数理解析研究所講究録
第1780巻 2012年 232-242

数式を省略して表示する方法の提案と検討

電気通信大学・大学院情報理工学研究科　村尾 裕一 (Hirokazu Murao)
Graduate School of Information Science,
University of Electro-Communications
香川高等専門学校・情報工学科　近藤 祐史 (Yuji Kondoh)
Department of Information Engineering,
Kagawa National College of Technology
(株) アルファオメガ　兵頭 礼子 (Noriko Hyodo)
AlphaOmega Inc.
(株) アルファオメガ　齋藤 友克 (Tomokatsu Saito)
AlphaOmega Inc.

1　はじめに

軽妙で直感的な操作を実現したiPhone/iPadはインターフェイスに大きな変革をもたらした．すなわち，タッチパネルは見たいところへ素早く移動することを，マルチタッチはズーム（＝観測の詳細さ）等の変更を，我々の日常的な動作に一致する指先の動きで可能とする．では，数式をそのような感覚で操作するとしたら（どのようになるのか），そのことを考えてみようというのが我々の発想である．これも数式の処理である．また，近い将来J.Han(NYU Courant Institute)が開発したPerceptive Pixel社によるマルチタッチの巨大ディスプレーやその類が教育や教育の現場で使われるようになることも考慮して，数式や数学の表示に活用することも考えてみたい．

数式という表現は，文字や記号（独自に定義するものも含む）を，何段階かの大きさと字体を変えて2次元的にレイアウトしたものである．数式処理システム（Computer Algebra System．以下CAS）は式の大きさを気にせずに計算し出力するのが普通であり，人間（利用者）には把握しきれないような長大な式が表示されることもしばしばである．表示についても，結果を表示する/しないの二者択一しかないというインターフェイスが現在では一般的である．教育現場ではそのような手に余るような式を扱うことは稀だが，簡単な式でも展開すると膨れ上がるということは往々にしてある．数式の部分部分にはシンボリックな情報があり，計算の結果でもそうした一部分だけが目的とする結果であったり，長大な式でも例えば項数やある項の係数だけを知りたいということもしばしばある．そのような状況において数式をどのように扱いたいかを考える時，(今や誰もが使いこなしている）Google Mapsのような地図情報の表示・操作をイメージしなぞらえるの良いだろう．地図では，目当てとする場所をキーワードによる検索やスクロール操作で探し出し，そこをズームインすることで詳しく調べる．では，数式に対してはどのような操作が望まれるだろうか？数式をレンダリングした結果のイメージの拡大・縮小だけというのは考慮に値しない．

目標：どんな操作をしたいか ：先ず数式の検索は今日的な課題としては，単なるパターン照合としてではなく形の類似度と順位づけというもうひとつの難しい問題として捉える [11] べきなので，ここでは考えないことにする．また，数式の筆記は，べき乗の指数を右肩に普通に書くことからもわかるように 2 次元的にレイアウトするのが普通だが，コンピュータ上での数式の 2 次元入力や編集はもうひとつの別の課題となるのでここでは扱わないことにする．つまり，ここでは数式の表示における interaction （対話的操作＝見せ方の変更）に限定して考える．

文書中において数式は，TEX などに見られるように，文中に埋め込んで文章の一部として表示されるが，処理そのものは普通の文章とは別のモードで扱われる．また，どんなに長大な文書中においても何行にも及ぶ長い式が文中に置かれることなど普通はありえない．数式は，並びの規則性に依って記述したり，略記したり，特徴的な部分だけを示して説明したりするのが普通である．つまり，文書中における数式の表示は，ある一定内の大きさに収まることが望まれる．そこで，数式については，文章のように文字列すべてを表示するのではなく，文章なりを表示する領域の中に数式表示用の箱 (*mathbox* と呼ぶことにする) を置き，その中に式の情報を表示すればよいという考えに到る．更に，*mathbox* やその内部での式の表示情報は次のように扱えばよいだろう．

- *mathbox* 内に数式を表示し，式はその中でスライド（移動）可能とする
- 数式は，箱の数倍に収まるように適宜省略する
- 省略しても意味を失わないように情報を圧縮して表示する
- 表示しなくてよい情報を filtering で選択する

