分としての意味記述が可能である。また表示からもその違いが分かるように、MathMLでは分数式と微分記号を区別して表示している。具体的には、MathMLが表示可能なブラウザ上では、斜体のdと直立記号のdでその違いが分かる。しかし、現在のところMathMLを表示するブラウザが少ないため、MathMLの機能が十分発揮されていない。さらに表示可能なブラウザにおいても、LaTeXの表示に較べ、数式の表示バランスが見劣りするものが多く、美しい数式の表示ができていない。MathMLでは厳密な意味形式の記述が可能ではあるが、数式の表示面では難点がある。意味記述形式を表示した場合、数式の見栄えにはバランスが悪いという難点はあるが、数式の意味を厳密に記述できるのは強力で、その利用価値は大きい。例えば、数式において乗算記号は省略されることが多いが、省略されている乗算記号がスカラー、複素数、ベクトル、行列・行列式等の区別が可能であるため、演算子・記号・変数・定数の種別を把握[注1]することができる。XMLおよびMathMLはテキストファイルであるため、視覚障害者がパソコン上で使用するスクリーン・リーダは、読み上げが可能である。これまで数式は、HTMLではGIFなどの画像ファイルで表示されていたため、視覚障害者が使用しているスクリー

表示（present）形式

```
<mrow>
    <mi>x</mi>
    <mo>=</mo>
    <mfrac>
        <mrow>
            <mrow>
                <mo>-</mo>
                <mi>b</mi>
            </mrow>
            <mo>&PlusMinus ; </mo>
            <msqrt>
                <mrow>
                    <msup>
                        <mi>b</mi>
                        <mn>2</mn>
                    </msup>
                    <mo>-</mo>
                    <mrow>
                        <mn>4</mn>
                        <mo>&InvisibleTimes ; </mo>
                        <mi>a</mi>
                        <mo>&InvisibleTimes ; </mo>
                        <mi>c</mi>
                    </mrow>
                </mrow>
            </msqrt>
        </mrow>
        <mrow>
            <mn>2</mn>
            <mo>&InvisibleTimes ; </mo>
            <mi>a</mi>
        </mrow>
    </mfrac>
</mrow>
```

意味（content）形式

```
<apply>
    <eq/>
    <ci>x</ci>
    <apply>
        <divide/>
        <apply>
            <mo>&PlusMinus ; </mo>
            <apply>
                <minus/>
                <ci>b</ci>
            </apply>
            <apply>
                <root/>
                <apply>
                    <minus/>
                    <apply>
                        <power/>
                        <ci>b</ci>
                        <cn>2</cn>
                    </apply>
                    <apply>
                        <times/>
                        <cn>4</cn>
                        <ci>a</ci>
                        <ci>c</ci>
                    </apply>
                </apply>
                <cn>2</cn>
            </apply>
        </apply>
        <apply>
            <times/>
            <cn>2</cn>
            <ci>a</ci>
        </apply>
    </apply>
</apply>
```

図１：数式 $x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ **の記述例**

ン・リーダでは解釈不能であった。MathMLではテキスト形式で記述されているため、スクリーン・リーダでの対応が可能になり、視覚障害者にとり有効な情報となる。

既存LaTeXファイルや数式構造を持ったファイルをMathML形式に変換すれば、LaTeXの処理系や特別なアプリケーションを要することなく、既存ブラウザを用いて数式の表示が可能である。LaTeXの記述はMathMLにおけるpresentaion方式に当り、単なる表示形式のみへの変換では応用範囲が限定されるため、MathMLへの変換ではcontentで記述する意味形式記述に表現できれば利便性が高まる。前述のようにMathMLの記述方法には3種類（3.1節）存在するが、次節では表示（presentaion）形式と意味（content）形式の2つを扱い、表示形式と意味形式の混在形式は扱わない。

4　意味記述変換

2次方程式の解の公式をLaTeXでは1行で記述（3.1節）できるが、MathMLでは表示形式（presentation）および意味形式（content）のどちらにおいても、前述（図1）のとおり膨大な記述量（行数）になる。これらはLaTeXのように、手作業で容易に記述できるものではない。手作業でキーボードからタイプ入力する以外に、数式を得るには次の方法がある。

