数理解析研究所講究録
第1568巻 2007年 182-184

数式入力のための動的な入力インタフェースの紹介

出口博章
DEGUCHI HIROAKI
神戸学院大学
KOBE GAKUIN UNIVERSITY *

1 はじめに

数式をコンピュータ・ディスプレイ上で扱うための方法については、文献 [1] によると 1960 年代から議論されてきているが、どのように数式の二次元構造を扱うかということは未だに解決していない問題の一つである。例えば、様々な機能を持つワープロに対しては、最小限の機能だけを備えたプレーン・テキスト・エディタが存在するが、数式の入力・編集において「最もシンプルな形態はどのようになるか」という問いに答えるためのシステムは既製品においては存在していない。

従来の GUI 環境において、数式処理システムやワープロを数式入力のために使用する際は、テンプレートを利用することによって二次元構造を記述することが一般的となっている。テンプレートの利用によりテキスト入力可能なボックスが二次元の位置関係を持って配置され、利用者はそのボックス内にテキスト入力することにより数式を構成していくことになる。テンプレート利用に基づくシステムにおけるテキスト・カーソルの様々な挙動について、文献 [5] に記述があるが、統一的な操作方法は存在しておらずシステムによって異なる挙動となっている。つまり、テンプレートによる数式入力が広く普及しているようには見えるものの、利用者にとっては、統一された入力インタフェースが提供されているとは言い難いという状況である。

2 MathBlackBoard

我々は、ポインティング・デバイスのみで利用可能な「使いやすい」数式処理システムとして提案された黒板アプレット [4] を拡張し、数式処理システムのユーザ・インタフェースとして利用可能な MathBlackBoard[2] を開発してきた。MathBlackBoard において数式はドラッグ&ドロップによって編集画面上に自由に移動可能であり、ワープロにおけるテキストのようにベースラインに縛られることはない。

黒板アプレットや初期の MathBlackBoard では数式をポインティング・デバイスで直感的に扱うことが可能であったが、オブジェクトの単位はひとかたまりの数式であり、数式オブジェクト内部においては一次元のテキストを用いて数式が表現されていた。そのため、これらのシステムにおける数式の編集操作にはテキストの一次元構造に由来する限界があった。具体的には、例えば、数式 A を別の数式 B に接続する場合には、数式 B の後に数式 A（あるいは数式 A の左辺または右辺）を接続することのみが提供されていた。また、入力済みの数式を訂正する際には、数式オブジェクト内部に保持された一次元のテキストを最後の文字から順に消していくという操作が必要となった。

*deg@human.kobegakuin.ac.jp

そこで我々は、MathBlackBoard をゼロから再構築し、MathBlackBoard ver. 2 を新たに作成した。MathBlackBoard ver. 2 においては、数式がオブジェクトの集合として構成され、オブジェクトは文字や記号（以下、シンボル）ごとに用意されている。それ以前のものと比較して、オブジェクトの単位が、操作の単位であるシンボルと一致するため、数式の編集操作はオブジェクトの操作として後から拡張可能となった。

また、MathBlackBoard ver. 2 の作成に先立って、オブジェクトの集合としての数式を扱うための新たな入力インタフェース [3] を考案し、特許出願を行なっている。この新入力インタフェースによって、ポインティング・デバイスによって得られる二次元情報をより有効に利用して、数式の二次元構造を扱うことが可能となった。

3 Dynaput 操作

MathBlackBoard の新たな入力インタフェースにおいては、ドラッグ&ドロップ操作によってオブジェクトを組み合わせていくことにより入力・編集を行なっている。入力操作はパレットからオブジェクトをドラッグすることにより行ない、編集操作は入力済みのオブジェクトをドラッグすることによって行なうため、入力操作と編集操作が同じ GUI 操作として提供されている。また、オブジェクトのドラッグ中には、ドロップ後の結果がプレビュー表示されるため、ポインティング・デバイスの移動により画面表示が動的に変化する。以上のように、入力・編集操作が同様の操作で提供され、また動的なプレビュー表示を備えていることから、この操作を単なる入力（input）操作とは呼ばずに、dynaput 操作と呼ぶことにした。dynaput とは、dynamic と input（または put）からの造語である。

3.1 テンプレートとの比較

dynaput 操作による入力では、オブジェクトはパレットから編集エリアまでドラッグされ、その後ドロップされる。詳細には、(1) 入力対象のシンボルを選択し、(2) そのオブジェクトを入力位置までドラッグし、(3) 目的の位置でドロップする、という一連の操作になるが、その際には画面表示が連続的に変化する。このように入力・編集操作の間の画面が連続的に変化するということが、dynaput 操作における一つの特徴となっている。