想定されるプラットフォームと環境 ：インタフェースとしては，TeXmacs のように非常に優れたソフトウェアもあるが，実際に数式を扱う人の多様さを考えるとブラウザが妥当なところだろう．数式データとドキュメントが処理の対象となるが，その形式は，当然，MathML[9] (OpenMath[5] は，一般には，補強するための機能として扱えば十分であろう) と XHTML (eXtensible HTML) が最優先である．ここで，旧来の典型的な文書処理はテキストの文章を主体とし，それ以外は補遺的なものとして埋め込むという考え方（数式はレイアウトの指示の入った文字列として別処理をし結果を埋め込む）に基づいており，我々の目的に適わない．また，ひとつの文書の中には様々な XML 要素が含まれる場合も想定されるが，数式を表す MathML とそのグラフを表す SVG 等，XML 要素間で参照することも考慮に入れておく必要がある．それには XHTML の modularization [10] で対応していくことになる．更には，CAS の出力の変換表示装置として利用するために，CAS との連携法も検討するとよいだろう．

本稿の概要と関連研究 本稿では，上述の考え方に基づき，技術的な課題と解決法をできるだけ具体的に検討する．より具体的には，GoogleMaps をひとつのお手本として，できるだけ手軽に実現することを目指して，MathML 等の XML 関連の既存技術を活用することを検討する．その際，処理の対象となるドキュメントと数式をどのように捉えど

のようなデータとして扱うか，特に数式については略記の方法とデータ表現の方法を検討する．

関連するか同様の目的を持った研究は，我々の目標が当たり前過ぎるためか，MKM, Calculemus, CICM, Math UI, OMW(OpenMath ワークショップ) などの国際会議の発表にも殆ど見られない．省略という意味では冗長な括弧等の処理法の研究として，I.Stoyanova と J.H.Davenport による OpenMath ブラウザ上での実装 [7] と，M.Kohlhase と C.Lange による [3] がある．この後者の Jacobs University, Bremen の人達がによる JOBAD プロジェクト [1] では，より一般的な方向（active なドキュメントを作成する環境の構築）を目指して技術開発を進めており参考になる．

2 文書と数式の関係について

次の図（枠内の文書）は式を含んだ典型的なテキストの例だが，数式が主体であり，数式だけを見れば凡そ何を記述しているのかがわかるだろう．また，式を他の場所から参照したり，更には右辺・左辺など式の一部を引用し再表示して用いていることに注目して欲しい．

数式を含む文書の例

$$\sum_{k=1}^{n} k = \frac{1}{2}n(n+1) \qquad (1)$$

帰納法による証明：$n=0$ の場合，(1) の左辺の和は 0 で右辺は $=\frac{1}{2}0(0+1)=0$ ゆえ等号が成り立つ．次に (1) が n で成立つと仮定し，$n+1$ の場合を考える．

$$\begin{aligned}\sum_{k=1}^{n+1} k &= \sum_{k=1}^{n} k + (n+1) \qquad \text{..... (1) を用いて} \\ &= \frac{1}{2}n(n+1) + (n+1) = \frac{1}{2}(n+1)(n+2)\end{aligned}$$

上の例では，ひとつの数式についてその一部（変数，左辺・右辺）を引用したり，その式を用いた簡単な計算を行った結果を文章中で表示して用いている．つまり，ひとつの式を多様な形で文章中で参照しており，元の式に変更があった場合は参照している部分も連動して変更する必要がある．これをコンピュータで扱うとすれば，数式というひとつのオブジェクトが存在し，複数の覗き窓からその全体や一部分を必要に応じて変換をして表示していると捉えるのが自然であろう．より一般的には，ドキュメントはテキスト主体であったり，単に清書・レンダリングした結果を表示するものではなく，数式・グラフ等々様々なオブジェクトが共存する composite な表現形態をとるのが望ましい姿であろう．HTML はそのような目的を一部適えようとする形式とみなすことができ，より多様なオブジェクトを扱うべく XML や XHTML へと発展してきた．数式を記述する XML が MathML であり，そこに active な要素を付け加えることが我々の目標である．ここで active とは，内容は不変でも見せ方は多様にあり，見る人の指定で変更が可能ということである．我々は数式という文脈に限定し，前述のとおり，*mathbox* とい

