Maple's 2-D Math Interface

Paulina Chin, Maplesoft
June 24, 2009

Introduction

Maple is a software package for mathematical manipulation and modelling. Originally a computer algebra system, it has expanded into a tool for mathematicians, scientists and engineers and encompasses both symbolic and numeric computational capabilities. In recent years, there has been much development in Maple's interface, with a focus on tools for creating interactive Mathematical documents.

Maple allows users to enter executable standard math notation with its 2-D math interface. An older command-line-style 1-D math interface is also available. This paper will present an overview of the 2-D math system and discuss some of the technical challenges associated with both parsing and displaying typeset math expressions.

2-D Math in Maple

Maple provides the ability to display the output of computations as typeset math. It also allows a user to enter the commands as typeset math, and then it parses and executes them. Below is an example of computing an integral using 1-D and 2-D input.

```
> int(x/(x^3-1), x);
```

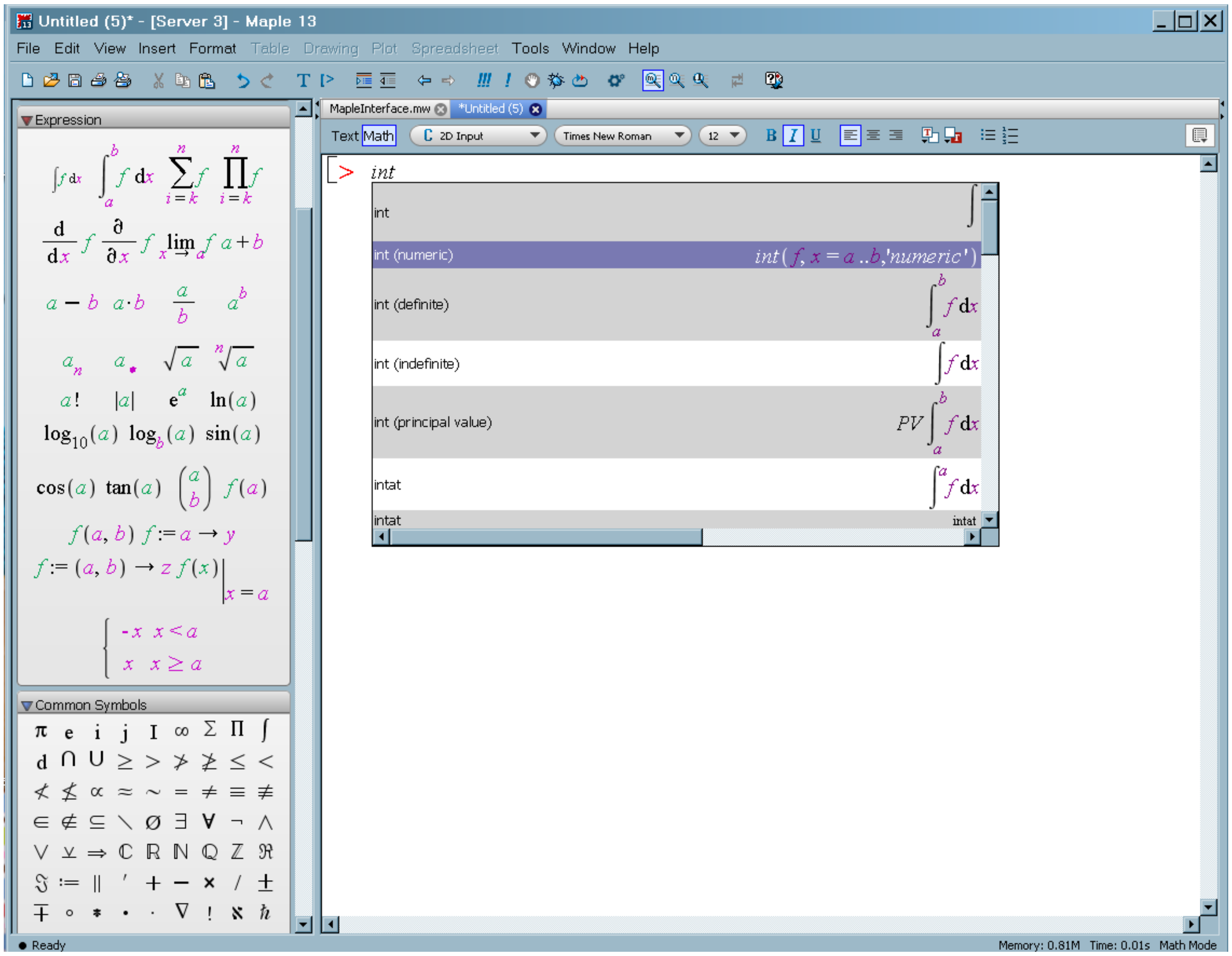
$$-\frac{1}{6}\ln\left(x^2+x+1\right)+\frac{1}{3}\sqrt{3}\arctan\left(\frac{1}{3}(2x+1)\sqrt{3}\right)+\frac{1}{3}\ln(x-1) \tag{2.1}$$

