Data Representation in a System for Understanding Mathematical Expressions
数式のデータ表現と意味解釈システム

3 J − 9

趙燕結 杉浦洋 鳥居達生(名古屋大学 情報工学科) 桜井鉄也(筑波大学 電子情報系)

The researches of understanding mathematical notations by computer have continued more than 30 years in several fields (programming languages, pattern recognition, document processing, software specification and interfaces of mathematical software). In recent years, a research upsurge was set off again. Many famous computer algebra systems changed their text interface into formulary and graphical interfaces.

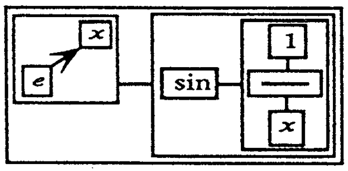
We can find that all existing systems and prototypes accept necessarily minimal sub-set of mathematical notations to avoid ambiguities, nevertheless, they sacrifice some powerful representations in mathematics. They also need big user manuals because of lacking universality.

We develop a general and natural method to represent the structures with meanings of mathematical expressions on computer. This method will be applied in many fields, such as interfaces of mathematical software, computer aided education, algorithm representation, document processing and proofreading.

We find that the visual image of a mathematical expression is constructed by characters through several kinds of geomeric adjacent relations which express certain meaning connection between symbols in specified locations. These relations are defined as several fundamental structures abstractly. Through these fundamental structures any mathematical expression can be generated recursively and aptly. Therefore, mathematical expressions can be understood recursively through these structures and with knowledge of mathematics.

To make computer understand mathematical expressions, we design data structures to implement these fundamental structures. We show the mechanism of our scheme through an example $e^x \sin \frac{1}{x}$, where x is on the domain of real numbers.

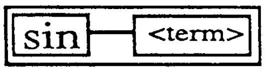
In the first stage, through the fundamental structures, this expression can be input as the following list-structure recursively in our system.



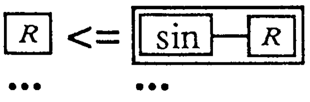
Namely, this expression is constructed by two part: e^x and $\sin \frac{1}{x}$. Especially, the x on the back superscript of e in e^x is connected with e in meaning, not the sin. The sin is connected with the $\frac{1}{x}$. Therefore, through the list-structure, the left part is a superscript structure to express an exponential operation. The right part is a sinusoidal operation which contains a sub-structure $\frac{1}{x}$ to express a fraction operation. This list-structure describes its structures and meanings boundaries correctly and naturally.

In the next stage, the understanding stage, computer translates a mathematical expression into its equivalent recursive function representation. To make our method be extensible and able to manage by user, we divide the understanding program into a knowledge base and a parser. The knowledge base contains the necessary knowledge about the syntax and semantics of mathematical notations. The parser is an algorithm to accomplish top-down syntactic analysis and bottom-up semantic interpretation in terms of a series of rules in the knowledge base.

For example, to accept the sinusoidal part of the example above, the parser finds out a rule which contains the following structure pattern (where the <term> means multiplication of a series of factors) from the knowledge base.