う覗き窓を通して数式のオブジェクトを表示すると考える．*mathbox* と数式オブジェクトの表現法に関する設計方針を列挙する．

- *mathbox* 内に数式を表示し，式はその中でスライド（移動表示）可能とする
- *mathbox* の大きさは可変とする
- 数式は，箱の数倍に収まるように適宜省略する... どの程度省略するかは，*mathbox* 毎に設定するズームの倍率で決める
- 省略しても意味を失わないように情報を圧縮して表示する
- 数式を表す木構造において，部分式毎の何段階かの表示すべき情報と部分式間の優先度を再帰的に求め，表示領域のズームの倍率との兼ね合いでどこまでを表示するかを決める
- 表示しなくてよい情報を filtering の対象とし，その具体的な条件を *mathbox* 毎に設定する
- 何段階かの省略記法と意味的なズームを用意する
- 式の内部構造では，部分式ごとのラベルづけと他との参照関係を可能とする

3 数式データの構成

先ず，一般的な CAS や LaTeX などの文書清書システムを元に，対象となる数式を 2 次元的なレイアウトの構成部品として区別してまとめると次のとおり．

- 数：整数（多倍長），浮動小数（倍精度、任意精度），有理数，複素数，区間数
- 変数（添字を持つ場合もある）．記号．文字列
- 多項式，級数：1 変数または多変数．表現（再帰的/分散．項順序）
- 有理式
- 関数形（＝関数名と引数の並び）．添字がつくこともある
- 行列とベクトル．リストや集合
- 積分．級数の和と積． 集合． 最大と最小 ... 添字がつく
- 根号．場合分けなど

これらを構成要素とするデータを表現する形式として MathML は十分であり，入出力の標準形や内部表現として有力候補となる．その概要を次節にまとめる．

3.1 MathML(3.0) について

3.1.1 Presentation Markup と Content Markup

数式を表現するための WEB 技術として MathML [8] が XML として定義されている．MathML の特徴として，数式を表示の仕方指定する表現と意味を表す表現法の 2 種類の記術法を提供していることがあり，これは OpenMath に由来する．前者を presentation markup，後者を content markup と呼び，MathML 3.0 の仕様書 [9] でも第 3 章及び第 4 章と別々の章に分けて仕様記述がされている．

Presentation markup の要素には，トークン要素（記号，名前，数，ラベル等）とレイアウト スキーマ（分数・根号などの一定の配置，添字，行列等）の 2 つのクラスがあり，表示法を制御する指示（displaystyle と scriptlevel．改行等）が用意されている．

一方 content markup は，数・変数・オペレータ等を葉とし，関数の適用等の数学的構成物を表す途中のノードで構成される木構造で，数学的な構造を記述するための markup である．Content markup は，semantics（意味論）を記述・把握するための外部（content dictionary）を参照する機構等，大部分は OpenMath に由来する．Markup のための要素は，幼稚園から大学教養課程程度までで用いる数学[1]には十分な基本セットを含み，特に MathML の 3.0 では，極限・積分・和などの厳密とは限らないが慣用的に使われている形と，束縛変数（bound var.: $\sum_i$ の i とか）の概念などが導入されたことが新しい．Content markup で記述された数式をいかに表示 (rendering) するかは，規格では指定されておらず典型例で例示されるにとどまっているが，XSLT（記法を宣言で定義）で実装した例 [4, 6, 2] が発表されており参考になる．

3.1.2 MathML の構成要素

前述のとおり，具体的な要素は [9] の第 3 章と第 4 章で定義・記述されている．その代表的なものを以下にまとめる．

- presentation markup
  - token 要素：mi(名前), mn(数), mo(オペレータ), mtext, mspace, ms(文字列)
  - general レイアウト スキーマ：mrow(横並び), mfrac, msqrt, mroot, mstyle(スタイルの変更), merror, mpadded, mphantom, mfenced, menclose
  - 添字：msub, msup, msubsup, munder, mover, munderover, mmultiscripts
  - 表や行列：mtable, mlabeledtr, mtr, mtd, malignmark と malignmark
  - elementary math 用: mstack, mlongdiv, msgroup, msrow, mscarrries, mscarry, msline
  - maction
- content markup

[1] それより先は，記法を自由に導入するので手に負えない．

- ci, cn, csymbol, cs, apply, bind, bvar, share, semantics と annotation と annotation-xml, cerror, cbytes
- 一般的な/広く使われている記法：必ずしも厳密には定義されていないが常用されている慣用句（例：偏微分、場合分け、…）

