ヒューマンインタフェース 61－8
（1995．7．20）

自然な数式ヒューマンインタフェースに関する研究

趙燕結* 桜井鉄也† 杉浦洋‡ 鳥居達生‡
* 久留米工業大学 知能工学研究所
† 筑波大学 電子・情報工学系
‡ 名古屋大学 工学部

概要

代数計算と数値計算のための汎用ヒューマンインタフェースを実現した．これによって，自然な数式の入力，ディスプレー，及び計算の実行が直接できる．また，数式の文書構造と意味解釈両方のための汎用形式表現，及び実行と独立するための共通の意味表現を定義した．これらに基づき，科学文書整形と数学計算を密接に統合した．

A Research in Natural Mathematical Expression Human Interface

Yanjie Zhao* Tetsuya Sakurai† Hiroshi Sugiura‡ Tatsuo Torii‡
* Intelligence Engineering Laboratory, Kurume Institute of Technology
† Institute of Information Sciences and Electronics, University of Tsukuba
‡ Department of Information Engineering, Nagoya University

Abstract

We realize a universal human interface of algebraic and numerical computation to input, display, and calculate natural mathematical expressions directly. We define a universal abstract representation for description of both the document structure and the meaning understanding of mathematical notation, and also define a common meaning representation being independent of different executions. Based on them, we combine scientific document preparation with mathematical computation closely.

* zhao@cc.kurume-it.ac.jp † sakurai@is.tsukuba.ac.jp ‡ sugiura@urus.torii.nuie.nagoya-u.ac.jp

1. 目的と問題

これまで，グラフィックス環境の発展にもかかわらず，数学計算用コンピュータ言語の表現力は，キーボードから入力される文字セットに制限されており，数式の記述と実行できるプログラム記述の間にまだ大きなギャップが存在している．

数式を記述表現のままで一般のプログラム (FORTRAN 等) の中に挿入することにより，計算処理を可能とするようなプログラム言語の開発は１９６１年より行われている．いままで，幾つかの分野 (高級言語設計，パターン認識，文書処理，ソフトウェア仕様言語，数学ソフトウェアのインターフェイス) で，様々な成果をあげた．最近１０年間にこの研究はもう一度活発化している．しかしながら，数式の構文と意味解釈を可能とするような自然な”数式ヒューマン・インターフェース”については，現在までに多数の研究や実用システムの提案，あるいは商品化されたものも存在するが [1]，まだ十分な状態であるとはいえない．

我々の研究は，数式を含んだ一つのコンピュータ言語を設計するのではなく，数式自体の意味と構造を研究し，数学計算と数式文書処理両面にわたる入力編集インターフェースと意味解釈メカニズムを構築することにある．これにより，様々な応用分野 (数値計算，代数計算，定理証明，オープン公式ライブラリー，算法の表現，数学の教育，科学文書処理など) に応用することである．

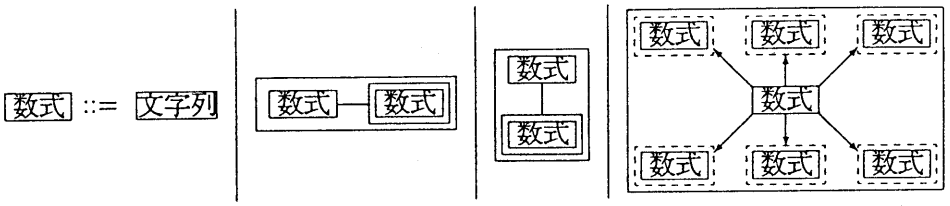
現存のシステムと比べて，自然な数式は高い表現力，普遍性，読みやすさや，拡張性などを持つ．現存システムには下記の問題点があると考える．

1. 数式の入力編集と意味解釈の両方のための普遍的な抽象表現がない．また，数式の記号法の使用も制限されている．
2. 数式表現中において多用する各種字体がほとんど扱えないため，多種な数学対象と概念を表現し難い．
3. 数式のディスプレイ品質は低いので，表示された数式の可読性は低い．
4. 数式の構文解析，意味解釈と記号法拡張に関する研究成果がまだ少ない，もしくは不十分である．
5. 本来の数式表現ではない表現手法を強いられる場合が存在する (例えば，かけ算記号を省略できない，$\sin^n x$ のような表現ができない，$f(x)$ と $f[x]$ の区別ができないなど)．

