数理解析研究所講究録
第1865巻 2013年 137-145



# 導出規則に着目した証明視覚化・式変形支援システムの提案

静岡大学・情報学研究科　渡部 孝幸 (Takayuki Watabe)
Graduate School of Informatics,
Shizuoka University
静岡大学・情報学部　宮崎 佳典 (Yoshinori Miyazaki)
Faculty of Informatics,
Shizuoka University
静岡大学・情報学研究科　林 佳樹 (Yoshiki Hayashi)
Graduate School of Informatics,
Shizuoka University

## 1　はじめに

数学学習において，ある結論を得るための導出過程を学ぶことは重要である．導出過程を学ぶことは結論を得るための方法を学ぶということであり，方法を学ぶことによってはじめて，学習者自身が自らの手で数学的な問題を解くことができるようになる．本論では，この導出過程に着目し，高等学校から理工系大学学部初年度程度の数学学習者を対象とした，導出過程の理解を支援するためのシステムについて述べる．

本論で対象とするのは，数式変形と，証明における推論である．変形および推論はステップ・バイ・ステップで進められるものである．このため，変形および推論を理解するためには，ある数式から別の数式への変形の根拠，あるいは，ある命題から別の命題への推論の根拠を一つ一つ明確に理解する必要がある．そこで我々は，変形や推論の根拠，すなわち導出規則を，ユーザに明示するようなシステムの開発を試みる．数式変形に関しては，学習者による変形を支援するという方法によって，理解を促す．証明に関しては，その論理構造と推論の根拠を一目で理解できるよう，証明を図示するという方法をとる．

以下，2章で，我々が数式の表現手段として利用しているMathMLについて概説し，3章で変形支援システムについて，4章で証明の図示について，それぞれ述べた後，5章にて本論の総括を示す．

## 2　MathML

数式変形や証明を扱う上で，数式は不可欠である．我々は，数式を記述するためのデータ形式として，W3CによってXMLボキャブラリとして策定されているMathML[1]を利用する．

MathMLには，数式の見た目を記述するためのPresentation Markupと，数式の意味を記述するためのContent Markupの2種類の形式が存在する．我々のシステムの処理に直接的に関係する形式はContent Markupであるため，本章では，Content Markupについて述べる．また，以後，本論においてMathMLとはContent Markupを指すものとする．

MathMLでは，"種別"と"対象"を明示することによって演算や関係を表現する．例えば，$a+1$という数式は，「可算を，識別子$a$と数1に対して適用する」という形で記述される．これをMathMLのコードで書くと次のようになる．

```
<apply>
  <plus/>
  <ci>a</ci>
  <cn>1</cn>
</apply>
```

タグapplyで囲まれた部分が，1つの演算や関係を表現している．applyの最初の子が"種別"を表し，それに続く子が"対象"を表す．すなわち，ここでは"種別"はplus(加算)であり，"対象"はaと1である．ciはその子が識別子であることを，cnはその子が数であることを表すためのタグである．<plus/>の箇所を，sin関数であれば<sin/>，等号であれば<eq/>など，他のタグに置き換えることによって，様々な数式を表現できる．また，apply自身を"対象"として記述し，入れ子構造を構成することも可能である．これによって，複雑な数式を表現することができる．

## 3 数式変形支援

学習者による数式の変形を支援する上で，我々は2つの目標を定める．

1. 変形の方針が定まっていない場合や変形の知識が不足している場合にも変形を継続することができる．
2. 変形のすべてのステップにおいて，その根拠を理解できる．

1.に関して，学習者は多くの場合，どのように変形すれば目的(数式中の変数を減らす，方程式を解く，微分しやすい形にする，など)を達成できるのか，その指針を把握していないと考えられる．また，目的を達成するための変形の方法をそもそも知らない可能性もある．数式変形に関して，誤解をしている場合もあり得る[2]．本システムの支援により，そのような学習者でも，試行錯誤によって数式を様々に変形することができるようにする．この試行錯誤によって，数式変形についての知識の習得や，目的を達成できるような変形の発見を促すことを目指す．