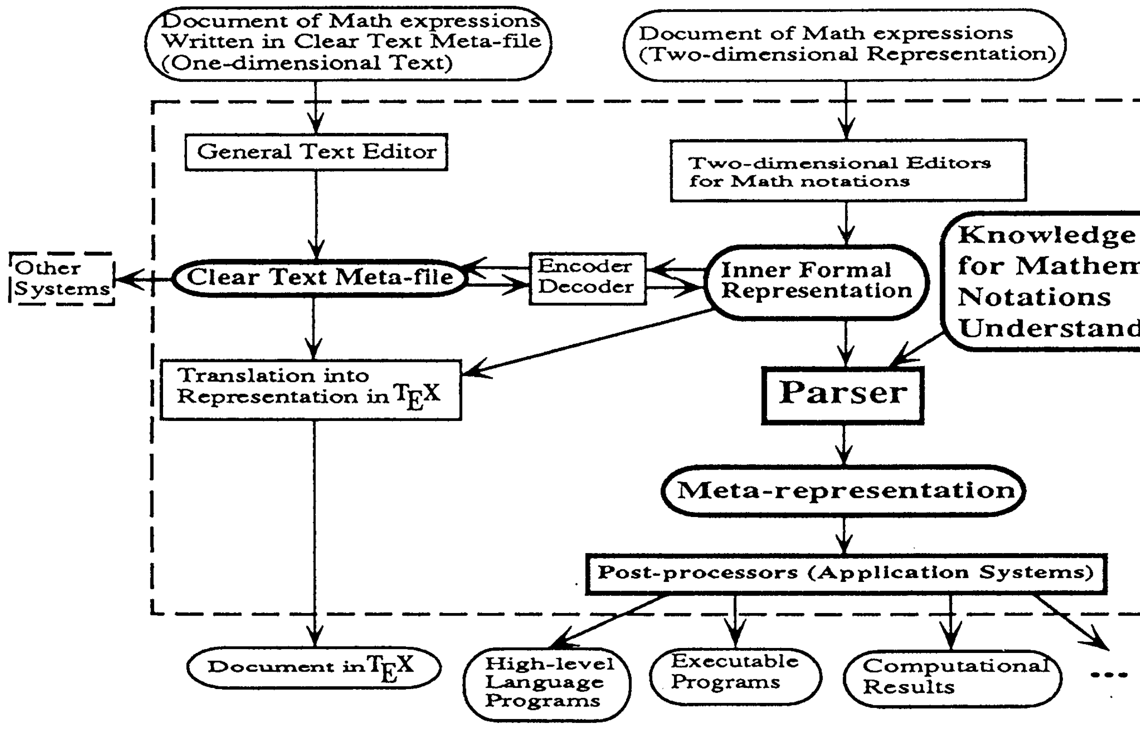
The parser do matching between the right part of the example and this structure pattern. Then, the parser matches the fraction part recursively. At the same time, the parser obtains the domain of the $\frac{1}{x}$, i.e., the set of real numbers $\boldsymbol{R}$. After matching one of the following domain rules in a the same rule in the knowledge base, the parser can determine a meaning of this part, i.e., the function sin(< *term* >), and the domain of this part, i.e., $\boldsymbol{R}$.



Finishing the whole understanding, we can obtain the following recursive function on the domain of real numbers. It is independent of any applications and can be used by application systems directly.

```
multiple(exp(variable(x)), sin(frac(number(1), variable(x))))
```

The wohle scheme of our system is shown in the following Figure.


Document of Math expressions Written in Clear Text Meta-file (One-dimensional Text)
Document of Math expressions (Two-dimensional Representation)
General Text Editor
Two-dimensional Editors for Math notations
Other Systems
Clear Text Meta-file
Encoder Decoder
Inner Formal Representation
Knowledge Base for Mathematical Notations Understanding
Translation into Representation in TEX
Parser
Meta-representation
Post-processors (Application Systems)
Document in TEX
High-level Language Programs
Executable Programs
Computational Results
•••

Formal Representation is list-structured data of mathematical expressions. **Clear Text Meta-file** is a mechanism for storing and transmitting the formal representation, a readable text which is equivalent to the formal representation. **Knowledge Base** is composed of a series of rules. Each rule contains a structure pattern, a series of domain rules, and a function as the understanding result. **Parser** is an algorithm to accomplish top-down structure matching and bottom-up domain reduction according to the knowledge base. **Meta-representation** is the understanding result of the formal representation, the one-dimensional recursive function of the equivalent two-dimensional mathematical expressions. **Post-processors** are the application systems using mathematical notations (e.g., for mathematical computation, it contains selection of data types, selection of algorithms, and translation into programs).