D-024

アプリケーション開発における要素技術としての利用に着目した数式検索手法の提案

A Search Method for Mathematical Expressions as a Component of Application Software

渡部 孝幸† 宮崎 佳典‡
Takayuki Watabe Yoshinori Miyazaki

1. はじめに

数式は，数理的な手法を用いる様々な学問領域において，概念を数学的に表現するために利用されており，紙媒体の書籍等に記載されることはもちろん，計算機上においても広く用いられている．計算機上での数式利用の目的は，2 つに大別される．1 つは，数式の表示である．Web ページや電子書籍，pdf 等において文書中に数式を記述するといった利用方法がこれに該当する．もう 1 つは，数式に対して，数学的な処理を行うようなものである．例えば，数式に対して代数的な変形を適用する数式処理システムや，数学学習者に対して演習問題を出題しその解答として数式の入力を受け付けるシステム[1]などが挙げられる．計算機上での数式利用が進むにつれ，数式データの量も増している．数式データを多数集積したデータセット[2]も公開されている．これらの数式データを活用するためには，数式を検索するための技術が不可欠となる．そこで本論文では，数式を効果的に検索するための手法を提案する．

数式検索には，いくつかの用途が想定される．典型的には，数式を問い合わせとして，数式が記述された文書の集合から目的の文書を検索するという方法が考えられる．他には，数式が記述された単一の文書から，特定の数式が記述された箇所を探すという，テキストにおける文字列探索に類似するような方法も考えられる．これらの手法は，ユーザが直接，問い合わせを記述することを想定したものである．この他に，検索という処理を，より高次の処理を実現するための要素技術として利用することも想定される．具体的には，例えば，システムの内部に静的な問い合わせを用意しておき，入力された数式がその問い合わせを満たすか否かを判別するフィルタリング処理や，数式のある特徴を問い合わせの形で表現しておき，数式が問い合わせを満たす場合にはその特徴を持っていると判断するという，情報抽出などが挙げられる．このような処理の要素技術として利用するという用途においては，ユーザが直接問い合わせを記述するような検索で用いられるアプローチが最適であるとは限らない．本論文における数式検索は，上記のうち，数式を利用したアプリケーションを開発する際の要素技術として特に効果を発揮するようなものとする．

要素技術としての数式検索を開発する上でいくつか，重視すべき点がある．まず，検索対象となるデータ形式を適切に決定する必要がある．計算機上で数式を表現するために複数のデータ形式が提案されており，それらは，数式の見た目を指定するためのものと，数式の意味を指定するためのものの 2 種類に大別できる．例えば，x^2という数式は，見た目を指定する形式では，「x の右上に 2 が配置されている」と表現され，意味を指定する形式では，「底 x，指数 2 に対して，べき乗を適用する」と表現される．数式の変形や数式からの情報抽出など，数式に対して数学的な処理を行う場合には，意味の情報が必要となる場合が多い．これは，数式が多義性を持つためである．例えば，$f(x)$という数式は，fとxの掛け算であるとも，変数xに対する関数fの適用であるとも解釈できる．このため，数式の見た目を指定する情報からは，数学的な意味を解釈できない場合が多い．したがって，数学的な処理を実現するための要素技術としての数式検索は，意味の情報で表現された数式を検索の対象とすべきである．本論文では，意味を表現するためのデータ形式として，W3C によって標準化されている MathML Content Markup を対象とする．

次に，問い合わせ言語の一貫性を重視する．そもそも，一般的な数式の記法は，一貫性に欠ける場合がある．例えば，aとbの加算を，加算演算子を用いた 2 項演算として記述する場合は$a+b$となり，2 引数をそれらの和へと写す 2 変数関数として記述する場合は$f(a,b)$となる．これら 2 つの表現を見ると，演算とその適用対象の記述の順序が一貫していない．このため，本研究では，数式の記法にしたがうのではなく，S 式を基礎とした問い合わせ言語を設計することで，一貫性を実現する．問い合わせ言語の一貫性によって，問い合わせを自動的に生成するような処理を実現することが容易になる．この特性は，要素技術としての数式検索において非常に重要なものである．一方，純粋に一貫性のみを追求すると，問い合わせが冗長になるという欠点がある．そこで，省略記法を豊富に用意することにより，一貫性を維持しつつ，必要に応じて冗長さを低減させることができるような問い合わせ言語を実現する．問い合わせ言語に関しては，表現力も重要な要素である．例えば，数式においては，xとθが共に変数を表現したものであると考えた場合，$\sin x$と$\sin\theta$は，本質的に同じ意味を表現していると捉えられる．そのため，$\sin x$と$\sin\theta$を同時に取得できるような問い合わせを記述できることが望ましい．我々は，このような，数式検索に特有の問い合わせも行うことができるような問い合わせ言語を提案する．

最後は，互換性である．様々なアプリケーションの要素技術として利用できるためには，多様な環境で動作する必要がある．これを実現するために，本論文では，マッチング処理に XQuery を利用する．XQuery は，W3C により策定されている XML のための問い合わせ言語である．XQuery を処理するエンジンは様々なプラットフォームに存在し，多くの XML 向けデータベース管理システム(DBMS)にも備えられている．つまり，マッチング処理に

† 静岡大学大学院自然科学系教育部 Shizuoka University, Graduate School of Science and Technology
‡ 静岡大学大学院情報学研究科 Shizuoka University, Graduate School of Informatics