1. 印刷物をスキャナで読み込み、文字・記号・数式認識を行う。
2. 手書き（フリーハンド）入力された文字・記号・数式認識を行う。
3. 既存の他マークアップ言語（LaTeX等）を利用する。
4. 音声により数式を入力する。

上記1～3の方法において、入力された数式からMathML形式のファイルを得るには、内部にMathML用の変換プログラムを持つか、意図的に変換を行う必要がある。LaTeXが手作業で容易にタイプ入力できるのに対し、MathMLファイルをエディタ等を用いた手作業で作成することは困難であるため、MathML作成支援ツールが必要となる。項目1のスキャナを用いた読み込みと、項目2の手書きによる数式の入力は、すでに開発済のInfty［7］で実現している。項目4の音声による数式入力は、現在の音声認識に数式理解を盛り込むことは、4項目の中では効率が悪く現実的ではない。ここでは項目3によるMathML作成支援ツールを用いて、意味形式MathMLファイルの生成を行う。項目3による手法が実現できれば、開発済の項目1と2に、項目3の機能を取り入れることが可能である。

4.1　MathML変換プログラム

LaTeXは見かけ上の表示が正しければ書き方は自由であるため、MathMLの2つの定義法の1つである表示（presentation）形式記述に、LaTeXファイルから変換すれば、数式を含む
6 XMLファイルを生成できる。現在、LaTeXからの変換以外にも数式エディタやスキャナを用いた方法で、MathMLを含む数式記述を出力するプログラムが既に存在する。W3Cのホームページ［3］には、IBM techexplorer, Mathcad, MathType, Infty, TtM, TtH, LaTeX2HTML, BraMaNet, ConTeXt, MathPlayer, WebEQ, EzMath, Maple, mathmled, GtkMathView, Meditor, Publicon などが掲載されている。これらのプログラムはWindowsあるいはDOSベースで使用するもの、UNIX上でperlとの組み合せで動作可能なものである。ただし、そのほとんどのMathMLへの変換プログラムは、MathML表示

(presentation) 形式への出力である。したがって、表示 (presentation) 形式部分を、意味 (content) 形式へ自動変換できれば、視覚障害者が使用するスクリーン・リーダに解釈可能なMathMLの数式意味記述が使用できる。

上記応用ソフトウェアを用いれば、XML形式ファイルが得られ、いづれも数式部分は表示形式によるMathMLとして出力される。本システムではIBM techexplorerとMathTypeが出力する表示形式MathMLを入力として、意味形式MathMLへの出力を行った。本システムでは使用しなかった応用ソフトウェアでも、LaTeXからMathMLへの変換プログラムを利用すれば、これまで資産として蓄積されたLaTeXファイルを入力とし、容易にMathMLが得られることになる。ただし上記のいずれのソフトウェアにおいても、完全に正しい表示形式の出力を得られるものではない。例えば、次の2式があった場合、

$$a\chi^2 + b\chi + c = 0 \qquad D = b^2 - 4ac$$

右式を判別式とみなし、4acの部分は4×a×cと解釈するが、左右2つの式が全く無関係であれば、acという1変数を4倍したものであれば4×acとなる。4とaの間に乗算記号×が省略されていることは明らかだが、aとcの間に乗算記号×が省略されているか否かは明らかではない。4acの部分に関し、この違いを表示形式で示すと以下のようになる。

```
<mrow>
    <mn>4</mn>
    <mo>&InvisibleTimes ; </mo>
    <mi>a</mi>
    <mo>&InvisibleTimes ; </mo>
    <mi>c</mi>
</mrow>
```

```
<mrow>
    <mn>4</mn>
    <mo>&InvisibleTimes ; </mo>
    <mi>ac</mi>
</mrow>
```

表示的な記述のみから、acが1変数か2変数の乗算であるかは読み取れない。しかし、MathMLの表示形式から意味記述への変換を行うには、表示形式が正しく記述されていることが前提となる。したがって、正しい表示形式を得るには、手作業での修正が不可欠となるが、これを効率良く行うために、手作業による変更が必要となる個所を効率的に見い出し自動化できれば、修正作業を減らすことができる。以下の節において、修正手順および意味記述への自動変換について述べる。

