Vol.2021-CE-161 No.2
2021/10/23

MathMLにおける一意な表現形式記述への変換ツールの開発

八巻 澄奈[1]　浅本 紀子[2]

概要：MathML とは数式を記述するマークアップ言語であり，意味形式記述と表現形式記述の 2 種類の記述方法がある．MathML の表現形式記述をサポートするブラウザは多くなってきているが，表現形式では同一の見た目に複数の記述方法が存在するため一意でない表現で記述できてしまい，数式の意味が正しく伝わらないという問題点が存在する．本研究ではこの問題点に着目し，見えない演算子が省略されて書かれた部分に適切な見えない演算子を加えるなどの適切な処理を行うことで，MathML の文書を正規的な記述に変換し出力するアプリケーションの開発を行った．この処理により，表現形式の利点を持ちながら正しい意味を伝える数式記述が可能となる．

1. はじめに

Web ページ上に数式を記述する方法として，これまでは，数式の画像を貼り付けるという方法がよく利用されていた．しかし，現在では，Web ページ上に直接数式を記述するという方法が多く利用されるようになった．画像を貼り付ける方法と比較すると，数式処理ソフトに入力して再利用するなどのことが可能になり，利便性が向上した．この方法のうちの一つが数式を記述するためのマークアップ言語，MathML である．MathML は HTML に埋め込むことで HTML 文書として扱うことができる．他の方法と比較した際の MathML の強みとしては，数学的表記における意味と外観の両方の情報がコード化できるという点が挙げられる．これにより，Web ページ上で数学的情報を公開・閲覧するだけでなく，処理をすることが可能になる．本研究では MathML の問題点を解決する Web アプリケーションを提案し，開発を行なった．

2. MathML

2.1 表現形式記述と意味形式記述

MathML には，表現形式記述と意味形式記述の 2 種類の記述方法が存在する．表現形式記述は表現に特化した記述方法であり，数式を表示することのみが重要である状況，例えば閲覧のみを目的とした Web ページに数式を掲載する場合等に便利である．一方で，意味形式記述は意味に特化した記述方法であり，数学的意味をコード化することが重要な状況で便利である．例えば，読者が数式をコピーして数式処理のソフトウェアにペーストして処理を行うといったことが可能になる．表現形式記述は多くのブラウザでサポートされているが，意味形式記述をサポートするブラウザは少ない．そのため，多くの Web ページでは表現形式記述で数式が記述されている．

2.2 表現形式記述の問題点

表現形式記述では同一の見た目に複数の記述方法が存在するため一意でない表現で記述できてしまい，数式の意味が正しく伝わらないという問題点が存在する．例えば，「$ax+1$」という見た目の数式には表 1 に示したように 3 通りの記述方法が存在する．A の記述方法では，乗算を表す演算子`&InvisibleTimes;`を記述することで「ax」が「$a \times x$」を意味することが明確に表現されている．B の記述方法では，`&InvisibleTimes;`が省略されており，「a」と「x」が乗算されているのかどうかが明確で無い．C の記述方法では，「ax」が 1 つの変数として扱われている．このように Web ページ上では同じ式に見えても実際の意味は異なるため，数式の意味が正しく伝わらないという問題が発生する．この問題が発生する演算子としては，`&InvisibleTimes;`の他にも表 2 に示したものが存在する．本研究ではこの問題点に着目し，見えない演算子が省略されて書かれた部分に適切な見えない演算子を加えるなどの適切な処理を行うことで，MathML の文書を正規的な記述に変換し出力するアプリケーションの開発を行った．

[1] お茶の水女子大学大学院 人間文化創成科学研究科 理学専攻 情報科学コース
[2] お茶の水女子大学大学

表 1　数式 $ax+1$ の記述方法

A	B	C
<mi>a</mi> <mo>⁢</mo> <mi>x</mi> <mo>+</mo> <mn>1</mn>	<mi>a</mi> <mi>x</mi> <mo>+</mo> <mn>1</mn>	<mi>ax</mi> <mo>+</mo> <mn>1</mn>