XQuery を用いると，我々の策定する数式検索用問い合わせ言語から XQuery による問い合わせ言語へと変換する単純なフロントエンドプログラムを複数のプラットフォームに対して用意するだけで，本手法を広範に利用することが可能となる．

本論文の構成を示す．2 章で，本研究で対象とするデータ形式である MathML Content Markup について概説する．3 章で数式検索に関する先行研究について述べる．4 章で本研究で提案する数式検索のための問い合わせ言語である cmmlQ について詳述する．5 章で，cmmlQ による問い合わせを XQuery に変換する手順を示す．6 章で本検索手法について評価を試みる．7 章で本論文を総括し，今後の展望を示す．

2. MathML Content Markup

MathML Content Markup は，W3C によって策定された XML ボキャブラリである．XML ボキャブラリであるため，データはタグを用いて表現される．また，データは木構造に一対一で変換できる．以下，親，子，兄弟など，木構造の用語によってデータを表現する．

MathML Content Markup では，apply 要素という要素を 1 つの単位として数式を記述していく．apply 要素の第一の子が演算，第二以降の子が演算対象となる．なお，apply 要素が子として apply 要素を持つことも可能である．例えば，$2 + \sin x$という数式は，次のように記述される．

```
<math>
    <apply>
        <plus/>
        <cn>2</cn>
        <apply>
            <sin/>
            <ci>x</ci>
        </apply>
    </apply>
</math>
```

上の例を木構造で表現すると図 1 のようになる．

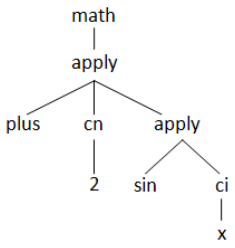

図 1　木による$2 + \sin x$の表現

上の例に出現するそれぞれの要素について説明する．apply 要素は演算子や関数の適用を示し，はじめの子をそれ以降の子に対して適用するという効果を持つ．plus 要素は加算を意味する．plus 要素は<plus/>と表記されているが，これは子を持たない要素を表す記法であり，<plus></plus>と同じものと考えて良い．sin 要素は正弦関数を意味する．また，cn 要素は数，ci 要素は識別子を，それぞれ表す．

3. 先行研究

数式検索に関する研究は数多く行われている．本章では，それらのうちのいくつかについて述べる．

まず，本論と同じく，数式の意味の側面に着目した手法について述べる．[3]は，問い合わせと検索対象のマッチングを行い，検索結果を順序付けする手法を提案している．問い合わせ言語に注目すると，演算子を'+', '-'といった記号で表現する．このような形式は，可読性には優れるが，論理演算や集合演算を表現する際に，ASCII 文字だけでは四則演算と共通した記法を用いることができないため，多様な数式に対して一貫した記法で問い合わせを記述することができない．[4]で提案されている手法は，MathWebSearch という数式を問い合わせとして論文を検索するシステムとして実装および公開されている．検索は基本的に部分一致検索によって行われる．問い合わせの記述方法は豊富に用意されており，GUI による入力をはじめ，LaTeX や Maxima による記法にも対応している．一方，機能の面では，x や y などの識別子を任意の数式とみなすといったワイルドカードに相当するもののみしか用意されておらず，複雑な問い合わせを記述することは難しい．[5]では，検索システムを実装することを重視し，LaTeX を拡張するという方法で問い合わせ言語を設計している．キーワードを用いた簡易なシソーラスも提案している．[6], [7], [8]は，数式の意味の側面に着目しつつ，類似検索を行う方法について述べている．[6]は，数式における演算子や変数の位置に着目した索引を作成することで，数式の構造を反映するという類似検索を提案している．我々の研究と同じく，問い合わせの数式を表現するために S 式を用いている．[6]では数式そのものを S 式として記述するが，我々の研究では，S 式を拡張し，数式そのものに加えてワイルドカードや論理和といった情報も表現する．[7]では，数式から得られる木構造のパスの集合を用いて，数式間の類似度を算出する．[8]は，Latent Semantic Indexing (LSI)を用いた数式の類似検索手法を提案している．LSI を適用する上で，MathML Content Markup による数式データを，タグの出現頻度に着目したベクトルとして表現している．

次に，数式の見た目に着目した手法について述べる．[9]による方法では，n-gram を用いた検索を行う．n-gram は自然言語処理の分野においては連続する n 文字または n 単語のまとまりを指すが，この概念の数式に対する定義を試みている．[10]では，MathML Presentation Markup (数式の見た目に関する情報を記述するための，MathML のサブセット)によるデータから得られる木構造のパスの一部に着目することで，数式を特徴付ける方法が述べられている．[11]は，数式の見た目を表現するためのデータに対して，意味的な構造を考慮したパターンマッチングを実現することを目指している． [12]は，分数やべき乗といった，数式の持つ記号の二次元的な位置構造を，一次元的な文字列の入れ子構造であると捉えることで，テキストのパターンマッチングに類似するマッチングアルゴリズムを提案している．

4. 問い合わせ言語 cmmlQ

我々の提案する数式のための問い合わせ言語である cmmlQ (Content MathML Query)について述べる．はじめに概要を述べた後，問い合わせを一貫した方法で記述するた

めの記法である条件記法と，問い合わせを簡潔にするための省略記法を示し，最後に cmmlQ の設計の根拠や利点について述べる．

4.1 概要

問い合わせは，S 式を用いて記述する．S 式によって演算と演算対象の組を入れ子構造として表現することで，複雑な数式を表現することが可能である．例えば，$x^2 + \sin y$ を検索するための問い合わせは，次のように記述する．

```
(plus (power x 2) (sin y))
```

演算対象の役割は，演算対象を記述する順序によって指定する．例えば，べき乗の場合は，S 式の 2 番目の要素が底，3 番目の要素が指数となる．対数や微分などについても，役割と順序の対応が設定されている(その一覧は省略する).

