Dynaput: Dynamic Input Manipulations for 2D Structures of Mathematical Expressions

Deguchi Hiroaki

Kobe University, 3-11 Tsurukabuto, Nada-ku, Kobe 657-8501, Japan
deg@main.h.kobe-u.ac.jp
http://wwwmain.h.kobe-u.ac.jp/MathBB/

Abstract. This paper describes a prototype of input interface of GUI for mathematical expressions. Expressions are treated as sets of objects. To handle 2D structures, we have added new areas, called "peripheral area," to the objects. Based on new operations for these areas, the system provides dynamic interactive environment. The internal data structure for the interface is also presented in this paper. The tree structure of the objects is very simple, and has high potentiality for being converted to a variety of formats. Using this new dynamic input interface, users can manipulate 2D structures directly and intuitively, and they can get mathematical notations of wished format.

Keywords: GUI, input interface, direct manipulation, text entry, pen-based computing.

1 Introduction

How to handle mathematical expressions on screens of computers has been discussed from 1960's[1]. While many systems have been proposed, there are no easy-to-use systems for beginners. Users must choose "not too bad(wrong) one" from existing systems. One of the reasons for the situation is that APIs of the existing systems are not designed adequately in consideration of handling 2D structure.

For example, a cursor(caret) of GUI text editors has 2D coordinates, but the information is not used effectively. A Cursor is placed on or before or after one of characters. Because of texts' linear structure, the position of a cursor is able to be converted to 1D information. Therefore, although the cursor of GUI text editors looks like 2D cursor, it is 1D cursor in practice. And, many systems which handle mathematical expressions are based on this type of cursor models.

1.1 Input Interfaces of Computer Algebra Systems

Computer algebra systems handle mathematical expressions, but their input interfaces are generally based on template models, such as word processors' equation editor part which are treated like sub programs. An equation editor with

Y. Shi et al. (Eds.): ICCS 2007, Part II, LNCS 4488, pp. 243–250, 2007.

template model is able to handle mathematical expressions by using templates of 2D structures. And the input interface of the system is designed under the influence of a paradigm based on the cursor model as mentioned above.

Templates for 2D structures have box structure, each box of which is designed for cursor model based input interfaces. Box structures can be nested inside other boxes. 2D structures are constructed from such nested boxes. Usually these boxes are displayed on the screen. There are symbols users would like to display, and boxes these systems have to display, on the screen. Such systems are not intuitive, because of these boxes users don't necessarily wish to display. Therefore, user interfaces of many computer algebra systems are not suitable to handle 2D structures.

As for input devices, template based systems require both pointing devices and keyboards. Keyboards are required for inputting texts, and pointing devices are required for selecting templates of 2D structures. It is not easy-to-use that users are forced to use two or more devices.

1.2 The Present Work

The goal of this work is to provide easy-to-use environment to novice users, such as students studying mathematics of primary (or higher) education. In our former researches[2,3,4,5], we have developed MathBlackBoard as a user interface of computer algebra systems. In MathBlackBoard, only a pointing device is required, and keyboards are not necessarily indispensable devices. And mathematical expressions located in the editing area of MathBlackBoard are able to be dragged and be dropped. The user interface of MathBlackBoard is easy-to-use at least for students of junior high schools[6,8].

This paper describes a new input interface[7] which replaces template based interfaces. In this paper, we introduce a dynamic input interface and its data structure. The interface has patents pending in Japan. The new version of MathBlackBoard(Fig. 1) has been developed as a prototype of the interface. In the Section 2 of this paper, the new model of GUI operations is shown, and operations of existing systems and of the new system are discussed. In the Section 3, data structure for dynamic input interface is described. Finally the Section 4 describes a conclusion.

2 GUI Operations

Using computers with GUI, users can manipulate objects on screens directly. Especially, drag and drop operations are intuitive. However, it is hard to say that they are effectively utilized in interfaces treating mathematical expressions. For example, in some systems, to connect a selection to a target, users can select and drag the selection, but can drop them only onto prepared boxes related to the target. These boxes have to be prepared by using templates before drag and drop operations.