表 2　問題の原因となる演算子一覧

Unicode 名	コード番号	演算子	短縮形	数式の例
FUNCTION APPLICATION	U+2061	&InvisibleTimes;	&af;	ax
INVISIBLE TIMES	U+2062	&ApplyFunction;	&it;	$f(x)$
INVISIBLE SEPARATOR	U+2063	&InvisibleComma;	&ic;	a_{ij}
INVISIBLE PLUS	U+2064			$5\frac{7}{2}$

3. 関連研究・先行研究

3.1 関連研究 1

『数式記述言語 MathML の表現形式から意味形式への変換およびオンライン小テスト作問への応用』（内橋夏実，2021）[1]
表現形式を意味形式に変換するツールの開発を行なっている．さらに，コンバータツールの応用の例として，数式部分が表現形式で記述されている数学の問題から，LMS である Moodle 上の STACK を利用した数学オンラインテストの問題を自動生成するツールを開発している．本研究のツールと併用することで，より実用的な機能を実現することが可能となる．

3.2 関連研究 2

『視覚障害者学習支援のための MathML 変換』（渡辺千晶，2018）[2]
視覚障害者の数学学習支援のために，数式を含む文書を既存のスクリーンリーダーを用いて学習可能にするための文書変換アプリケーションの提案を行っている．TEX や表現形式の MathML では 2 通りの解釈ができてしまう数式が存在するため，これがスクリーンリーダーで読み上げを行う際の問題点となる．これを解決するための手法として，TEX 文書を意味形式の MathML に変換するという方法をとっている．2 通りの解釈がある数式を変換する際には，機械学習を用いて，前後の文脈から自動的に数式の意味を推定するという手法をとっている．本研究でも同様に，一意な数式にするための変換を行なっているが，その数式の対象範囲が異なる．この研究では，スクリーンリーダーで読み上げることを目的としているため，それぞれの解釈で読み上げる際の音が異なる場合のみに変換を行えば良い．これに対し，本研究では数式処理などに利用することを目的としているため，音が同じでも解釈が複数ある数式も対象として変換を行なっている．

3.3 先行研究

「MathML における一意な表現形式記述への自動変換」（渡辺裕美，2020）[3]
この研究では，一意な表現形式記述への変換を行う CUI のツールを開発している．一方で本研究では，これを GUI の Web アプリケーションに発展させることで，誰でも利用できる操作性の良いツールを開発した．また，先行研究では同じ変換ルールを適用できる複数の箇所があった場合に，それぞれに対して毎回ルールの選択を行なう仕様となっていたが，これを一括でルールを選択できる仕様に変更することで，変換効率を向上させた．

4. 教育分野への応用

3.1 の関連研究 1 で紹介した，表現形式を意味形式に変換するツールと本ツールと併用することで，一意でない表現形式の文書を正しい意味形式の文書に変換することが可能になる．この一連のシステムを利用することにより，Web ページに記載された数式を活用した数学教育ツールの開発など，教育分野への応用も期待できる．

5. 研究概要

5.1 変換対象

変換の対象とする数式の範囲は，高等学校までの算数及び数学の学習指導要領 [4] に記載されている範囲に含まれる数式とし，指導要領は平成 20 年及び平成 21 年に改訂されたものを参照する．具体的には，変換が必要になるケースには大きく分けて以下の 3 つのケースがある．本研究では 1 つ目と 2 つ目のケースに対応したアプリケーションを開発した．3 つ目のケースについては今後の研究で対応すべき課題としている．

5.1.1 変数

複数の文字が 1 つのタグにまとめて記述された場合には，それらを一つの変数として扱うか，複数の変数として扱うか，という問題が発生する．例えば「ab」という変数があった場合には，これを 1 つの変数とするか，「$a \times b$」という 2 つの変数の乗算の数式とするかという 2 つの解釈ができてしまう．後者の場合にはまず「a」と「b」という 2 つの変数に分けるという処理が必要になる．