$$> \int \frac{x}{x^3-1}\,\mathrm{d}x$$

$$-\frac{1}{6}\ln\left(x^2+x+1\right)+\frac{1}{3}\sqrt{3}\arctan\left(\frac{1}{3}(2x+1)\sqrt{3}\right)+\frac{1}{3}\ln(x-1) \tag{2.2}$$

There are various ways of entering 2-D math as input. The easiest way is to select the 2-D input mode, and the characters will be typeset as you type. For example, typing 'x', '^' and '2' in sequence results in x^2 appearing on the input line. You can also select output that is already in typeset form and then copy and paste it into an input line, or you can enter an expression in 1-D form, select it and convert to 2-D form.

For larger and more complicated expressions, Maple provides two useful tools to facilitate the input process, and both are shown in the screenshot below. First, there is a command-completion facility which provides both 1-D and 2-D versions of Maple commands. There is also a large collection of palettes, two of which are shown in the screenshot. The user simply clicks on the palette entry to insert it in a document and then tabs through the fields to fill the placeholders with the desired values or expressions.


Untitled (5)* - [Server 3] - Maple 13
File Edit View Insert Format Table Drawing Plot Spreadsheet Tools Window Help
Expression
Common Symbols
MapleInterface.mw
*Untitled (5)
Text Math
2D Input
Times New Roman
12
int
int
int (numeric)
int(f, x = a ..b,'numeric')
int (definite)
int (indefinite)
int (principal value)
intat
intat
Ready
Memory: 0.81M Time: 0.01s Math Mode

Dealing with Ambiguity

The biggest challenge with parsing mathematics notation is dealing with ambiguous expressions, or at least ones that look ambiguous to the user even if there are set rules for dealing with them in the parser. The 1-D command-line style Maple syntax is unambiguous but generally difficult for users to learn, as it often bears little resemblance to real mathematical notation.

The most common problems come from the use of implicit multiplication. The first example below is treated as a function call while the second is considered a multiplication.

> $s(t+u)$

$$s(t+u) \qquad \textbf{(3.1)}$$

> $eval(\textbf{(3.1)}, \{t=3, u=4\})$

$$s(7) \qquad \textbf{(3.2)}$$

> $s\ (t+u)$

$$s\ (t+u) \qquad \textbf{(3.3)}$$

> $eval(\textbf{(3.3)}, \{t=3, u=4\});$

$$7\ s \qquad \textbf{(3.4)}$$

Maple has specific documented rules describing when implicit multiplication is assumed and when it is not. In addition, we advise users to use explicit multiplication (a dot operator) if there is any chance of ambiguity. Even so, it is easy for users to forget the rules or not see them in the first place.