2.に関して，システムによって数式の変形を進めることができたとしても，なぜそのような変形を行うことができるのか，その根拠がわからなければ，知識の習得にはつながらないと考えられる．このため，本システムは，定義，定理や公式の適用を1ステップと捉え，1ステップずつの変形のみを許す．これにより，学習者は，各ステップの変形がどのような根拠で成り立つのか理解しながら変形を進めることができる．

本システムによる処理の流れを，次の図1に示す．

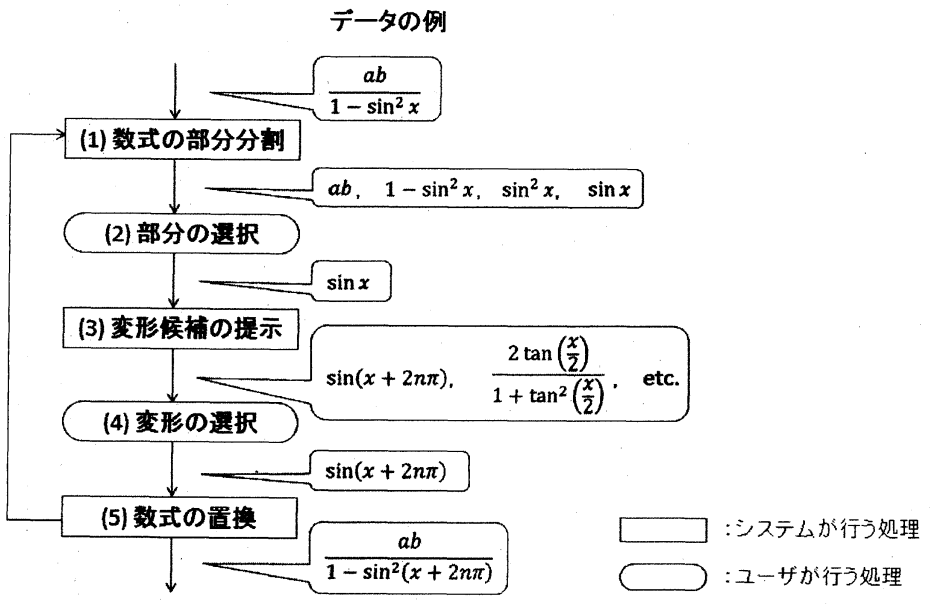


図 1: 数式変形支援システムの処理

システムは数式を受け取り，その数式を小部分に分割し，ユーザに提示する (1)．ユーザは，分割して得られた数式から，変形を適用する数式を選択する (2)．さらにシステムは，選択された数式から変形して得られる数式の一覧をリストとして提示する (3)．そのリストを見たユーザは，実際にどの変形を行うかを決定する (4)．それをもとに，システムが数式を置換する (5)．この過程を経て，1 ステップの変形が完了する．こうして得られた数式に対して，再び変形を行うこともでき，これを繰り返すことで，1 ステップずつ変形を進めていくことが可能となる．

以下で，数式の部分分割，変形候補の探索，および数式の置換について述べる．また，本システムのプロトタイプのインタフェースを紹介する．

## 3.1 数式の部分分割

数式の分割では，それによって得られるまとまりが演算や関数を含む数式となるように分割を行う．このような分割で得られたまとまりを，数式の部分と呼ぶ．分割は，MathML の「1 つの演算や関係が，1 つの apply 要素として記述される」という性質を利用することで実現できる．すなわち，入力された MathML データに含まれる apply 要素を抜き出すことによって，数式を部分に分割することができる．