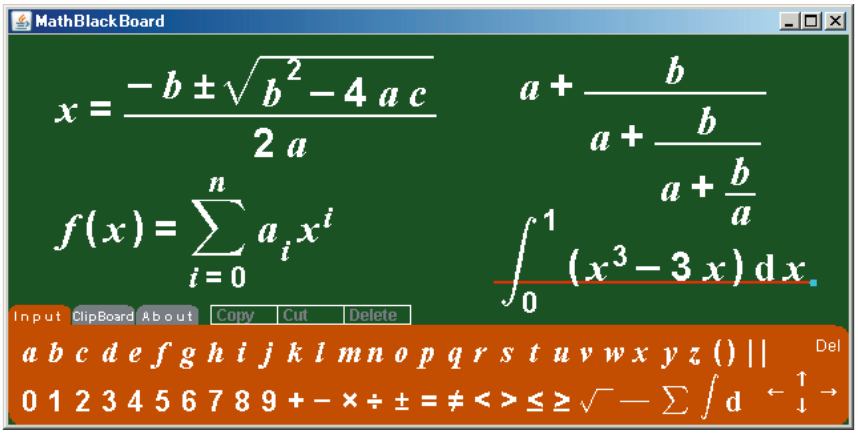


Fig. 1. MathBlackBoard

2.1 Drag and Drop

In general, drag and drop means "drag an object and drop it to different places or onto other objects." Each object on GUI screens has its own selection area which is used to be selected or to be dropped onto. And when a dragged object is released by manipulation of pointing devices, all of the selection areas are scanned. If the pointer lies inside the selection area of any object, the dragged object is dropped onto such pointed object. Otherwise, the dragged object is dropped to the place marked by the pointer.

Using these operations, users can drag and drop expressions onto a box prepared or into an insertion point which is used in linear structure of texts. Insertion points have 1D information (or convertable to 1D information) of their position, and are used with text stream, where text includes characters and symbols. When a dragged object is released, all of the boxes are scanned. And then, if a box is pointed, all of the insertion points in the box are scanned.

In the model of templates and their box structures, that is used with drag and drop operations generally, mathematical expressions are constructed from nested boxes which contain other boxes or text stream as contents.

2.2 Drag and Draw

As mentioned above, existing drag and drop operations and template models are not suitable for 2D structure of mathematical expressions, because a mathematical formula is constructed as box-structured texts.

To extend drag and drop operation, we have added new elements to objects. The most important element added is area for target of drag and drop operations. Notice that the areas are used to be targets of drag and drop, not to be box-structured containers. These new added areas are called "peripheral area." Left

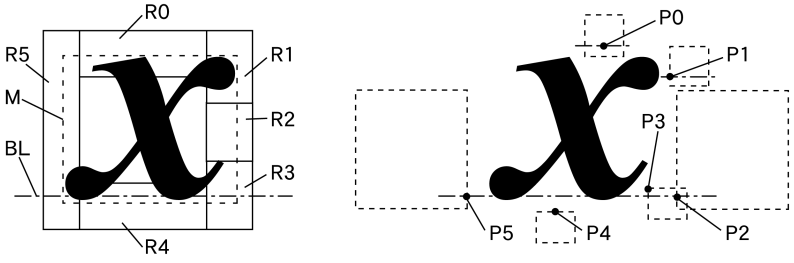


Fig. 2. Left: A symbol object "x" with a selection area, peripheral areas, and a baseline. **Right:** Base points and their link relations.

hand of Fig. 2 shows the symbol object "x" with M as a selection area, R0–R5 as peripheral areas, and BL as a baseline.

By using objects with peripheral areas, users can drag mathematical expressions and drop them onto the peripheral area of the target, for connecting the selection to the target with the desired position. Each object of the system has information of its connected child-nodes. And, it has information of location and display size of its child-nodes. Right hand of Fig. 2 shows the symbol object "x" with P0–P5 as base points. Each base point of P0–P5 is related to each peripheral area of R0–R5. Base points are used as link points to connect their child-nodes. For each base point, information of link relation for its child-node is configured(right hand of Fig. 2). Size of a dotted line square means child-node's display size. And relation between a dotted line square and related base point means the relative point of the child-node.

The operations for objects with these new elements are able to be explained as drag and drop operations. But, in this case, drag and drop is an operation between the selection and the peripheral area of the target. It is not an operation between the selection and the target itself. Therefore it could be a new GUI operation. In case the selection is dropped onto peripheral areas, the GUI operation is called "drag and draw." The drag and draw operation is an operation between the selection and the target. In this case, drag and draw means "drag the selection and draw it near to the target." The system with "drag and draw" operations is suitable for mathematical expressions, because objects are associated with other objects in the nature of 2D structures by using peripheral areas and their related elements.

2.3 Dynaput Operation

The "drag and draw" operations are suitable to handle 2D structure of mathematical expressions. And there are other elements to extend GUI operations for beginners. The new elements are feedbacks from drag operations and feedbacks from draw operations.

Feedbacks from drag operations should show what object is dragged. Such information helps novice users to know which object is selected and is dragging. MathBlackBoard is the system which provides this kind of information, from the early stages of development.

