

## Re: Lisp syntax vs. Mathematica syntax

---

*Source:* <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2005-08/msg01584.html>

---

- *From:* Jon Harrop <[usenet@xxxxxxxxxxxxxxxx](mailto:usenet@xxxxxxxxxxxxxxxx)>
  - *Date:* Sat, 20 Aug 2005 15:52:35 +0100
- 

josephoswaldgg@xxxxxxxxxxx wrote:

```
> My personal "favorite" is Apply, where I had not defined a function
> "g". Check out Mathematica's documentation
>
> Apply[Plus, g[a,b]] --> a+b
>
> WTF! Hey, what happened to my function g? And, amazingly enough, at
> least some folks at Wolfram think this is what Lisp does,
```

Let's try writing a Lisp macro equivalent to Mathematica's Apply.

In Mathematica, Apply may be defined as:

```
Apply[h_, _[args___]] := h[args]
```

For example:

```
In[1]:= Apply[Plus, g[3]]
```

evaluates Plus[3] to give:

```
Out[1]= 3
```

Effectively, Mathematica's Apply replaces a call to one function (g) with a call to another (h).

Here's my puny attempt at a Lisp conversion that only works for single-argument functions:

```
(defmacro mapply (`f` (g arg))
  `(f ,arg))
```

Repeating the same example, Lisp gives a few warnings but it seems to work:

```
* (mapply `+ `(g 3))
```

```
; In: LAMBDA (#:WHOLE-1516 #:ENV-1517)
```

```
; (LET* (#####)
```

## Re: Lisp syntax vs. Mathematica syntax

```
; `(F,ARG))  
; Note: Variable G defined but never used.  
;  
; Note: Variable QUOTE defined but never used.  
;  
3
```

> and they'll start explaining to you about the "head" of an expression....  
>  
> You can also write a Mathematica-like syntax parser in Lisp. Look up  
> Fateman's "mma" if you want to.

I've already looked at it.

>> Just to clarify, many of the symbols that you've listed are simply infix  
>> functions. You've also omitted a lot of Mathematica's syntax (which is  
>> unusually complicated).  
>  
> And some of those "functions" affect the pattern \*transformations\* of  
> the expressions they define. That's not what Lisp folks mean when they  
> talk about functional programming.

If you mean the difference between Rule and RuleDelayed, for example, then I believe that is probably just shorthand for quoting and not quoting in a Lisp macro.

>> Mathematica can also be very concise. For example, the following squares  
>> each element in a list "l":  
>>  
>> #^2&/@l  
>  
> Yes, because Mathematica has the "feature" that it automatically  
> distributes functions over lists.

No. If you want a function to be threaded over its arguments when they are lists then you must set the Listable attribute for that function. There are many possible attributes. This is one piece of functionality that makes Mathematica's pattern matching much more powerful than most other forms of pattern matching. However, Mathematica's pattern matching can be far from linear.

In that example, I used the map function which can be written "func /@ list" in Mathematica's infix notation. The same infix notation can be defined in OCaml or SML, for example:

```
# let ( /@ ) f l = List.map f l;;  
val ( /@ ) : ('a -> 'b) -> 'a list -> 'b list = <fun>
```

> What you are doing is not the same as other programming environments.

Yes. Equivalently, what Lisp macros do is "not the same as other programming

Re: Lisp syntax vs. Mathematica syntax

environments".

- > Yes, you can write programs in Mathematica very easily in a huge number
- > of various approaches. But will they come when you call them? Can you
- > really be sure what they do? That you can avoid name capture? That
- > things don't depend on the exact structure of the expressions you pass
- > in? That you really made the right choice between Hold and Unevaluated?

I wrote most of my PhD in Mathematica and it worked superbly. Mathematica also has many happy users. Indeed, my old department are apparently looking into a site license. From my point of view, the main source of bugs in Mathematica code is due to the lack of static type checking.

--

Dr Jon D Harrop, Flying Frog Consultancy  
<http://www.ffconsultancy.com>

.

---

• *Follow-Ups:*

- ◆ **Re: Lisp syntax vs. Mathematica syntax**  
◇ From: josephoswaldgg@xxxxxxxxxxxx
- ◆ **Re: Lisp syntax vs. Mathematica syntax**  
◇ From: Wade Humeniuk

• *References:*

- ◆ **Re: Very poor Lisp performance**  
◇ From: Förster vom Silberwald
- ◆ **Re: Very poor Lisp performance**  
◇ From: Michael Sullivan
- ◆ **Re: Very poor Lisp performance**  
◇ From: Jon Harrop
- ◆ **Re: Very poor Lisp performance**  
◇ From: Hartmann Schaffer
- ◆ **Re: Very poor Lisp performance**  
◇ From: Jamie Border
- ◆ **Re: Very poor Lisp performance**  
◇ From: Jon Harrop
- ◆ **Re: Very poor Lisp performance**  
◇ From: Christophe Rhodes
- ◆ **Re: Very poor Lisp performance**  
◇ From: Joe Marshall
- ◆ **Re: Very poor Lisp performance**  
◇ From: Jon Harrop
- ◆ **Re: Very poor Lisp performance**  
◇ From: Tayssir John Gabbour
- ◆ **Re: Very poor Lisp performance**  
◇ From: Jon Harrop
- ◆ **Re: Very poor Lisp performance**  
◇ From: Joe Marshall

Re: Lisp syntax vs. Mathematica syntax

- ◆ **Re: Very poor Lisp performance**
  - ◇ From: Jon Harrop
- ◆ **Re: Very poor Lisp performance**
  - ◇ From: Ulrich Hobelmann
- ◆ **Re: Very poor Lisp performance**
  - ◇ From: Jon Harrop
- ◆ **Lisp syntax vs. Mathematica syntax**
  - ◇ From: josephoswaldgg@xxxxxxxxxxxx
- ◆ **Re: Lisp syntax vs. Mathematica syntax**
  - ◇ From: Jon Harrop
- ◆ **Re: Lisp syntax vs. Mathematica syntax**
  - ◇ From: josephoswaldgg@xxxxxxxxxxxx
- Prev by Date: **Re: Lisp/Unix impedance [a programming challenge]**
- Next by Date: **Re: Lisp syntax vs. Mathematica syntax**
- Previous by thread: **Re: Lisp syntax vs. Mathematica syntax**
- Next by thread: **Re: Lisp syntax vs. Mathematica syntax**
- Index(es):
  - ◆ **Date**
  - ◆ **Thread**