5.1.2 見えない演算子

2.2 表現形式記述の問題点でも触れたように，MathML には「見えない演算子」が存在する．これらを使用した数式には表 3 で示したようなものが挙げられる．これらの式は，見えない演算子が記述されていない場合には，関数・

乗算・座標など複数の意味で取ることができてしまうので，必要に応じて見えない演算子の挿入を行う必要がある．

表 3　見えない演算子の使用例

ApplyFunction		InvisibleTimes	
コード	数式	コード	数式
<mi>sin</mi> <mo>⁡</mo> <mi>θ</mi>	$\sin x$	<mn>2</mn> <mo>⁢</mo> <mi>a</mi>	$2a$
<mi>f</mi> <mo>⁡</mo> <mo>(</mo> <mi>x</mi> <mo>)</mo>	$f(x)$	<mi>x</mi> <mo>⁢</mo> <mo>(</mo> <mi>x</mi> <mo>+</mo> <mn>1</mn> <mo>)</mo>	$x(x+1)$

5.1.3　複数の解釈が可能な数式

MathML には同一の見た目の数式で複数の解釈が可能な数式が存在する．高等学校学習指導要領に記載されている数式の中では表 4 に示した 6 つの数式がこれに当たる．例えば「f^{-1}」は「変数 f の－ 1 乗」と「関数 f の逆関数」の２通りの解釈が考えられる．

表 4　複数の解釈が可能な数式

数式	解釈 1	解釈 2
x'	微分式	記号 ′
$\frac{dy}{dx}$	微分式	分数式
A^t	転置行列	指数
e	ネイピア数	文字 e
!	階乗記号	エクスクラメーションマーク
f^{-1}	逆関数・逆元	指数式

5.2　変換方法

変換には，Python の WEB スクレイピング（HTML や XML ファイルからデータを取得し，解析する）用のライブラリである Beautiful Soup を用いた．変換の主な手順は以下の通りである．

(1) 変換対象の入力ファイルをユーザーから受け取り，MathML が記述された箇所を MathML ファイルとして抽出する．

(2) タグとそのテキスト部分を解析し，変換対象となる箇所を抽出する．

(3) ユーザーに変換ルールの選択を一括で求め，その結果をリストにする．

(4) 変換ルールのリストに基づき変換し，変換後の MathML ファイルとして出力する．

(5) 変換後の MathML ファイルを入力ファイルの MathML 箇所に置き換え，入力ファイルと同様の形式のファイルとして出力する．

（1）と（2）を実行するプログラムと（4）と（5）を実行するプログラムの 2 つを Python で作成した．（3）の処理は 1 つ目のプログラムから受け取ったデータをもとに ruby で実行し，ユーザーから受け取ったデータを 2 つ目のプログラムに引き渡している．

5.2.1　変数の変換

複数の文字が 1 つのタグにまとめて記述された変数の変換は次のような方法で行なった．

(1) 変数のうち，複数の文字が 1 つのタグにまとめて記述されているものを抽出する

(2) ユーザーが一つの変数として扱うか，複数の変数として扱うかを選択する

(3) 複数の変数として扱う場合にはそれぞれの変数の頭文字を選択する

(4) 変換ルールのリストに基づき変換する．

例として，「abc」という変数を含むファイルを変換した際には，図 1 のような画面が表示される．ここでは「abc」を複数の変数として扱うこととし，変数の頭文字は「a」と「c」とした．この結果，変換前と変換後のコードは表 5 のようになった．