これらの問題を解決するために，我々は形式表現という普遍的な抽象表現を定義し，これを用いた数式入力インターフェイスを試作した．更に，知識ベースの手法によって数式の構文解析と意味解釈を実現することで，直接の数式入力から各種プログラミング言語上で実行可能なプログラムに変換することができるような数学計算のヒューマン・インタフェースを基本的に作成した．

2. 形式表現

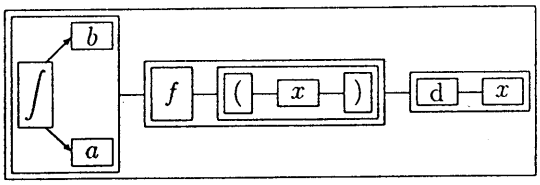
数式の印刷イメージは記号の位置関係の記述によって得られる．この記述を抽象して，下記の形式表現 (formal representation)[2][3] という表記を定義する．



中には，数式はボックス □ を占めて，ボックス中の任意の数式はアトム・ボックス中の文字列から横並び，縦並び，六本木という三種類の構造によって構成する．二重ボックス ▣ は同じ構造の繰り返しを意味する．⬚ は構造のオプションを意味する．

形式表現は，数式の印刷イメージから，意味解釈と関係ない情報 (記号の大きさや，記号の位置調整や，各種な空白や，色々な組版など) を無視して，抽象された表現である．形式表現によって，全ての数式の記号及び各記号のつながりや，意味の限界などを同時にコンピュータ上に入力，保持できる．形式表現に従った数式入力は，意味に従った数式を構成できるため，人間にとって自然である．

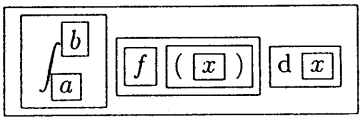
形式表現の例 $\int_a^b f(x)\mathrm{d}x$ を示すと，下記のようなものである．



形式表現の目的は，数式全体の形式化，数式構造の描写，数式文法の記述，数式意味の限界の表示，及び数式入力用テンプレート (Template) のモデルである．

3. 形式表現の入力

上例のような形式表現の入力は数式の WYSIWYG のような入力ではない．また，ボックスを多用する．これらのために，下記のような実際の数式に近い"準形式表現"を用意する．



準形式表現の表示では，ボックス構造の表示は最小限にする．ボックス間の連結線とパターン中の終端記号のボックスも表示しない．

準形式表現は人間の意図を表現するための入力と出力インターフェイスになる．形式表現は意味解釈のための数式の内部表現になる．

4. 入力方法

数式準形式表現の入力方法は下記の3種がある．

1. テンプレート (Template) によって入力する．例えば，$\square^{\square} \Rightarrow \boxed{x}^{\square} \Rightarrow x^{\boxed{y}} \Rightarrow x^{y}$．
テンプレート・ライブラリーに依頼や，不自然な入力手順などが研究問題になる．

2. オーバーレイ (Overlay) によって入力する．例えば，$x \Rightarrow \boxed{x}^{\square} \Rightarrow x^{\boxed{y}} \Rightarrow x^{y}$．
テンプレート中に未入力所の入力順序の決定や，テンプレート・ライブラリーに依頼などが研究問題になる．

3. 指定位置で挿入する (Insert)．即ち，既入力の式中に，記号や，テンプレートなどを挿入する．

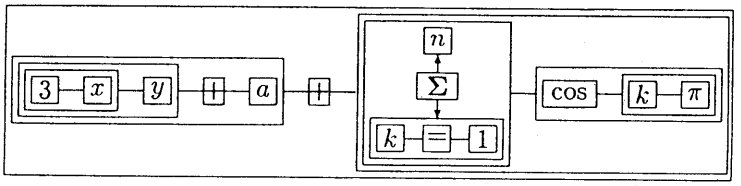
更に，数式の表現力を高めるために１２種なフォント字体も導入して，フォントから数式への構成を実現した．これらの入力方法と，フォント・ライブラリーから構築したテンプレート・ライブラリーによって，文書処理と意味解釈の両方のための数式入力ができ，入力した準形式表現から形式表現を抽出し，意味解釈を行う．