MathML が presentation と content の 2 種類の要素で構成されていることは良く知られているが，次節で述べるように，両方の markup 或いは複数の記述を混在させたり選択して利用したりする機能も用意されている．

3.1.3 色々な markup の混在

MathML には，従来あまり注目されてこなかったが，ひとつの式に対し複数の markup 記述を混在・併存させる機能が用意されており，現在も発展途上にある．実際，MathML の規格を記述する TR(Technical Report) でも，第 5 章が MathML2 では「Combining Presentation and Content Markup」と単に present と content の両 markup の混在のみを想定していたのに対し，3.0 では「Mixing Markup Languages for Mathematical Expressions」となり拡張の形態を増やすと共に記述も詳細さを増している．

MathML の要素として意味注釈 (semantics annotation) のための要素が用意されており，これを用いることにより MathML の markup を他の markup 言語 (OpenMath 等) と結合したり，式を替の表式と結びつけたり，意味的な性質やその他の属性を MathML 表現と結びつけたりすることが可能である．また，presentation markup と content markup は，一定の条件の元でひとつの式の木構造の中に散りばめて混在させることができるが，もうひとつ，各々の markup を一組用意して semantics 要素で結合するという方法もある．前者は mixed markup，後者は parallel markup と呼ばれる．

意味注釈の要素としては，annotation 要素 (`<annotation>` ... `</annotation>`) と annotation-xml 要素 (`<annotation-xml>` ... `</annotation-xml>`)，及び，その記述の入れ物となる semantics 要素 (`<semantics>` ... `</semantics>`) がある．semantics 要素は presentation と content のどちらにも認識される．

注釈 (annotation) のフレームワーク [9, § 5.1] は，前述の各種の結合を実現するための一般的なフレームワークを提供しており，MathML の式に対し annotation の種類を示すキー (key) とその値 (value) の組を任意個付加することができる．annotation/annotation-xml 要素は semantics 要素の中に置くが，semantics 要素の中では先ず MathML の式を書き，その後に annotation/annotation-xml 要素を任意個書き並べる．キーは，代替表現，semantics の指定や解説，型の情報，レンダリングのヒント，特定の処理系向けのデータ等，式と annotation の様々な関係を指定するのに用いられる．キーそのものは Content Dictionary 中で symbol として定義され，annotation/annotation-xml 要素の cd 属性と name 属性によって指定される．その symbol の定義には，“attribution” や “semantic-attribution” などの role 属性を付け加えることにより，その annotation を無視しても平気か（意味が変わってしまわないか）を示すこともできる．キーの例として，mathmlkeys という CD で定義された alternate-representation と contentequiv があり，

この alternate-representation が標準のキーである．また，encoding 属性が annotation のデータ形式が何であるか（MathML の presentation, OpenMath, TEX, Maple など）を示す．

presentation & content markup の混在の制約 [9, § 5.3]：先ず，曖昧さを生じないこと．content 中に置ける presentation は，(1) token 要素 <ci> や <cn> の中，(2) csymbol 要素中，(3) semantics 要素中の 3 つの場合に限定される．逆の presentation 中の content の場合は，content が well-formed な presentation に変換されることが条件である．

Parallel Markup [9, § 5.4] は，同一の数式に対する複数の markup を結合する方法のことで，semantics 要素中で encoding 属性を指定した annotation-xml 要素を用いて指定する．木構造の一部として用いることもでき，id 属性と xref 属性を用いて，ひとつの semantics 要素内で相互参照をすることができる．

4 数式の省略表示について

では，数式はどのようにして省略すればよいだろうか．言うまでもなく，単に表示の時に占める面積だけを問題にするわけではない．面積については，*mathbox* という大きさが一定で内部をスライドさせることのできる覗き窓で対処することは前に述べた．そしてその中に表示する数式は，式がもつ情報を圧縮して把握し易くすることが重要である．単に一定の略記法を定義しそれに従って式を変換することが目的なのではなく，情報量をできる限り減らさずに，対話的に略記法を動的に変化させられるようにしたいのである．その方策を基本的な方針から検討する．