例えば，$\frac{ab}{1-\sin^2 x}$ という数式を，MathML のコードに対応する木構造で表現すると，次の図 2 のようになる．

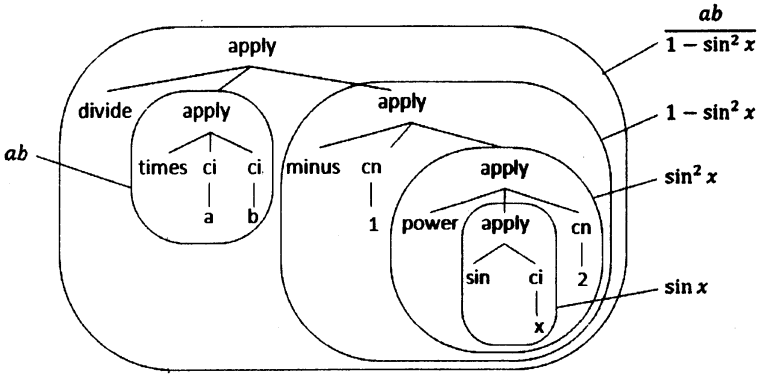


図 2: $\frac{ab}{1-\sin^2 x}$ の木

図 2 のうち，枠で囲んだ部分がそれぞれ，1 つの apply 要素を構成している．それぞれ，$\frac{ab}{1-\sin^2 x}$，$ab$，$1-\sin^2 x$，$\sin^2 x$，$\sin x$ に対応している．これらが部分分割によって得られる数式である．すべて，1 つの完結した数式を構成していることがわかる．

## 3.2 変形候補の探索

数式の変形候補の探索は，数式データセットに対して数式のマッチング処理を行うことによって実現される．数式データセットは，その内部に多数の等式を保持している．

部分分割で得られた数式のうちユーザに選択された数式と，数式データセット中の等式の左辺および右辺との間で，マッチング処理を行う．マッチングに成功した場合，マッチした等式のもう一方の辺を，変形候補としてユーザに提示する．プロトタイプとして，現在，数式データセットには四則演算と実関数を中心とした 300 程度の数式を収録している．

### 3.2.1 数式のマッチング処理

数式のマッチング処理においては多くの研究がなされているが，数式のセマンティクスに着目するか否かによって，そのアプローチが大きく異なる．本研究で扱う数式変形では，数式のセマンティクスを扱う必要がある．この理由として，例えば，数式のセマンティクスを扱わない場合，$f(a+b)$ という数式が，$a+b$ に対する関数 $f$ の適用を表すのか，$f$ と $a+b$ との積を表すのか，区別することができない．一方，数式の変形を行うためには，数式を一意に解釈することが不可欠となる．この点に関して，我々のシステムは内部的に Content Markup を用いており，数式のセマンティクスを扱うことができるため，上述のような数式の多義性の問題を避けることができる．

また，数式のマッチングが備えるべき性質は，[3]，[4] などで指摘されている．これらの指摘の多くは，数式の同義性や類義性に関するものである．実際，数式のマッチングに関する研究においては，同義性や類義性の定義がその主題となることも多い．一方，本システムのマッチング処理で数式の同義性や類義性を考慮すると，誤った変形候補を取得してしまう可能性がある．例えば，数式検索の研究では，sin と cos を，”三角関数

である”という共通性を根拠として，類似していると判断するものがあるが，我々のシステムにおいて sin を cos にマッチさせるような処理を行うと，変形候補として誤った結果が得られることになる．

上述の理由により，現時点でのマッチング処理は，数式の木構造が完全に一致する場合にのみマッチしたとみなすという処理をその基本としている．ただし，識別子の一般化は行っている．識別子の一般化とは，識別子の記号としての見た目を無視するための処理を指す．これにより，例えば，$\theta$，$x$，$a$ などはすべて，同じものと判定されるようになる．なお，同一の数式中に複数種類の識別子が出現した場合は，それらを区別する．

## 3.3 数式の置換

数式の置換の方法を述べる．ユーザによって部分の選択がなされることによって，置換されるべき apply 要素が定まる．また，ユーザによって変形の選択がなされることによって，置換すべき数式が定まる．なお，変形候補はすべて完結した数式であるため，1 つの apply 要素を構成する．したがって，部分の選択によって定まった apply 要素を，変形の選択によって定まった apply 要素に置換することによって，数式を適切に置換することができる．なお，入力された数式と数式データセット中の数式との間で識別子に相違があった場合，置換する際に識別子が修正される．例えば，$\sin x$ に対し $\sin\theta = \sin(\theta+2n\pi)$ という変形を適用する場合，$\theta$ が $x$ に置き換えられ，$\sin x$ は $\sin(x+2n\pi)$ に置換される．