問い合わせ言語としての機能には，ワイルドカード，部分一致，論理和の 3 つを用意している．

ワイルドカードは，任意の数式を意味するものであり，アスタリスク(*)で表現される．例えば，「平方根を任意の対象に対して適用する」という問い合わせは，次のように記述できる．

```
(root *)
```

この問い合わせには，$\sqrt{2}$や$\sqrt{a+b}$，$\sqrt{1/x}$ などがマッチする．

部分一致は，指定された数式を含むような数式を問い合わせるためのものである．数式を波括弧('{', '}')で囲んで表現する．例えば，「$\sqrt{2}$を含む」という問い合わせは，次のように記述できる．

```
{(root 2)}
```

この場合，$a+\sqrt{2}$や，$\sqrt{2}/x$ などの数式がマッチする．また，「a と，内部に 2 を含む平方根との和」のような問い合わせも可能である．これは，次のように記述される．

```
(plus a (root {2}))
```

この問い合わせには，$a+\sqrt{x-2}$や，$a+\sqrt{b^2}$ などがマッチする．

論理和は，「A あるいは B」という形で問い合わせを記述するためのものである．{A|B}のように記述する．例えば，「sin あるいは cos を x に対して適用する」という問い合わせは，次のように記述できる．

```
({sin|cos} x)
```

この問い合わせには，$\sin x$あるいは$\cos x$がマッチする．

さらに，演算および演算対象に対して詳細な情報を付与するために，条件という機能を用意している．条件は角括弧('[', ']')を用いて表現し，オペランドあるいはオペレータの直後に記述する．角括弧の中には，「条件名=値」という形式で，条件とその値との組を記述する．条件の例として，識別子(x, θ, a, fといった，変数，定数，関数を表すために用いられる記号)の種別が挙げられる．この条件は，オペランドに対して付与することができ，変数，関数，実数，複素数，といった識別子の種別を指定することができる．例として，xが複素数であるような$\sin x$は，次のように問い合わせることができる．

```
(sin x[type="complex"])
```

4.2 一貫した記法—条件記法

4.1 節では，演算の種類を指定するために plus や sin といった文字列を S 式中に直接記述した．また，部分一致，論理和，ワイルドカードを実現するために，特別な記法を用いた．これらは，省略記法を用いて記述した問い合わせである(省略記法については 4.3 節で後述する)．省略記法を用いない場合には，S 式と条件の 2 つの記法のみを用いることで，問い合わせを記述する．以後，S 式と条件のみを用いる記法を条件記法と呼ぶ．例えば，(plus (power x 2) (sin y))は，条件記法では次のように記述される(読みやすくするためにインデントしている).

```
([operation="plus"]
 ([operation="power"]
   [class="identifier" content="x"]
   [class="number" content="2"])
 ([operation="sin"]
   [class="identifier" content="y"]))
```

上の例を見るとわかるように，演算の種別は条件 operation の値として指定されている．また，演算対象の文字は条件 content の値となっている．このように，条件記法では，S 式を用いて数式の構造を表現し，演算と演算対象に関する情報はすべて，条件で記述される．

条件は，条件の名称と値を指定するという形で記述されるが，条件が取ることのできる値の種類を型と呼ぶ．型には，真理値型，文字列型，演算型，演算対象型の 4 種類がある．真理値型は，true あるいは false の 2 つの値を持つ．真理値型の条件のデフォルト値は false である．文字列型の値は文字列であり，これは二重引用符に囲まれた形で記述される．条件によっては，値として取ることのできる文字列が決まっている場合がある．演算型と演算対象型の値は，それぞれ，cmmlQ によって記述された演算および演算対象である．

条件に関しては，型の他に，対象という概念も存在する．条件の対象は，演算か，演算対象のいずれかである．条件が S 式の第一引数として記述された場合，その条件は演算を対象としている．第二以降の引数として記述された場合は，演算対象を対象としている．本節冒頭の例においては，条件 operation の対象は演算であり，条件 type, content の対象は演算対象である．条件によっては，演算あるいは演算対象のどちらか一方としてしか指定できないものがある．また，同じ条件でも，対象が演算であるか演算対象であるかによって，値の型が異なる．

表 1 に，条件の一覧を示す．それぞれの条件について，名称，対象，型，およびその条件の役割を示す．型が文字列型である場合は，取りうる値の詳細も併記する．なお，条件によっては，1 つの条件に対して複数の値を設定することができる(値をカンマで区切って記述する)．表 1 では，条件が複数の値を取ることができる場合，それを明示する．