5. 入力と意味解釈の関係

数式の入力単位は，逐一の文字入力，構造として入力 (例えば，$\sin\square$, $\square \times \square$ など)，概念によって入力 (例えば，実行列のかけ算，複素定積分など)，という3種がある．数式の入力は，まず少なくとも，逐一の文字入力と構造として入力が必要である．本研究はいままでこれらの二つの入力と意味解釈を実現した．逐一の文字入力と構造として入力は，文字の体や，組版などの知識をある程度知っていることをユーザに強いる．しかし，ユーザは表現手法を自由に選ぶことに限りない．

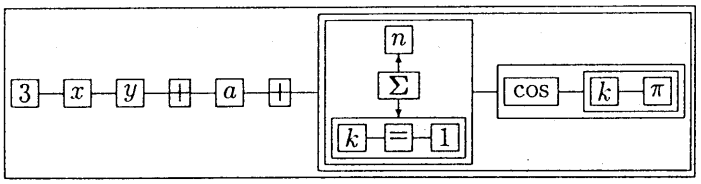
形式表現は，数式の構造を明らかに示す程度によって，下記の3種に分けることができる．ここで，"$3xy + a + \sum_{k=1}^{n} \cos k\pi$" を例として示す．

強形式表現 では，全ての構造を必ずボックスで明らかに示す．全ての数式は，構文解析の時，唯一的な構文変数 "式" から導出されることができる．そして，ユーザからの拡張も容易にする．強形式表現は暗黙がないように数式の一般的な形式表現であるため，厳密な意味解釈ができる．上例の強形式表現は下のようである．

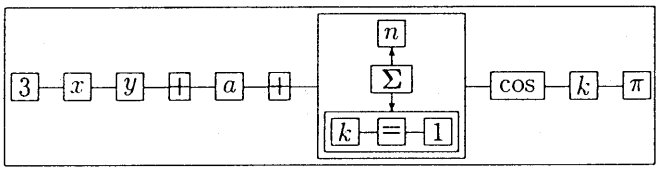


弱形式表現 では，日常計算中の演算子の優先度や，結合法則などを暗黙規則として仮定したので，"アトム"と"因子"だけを必ずボックスで明らかに示す (因子内の演算

子の優先度を仮定しない)[2][3]．全ての数式は，構文解析の時，7種の構文変数(アトム，因子，項，加算式，関係式，論理式，及び式)から導出されることができる．弱形式表現は日常計算のため意味解釈を行える．弱形式表現の横並び一部の入力はキーボードからもできる．上例の弱形式表現は下のようである．



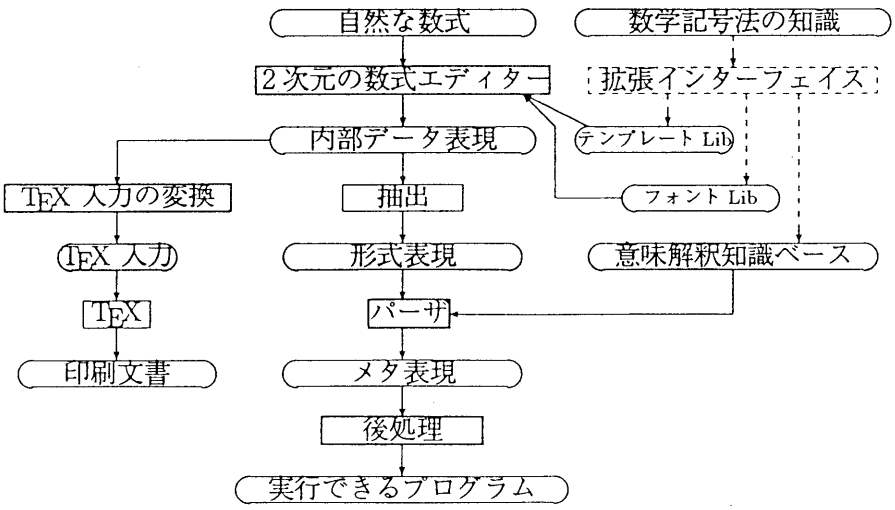
自由形式表現 では，因子内の演算子の優先度と他のいろいろな暗黙規則も仮定して，"アトム"だけを必ずボックスで明らかに示す．分野によって他の仮定があるかもしれない．構文解析の時，もっと多種な構文変数が必要である．キーボードから，横並びのもっと多い部分を入力できる．しかし，記号法を限らないならば，意味解釈ができない可能性がある．上例の自由形式表現は下のようである．