## 3.4 システムのインタフェース

本システムのプロトタイプのインタフェースを図 3 に示す．

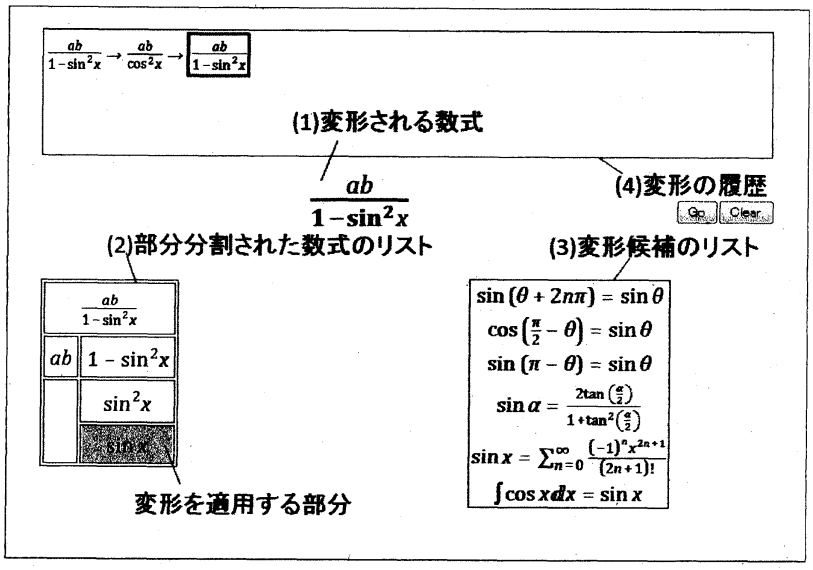


図 3: インタフェース

(1) にはまず，変形される数式が表示される．(2) は，(1) を分割して得られる数式のリストである．ここから変形したい数式をクリックすることで，変形候補の一覧 (3) が得

られる．図3の(3)では，$\sin x$をクリックした場合の例を挙げている．(3)のうち，変形に利用したい数式を選択することで，(1)の数式が変形される．(4)は，変形の履歴である．変形の履歴は，1ステップの変形が終わるたびに更新されていく．前のステップに戻りたい場合には，(4)のうちのいずれかの数式をクリックすることで，その数式が(1)に表示される．これにより，数式変形における試行錯誤を円滑に行うことが可能になると期待される．

# 4 証明の図示

数学における証明は，自然言語によって記述されることが多い．自然言語はその線状性により，情報が一次元的に表現される．しかし，証明は，2つの前提から1つの結論が推論される場合などがあり，その構造は単純ではない．このため，自然言語によって記述された証明から証明の論理構造を適切に解釈することは，初学者にとって容易ではないと考えられる．実際，証明を構造化して学習者に提示することで，教育効果が向上した例も報告されている[5]．そこで本研究では，初学者が理解しやすいように，構造化された形で証明を表現するための体系を提案する．

証明の図示では，ある命題から別の命題への推論について学習者が容易に理解できるようにするために，推論の根拠を明示する．それと同時に，証明における論理構造を木構造の形で示し，自然言語よりも直感的な形で論理構造を把握させることを目指す．

## 4.1 図の構造

次の定理の証明について考える．

$a, b, c$を自然数とする．

$a, b$がともに3の倍数でない時，$a^2+b^2=c^2$は成り立たない．

というものである．また，補題1として，

自然数$a$が3の倍数でないなら，$a^2-1$は3の倍数である．

を利用することを許す．

この証明を自然言語で記述すると，次のようになる．

> $a$と$b$はともに3の倍数ではない．補題1より，$a^2$と$b^2$をそれぞれ3で割った余りは1となる．したがって，$a^2+b^2$を3で割った余りは2である．一方，$c$について考えると，$c$が3の倍数の時，$c^2$も3の倍数であるため，$c^2$を3で割った余りは0である．$c$が3の倍数でない時，補題1より，$c^2$を3で割った余りは1である．$a^2+b^2$を3で割った余りは2であり，$c^2$を3で割った余りは0もしくは1であるため，$a^2+b^2=c^2$は成り立たない．