表 1　条件の一覧

条件名	対象	型	役割
	取りうる値(型が文字列の場合のみ記述)		
operation	演算	文字列	演算の種別を指定
	MathML3.0 の仕様書[12]のうち，A.2.5 The Grammar for Content MathML で定められている，演算を意味する要素名．		

useroperation	演算	文字列	演算の種別を指定
	任意の文字列.		
commutative	演算	真理値	オペランドの順番を無視する(true)
class	演算対象	文字列	演算対象の分類を指定
	identifier , number , symbol , arbitrary, tuple の 5 種類の値のいずれか. identifier, number, symbol は，それぞれ，識別子，数，シンボル(円周率π や実数集合$\mathbb{R}$など，数学的概念を表現する記号)を意味する. arbitrary は，任意の演算対象であることを意味する. tuple は，演算と演算対象との組であることを意味する.		
content	演算対象	文字列	演算対象の内容を指定
	任意の文字列. ただし，条件 class が symbol に設定されている場合は，MathML3.0 の仕様書における 4.4.10 Constant and Symbol Elements に基づく，数学的概念に対応するキーワード(e.g., 円周率の場合は pi, 実数集合の場合は reals).		
type	演算対象	文字列 (1 つ以上)	識別子あるいは数の種別(e.g., 実数，複素数，ベクトル)を指定
	MathML3.0 の仕様書 4.2.2 Content Identifiers において，type Attribute Values として指定可能な文字列.		
or	演算	演算 (2 つ以上)	条件 or が指定された演算は，その値のうちのいずれかを表す
	演算対象	演算対象 (2 つ以上)	条件 or が指定された演算対象は，その値のうちのいずれかを表す
partial	演算, 演算対象	真理値	部分一致を適用する(true)

4.3 省略記法

条件記法を用いて問い合わせを記述すると冗長になるため，省略記法を用意している(4.1 節で示した問い合わせは，省略記法を用いた問い合わせである)．省略記法による記述はすべて条件記法に置き換えることができる．

表 2 に，省略記法と，それに対応する条件記法を示す．

表 2　省略記法の一覧

記法の概要		
省略記法	対応する条件記法	省略記法の例
オペレータとして演算の名称を直接記述する (MathML Content Markup で定義されている演算名)		
value	[operation="*value*"]	plus
オペレータとして演算の名称を直接記述する (MathML Content Markup で定義されていない演算名)		
value	[useroperation="*value*"]	f
オペランドとして識別子(e.g., x, a, θ)を直接記述する		
value	[class="identifier" content="*value*"]	x
オペランドとして数を直接記述する		
value	[class="number" content="*value*"]	2
オペランドとして，シンボル(e.g., 円周率π，実数集合$\mathbb{R}$)の名称を直接記述する		
value	[class="symbol" content="*value*"]	pi
真理値型の条件で値が true である場合は，条件名のみを記述する		
[*conditionname*]	[*conditionname*=true]	[commutative]
論理和には記法{A|B}を用いる		
{*value1*|*value2*|*value3*|...}	[or=*value1*,*value2*,*value3*,...]	{sin|cos|tan}
部分一致には記法{}を用いる		
{ *operand* }	[partial=true]	{(sin x)}
任意のオペレータは*で表現する		
*	[class="arbitrary"]	*

4.4 cmmlQ の設計について

本章では，ここまで，cmmlQ の仕様について説明した．本節では，cmmlQ の仕様について，そのように設計した理由や利点について述べる．

4.4.1 S 式と条件

cmmlQ では，S 式を用いて演算と演算対象の組を表現し，角括弧を用いた条件の記法を用いて，演算および演算対象そのものに関する情報を表現する．一方， cmmlQ において条件を用いて表現した情報も，すべて S 式を用いて記述するような問い合わせ言語に設計することも可能である．本節では，あえて S 式と条件を別個に扱った理由を述べる．

cmmlQ では，例えば「複素数 x と複素数 y の和」は，(plus x[type="complex"] y[type="complex"])と表現される．これを S 式のみを用いて表現するような設計にした場合，例えば(plus (type x complex) (type y complex))のような問い合わせが考えられる．このような設計にすると，構文が S 式に一本化されるため，構文解析が容易になるという利点がある．しかしあえて条件という記法を導入した理由は，数学的な演算と演算対象を表現する機構とそれ以外の情報を表現する機構とを区別することで，問い合わせの解釈が容易になると考えたためである．前述の，「複素数 x と複素数 y の和」を S 式で表現した問い合わせである(plus (type x complex) (type y complex))について考える．この問い合わせのうち，plus は数学的な演算であり，type は識別子の情報を示すためのものであるため，その性質は大きく異なる．しかし，S 式

を見る限りでは，その違いを一目で判別することができない．この欠点は，cmmlQ を利用するユーザにとって，cmmlQ の学習を困難にするものであると考えられる．したがって，本研究では，S 式は数式を表現するために用い，検索のための情報は条件として表現するという方法を採った．

4.4.2 条件記法と省略記法

本論文では，同一の問い合わせを記述するために，条件記法と省略記法という 2 種類の記法を用意した．この理由は，問い合わせの一貫性と可読性を両立するためである．1 章で述べたように，本検索手法は，アプリケーション開発等の要素技術として利用されることを想定している．つまり，アプリケーションによって問い合わせが自動的に生成されるような用途も視野に入れている．このような処理の実装を行う上で，問い合わせ言語に一貫性があると，実装が容易になると期待できる．このため，条件記法という一貫性を持つ記法を策定した．一方，数式のフィルタリング処理などを実装する上では，人間の手で問い合わせを直接記述すると推察される．このような用途においては，問い合わせの可読性ないし記述容易性が要求される．これに応えるために，省略記法を用意した．