4.2　修正

数式において変数は好みの記号を選んで使え自由度が高いが、固定的に使われるものもあり、添え字等には i, j といったものがよく使われる。また数学の専門分野ごとに、頻繁に使われる特定の記号が存在する。LaTeXおよびMathMLの表示形式では、分数として表記するものが微分であったり、上付き添字は乗数を表すが、乗数ではない意味を持つ数式も存在する。以下の例は、LaTeXでは分数および乗数の命令文を使用し記述されるが、実際の数式の意味は、微分や三角関数の逆関数を表している。

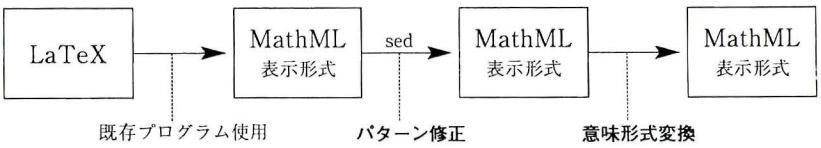
$$\frac{\partial^2 f}{\partial \chi^2}, \qquad \frac{d}{d\chi}\left(\frac{d\,y}{d\chi}\right), \qquad \sin^{-1}\chi, \qquad f^{(n)}(\chi) = \frac{d^n y}{d\chi^n}$$

数式の変数には自由な記号を用いることが可能だが、数式中における e は変数と見るより

自然対数の底としての e とみなす方が自然である。しかし前節で挙げた、4acが 4・a・cであるか4・ac であるかの判定はできないので、一般的に使われる記述パターンから意味を見い出し、意味記述への修正を行う方法を用いる。具体的には、分数に現れる∂は、分子・分母の要素ではなく微分記号と見なせ、$\sin^{-1}\chi$ の－1はLaTeXでは乗数として記述されるが、数式の意味は逆関数と見なすほうが自然である。ただし、以下の例のように、縦棒2本の間に記号が挟まれた数式表記のみからでは判断がつかないものもある。

$|X|$　絶対値　　$|Y|$　行列式　　$|Z|$　濃度

LaTeXの数式記述のみから、意味を完全に抽出するのは困難である。上記の例で、どの意味が当てはまるかを判断するには、そこに書かれている文章から文脈を読み取らなくてはならない。また、LaTeX以外の他マークアップ言語で記述された場合も同様で、表記された数式からのみでは意味抽出は完全には行えない。従って、MathMLへの変換ソフトウェアおよび類似ソフトウェアにおいても、MathMLにおける正しい表示形式に自動的に変換するには無理がある。いずれの処理過程においても、人手による修正は必要であるが、手作業による修正を最小限にするために、典型的パターンで書き換えが可能と見られる部分を、sed（stream editor）[8] のスクリプト・ファイルとして生成する。



修正個所をsedのスクリプト・ファイルとして生成するのは、自動的に修正を加えると、修正個所の確認作業が行えないことと、修正が不必要なところに修正を加えてしまう可能性があるため、修正すべき個所の記録ということでスクリプト・ファイルを生成する。典型的パターンで書き換えが可能と見られる部分の抽出のみでは不備があるため、手作業による修正は避けられない。そのため、修正個所の有無をスクリプト・ファイルで確認を行いながら、不備も見出す方法を用いる。

$\frac{d}{d\chi}$（分数）　⟶　$\frac{d}{d\chi}$（微分）

```
<mfrac>                          <mfrac>
   <mi> d </mi>                     <mo> &DifferentialD ; </mo>
   <mrow>                           <mrow>
      <mi> d </mi>                     <mo> &DifferentialD ; </mo>
      <mi> x </mi>                     <mi> x </mi>
   </mrow>                          </mrow>
</mfrac>                         </mfrac>
```

```
2s/<mi>/<mo>/
2s/d/\&DifferentialD ; /
2s/<mi>/<mo>/
4s/<mi>/<mo>/
4s/d/\&DifferentialD ; /
4s/<mi>/<mo>/
```