この証明を図によって表現したものを，図4に示す．

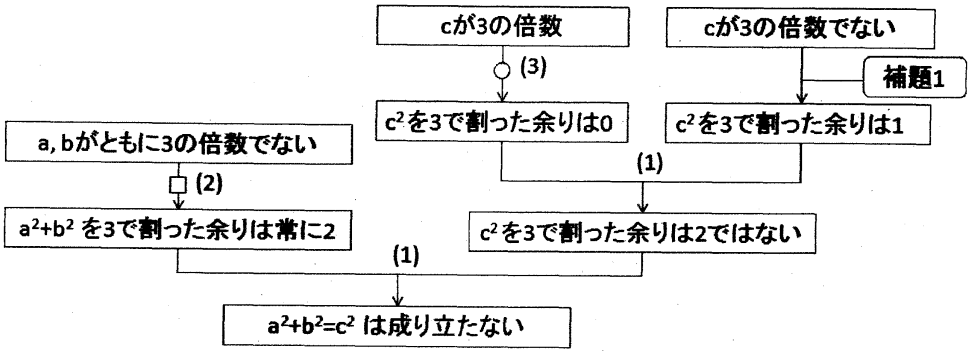


図 4: 証明の図の例

1つの命題は1つの矩形で囲まれる．矩形同士は線で結ばれる．この線は推論を意味する．位置的に上側に配置されている命題が前提，下側に配置されている命題が結論を意味する．命題を結ぶ線には，その推論規則をラベル付けすることができる．このラベルは推論の根拠を意味し，角が丸い矩形で表現される．図4の(1)で示した箇所のように，複数の前提から1つの結論を得るような推論も表現することができる．

複雑な証明になると，命題や推論の根拠の数が多くなり，図の視認性が低下する．このため，命題や推論の根拠をクリックすることで，それを一時的に折りたたむことができる．図4の(2)に示した箇所では命題を折りたたんでおり，(3)では推論の根拠を折りたたんでいる．

なお，図の構造は，シークエント計算[6]における証明図をその基礎としている．シークエント計算では，シークエント(本研究において命題と呼んでいるもの)として認められる形式が厳密に定められているが，本研究における図では，教師などのコンテンツ制作者の判断により，シークエント(命題)を自由な形式で記述することを許す．

## 4.2 SequentML

証明は，独自に定義したXMLボキャブラリを用いて表現される．そのXMLボキャブラリを用いて記述されたデータに変換を施すことにより，証明の図が得られる．証明を表現するためのXMLボキャブラリを，SequentML[7]と呼ぶ．SequentMLの基本的な記法について述べる．SequentMLによる推論のデータを示し，図5に，そのデータから得られる証明の図の例を示す．

```
<deduction>
  <sequent><math><power/><ci>c</ci><cn>2</cn></math>
          を 3 で割った余りは 2 ではない</sequent>
  <deduction>
    <rule><math><power/><ci>c</ci><cn>2</cn></math>も 3 の倍数</rule>
    <sequent><math><power/><ci>c</ci><cn>2</cn></math>
            を 3 で割った余りは 0</sequent>
    <sequent>c が 3 の倍数</sequent>
  </deduction>
  <deduction>
    <rule>補題 1</rule>
    <sequent><math><power/><ci>c</ci><cn>2</cn></math>
            を 3 で割った余りは 1</sequent>
    <sequent>c が 3 の倍数でない</sequent>
  </deduction>
</deduction>
```