5. cmmlQ から XQuery への変換

本章では，cmmlQ から XQuery への変換方法について述べる．本検索手法が対象とする数式データ形式である MathML Content Markup，および XQuery について概説した後，変換の手順を示す．

5.1 XQuery

本論文では XQuery のうち，XPath と呼ばれるサブセットを主に利用する．本節では，XPath の基本的な動作と，本研究で利用する，述語と呼ばれる記法について述べる．以後，図 1 の MathML Content Markup のデータを例に説明する．

XPath では，XML データの木構造を根から順番に辿っていくことで要素を取得する．例えば，図 1 に示したデータから<sin/>を取り出したい場合，次のように記述する．

```
math/apply/apply/sin
```

これは，XML データを根から順番に，math, apply, apply, sin という順番で辿るということを表現している．

XPath には，述語と呼ばれる記法がある．述語は，要素に対して条件を指定するという役割を持つ．述語を記述した場合，述語で指定された条件を満たす要素のみを取得する．例えば，「math 要素直下の，plus 要素を子に持つ apply 要素」は，次のようにして取得する．

```
math/apply[./plus]
```

この場合，取得されるのは plus 要素ではなく apply 要素である．なお，ドット記号'.'は，述語が指定されている要素を意味する．つまり，上の例では，ドットは `math/apply` によって取得される要素を意味する．述語を用いることで，ある要素が sin 要素であるか否か，といった問い合わせだけでなく，ある要素が x を演算対象とする sin 要素か否か，といった，その要素の周辺の要素にまで着目した問い合わせを記述することが可能となる．これによって，cmmlQ において主に S 式を用いて表現された数式の構造を，XQuery を用いて表現することが可能となる．詳細は省略するが，述語においては，if 文や関数なども利用することができる．このため，cmmlQ における論理和や可換則といった検索の機能も，述語によって実現することができる．

定型的な表現として，「n 番目の要素がある特定の要素である」ことを表すための述語を用いる．例として，「2 番目の要素が ci 要素である」という述語は，次のように記述する．

```
[./node()[2]/self::ci]
```

`node()`は，要素のすべての子を取得するという役割を果たす．次に，`[2]`によって，そのうちの 2 番目のもののみが取得される．その次の `self::`は軸と呼ばれる記法であり，これによって自分自身を指定することができる．したがって，この場合，要素の 2 番目の子が参照される．軸 self の後には ci 要素が指定されている．このため，もし 2 番目の子が ci 要素であればその要素が取得されるため述語は真となり，そうでなければ何も取得されないため述語は偽となる．これにより，「2 番目の要素が ci 要素である」ということを表現している．

5.2 変換の手順

cmmlQ から XQuery へと変換された問い合わせは，XPath における述語となる．これを，MathML Content Markup の math 要素の述語として記述することで，その math 要素が表す数式が cmmlQ による問い合わせを満たすか否かを判定することができる．

cmmlQ から XQuery への変換は，cmmlQ に対応する XQuery の雛形を用意しておき，条件に応じて雛形の一部を書き換えるという方法で実現する．雛形は，演算を表現するためのものと，演算対象を表現するためのものが存在する．演算対象の雛形には，識別子や数を表現するための雛形と，演算と演算対象との組を表現するための雛形がある．以下，前者を原子，後者を組と呼ぶ．すなわち，演算，原子，組の 3 つの雛形が存在する．例外的に，条件 or (論理和)や条件 commutative (可換則)などの一部の機能は，3 つの雛形を条件に応じて書き換えるという方法では実現できない．これらについては，その機能のための特別な雛形を用意し，それを書き換えるという方法を取る．これらの特別な雛形については，論理和についてのみ示す．

以下，本節では，cmmlQ による記述に対応する XQuery の雛形を示す．また，条件によって雛形がどのように書き換えられるかについても述べる．雛形のうち，条件によって内容が変化する箇所はイタリック体で示す．条件によっては記述されない可能性がある箇所は下線を引いて示す．なお，cmmlQ から変換して得られる XQuery の具体例およびその動作の詳細は，本論文の最後に付録として掲載する．

5.2.1 演算の雛形

演算の雛形は，次の通りである．

```
[./node()[1]/self::operation]
```

operation は，条件 operation が設定されている場合，その値となる．条件 useroperation が設定されている場合，`ci[.="useroperation"]` となる (*useroperation* は条件 useroperation の値となる)．条件 operation および条件 useroperation が指定されていない場合は，`node()`となる．

5.2.2 原子の雛形

原子の雛形は，次の通りである．

```
[./node()[n]/axis::class[.="content"]
                        [./@="type"]]
```

n は，原子が何番目の演算対象であるかに応じて書き換えられる．原子が x 番目の演算対象である場合，*n* は x-1 となる．

axis は，条件 partial の値が true である場合は `descendant-or-self`，false である場合は `self` となる．

class は，条件 class の値が identifier である場合は `ci`，number である場合は `cn` となる．symbol である場合は，条件 content の値となる．条件 class の値が symbol であり，かつ条件 content が指定されていない場合，class は `node()[fn:local-name() != "ci" and fn:local-name() != "cn" and fn:local-name() != "apply"]` に書き換えられる．この XPath は，「ci でも cn でも apply でもない要素」を表すものである．条件 class が指定されておらず，条件 content が指定されている場合は，class は `node()` となる．条件 class も条件 content も指定されていない場合は，`node()[fn:local-name() != "apply"]` となる．この XPath は，「apply ではない要素」を表す．

