4ZJ-02

# 代数的変形に対応した数式変形依拠公式提示 Web アプリケーションの改良
# Improving a Web Application Annotating Basis of Mathematical Transformations with an Extension of Algebraic Transformations

加藤 駿弥[*1], 宮崎 佳典[*2], 中村 泰之[*3], 田中 省作[*4], 新谷 誠[*2]
Shunya KATO[*1], Yoshinori MIYAZAKI[*2], Yasuyuki NAKAMURA[*3], Shosaku TANAKA[*4], Makoto ARAYA[*2]
[*1] 静岡大学 情報学部
[*1]Faculty of Informatics, Shizuoka University
[*2] 静岡大学学術院 情報学領域
[*2]College of Informatics, Shizuoka University
[*3] 名古屋大学 大学院情報学研究科
[*3]Graduate School of Informatics, Nagoya University
[*4] 立命館大学 文学部
[*4]College of Letters, Ritsumeikan University
Email: kato.shunya.18@shizuoka.ac.jp

**あらまし**：MathML 記述の数式検索ツールを基盤として，我々は数式変形された（不）等式に対して公式適用箇所をその公式名とともにアノテートする数式変形依拠公式提示機能を開発している．しかし，データベースに事前登録した公式に対し，代数的に変形された上で適用される場合は抽出できないため，本研究では先に採用した Wolfram Alpha API に加え，Python の記号計算用ライブラリである SymPy を相補的に用い，データベース管理の効率性や処理速度の変化について実験結果と共に調査・評価を行う．
キーワード：数式検索システム，正規表現，変形依拠公式提示機能，Wolfram Alpha API，SymPy

## 1. はじめに

STEM 教育ならびに数理・データサイエンス教育が世界の潮流となってきた．しかし，それらの教育において必要不可欠な数式に対し，苦手意識を抱く学習者は多く存在する．一方で，MathML などを代表に計算機上で数式を利用するためのデータ形式の普及が進み，STEM 教育で利用されるデジタル教科書の普及などが今後予想される．そこで我々研究グループは，計算機上の数式を含む文書を対象とした数式検索システムを開発し(1)，将来の数学学習に役立てる機能拡張を行ってきた．さらにはその機能の 1 つである，数式変形の拠り所となっている公式を検索・提示する変形依拠公式提示機能を数式処理システムと連携することで拡張し，数学学習支援においてより高い実用性を持つ数式検索システムの構築を目指した(2)．(2)は公式データベースのクエリを削減するために Wolfram Alpha（API）と SymPy(3)両者を相補的に用いることでデータベースに登録する必要のあるクエリ数の削減を目指し，SymPy 単体での数式変形の結果からクエリの削減を推測したが，これに対し本研究では，実際に既存システムに実装することで削減率の検証を行うことを目的とする．数式検索システムを 2 つ用いることで処理速度が遅くなる可能性がある問題についても調査を行う．

紙面のスペースの都合上，関連研究として以下 2 点を挙げる．(4)は紙折り数列による数学の導入教材の提案として具体的に Mathematica を活用した ICT 教材について紹介している．(5)では中学数学の文章題に対する立式補助アプリ開発が行われている．

## 2. 変形依拠公式提示機能

Web 上の文書に数式を記述する場合に数式表示用のデータ形式が用いられる．そこで(1)のシステムは MathML 内の表示の情報を記述するタグセットである Presentation Markup で記述された（以下，“MathML 記述”と表記）数式を対象とし，正規表現とパターンマッチングによるページ内数式検索を基本的な機能として実装した．文書中の MathML と検索窓に入力した検索数式を正規表現へ置換ののち照合し，マッチした文書中当該箇所の数式をハイライトする．

数式検索システムの機能として実装されたのが変形依拠公式提示機能である．機能の適用例として図 1 を示す．文書中の数式変形を読み込み，公式クエリ（あらかじめシステムに登録されている MathML 記述の公式データ）と照合することで，式変形の拠り所となっている公式名を提示する．

$$\frac{\cos 2x+2\cos x+1}{\cos x+\cos^2 x+\sin^2 x+\sin x}=\frac{\cos 2x+2\cos x+1}{\cos x+1+\sin x}$$
$$=\frac{\cos^2 x-\sin^2 x+2\cos x+1}{\cos x+1+\sin x}$$
$$=\frac{(\cos x+1)^2-\sin^2 x}{\cos x+1+\sin x}$$
$$=\frac{((\cos x+1)+\sin x)((\cos x+1)-\sin x)}{\cos x+1+\sin x}$$
$$=\cos x+1-\sin x$$

$\frac{cos2x+2cosx+1}{cosx+cos^2x+sin^2x+sinx}=\frac{cos2x+2cosx+1}{cosx+1+sinx}$ ←∵ピタゴラスの定理($cos^2x+sin^2x=1$)
$=\frac{cos2x+2cosx+1}{cosx+1+sinx}$
$=\frac{cos^2x-sin^2x+2cosx+1}{cosx+1+sinx}$ ←∵cosの倍角の公式($cos2x=cos^2x-sin^2x$)
$=\frac{(cosx+1)^2-sin^2x}{cosx+1+sinx}$
$=\frac{((cosx+1)+sinx)((cosx+1)-sinx)}{cosx+1+sinx}$
$=cosx+1-sinx$

図 1 変形依拠公式提示機能



Copyright ©2022 Information Processing Society of Japan.
All Rights Reserved.