従来のテンプレートを利用した入力においては、テンプレートの選択を行なうパレットと編集エリアはピクセルを隔てて離れており、その間を連続的につなぐものは存在していなかった。入力操作においては、(1) テンプレートの選択をクリック操作によって行なうと、(2) 離れた位置である編集エリアに直ちに操作の結果が現われる。その際、コンピュータに慣れた利用者であれば、パレットと編集エリアの関係を把握しているため、パレットから編集エリアへと視線や意識をスムーズに移動させることが可能であるが、コンピュータ初心者にも同様のことを求めることはできない。

一方、dynaput 操作における連続的な画面変化は、利用者の視線や意識をスムーズに導くことに役立つため、初心者が「これから何が起こるのか」を理解することを助ける要素となる。以上のように、dynaput 操作のために用意されたパレットは、テンプレートのような間接的な操作のために用意されたものではなく、直接的に操作されるオブジェクトを配置するために用意されたものである。ただし、パレット上のオブジェクトをクリックすることによって、ソフトウェア・キーボードのように利用することも可能であるため、ある程度システムを把握したユーザが間接的な入力操作のために利用することも可能となっている。

3.2 手書き文字認識との比較

dynaput 操作は、ポインティング・デバイスのみで利用することを前提としているが、ポインティング・デバイス利用によって数式を入力する方法には、手書き文字をオンラインで認識すること（以下、手書き文字認識）を利用したものがいくつか提案されている。手書き文字認識による入力においては、ポインティング・デバイスによって入力されたストロークを解析することにより、利用者が入力しようとしたシンボルが推測され、候補が決定される。そのため、得られたストロークに関する情報の量に応じて、入力されるシンボルとなるべき候補を絞り込む精度が向上する。しかし、扱うシンボルの種類が多くなるに従って、ストロークの微妙な違いに多くの解釈を与えたり、シンボルの前後関係による判定を必要とするなど、処理は複雑になる。

一方、dynaput 操作においては、入力対象となるシンボルがパレット上に用意されていれば入力が可能となる。必要となるのは、事前にシンボルがパレット上に用意されていることである。シンボルに対しての準備を整えておく必要があることは、どちらのシステムでも同じことであるが、利用時の処理では大きな違いがある。手書き文字認識では、より多くの情報を得るためにストローク全体の解析が必要であることに対して、dynaput 操作ではドラッグ開始位置とドロップ位置の二点の座標のみがあれば十分である。ドラッグ開始位置によって「どのオブジェクトが入力・編集対象なのか」が決定され、ドロップ位置によって「どの位置に配置されるのか」あるいは「どのオブジェクトとどのような関係で接続されるのか」が決定される。その処理は、ポインタの示す座標が、ある領域に含まれるか否かという条件分岐をいくつか組み合わせたものとなり、手書き文字認識の際の処理と比較すると極めて単純な処理となる。

4 まとめ

dynaput 操作を入力操作として考える場合は、テンプレートや手書き文字認識などの他の方法と比較すると、直接操作であって、処理が単純であることが分かった。事前にシンボルを用意しておく必要がある点や、その配置に工夫が必要である点など、従来の GUI におけるメニューなどと同様の課題は存在している。

ただし、dynaput 操作を編集操作として考える場合は、従来の他の方法との融合も可能である。キーボード操作や手書き文字認識によって入力されたシンボル、あるいはそれらシンボルが集まってできた数式に対して、追加や削除などの変更を加える場合に、それらシンボルが dynaput 操作可能なオブジェクトであれば、ポインティング・デバイス操作による直接的な編集が可能となる。

MathBlackBoard のみで、完結したシステムを目指して開発を続けるということと併せて、従来の方法との融合という観点でも開発を進めていきたい。

[1] Kajler, N., Soiffer, N., A Survey of User Interfaces for Computer Algebra Systems, *J. Symb. Comput.*, **25**(2), pp. 127–159, 1998.

[2] 出口博章: MathBlackBoard, 数式処理, **11**(3,4), pp. 77–88, 2005.

[3] 出口 博章, 数式入力のための動的な入力インタフェース, ヒューマンインタフェースシンポジウム *2006* 論文集, pp. 627–630, 2006.

[4] 松嶋純也: *Java* を用いた使いやすい数式処理システム, 神戸大学大学院教育学研究科修士論文, 1998.

[5] Padovani, L., and Solmi, R.: An Investigation on the Dynamics of Direct-Manipulation Editors for Mathematics, *MKM 2004*, Springer LNCS **3119**, pp. 302–316, 2004.