content は，条件 content の値となる．ただし，条件 class の値が symbol である場合は，content を含む下線部は記述されない．また，条件 content が指定されていない場合にも，content を含む下線部は記述されない．

type は，条件 type の値となる．条件 type が記述されていない場合には，type を含む下線部は記述されない．

5.2.3 組の雛形

組の雛形は，次のようになる．

```
[./node()[n]/axis::apply
  operator
  operand
  ...
]
```

n および *axis* の書き換え規則は，5.2.2 節で述べた原子の書き換え規則と同様である．

operator は，5.2.1 節で述べた方法で得られる，演算を表現するための XPath に書き換えられる．

operand および ... は，5.2.2 節で述べた方法で得られる原子の XPath か，本節で述べた方法で得られる組の XPath に書き換えられる．演算対象が複数存在する場合は，その数だけ記述される．

5.2.4 特別な雛形(論理和)

演算，原子，組の 3 つの雛形を書き換えるという方法では実現できない機能については，特別な雛形を用意することで対応する．ここでは，特別な雛形のうち論理和について述べる．論理和の雛形は次のとおりである．

```
[.
  value
  or .
  value
  or .
  ...
]
```

value は，条件 or の値である問い合わせを変換して得られる XPath となる．値が 3 つ以上設定されて場合は，その数に応じて，条件値から得られる XPath と `or` が繰り返し記述される．これは，条件 or が演算として記述されている場合も演算対象として記述されている場合も同様である．

6. 評価

本手法の評価の方法について述べた後，本手法に関して，数式の問い合わせ言語としての表現力と，MathML Content Markup の問い合わせ言語としての表現力という 2 つの観点から評価する．

6.1 評価の方法

本研究における評価の方法について述べる．数式検索を含む検索処理の評価には，様々な方法がある．評価の方法を決める大きな要因として，検索結果の良し悪しの判断がユーザの主観に依存するか否かという点がある．ユーザの主観に依存する場合，ユーザによる評価に基づいて再現性や適切性を算出するといった定量的な評価が意味を持つ．一方，それぞれ検索対象について，問い合わせを満たすか否かを客観的な基準の元で決定できる場合，再現性や適切性を算出することは，実装したプログラムに誤りがないことを検証しているに過ぎない．

ここで，本検索手法について考えると，本手法では，検索対象である MathML Content Markup のデータが問い合わせを満たすか否かは，cmmlQ の仕様という客観的指標に基づいて決定することが可能である．このため，本手法については，定量的な評価は行わない．

本手法では，検索対象が問い合わせを満たすか否かを客観的に判断することができると述べたが，このような検索においては，”何を”問い合わせることができるのかという点，つまり問い合わせ言語の表現力が重要となる．そこで，評価として，問い合わせ言語 cmmlQ の表現力を検討する．ここで，cmmlQ は数式のための問い合わせ言語であるが，検索する数式は MathML Content Markup データで表現されている．このため，cmmlQ の表現力は，数式の問い合わせ言語としての表現力と，MathML Content Markup の問い合わせ言語としての表現力に分けて考えるべきである．以下，これら 2 つを区別して述べる．

6.2 数式に対する表現力

問い合わせることのできる演算の種類，および数式の問い合わせ言語としての機能という 2 つの観点から述べる．

数式における演算には，四則演算や，三角関数などの初等関数，集合演算など，様々な演算が用いられる．また，等号や不等号，包含といった関係も数多く存在する(本論文では，関係も含めて演算と呼ぶ)．このため，数式検索を実現する上では，これらの多様な演算を問い合わせることのできる能力が要求される．

cmmlQ では，演算の種別を条件 operation の値として表現する．この値には，表 1 に示したように，MathML3.0 で定義されている演算を指定することができ，具体的には前述した四則演算，初等関数，集合演算，論理演算，等号・不等号といった演算をはじめ，微積分や論理演算など，様々な演算が含まれている．つまり cmmlQ は，MathML Content Markup の表現可能な演算に立脚した多様な問い合わせを実現する．

次に，cmmlQ の機能を検討する．数式には，記法に独特の規則がある．例えば，数式においては，記号の違いは本質的には無視される．つまり，$\sin x$ という数式も，$\sin \theta$ という数式も，本質的に同じ意味を表す．別の例として，数式には等価性という性質がある．つまり，$\sin x + \cos x$ と

いう数式と，$\cos x + \sin x$という数式は，等価なものであるとみなされる．これらの数式の特徴を数式検索にも反映させるという点については，前者の例については[9]，後者については[5]で指摘されている．

cmmlQ を用いることで，このような問い合わせも記述することができる．識別子の違いの同一視については，演算対象の条件 class によって実現できる．上で例示した，$\sin x$と$\sin \theta$を両方とも検索するという処理を行うためには，(sin [class="identifier"])という問い合わせを記述すれば良い．さらに，条件 class の値を number にすれば，sin の引数が数であるようなもののみを問い合わせることができ，値を arbitrary にすれば，$\sin(a+b)$といった，sinの引数が数式を構成しているようなものまで取得することができる．可換則ついては，条件 commutative で対応している．$\sin x + \cos x$と$\cos x + \sin x$を両方とも取得するような問い合わせは，(plus[commutative] (sin x) (cos x))によって実現できる．なお，等価性については，可換則に基づくもののみではなく，例えば$1+1$と2の等価性を考慮するような機構を備えることで，利便性が向上する可能性がある．これについては 7 章で後述する．