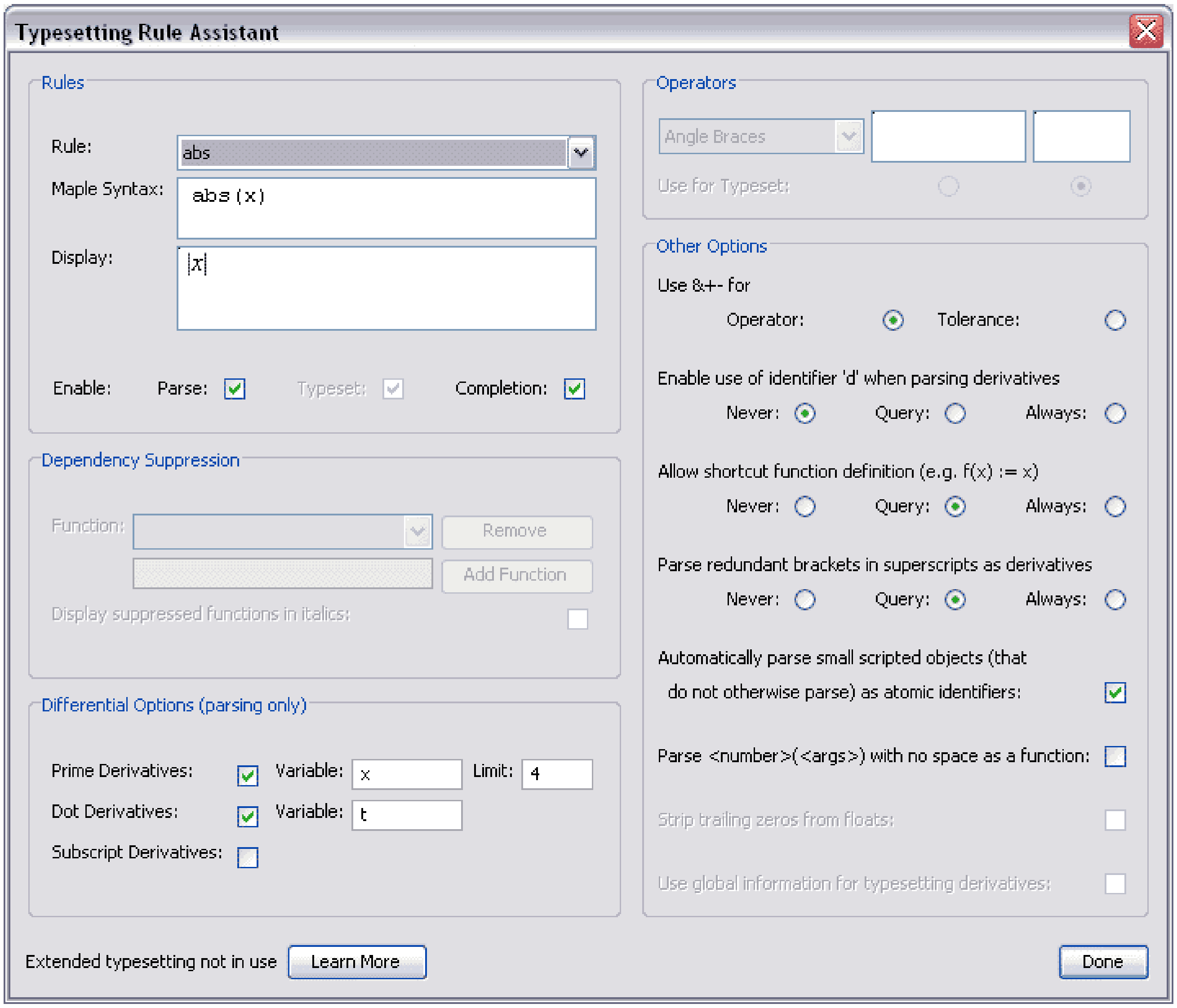
Other problems arise from attempts to be compatible with the existing Maple 1-D language. For example, angle brackets can be used to create vectors. The following expression, "$x < 1, 2 > y$", is parsed as a sequence of inequalities, but a user might expect this to mean the product of x, a vector and y.