図2：生成sedスクリプト（下段）による変換例（中段）

4.3　数式意味形式変換対象

本節では、LaTeXから変換されたMathML表示形式をMathML意味形式に変換し、利用範囲の広い数式データを得るプログラムについて述べる。ただし、MathMLの定義書［3］は非常に大きなものであるため、対象となる数学の範囲を大学1～2年程度までとする。MathMLはLaTeXと同様に文字サイズや空白の間隔等を微妙に調整できるが、これらは数式の意味を表現するものではないため、意味変換への対象とはしない。また、テキストや記号の色指定も可能であるが、これらも対象とはせず、情報として残すだけとする。数式意味記号は一項演算子（unary）、二項演算子（binary）、多項演算子（n-ary）に分類され、各演算子に対し表1の修飾子が対応している。したがってMathML意味形式変換には、前述の条件で表示形式を解析対象とし、意味構造を持つ解析木を生成すればよい。ここでは、この解析木をMathML意味形式に合った形で出力するが、他のマークアップ言語や、特定アプリケーションのデータ形式に変換出力することも可能である。

表示形式から意味形式への変換は、
表示形式が不完全であっては意味を成さないため、表示形式の完全な形式への修正は4.2節で述べた方法で行う。意味形式への変換を自動的に行うには、意味形式の記述方法が一意に定められていなければならない。しかし図3にあるように、数式を記述するには、1つの式に対し意味形式の記述方法が必ずしも一意ではない。演算子とその修飾子（表1）の記述方法は全て一意には決まらないため、一定の出力パターンを取るという統一を図るため、MathML定義書［3］とW3Cで照会されている関連サイトの例を元に、使用度の高いものと、サンプルとして示される場合が多いものを、本プログラムでの出力形とした。

$$\int_a^b f(\chi)\,d\chi$$

表示形式記述

```
<mrow>
    <msubsup> <mo>&int ; </mo> <mn>a</mn> <mn>b</mn> </msubsup>
    <mrow>
        <mi>f</mi> <mo>&ApplyFunction ; </mo>
        <mrow> <mo>（</mo> <mi>x</mi> <mo>）</mo> </mrow>
        <mo>&InvisibleTimes ; </mo>
        <mrow> <mo>&DifferentialD ; </mo> <mi>x</mi></mrow>
    </mrow>
</mrow>
```

意味形式記述　1.

```
<apply>
    <int/> <bvar><ci>x</ci></bvar>
    <interval><cn>a</cn><cn>b</cn></interval>
    <apply><fn><ci>f</ci></fn><ci>x</ci></apply>
</apply>
```

意味形式記述　2.

```
<apply>
    <int/> <bvar><ci>x</ci></bvar>
    <lowlimit><ci>a</ci></lowlimit>
    <uplimit><ci>b</ci></uplimit>
    <apply><fn><ci>f</ci></fn><ci>x</ci></apply>
</apply>
```

図3：積分の例

9

本プログラムでの出力形とした。

演算子	int, sum, product, root, diff, partialdiff, limit, log, moment, min, max, forall, exist
修飾子	lowlimit, uplimit, bvar, degree, logbase, interval, condition, domainfapplication, momentabout

表１：演算子・修飾子

4.4 数式意味形式自動変換

表示形式から意味形式に変換するには、表示形式の数式から解析木を生成し、生成された解析木から意味表示に書きかえる手順をとる。各要素（タグ）は固定（1,2または3個）か任意個の引数からなっている。数式は解析木の形式でメモリー上に保持するが、任意個の引数はリスト形式であるが解析木の節の一部となる部分木として扱う。引数が同じであっても、表示形式と意味形式では引数の並びが同じものもあれば、異なるものもある。例えば、分数は分子・分母の順で表示形式と意味形式が同じであるが、根号（$\sqrt[n]{\chi}$）では、次数（n）が表示形式は後で意味形式は先である。そこで解析木は表示形式の並びに従った引数の順とし、意味形式変換の際に並びの変更を行う。また、表2のように、表示が等しいが数式の意味が違う場合は、使用頻度の高いもの及び記述が簡単なものを出力する。一般に数式の1要素も数式であるため、表示形式のタグ解析は再帰的に行う。なお本システムは、汎用性を高めWindows, UNIX, Linux等で使用可能なように、C言語で記述した。