Feedbacks from draw operations also show useful information, categories of which are where the selection object will be connected to, and which size the selection object will be changed. With information from feedbacks, users can check the place where the dragged object will be connected to, or the preview of size, before dragged object is released by manipulation of pointing devices. Thus users can check results of various cases by moving pointing devices, without carrying out determination operation that is "to release the pressed button of the pointing device" in general. If the preview is what the user wished, determination operation is to be performed by manipulating pointing devices. After the determination, the selection is connected to the target, and is placed on the desired position with the previewed size.

The environment with these two kinds of feedbacks provides interactive dynamic input interface. In the dynamic input interface of MathBlackBoard, operations include not only an inputting aspect but also an editing aspect. Users can input expressions and edit expressions in the same way. For example, input operation in MathBlackBoard is performed as follows: drag the object in the palette area of Fig. 1 and drop it onto other objects, peripheral areas, or the blackboard area of Fig. 1. And edit operation is performed as follows: drag the object in the blackboard area and drop it onto other objects, peripheral areas, or the blackboard area.

Since same operations in the user interface mean both of inputting and editing, as described above, that could be new GUI operations. Because the operations are not simple input operations but dynamic input operations, the operations are called "dynaput" instead of input. "Dynaput" is a coined word combining "dyna" (from "dynamic") and "put" (from "input" or "put").

3 Data Structure

3.1 Layout Tree

Symbol objects for dynaputting are structured as tree. Each node has information of numbered direction for linking to child-nodes(left hand of Fig. 3). In Fig. 3, the thick lines mean the default direction. In this case, the default direction is "2." Right hand of Fig. 3 shows a layout tree of the following expression:

$$f(x) = \sum_{i=0}^{n} a_i x^i$$

The structure of layout tree is based on the layout of symbols. Mathematical semantic representation is not used in the layout tree structure. Our strategy for handling expressions on computer screens is that "what it means appears when you need." The selected tree is traversed when the user invokes a command.

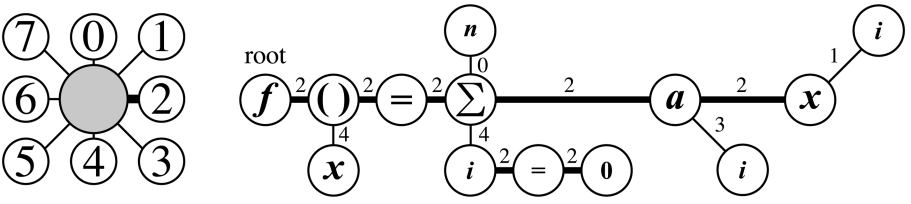
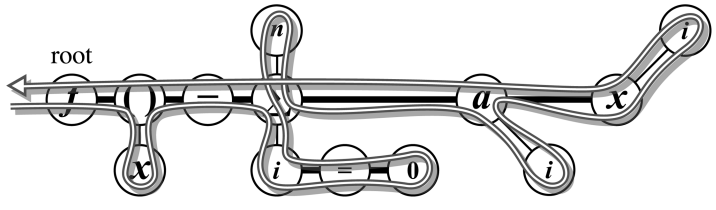


Fig. 3. Left: Numbered directions. (The thick line means the default direction.) **Right:** An example of layout tree. (The numbers beside lines mean directions.)

The layout trees can be converted easily to presentation markup languages, such as TeX and MathML. Fig. 4 shows tree traversal and TeX output. The child-node in default direction is visited after all of child-nodes in other directions are visited. Parent-node performs pre/post processes. For example, when it has child-node in the non-default directions, the parenthesis "{" is outputted before visit, and "}" is outputted after all child-nodes in the direction are visited. The outputs of other symbols (such as "^" or "_") depend on conditions of the content of the symbol object and the direction number of the child-node.




```
f ( x ) = \sum_{i=0}^{n} a_{i} x^{i}
```

Fig. 4. Tree traversal and TeX output

The MathML (Presentation Markup) output is similar in method to the TeX output. The "∑" part of a MathML output example is as follows:

```
<munderover>
  <mo>&sum;</mo>
  <mrow>
    <mi>i</mi>
    <mo>=</mo>
    <mn>0</mn>
  </mrow>
  <mrow>
    <mi>n</mi>
  </mrow>
</munderover>
```

Layout tree traversal with outputting text characters is the common method of these. The tasks for outputting are simple because of the simple structure of layout trees. Thus layout trees are converted to presentation markup languages easily.

3.2 Binary Tree of Symbol Objects

Layout trees can be converted to mathematical expressions for computer algebra systems to evaluate, in another way. Layout trees are converted to binary trees of symbol objects before converting to expressions for evaluation.

At first, a layout tree is parsed into "reverse Polish notation." The layout tree is scanned from root in the default direction. An example of reverse Polish notation of the layout tree in Fig. 3 is as follows:

```
[f][x][apply function][∑][a][x][i][power][invisible times][apply][=]
```

As to "$\sum$" and "a", only child-nodes of direction "2" are visited in that scan process. The ignored child-nodes are called after the process, if it is needed. That is decided by the parser. Temporary objects (such as "power" or "apply") are generated in the scan process.