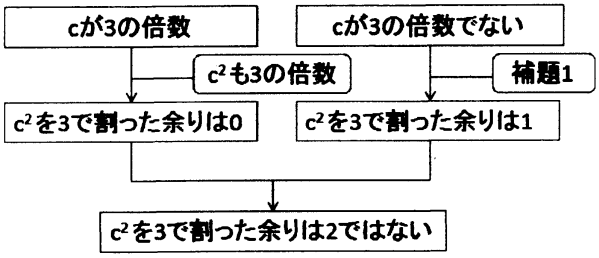


図 5: SequentML によるデータに対応する証明の図の例

中心的な役割を果たす要素は，deduction 要素である．1 つの deduction 要素が 1 つの推論を表す．deduction 要素の 1 つ目の子は，rule 要素である．rule 要素には推論の根拠が記述される．図の作成者が推論の根拠を示す必要がないと判断した場合には，rule 要素を省略することができる．次の要素は sequent 要素であり，これは推論の結論に相当する命題である．それ以降の子は，sequent 要素あるいは deduction 要素となる．これらは推論の前提にあたる．deduction 要素を入れ子状にしていくことにより，推論の列を表現することができる．SequentML のデータからは，それに対応する図を一意に定めることができるため，SequentML のデータから証明の図への自動的な変換が可能である．

## 5 おわりに

本論では，数学学習における導出というプロセスに着目した，数式変形支援，証明の図示という 2 つの研究についてその枠組みを示した．現時点では，数式変形支援に関してはプロトタイプの開発を進めており，証明の図示に関してはそのデータ形式の定義が完了しているという段階である．

本研究の今後の課題として，本手法による教育効果を確認しなければならない．数式変形支援に関しては，本稿で述べた機能は基幹的なもののみであり，学習者を対象とした実験を行うには不十分である．実験を行うためには，本稿に述べたものに加えて，2 つの機能を追加する必要があると考えている．1 つは，数を処理するための機能である．これは，端的に述べると，1+1 を 2 へと変形するというような機能である．もう 1 つは，

数式の部分を1つの識別子に抽象化する機能である．これは，「$\tan x$ を $t$ とおく」といった手順を実現するためのものである．これらの機能の実装には，代数的処理や，識別子の条件 (識別子が意味するものが関数であるか変数であるかといった種別や，識別子が値として取りうる範囲など) を取り扱う必要があるが，数式処理や制約プログラミングに関する研究成果を応用することで実現することができると考えている．

利用者に向けたユーザインタフェースの開発も必要である．数式変形支援に関しては，数式を容易に入力することができるような機能が求められる．数式エディタについてはさまざまな研究が行われているが，本研究では数式のセマンティクスを取り扱う必要があるため，独自の数式入力機構を開発することが望ましい．また，証明の図示に関しては，教師などのコンテンツ制作者のための，SequentML のエディタを開発することを計画している．

証明の図については，さらに表現力を高めるべきである．現時点での SequentML は，証明を記述することを目的としたものであるため，そこから得られる図も，単に証明を構造化しただけのものに留まっている．今後は，証明に関する周辺的な情報（例えば，場合分けの簡明な表記など）を記述するために，SequentML を拡張した XML ボキャブラリを定義することを計画している．

## 参考文献


[1] MathML, http://www.w3.org/TR/MathML3/.

[2] 出口幸子，小原啓義：「数式変形に関する誤りの解析」，電気情報通信学会技術研究報告. ET, 教育工学，Vol.95，No.14，pp.145-152, 1995.

[3] B. R. Miller，A. Youssef：「Technical Aspects of the Digital Library of Mathematical Functions」，Annals of Mathematics and Artificial Intelligence，Vol.38，No.1，pp.121-136, 2003.

[4] M. Q. Shatnawi，M. T. Alquran, F. M. Quiam：「Expanded Grammar for Detecting Equivalence in Math Expressions」，International Journal of Computer Applications，Vol.43，No.15，pp.44-51, 2012.

[5] J. Aczel，P. Fung，R. Bornat，M. Oliver，T. O'Shea，B. Sufrin：「Software that Assists Learning within a Complex Abstract Domain: the Use of Constraint and Consequentiality as Learning Mechanisms」，British Journal of Educational Technology，Vol.34，No.5，pp.625-638, 2003.

[6] G. Genzen：「Untersuchungen ber das logische Schlie $\beta$ en, I」，Mathematische Zeitschrift，Vol.39，No.1，pp.176-210，1935.

[7] T. Watabe，Y. Miyazaki：「Visualization of Logical Structure in Mathematical Proofs for Learners」，Computer and Information Science 2012，Vol.429，pp.197-208，2012.