## 3. Wolfram Alpha と SymPy による DB 削減

先行研究(2)では Wolfram Alpha と SymPy を相補的に連携してデータベースのクエリの削減を行った．実験では高校数学レベルで学ぶ 103 個の公式に対して 5 種の変形パターンを適用した派生数式を用意した．以下は変形パターンである．

変形パターン：

・従前の登録済みと同一　・両辺を入れ替え

・両辺×N　・両辺÷M　・両辺×(-1)

（ここに N，M は適当な自然数）

“両辺×N”,“両辺÷M”については数式の MathML 木構造が崩れる場合のみ適用する．5 種の数式の変形結果がすべて一致するもの,2 式や 3 式,4 式一致するものがある．この共通の数式を基底式として公式クエリに登録した．実際に 5 種のルールにより得られた数式は 283 個である.Wolfram Alpha と SymPy を相補的に利用することで 95%のクエリを削減した．変形パターンから得られる対象数式の正当性を示すため本研究では変形パターンを定める上での変形ルール設定を明確にした.以下が変形ルールの例である．

変形ルール：

・項が 1 つである場合　・項が 2 つ以上である場合　・係数があるか　・分数が含まれているか

上記のルール設定をもとにパターンを適用した場合,実際に 103 個から得られる実験対象の数式が 180 個になるのかを調査した．以下は SymPy をシステムに導入した場合のシステム構成図である．

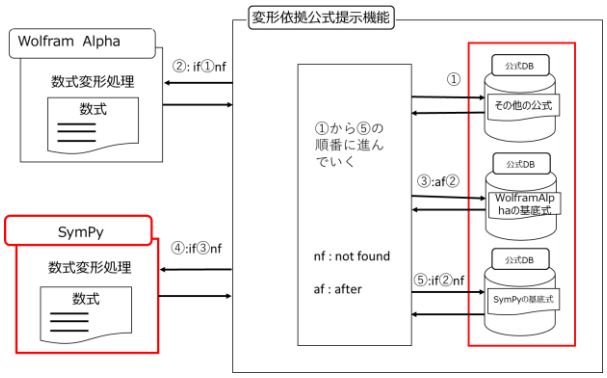



**図 2 システム構成図**

動作手順は①から⑤の順となる．まず，基底式を得られなかった公式のデータベースを参照する．マッチしない場合，Wolfram Alpha で数式変形を行い Wolfram Alpha で得られた基底式の入ったデータベースを参照する．SymPy についても同様に行った結果として対象の数式は 180 個であることが確認できた．先行研究では SymPy のモジュール単体でしか実験をしていないため実際にシステムに組み込んだ実験を行い，削減率を求めた．結果として対象となる数式 180 個に対して 94 個のクエリ削減ができ，52%のデータベースを削減できることを確認した．Wolfram Alpha で削減できなかった 31 個のクエリについては 15 個削減でき，Wolfram Alpha と SymPy を相補的に活用することで 91%のクエリの削減ができたこととなる．

## 4. DB 削減の問題点

本研究では Wolfram Alpha と SymPy を相補的に利用しているが数式処理システムを 2 つ用いることでマッチングにおいて処理速度が遅くなることが考えられる．マッチングでは 2 つの数式処理システムを通す前と各数式処理システムを通した後の計 3 回データベースを参照する必要があることから SymPy を導入する前と後で 1 回分のデータベースを参照する時間分処理速度が増加することが想定される．そこで本研究ではもともとある 1 つのデータベースを 3 つに分割し速度比較を行った.2 つの数式処理システムを通す前に参照するデータベースと各数式処理システムを通した後に参照するデータベースを作成することで何回も同じデータベース全体を参照する必要がなくなりマッチングの処理時間を減らすことが可能になる．SymPy を導入する前のマッチングまでにかかる時間は 6,095 ミリ秒だったのに対し，SymPy を導入した後のマッチングにかかる時間は 8,278 ミリ秒になった．SymPy で数式変形をする処理にかかる時間は 1,260 ミリ秒となった．

## 5. まとめ・今後の展望

本研究では変形パターンのルール設定を明確にし,対象数式を確認し,実験を行った．Wolfram Alpha と SymPy を相補的に活用することによって DB の削減に成功した．SymPy を導入した際の速度計測も行った．

今後の展望としてはまだクエリの削減は完璧ではなくデータベースの削減を目指していく．変形依拠公式提示機能 Web アプリレーションを一般のユーザが問題なく使用できるようにマニュアル作成をし,UI に組み込む．現在，システムが利用できる前提条件として Firefox である必要がある．実用化のためには Chrome など別のブラウザ上でも行う必要がある．

### 参考文献

(1) 渡部 孝幸, 宮崎 佳典, 正規表現を用いた数式検索手法の提案, 情報処理学会論文誌, Vol.56, No.5, pp.1417-1427 (2015).

(2) 加藤 駿弥, 宮崎 佳典, 中村 泰之, 田中 省作, 新谷 誠, 数式変形依拠公式提示アプリケーションの機能拡張への試み, 2020 年度 JSiSE 学生研究発表会（東海地区）, pp. 99-100 (2021).

(3) SymPy :https://docs.sympy.org/latest/modules/simplify/simplify.html

(4) 西林 大樹 ほか, 紙折り数列による数学の導入教育の提案-Mathematica を用いた ICT 教材, 2021 年度工学教育研究講演会講演論文集 69, pp. 84-85 (2021).

(5) 村瀬 輝, 伊藤 利明, 中学数学の文章題に対する立式補助アプリ開発, 情報科学技術フォーラム講演論文集(FIT), pp. 159-160 (2020).



Copyright ©2022 Information Processing Society of Japan.
All Rights Reserved.