数式意味形式自動変換は、数式表示形式解析木から数式意味形式解析木への内部構造の変換を行えばよいが、出力はMathMLの要素（タグ）単位を出力したファイル形式にしなければならない。XMLにおけるMathMLの部分は$から$の部分であるから、出力を見やすくするためにはインデンテーションを付ける必要がある。ただし、閲覧ソフトウェアであるブラウザにとりインデンテーションは無くても表示可能であるため、一意的なインデンテーションとして解析木の深さに対応したインデンテーションを出力する。タグの修飾子にある活字サイズや色指定等は、数式の意味としては自動変換の対象外であるが、本プログラムの今後の拡張のため、オプション指定部として分解析木に残しておく。

表　示	意　味	MathML　意味記述
f	変数 関数	<ci> f </ci> <ci type ="fn"> f </ci>
χ^n	冪乗 次数	<apply><power/><ci>c</ci><ci>n</ci></apply> <bvar><ci>x</ci><degree><ci>n</ci></bvar>

表２：意味修飾記述

4.5 オプション解析

一般のタグは＜tag-name＞という形をしているが、タグ名の後に空白を置き、属性を記述することができる。オプション付タグの形式は属性部が＜tag-name *attribute*＞で記述され、

属性は attribute-name="value"またはattribute-name='value'で指定される。このタグ名に続くオプション部分を解析するには、いくつかの問題がある。

次の分数式は、全く同じ意味であるが、右式では属性部で空白を付加することで、式が読みやすくなる。

$$\frac{a+b+d}{a+b+c+d} \qquad \frac{a+b\quad+d}{a+b+c+d}$$

空白を入れるオプションを付けることで、読みやすさの改善を図っているだけで、属性部がある式と、属性部が無い式では、数式の意味には相違点はない。しかし、次の行列では、オプションで文字（フォント）サイズを指定することで、1要素がゼロであることと、上三角部分全てがゼロであることの違いを、表示形式の中で表現している。文字サイズの変更により、行列の意味がまるで異なってしまい、MathMLの表示形式を読むだけでは意味を理解することは容易ではない。この場合は、実際に表示された数式を確認することで、はじめて意味内容をどのように表現しているか理解できる。

$$\begin{pmatrix} \ddots & & \\ & \ddots & 0 \\ & & \ddots \\ & & & \ddots \end{pmatrix} \qquad \begin{pmatrix} \ddots & & \\ & \ddots & 0 \\ & & \ddots \\ & & & \ddots \end{pmatrix}$$

分数と行列の例において、分数の例は意味表現が変わらないが、行列の例は、式の内容が異なっている。ただし、行列の2つの式は、共に行列であることには変わりはない。次の例は、丸括弧内に分数を記述した形で式が配列されている。右式にはオプション指定があり、分数の線の太さをゼロに指定しいる。右式ではタグの記述が分数でありながら、表示されている式はベクトルの形となり、オプションの指定により右式と左式では、全く異なった意味を持つことになる。

$$\begin{pmatrix} \frac{a}{b} \end{pmatrix} \qquad \begin{pmatrix} a \\ b \end{pmatrix}$$

```
<mfenced>
    <mfrac>
        <mi> m </mi>
        <mi> n </mi>
    </mfrac>
</mfenced>
```

```
<mfenced>
    <mfrac linethickness="0">
        <mi> m </mi>
        <mi> n </mi>
    </mfrac>
</mfenced>
```

上記3例のように、オプション部分の指定は、数式の意味解釈に影響を与えるものと、与えないものとがあり、これらを自動的に解釈することは困難である。従って、現状ではオプション部分の解析は行わずに、属性指定があることを知らせ、個々の式の対応は、表示を確認しながら修正を加えるしか方法はない。従って本プログラムは、半自動化変換の段階にある。

11

5　まとめ

次世代のWeb技術として注目を集めているのが、意味を理解するWebとしてのSemantic Webである。現在のWebが人間用の情報で、コンピュータが処理に適した形式になっていない。Webに対する知的コンピュータ処理が行えれば、Web情報の有効利用が可能となる。現

状では、サーチエンジンを用いてWebから、様々な情報を瞬時に得ているかのようであるが、実際は膨大な情報の山を検索し、必要としている情報を見逃していることが往々にある。コンピュータがWeb情報の意味を理解して処理が行えれば、Semantic Webは現状の技術の限界を打ち破るものとなる。