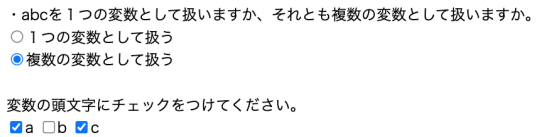

図 1　ルール選択画面

表 5　変換前後の比較

変換前	変換後
<mi>abc</mi>	<mi>ab</mi> <mi>c</mi>

5.2.2　見えない演算子の変換

見えない演算子を含む数式の変換は次のような方法で行なった．

(1) 見えない演算子が省略されている数式を抽出する．

(2) ユーザーがどのような解釈で変換するかを選択する．

(3) 変換ルールのリストに基づき，適切な見えない演算子を挿入する．

例として，「$f(x)$」という数式を含むファイルを変換した際には，図 2 のような画面が表示される．ここでは関数を表す式として扱うこととした．この結果，変換前と変換後のコードは表 6 のようになった．

・f(x)の数学的意味を１つ選択してください。
◉関数
○乗算

図 2 ルール選択画面

表 6 変換前後の比較

変換前	変換後
<mi>f</mi> <mo>(</mo> <mi>x</mi> <mo>)\</mo>	<mi>f</mi> <mo>⁡</mo> <mo>(</mo> <mi>x</mi> <mo>)\</mo>

5.3 GUI

本ツールをより利用しやすいものにするため，GUI を次のような設計とした．ルール選択時，同一のルールを適用できる箇所が複数ある場合には，一括で同一ルールを選択したり，そこから特定の箇所だけ別のルールを選択したりできるようにした．例として，「abc」という変数を複数箇所に含むファイルを変換した際には，図 3 のような画面が表示される．ここでは 1 ・ 3 箇所目では「abc」を複数の変数として扱うこととし，変数の頭文字は「a」と「c」とした．２箇所目では 1 つの変数として扱うこととした．この結果，変換前と変換後のコードは表 7 のようになった．この例では使用していないが，「全て選択」ボタンを押下することで一括で同一ルールを選択することも可能である．

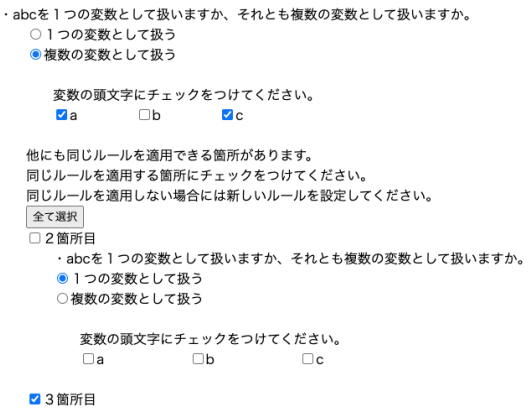

図 3 ルール選択画面

表 7 変換前後の比較

変換前	変換後
<mi>abc</mi> <mi>abc</mi> <mi>abc</mi>	<mi>ab</mi> <mi>c</mi> <mi>abc</mi> <mi>ab</mi> <mi>c</mi>

6. まとめと今後の課題

本研究では，表現形式記述の MathML を正規的で一意な記述に変換する Web アプリケーションの開発を行なった．現段階では変換を行うケースは「変数」と「見えない演算子」のみとなっているので，今後「複数の解釈が可能な数式」についても変換を行えるようにする必要がある．また，現在は文書ファイルのみを入力として受け入れているが，今後は URL からも変換が行えるようにし，利便性の拡張を目指したい．

参考文献

[1] 数式記述言語 MathML の表現形式から意味形式への変換およびオンライン小テスト作問への応用，内橋夏実，お茶の水女子大学大学院　人間文化創成科学研究科　理学専攻　情報科学コース，2021 第 161 回コンピュータと教育研究発表会 発表予定
[2] 視覚障害者学習支援のための MathML 変換，渡辺千晶，お茶の水女子大学大学院　人間文化創成科学研究科　理学専攻　情報科学コース　修士論文，2018
[3] MathML における一意な表現形式記述への自動変換，渡辺裕美，お茶の水女子大学大学院　人間文化創成科学研究科　理学専攻　情報科学コース　修士論文，2020
[4] 文部科学省（2012）「学習指導要領」<https://www.mext.go.jp/a_menu/shotou/new-cs/youryou/1356249.htm>（2021/9/2/閲覧）