And then, a binary tree of the symbol objects(left hand of Fig. 5) is constructed from the reverse Polish notation. Right hand of Fig. 5 shows another example. The parser can decide meaning of "$f(x)$."

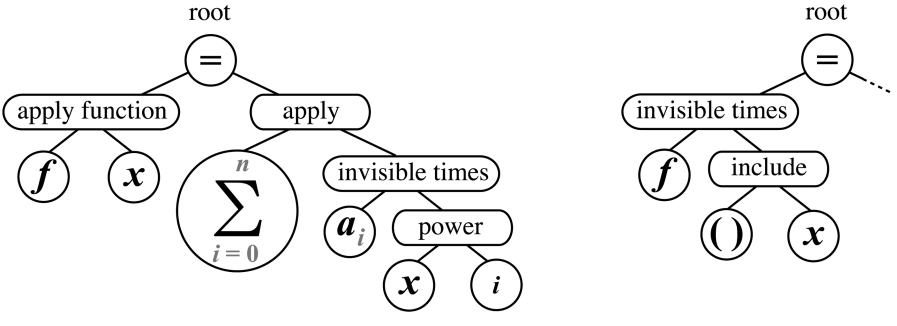


Fig. 5. Left: An example of binary tree. **Right:** A left sub tree of another example.

Finally, the binary tree of the symbol objects is traversed, and outputs which depend on contents of symbol objects are generated. Notice that the ignored child-nodes of "$\sum$" and "a" are called, when the binary tree is traversed.

The binary trees of symbol objects are used internally and temporarily in the process of the converting. Users can get mathematical notations of wished format after the process, because binary trees are able to be converted to various notations of mathematical expressions including infix, prefix, and postfix notations.

4 Conclusion

The new dynamic input interface has been developed. We have added new elements to objects of the system. For such objects, new GUI operations have been defined. "Drag and draw" is one of "drag and drop" operations. In case the selection is dragged and drawn near to the target, the operation is called "drag and draw." The exact actions of the operations are to drag over the peripheral areas and to drop onto these areas.

"Dynaput" has also been defined as operations which include input and edit, and provide dynamic input interface by previewing results. To provide dynaput operations, the interface should have operations "drag and draw" and peripheral areas for symbol objects. Using dynaput operations, users can input and edit mathematical expressions intuitively by using only pointing devices.

In this user interface, mathematical expressions are constructed from tree-structured symbol objects. Boxes as containers and text stream as contents, which are used in other systems with templates, are not used. The structure of layout tree is based on 2D structure of symbols on computer screens, and mathematical meanings of objects are ignored temporarily.

The presentation markup outputs, such as TeX and MathML, are generated easily because of the simple structure of layout trees. Mathematical notations for computer algebra systems' evaluations are also converted from layout trees via tree transformation. "What it means appears when you need" is our strategy for handling 2D structures of expressions.

Using this new input interface with the operation dynaput, users can manipulate mathematical formulas intuitively, and users are able to acquire mathematical expressions of various formats.

A demo video of MathBlackBoard is available on the following URL:
`http://wwwmain.h.kobe-u.ac.jp/MathBB/`

References

1. Kajler, N., Soiffer, N.: A Survey of User Interfaces for Computer Algebra Systems. *J. Symb. Comput.* **25**(2) (1998) 127–159
2. Matsushima J.: *An Easy-to-Use Computer Algebra System by Using Java*. Master Thesis, Kobe University (1998) [in Japanese]
3. Deguchi H.: Blackboard Applet. *Journal of Japan Society for Symbolic and Algebraic Computation* **9**(1) (2002) 32–37 [in Japanese]
4. Deguchi H.: MathBlackBoard. *Journal of Japan Society for Symbolic and Algebraic Computation* **11**(3,4) (2005) 77–88 [in Japanese]
5. Deguchi H.: MathBlackBoard as User Interface of Computer Algebra Systems. *Proceedings of the 10th Asian Technology Conference in Mathematics* (2005) 246–252
6. Deguchi H.: A Practical Lesson Using MathBlackBoard. *Journal of Japan Society for Symbolic and Algebraic Computation* **12**(4) (2006) 21–30 [in Japanese]
7. Deguchi H.: A Dynamic Input Interface for Mathematical Expressions. *Proceedings of the Human Interface Symposium 2006* (2006) 627–630 [in Japanese]
8. Deguchi H., Hashiba H.: MathBlackBoard as Effective Tool in Classroom. *ICCS 2006, Part II* Springer LNCS **3992** (2006) 490–493