我々は弱形式表現と強形式表現の入力と意味解釈を実現した．自由形式表現の入力と意味解釈，及びユーザの素早い入力方法は，今後の研究課題になる．

6. 入力と計算の統合

知識に基づいた数式の意味解釈を中心として，下に示す数式の文書処理と意味解釈，及びその計算の実行を統合するシステムを構築した．



数式のメタ表現 (meta-representation)[2][3] という共通の意味表現を下記のように定義する.

```
function: <関数名>( メタ表現 , ···, メタ表現 )
domain: <領域名>
```

例えば，前例のような実数領域上の定積分式のメタ表現は下記のように表される.

```
function: integral(
            function: integral_element(□)
            domain: set_of_real_numbers ,
            function: lower_bound(□)
            domain: <subset_of_real_numbers> ,
            function: upper_bound(□)
            domain: <subset_of_real_numbers> ,
            function: integrand(□)
            domain: set_of_real_numbers )
domain: set_of_real_numbers
```

このようなメタ表現はデータ型，算法，プログラミング言語などとは独立している．数式の意味解釈は，形式表現からメタ表現までの翻訳を意味する.

このシステムの中で，**後処理**はメタ表現からある言語の実行できるプログラムまでの変換である．このためにデータ型と算法を選択することも含んでいる．後処理に対しては，更に深く研究が必要である．これまで，後処理は，メタ表現が実行できることを確認とテストするために，メタ表現から *Mathematica* プログラムへの変換をした．知識ベースと Parser，及び後処理を *CESP*[4](オブジェクト指向 PROLOG) で実現した．共同のフォント・ライブラリーに基づいて，入力インタフェースも *CESP* で Fig.1 のように一つ実現したが，もう一つの入力インタフェースを *Tcl/Tk*[5] で Fig.2 のように実現した.

手短にいえば，自然な数式ヒューマン・インタフェースでは，自然な数式を準形式表現として 2 次元の数式エディターによって入力して，形式表現という内部表現になって，意味解釈を経て，メタ表現になる．後に，後処理によって，各種な言語の実行できるプログラムへ翻訳する.

テスト対象とする数式を公式集 [6] より抜粋した．これまでに，約１００例についてテストを行い，その結果，２例 ([6] 中の 3.963-4 と 3.963-3) の誤植を確認した．Fig.1 中の "Calculation 2" はこれらの一つの訂正例である．このような誤植が存在するという事実は，今後オープン公式ライブラリー構築の必要性を示唆するものと考える．貴重な公式集は，科学研究とシステム構築のために，大変重要な道具である．分厚い公式集の誤植のチェックは大変難しい．何年間に一回再版の時，多数の誤りは見つけられた [6]．沢山の公式を，電子図書館の一部分として，コンピュータで処理すれば，公式の検索が速いし，公式の訂正と追加も速い．公式表記の正規化も推進できる．これは将来本研究の重要な応用の一つと考える.