- 無闇に表示領域が必要な部分は隠そう！ ただし，必要な情報（特徴的な量 etc）は欲しい
  （例）係数に現れた長大な整数 → 桁数程度で十分
- 数式の構成要素毎に、何段階かの略記法を決める． ただし，略記とは別に、（吹き出しのようなものを想定し）補足的な表示に使う情報は別途準備可能とする
- ひとつの数式に対し、何段階か（精々 5 段階程度）のレベルに分けた表現がありうる
  - 表示の倍率の度合いで選ぶ → ズームレベルと呼ぶことにする
  - ズームレベル毎に，式の木構造の深さに応じた略記レベルを設定する．ここで，木の深さのように単調に増加し部分構造どうし一律な値をそのまま用いるかと，用いない場合にどのように減衰と分布をさせるかは実証的な検討が必要である．
  - 同列にグルーピングされた部分式どうしで優先度を比較して劣勢の方をより省略する・しないモードを選択可能とする

- どこまでを表示するか（見えるようにするか）を決める，ズームの倍率を *mathbox* 毎に設定する．

4.1 数式の構成部品毎の略記の仕方

- 文字列/symbol：そのまま/先頭の数語/先頭の数文字/ ”...” だけ
- 変数や関数の名前：
  - 基本的に省略しない
  - 添え字：そのまま/先頭いくつか/添え字の有無がわかる程度（... で）
  - 長過ぎる（n 文字以上）場合は略記する・しない
- 整数：そのまま/10 進で最初の数桁と桁数（浮動小数程度の情報）/最後の方の数桁だけを追加表示
  〈補足〉10 進桁数，基数変換結果等
- 浮動小数：そのまま/仮数と指数（10 進）
- 多項式や級数
  - そのまま/（先頭＋ α）項と項数
  - （多項式の場合）＋最低次（末尾）の項をつけるか否か
  - 次数の分布（集合 (?)）も付けるか

  〈補足〉これらの情報．表現（再帰的 or 分散）．項順序
- 有理式：分子・分母の多項式
- 根号：そのまま
- 積分・和・積：そのまま/上下限：束縛変数のみ明記/上下限の有無のみ
- ベクトル，リスト，集合：そのまま/先頭のいくつか/形のみ
- 行列，ベクトル：表示のどの部分に重点をおくかの設定．要素毎に窓

これらの各々に略記のレベルを設定する

5 実装法について

前節までで考察した省略表示を実現しようとすると，(1) CASから整形した結果を直接出力しその結果をクライアントが表示，(2) MathMLを入力とし整形したMathML(+α)を出力するサーバを用意しクライアントと連携させる，(3) ブラウザ上のクライアントのJavaScriptプログラムがMathMLを(サーバでの処理機能も含め)整形・変換して表示，の3つの処理形態が考えられる．結果をrenderingし表示するにはMathMLのpresentation markupを用いて，対応可能なブラウザの機能に任せれば当面は充分だろう．

データ表現はJOBADに倣うのが適当と思われるが，それに加えて，どのデータをどこ（サーバとクライアント）に配置するかやその際にAjaxを用いるか等の検討は必要だろう．式そのものを表すcontentの表現には，CASの一般的な内部データ構造を用いるが，いくつかの拡張を行うことになる：何段階か（せいぜい2, 3段階）の省略のレベルに分けての(contentの)データ表現を用意し，presentationからはその構造（content中のラベル）への参照をする．この時，presentationでは位置や表示の大きさの情報を獲得しておき，また，content中のラベルの管理はJSON（＝リスト）形式で交換しておく．

contentのデータ表現の変換には，MathMLのcontentからpresentationへの変換の例があるように，XSL/XSLTの利用が考えられるが，変換法の記述を宣言的にどこまでできるのかは未知数である．また，規格の付録[9, 付録A]には，XMLとしての構文解析のためのRelaxNG Schemaの記述があり(DTDやXML Schemaもあるが)，これを改造して省略形への変換を行うことも可能だろう．いずれにせよ，省略形式への変換はアルゴリズムが重要であり，また，CAS，サーバ，クライアントのすべてにおいて実装が必要となる．

事例紹介 ：JOBAD(JavaScript API for OMDoc-based Active Documents) [1]は，WEBの種々のサービスを統合するmashupの考え方を数式に適用し，数学関連のドキュメントに対し，読ませるだけでなく，表示のカスタマイズ・表記法の変更・関連情報の検索といった対話性を導入する多目的なソフトウェア・アーキテクチャである．そのアーキテクチャは，XHTMLとMathMLをベースとして，数学的サービス，UIの要素，通信・文書操作のための関数群の3つから構成され，サーバとクライアント側のJavaScriptのmodule群から成る．notation definitionを導入しJOMDoc libraryの利用してcontent markupをレンダリング（表示イメージに変換）し，また，CASの内部表現では当然のことだがMathML表現等の内部表現の木構造において現れる余計な括弧を自動的に削除することが機能の特徴の一部である．数式の省略表示に関連しては，次のような特徴がある．