このように，cmmlQ は，多様な演算を問い合わせることができ，識別子の同一視や可換則への対応といった数式に特有の規則にも対応していると言うことができる．

6.3 MathML Content Markup データに対する表現力

MathML Content Markup では，変則的なデータが記述される可能性がある．そのようなデータに対する cmmlQ の対応について述べる．

変則的なデータとして，ci 要素による演算の表現が挙げられる．2 章で示したように，MathML Content Markup では，演算は apply 要素の第一の子として記述される．この際，正弦関数 sin のような，数学的に定義されている演算の場合は，<sin />というように，その演算を表すための要素が用意されている．一方，データの記述者が定義した関数 f のような演算の場合は，<ci>f</ci>のように，ci 要素を用いて識別子として表現される．ここで，演算を<ci>sin</ci>と表現した場合，データ記述者が定義した sin という名称の関数であるということになる．これは，記述される可能性が低い，変則的なデータであると言えるが，問い合わせ言語としては，このようなデータにも対応していることが望まれる．cmmlQ を用いてこのようなデータを問い合わせるためには，条件 useroperation を用いる．例えば，[operation="sin"]と記述した場合には<sin/>を，[useroperation="sin"]と記述した場合には<ci>sin</ci>を問い合わせる．これによって，MathML Content Markup で定義された関数を ci 要素で表現したような変則的なデータにも対応することが可能である．なお，<sin/>と<ci>sin</ci>を両方とも問い合わせるには，条件 or を用いれば良い．

次に，ci 要素と cn 要素の変則的な使用が挙げられる．2 章で述べたように，MathML Content Markup では，識別子を ci 要素，数を cn 要素を用いて表現する．しかし，例えば"2"のような，一般的には数であるとみなされる記号を，ci 要素を用いて識別子として記述することも可能である．このような，ci 要素および cn 要素が変則的に利用されたようなデータには，条件 class を用いて対応できる．つまり，[class="number" content="2"]と記述すると<cn>2</cn>を，[class="identifier" content="2"]と記述すると<ci>2</ci>を，それぞれ問い合わせることができる．なお，数としての"2"と識別子としての"2"を両方とも問い合わせる場合は，[content="2"]と記述すれば良い．

このように，cmmlQ は変則的なデータにも対応できるような記法を備えているため，Web 上から取得したデータを検索するなど，データの表現方法にばらつきがあることが想定される場合にも，正確な検索を実現可能であると期待できる．

7. 結語

本論文では，アプリケーション等の開発における要素技術としての役割を想定した数式検索手法を提案した．独自に定義した問い合わせ言語 cmmlQ では，一貫性を持つ条件記法と，簡潔な記述が可能な省略記法が用意されている．これにより，問い合わせを自動的に生成する処理を行うという用途にも，人間の手で問い合わせを記述するという用途にも対応するような検索手法を実現した．また，問い合わせには XQuery を用いているため，様々な環境で動作可能であり，互換性にも優れている．

cmmlQ は，算術演算や初等関数はもちろん，論理演算や集合演算など多様な演算に対応しており，また，識別子の同一視や可換則など，数式検索に要求される機能も備えている．さらに，検索対象である MathML Content Markup で記述され得る変則的なデータに対しても，問い合わせを適切に記述することで正しく検索することが可能である．

今後の展望として，等価性に関する機能の追加が挙げられる．現在，cmmlQ では，可換則に基づく等価性に対応しており，$x+y$と$y+x$を同一視することができる．しかし，数式においては，他にも様々な等価性が存在する．例えば，1+1 と 2 は等価なものである．また，$a(x+y)$と$ax+ay$も，等価であると言える．cmmlQ における条件を用いて，これらの等価性にも対応することができれば，検索の利便性が向上すると考えられる．一方，数式検索として，どのような等価性に対応すべきであるかを決定することは容易ではない．例えば，$\frac{\mathrm{d}}{\mathrm{dx}}x$ と1は等価であると考えることができるが，これら 2 つの数式を同時に取得するような問い合わせを行う機会は，可換則に基づく問い合わせほど多くないと考えられる．このため，まず，考慮すべき等価性を慎重に選定した上で，数式検索における機能として適切なものを実装することを計画している．

参考文献

[1] STACK, http://stack.bham.ac.uk/
[2] The Wolfram Function Site, http://functions.wolfram.com/
[3] 小田切 健一, 村田 剛志, "MathML を用いた数式検索", 第 22 回人工知能学会全国大会, 1F1-3, (2008).
[4] Corneliu C. Prodescu, Michael Kohlhase, "MathWebSearch 0.5 An Open Formula Search Engine", Wissens- und Erfahrungsmanagement LWA (Lernen, Wissensentdeckung und Adaptivität) Conference Proceedings, (2011).
[5] Bruce R. Miller, Abdou Youssef, "Technical Aspects of the Digital Library of Mathematical Functions", Annals of Mathematics and Artificial Intelligence, Vol.31, No.1, pp.121-136, (2003).
[6] 高田 真澄, 村尾 裕一, "数式の構造を反映した検索法", 第 2 回データ工学と情報マネジメントに関するフォーラム論文集, C7-4, (2010).