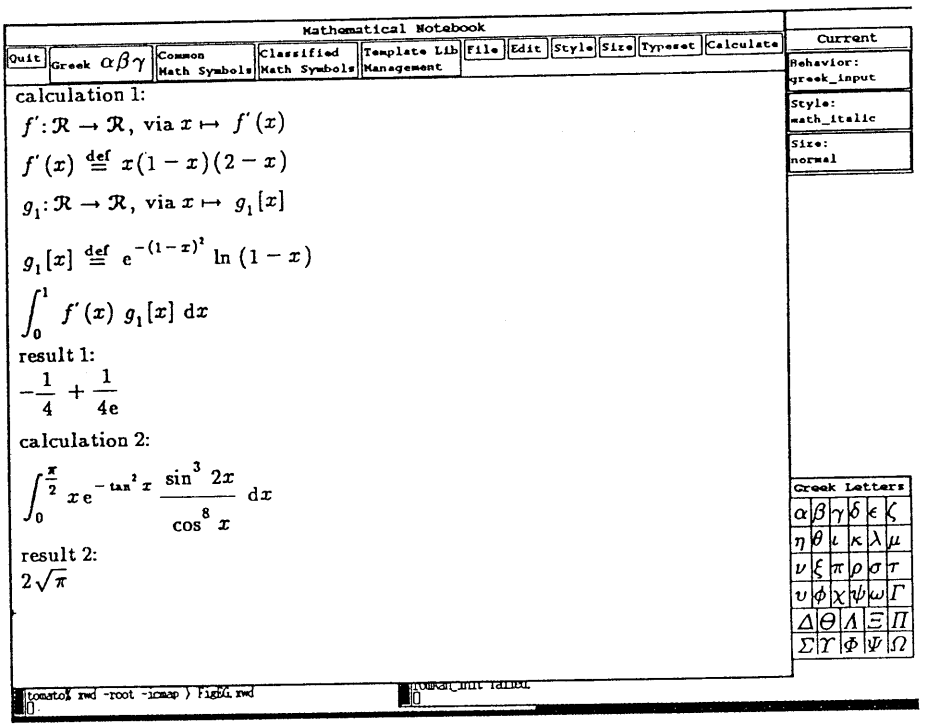


Fig.1 *CESP* で試作したインタフェース

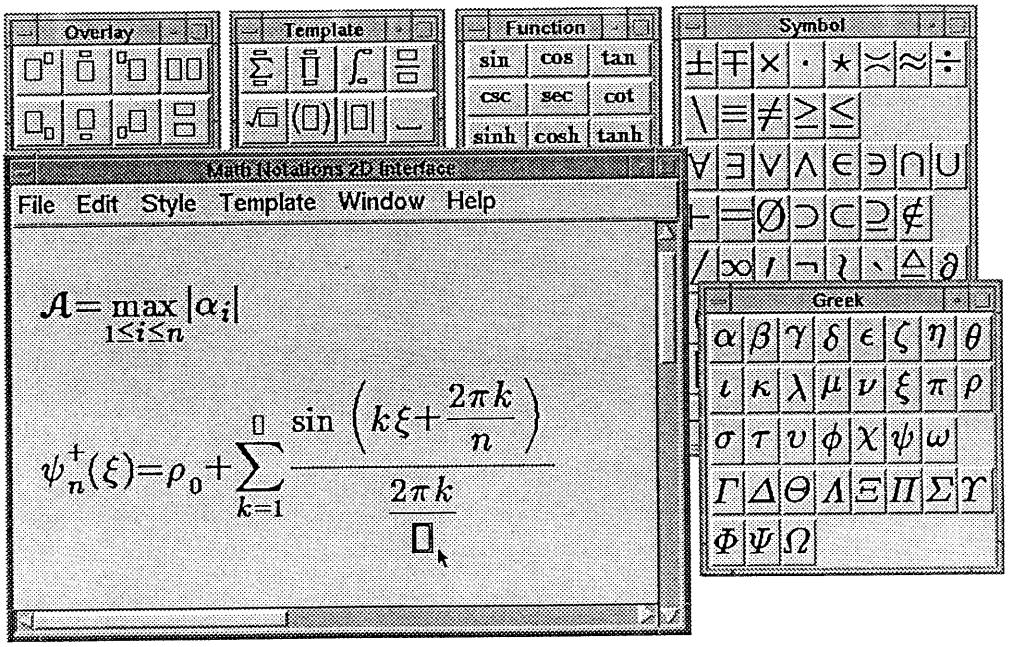


Fig.2 *Tcl/Tk* で試作したインタフェース

References

[1] Kajler,N. & Soiffer,N.: A Survey of User Interfaces for Computer Algebra Systems (to appear in *the Journal of Symbolic Computation*. Preprint is available as RIACA Technical Report #1) 1994.

[2] Yanjie Zhao, Hiroshi Sugiura, Tatsuo Torii, & Tetsuya Sakurai: A Knowledge-Based Method for Mathematical Notations Understanding. *Transactions of Information Processing Society of Japan*, Vol.35, No.11, pp2366-2381, Nov.1994.

[3] 趙燕結, 杉浦洋, 鳥居達生, 桜井鉄也: 知識ベースによる数式の意味解釈とその応用. 情報処理学会研究報告, Vol.94, No.83, pp1-10. 1994 年 10 月.

[4] *CESP A00.* AI 言語研究所, 〒 247 神奈川県鎌倉市大船 5-1-1 三菱電機 (株) 1994.

[5] Ousterhout,J.K.: *Tcl and the Tk Toolkit.* Addison-Wesley, 1994.

[6] Gradshteyn,I.S. & Ryzhik,I.M. (Jeffrey,A. Ed.): *Table of Integrals, Series, and Products.* Academic Press, 1994 (訳本: 数学大公式集. 大槻義彦 訳. 丸善, 1983).