> $\langle 1, 2 \rangle$

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad \textbf{(3.5)}$$

> $x < 1, 2 > y$

$$x < 1, y < 2 \qquad \textbf{(3.6)}$$



Typesetting Rule Assistant

Rules

Rule: abs

Maple Syntax: abs(x)

Display: $|x|$

Enable: Parse: [✓] Typeset: [✓] Completion: [✓]

Dependency Suppression

Function:

Remove

Add Function

Display suppressed functions in italics: [ ]

Differential Options (parsing only)

Prime Derivatives: [✓] Variable: x Limit: 4

Dot Derivatives: [✓] Variable: t

Subscript Derivatives: [ ]

Operators

Angle Braces

Use for Typeset:

Other Options

Use &+- for

Operator: [✓] Tolerance: [ ]

Enable use of identifier 'd' when parsing derivatives

[✓ Never:] Query: Always:

Allow shortcut function definition (e.g. f(x) := x)

Never: [✓ Query:] Always:

Parse redundant brackets in superscripts as derivatives

Never: [✓ Query:] Always:

Automatically parse small scripted objects (that do not otherwise parse) as atomic identifiers: [✓]

Parse <number>(<args>) with no space as a function: [ ]

Strip trailing zeros from floats: [ ]

Use global information for typesetting derivatives: [ ]

Extended typesetting not in use

Learn More

Done

A partial solution to the problems of ambiguity is to allow some user customization of the parsing and typesetting facilities. Maple provides the ability to adjust a subset of the rules through its Typesetting Rule Assistant, shown above. For example, in the 1-D Maple language, 2(x) is parsed as a call to the constant function 2, and there is no ambiguity since implicit multiplication is not supported by 1-D math. However, users of the 2-D language almost always expect 2(*x*) to mean 2 times *x*. This is the default behaviour, but the Typesetting Rules Assistant allows you to change the parse rule to respect the original 1-D meaning.

Maple also tries to use context information whenever possible to decide how certain operators should be parsed. Maple offers a number of packages that users can load to get quick access to its commands. An expression might be parsed differently with the Physics package or VectorCalculus package loaded. The parsing is often based on the types associated with the inputs. For example, if *A* and *B* are known to be matrices, then *A B* results in a call to the routine for matrix multiplication. A difficulty arises when the data types cannot be determined at parse time, but only later at evaluation time. In these situations, Maple attempts to delay commiting to a parsed result until evaluation takes place.

An ongoing challenge is to find the right balance between representing both "mathematics" and "commands". In Maple, users want the expressions to look like real mathematics as one would find in a textbook. However, for the most part, they are interested in executing a Maple command to produce a result. An example of this difficulty is representing a line integral over a path. How can we capture all the information in the single LineInt command in a 2-D representation that looks similar to how mathematicians would write it?

```
> with(VectorCalculus) :
   SetCoordinates(cartesian[x, y]) :
> LineInt(VectorField(⟨y,-x⟩), Circle(⟨0, 0⟩, r));
```

$$-2\pi r^2 \tag{3.7}$$

The example below shows the difference between exact and numeric integration. Currently, there is no simple way to provide the 'numeric' option with 2-D input.

> $int\left(\frac{x}{x^3+1}, x = 1..2\right)$

$$\frac{1}{18}\sqrt{3}\,\pi + \frac{1}{3}\ln(2) - \frac{1}{6}\ln(3) \tag{3.8}$$

> $int\left(\frac{x}{x^3+1}, x = 1..2, numeric\right)$

$$0.3502469061 \tag{3.9}$$

> $\int_1^2 \frac{x}{x^3+1}\,dx$

$$\frac{1}{18}\sqrt{3}\,\pi + \frac{1}{3}\ln(2) - \frac{1}{6}\ln(3) \tag{3.10}$$

Saving Semantic Information

Confusion often arises from two different objects that look identical in mathematical notation. For example, vector elements look the same as subscripted names even though they are fundamentally different objects. Below, the first v_3 was created from a palette entry for an indexed variable and is equivalent to v[3]. The second was created as an "atomic identifier" and is the name consisting of *v* with a subscript 3.

> $v := Vector([1, 2, 3, 4, 5])$:
> v_3

$$3 \tag{4.1}$$

> v_3

$$v_3 \tag{4.2}$$

Maple offers two modes for typesetting display, standard and extended. In the extended mode, it attempts to save more semantic information in a displayed expression so that it can be reparsed correctly after a copy-and-paste operation.