- ⋆ `<maction>`要素中に記述された複数の表式から選択して表示（actiontype属性, selection属性）を行う
- ⋆ 指定された式の一部を折り畳んで隠す．隠された表式は`<maction>`を追加して選択可能にする

⋆ 相互参照を行う parallel markup: content も保持して presentation 側から xref で参照する

これらは我々の目標の実現に大変参考になることは明らかだが，我々はもっと踏み込んで自動的にやってしまおうという立場をとる．

6 おわりに

iPhone/iPad を初めとした高い対話性を実現した機器が広く普及するようになり，筆者のひとりはその優れた操作性に触発されて，その操作感を数式に対しても実現したいと思い，それには式をどのように表現し，どのような技術を用いれば簡便に実現できるかを考え始めた．本稿は，その第一段階として，世の中の動向を調査し，既存技術の利用可能性を検討した結果のまとめである．

先ず，ドキュメントは様々な種類のオブジェクトから構成されるべき composite な表現形式と捉え，その中にあって数式というオブジェクトについて考察し，内容は静的であっても（CAS との連携をとり動的に変化させる方法の検討にまでは到っていない），interaction に耐えるという意味で active に扱うための方策を検討した．有用と思われる既存技術として MathML や　XML の関連技術（RelaxNG, XSL/XSLT, Ajax, DOM 木と JavaScript）に着目し，簡単なレビューを行った．特に，MathML については presentation/content markup ばかりが話題にされるが，semantic annotation の機能が active なオブジェクトを実現する上で重要な役割を果たすであろうことがわかった．また，先行研究として JOBAD プロジェクトがあり，実現技術を考案する上で大いに参考となることがわかった．今後，検討を実践的に進めることが何よりも大切であり，特に，省略形式への変換アルゴリズムが重要であり，試験的な実装が当面の課題である．この変換は単なる機械的な変換にとどまらずに，文章の要約と同様，意味を的確に表現するために，式の特徴量やひとつの式中でのその分布をモデル化する等，検索技術にも通ずる研究が必要となるかもしれない．更に，その変換のための技術には，XSL/XSLT，既存スキーマを発展・改造させて用いる RelaxNG Schema，DOM 木に対する JavaScript による操作などが考えられるが，どこまでをどの方法でどこで（サーバ/クライアント）やるのかの切り分けを検討することも今後の課題である．

参考文献

[1] J. Giceva, C. Lange, and F. Rabe. Integrating web services into active mathematical documents. In *Intelligent Computer Mathematics: Calculemus 2009/MKM 2009*, No. 5625 in LNAI, pp. 279–293, 2009.

[2] The W3C Math Working Group. Xslt stylesheets for mathml on the web. `http://www.w3.org/Math/XSL/`.

[3] M. Kohlhase, C. Lange, and R. Rabe. Presenting mathematical content with flexible elisions. In *OpenMath Workshop 2007*, 2007.

[4] E. Pietriga. Mathml content2presentation transformation (mathmlc2p). `http://www.lri.fr/~pietriga/mathmlc2p/mathmlc2p.html`.

[5] The OpenMath Society. The OpenMath standard. `http://www.openmath.org/standard/`.

[6] The OpenMath Society. xsl4mathml – xsl stylesheets for MathML. `http://code.google.com/p/xsl4mathml/`.

[7] I. Stoyanova and J.H. Davenport. Towards intelligent summarising and browsing of mathematical expressions. In *Mathematical User-Interfaces Workshop 2010*, 2010. `http://www.activemath.org/workshops/MathUI/10/`.

[8] W3C. Math home. `http://www.w3.org/Math/`.

[9] W3C. Mathematical markup language. `http://www.w3.org/TR/MathML3/`.

[10] W3C. Xhtml modularization. `http://www.w3.org/TR/xhtml-modularization/`.

[11] 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 DEIM2010 論文集, C7-4, 2010. `http://db-event.jpn.org/deim2010/proceedings/files/C7-4.pdf`.