[7] Keisuke Yokoi, Akiko Aizawa, "An Approach to Similarity Search for Mathematical Expressions Using MathML", 2nd Workshop towards Digital Mathematics Library, pp.27-35, (2009).
[8] 岸本 貞弥, 中西 崇文, 櫻井 鉄也, 北川 高嗣, 栃木 敏子, "MathML を用いた類似数式検索方式の実現", 第 14 回データ工学ワークショップ論文集, No.6-P, (2003).
[9] Robert Miner, Rajesh Munavalli, "An Approach to Mathematical Search through Query Formulation and Data Normalization", Towards Mechanized Mathematical Assistants, Springer, pp.342-355, (2007).
[10] Yoshinori Hijikata, Hideki Hashimoto, Shogo Nishida, "Search Mathematical Formulas by Mathematical Formulas", Human Interface and the Management of Information. Designing Information Environments, pp.404-411, (2009).
[11] Tetsuya Watanabe, Yoshinori Miyazaki, "Development of IR Tool for Tree-Structured MathML-based Mathematical Descriptions", International Conference on Computers in Education, pp.310-312, (2010).
[12] 渡部 孝幸, 宮崎 佳典, "二次元の位置構造に着目した数式のパターンマッチング手法", 情報知識学会誌, Vol.22, No.3, pp.253-271, (2012).
[13] MathML, http://www.w3.org/TR/MathML3/

付録 cmmlQ から変換された XQuery の例

cmmlQ から XQuery への変換の具体例を示すとともに，XQuery による問い合わせがどのように動作するのかを詳述する．

次の cmmlQ による問い合わせを考える．

```
({sin|cos} (plus {(root x)} *))
```

この問い合わせは，「x の平方根を含む数式と任意の数式とを足し合わせたものに，sin 関数あるいは cos 関数を適用したもの」を意味する．この問い合わせには，$\sin\left(\sqrt{x}+1\right)$ や，$\cos\left(2^{\sqrt{x}}+ab\right)$といった数式がマッチする．

この問い合わせを XQuery に変換したものは，次のようになる．

```
1  $var
2  [./node()[1]/self::apply
3    [.
4      [./node()[1]/self::sin]
5      or .
6      [./node()[1]/self::cos]
7    ]
8    [./node()[2]/self::apply
9      [./node()[1]/self::plus]
10     [./node()[2]/descendant-or-sel
   f::apply
11       [./node()[1]/self::root]
12       [./node()[2]/self::ci[.="x"]]
13     ]
14     [./node()[3]/self::node()]
15   ]
16 ]
```

この XQuery について，行を追って説明する．

1 行目の$var は，XQuery において変数を意味するものである．この変数には，math 要素が格納されていると仮定する．変数に math 要素を格納する方法は，MathML Content Markup によるデータが XML データベース中に存在するのか，XML 文書中に埋め込まれているのか，といった状況によって異なる．よって，本論文では，変数に math 要素を格納する方法には言及せず，変数に math 要素が格納されていることを前提とする．なお，2 行目以降が cmmlQ から変換されて得られる XPath の述語であるが，その述語が満たされる場合，変数$var に格納されている math 要素が，問い合わせを満たすものであると判定される．

2 行目は組に対応するものであり，「math 要素の 1 番目の子である apply 要素」を表す．node()[1]によって 1 番目の子を取得した後，軸 self によって，node()[1]自身を参照している．参照したものが apply 要素である場合，それを取得する．ここで，math 要素の 1 番目の子が apply 要素でない場合には，何も取得されず，述語は満たされないため，その math 要素は問い合わせにマッチしないということになる．

3 行目から 6 行目は，論理和に対応するものである．3 行目の括弧は，2 行目の apply 要素の述語を表すものである．括弧の後にドット.が記述されている．これは math 要素の第一の子である apply 要素を示す．

4 行目は，3 行目のドットに対する述語である．これは sin を表現するものである．3 行目のドットは 2 行目の apply 要素を表すものであるため，4 行目の述語は，実質的には，2 行目の apply 要素の述語として働く．

5 行目は，述語の論理和を表現するための or が記述されている．論理和を用いて複数の述語が記述された場合，いずれかの述語が満たされれば，or を用いた述語全体が満たされたとみなされる．or の直後にはドットが記述されているが，これも 3 行目と同様，2 行目の apply 要素を表す．

6 行目は，4 行目と同様，演算を表現するものであり，実質的に 2 行目の apply 要素の述語である．

すなわち，3 行目から 7 行目は，「2 行目の apply 要素の第一の子(つまり演算)が，sin であるか，または cos である」ということを表現している．

8 行目は組を表現するものである．sin 関数あるいは cos 関数の 1 番目のオペランドであるため，node()[*n*]における *n* の値が 2 となっている．

9 行目は，加算を表現するものである．

10 行目は，軸が descendant-or-self となっている．このため，10 行目は，「2 番目の子で，それ自身か，あるいはその子孫に apply 要素を持つ」ことを意味する．

11 行目および 12 行目は，10 行目の apply 要素の述語である．12 行の.="x"という記法は，「ci 要素の子であるテキスト要素が x である」ということを意味している．これはつまり，<ci>x</ci>という XML データを取得するものである．

14 行目は，9 行目の加算の 2 番目の演算対象である．node()[*n*]における *n* の値が 3 となっている．./node()[3]/self::node()という XPath は，「3 番目の子が何であっても良い」ということを意味しており，これによって cmmlQ におけるワイルドカードの機能を実現している．