> $interface(typesetting = extended)$:
> $\mathrm{BesselJ}(0, x)$

$$J_0(x) \tag{4.3}$$

> $J_0(x)$ # copy and pasted from previous line

$$J_0(x) \tag{4.4}$$

> $eval(\mathbf{(4.4)}, x = 0.5)$;

$$0.9384698072 \tag{4.5}$$

> $\mathrm{BesselJ}(0, 0.5)$;

$$0.9384698072 \tag{4.6}$$

The typeset expression for the BesselJ function includes semantic information that allows the J to be identified as BesselJ, and not as an ordinary J, after copy, paste and reparse. The Maple internal form of this typeset expression is shown below.

> $lprint(convert(Typesetting[Typeset](\mathrm{BesselJ}(0, x)), `global`))$

```
mrow(msub(mi("J", fontstyle = "normal", msemantics =
"BesselJ"), mn("0")), mo("&ApplyFunction;"), mfenced(mi("x")))
```

A challenge, which we hope to address in future versions of Maple, is to let users see easily the meaning of typeset expressions. Ideally, users should be able to immediately tell the difference between $J_0(x)$, the BesselJ function, and J with subscript 0 without having to resort to indirect means.

▼ Internal Representation

Maple's internal form for typeset expressions, referred to as TypeMK, is based on presentation MathML. TypeMK supports most of the elements of MathML and includes a small number of additions. The most significant of these is the msemantics attribute, as seen in the BesselJ example, which Maple uses to add content information in a limited way. The TypeMK representation of the integral used in a previous example is shown below.

> $lprint\left(convert\left(Typesetting[Typeset]\left(\int_1^2 \frac{x}{x^3+1}\,\mathrm{d}x\right), `global`\right)\right)$

```
mrow(msubsup(mo("&int;"), mn("1"), mn("2")), mfrac(mi("x"),
mrow(msup(mi("x"), mn("3")), mo("&plus;"), mn("1"))), mspace
(width = "0.3em"), mo("&DifferentialD;"), mi("x"))
```

When a user enters a 2-D expression, it is parsed initially by the GUI system to generate a simplified TypeMK form. In a second pass of the parser, context information is used and user settings are checked to refine the TypeMK result, and semantic information is added at this point, if any is available. This second parsing phase is done by the Maple library, which contains most of Maple's mathematical functionality and is written in the Maple language itself. During this phase, an internal representation of the expression or command that is understood by the Maple kernel is also generated and subsequently executed. The internal form of the integral used in the above example is shown below.

> $ToInert\left('int\left(\frac{x}{x^3+1}, x=1..2 \right)' \right)$

```
_Inert_FUNCTION(_Inert_ASSIGNEDNAME("int", "MODULE",                    (5.1)
    _Inert_ATTRIBUTE(_Inert_EXPSEQ(_Inert_NAME("protected",
    _Inert_ATTRIBUTE(_Inert_NAME("protected"))), _Inert_NAME("_syslib")))),
    _Inert_EXPSEQ(_Inert_PROD(_Inert_NAME("x"),
    _Inert_POWER(_Inert_SUM(_Inert_POWER(_Inert_NAME("x"), _Inert_INTPOS(3)),
    _Inert_INTPOS(1)), _Inert_INTNEG(1))), _Inert_EQUATION(_Inert_NAME("x"),
    _Inert_RANGE(_Inert_INTPOS(1), _Inert_INTPOS(2)))))
```

▼ Conclusions

Maple's 2-D math system allows users to enter expressions and see the results of computations in natural mathematical notation. This helps facilitate the use of Maple, without requiring the time normally needed to master the traditional command-line language. It also allows users to quickly and easily create technical documents containing complicated mathematical expressions.

However, there are still many challenges associated with parsing and displaying typeset math correctly and efficiently. These include resolving ambiguous situations, storing and using semantic information, and maintaining as much compatibility with the 1-D language as possible (so that long-time users can make an easy transition to the new language). In future work, we will continue to address these issues, as well as improve the interface so that users can enter mathematical expressions with as much ease as possible.

▼ Acknowledgements

I would like to acknowledge the assistance of my Maplesoft colleague, Allan Wittkopf, in the preparation of this paper.