Semantic Webが要求するタグ付を、一般のユーザが容易に行えるかという問題が残るが、Semantic Webで記述されるようになれば、Web上の情報は巨大な知識データベースとなる。またデータベースの応用としては、電子商取引に代表されるe-business（e-commerce）、効率の良いシステム開発におけるプログラムの自動合成、知識表現や推論機構を持った人工知能開発など応用分野の広がりが期待される。

W3CのホームページにおいているMathML変換プログラムは、LaTeX・スキャナ・手書き・記号写植入力から、MathMLの表示形式へのコンバータまたは変換フィルタ等を含むものである。これらはMathMLの表示形式タグを生成するが、全て完全なものではないため修正作業が必要である。同様に、ここで述べた意味記述形式への自動変換においても、表示形式だけからは完全にはMathML意味記述を生成できないなど、正しい表示形式のMathMLファイルを得るだけでも、誤り部分の後修正が必要となる点に改良の余地が残される。さらに、オプション部の属性指定により、表示される形式が変更されることで、数式の意味がまるで異なる式になるため、オプション部の自動解析は容易ではない。

現在のMathMLはVersion2.0（2001年2月）であるが、Version1.0と比較し意味要素が105個であったのが147個に増加し、かなり追加修正がなされている。また現状の版に至るまで、年に数回改定が公開されてきたため、次版でも大きな変更がなされる可能性もある。現状ではXMLがHTMLに代わるホームページ記述言語として普及していないため、XMLを認識する閲覧ソフトウェア（ブラウザ）が不足しており、Amaya, Mozilla, IBM techexplorerと少数で、それらの日本語化もなされていない。またブラウザにより表示される数式も、LaTeXの表示を見なれた者にとっては、バランスも悪く活字（フォント）自体も美しい表示とは言えない。MathML対応ブラウザであっても、その数式表示が貧弱であるため、添字等の活字のバランスが悪く、正しく表示されているか確認が取れないほど、MathMLに関してブラウザの対応は遅れている。

MathMLの定義書［3］や解説書等には、意味記述の有効性と応用ソフトウェアへの汎用性の高さ、視覚障害者への配慮が述べられているが、表示形式をサポートする支援ツールはあるが、意味形式記述用の応用ソフトウェアの実用例および支援ツールがほとんどない。MathMLの定義も全ての数式を記述できるものではないため、今後その仕様が強化されるであろう。これまでWeb上で、画像データとして表示されていたデータが、文字情報として記述されれば、スクリーン・リーダの読み上げで視覚障害者が理解できるが、記号の位置関係を記述した表示形式だけでは、正しい内容の理解をすることができない。ここではMathMLの意味記述への自動変換を与えたが、MathMLの仕様改定に合わせて、本プログラムに残された問題解決を図る必要がある。また意味記述への変換が行えるという利点は、他アプリケーションへの応用が可能なため、今後、本システムのMathML以外への応用も合わせて拡張する。

注

1.例えば、標準的な記述<ci>x</ci>と、オプションを含む<ci type="vector">v</ci>。

参考文献

[1] XML, http://www.w3.org/XML/

[2] W3C, http://www.w3.org/

[3] MathML, http://www.w3.org/pub/WWW/TR/REC-MathML/

[4] 野寺隆志：楽々LaTeX、共立出版、1990年

[5] Ohtake, N., Ogawa, Y. and Yonezawa, Y.：*Improved Accessibility System—Communication Environment for the Visually Handicapped,* Electronics and Comunications in Japan, Part 3, Vol.79, No.8, pp.56-63, 1996.

[6] Raman, T.V.：*Audio system for technical readings*, Thesis (PhD), Cornell University, 1994.

[7] Fukuda, R., Ohtake, N. & Suzuki, M.：*Optical recognition and Braille transcription of mathematical documents*, Proceedings of the 7th International Conference on Computers Helping People with Special Needs (ICCHP), pp.711-718, 2000.

[8] Dougherty, D. and Robbins, A：*sed & awk programming* (2nd. edition), O'Reilly & Associates, 1997.

（受理　平成14年8月13日）

13