

## Re: Very poor Lisp performance

---

*Source:* <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2005-08/msg01465.html>

---

- *From:* Jon Harrop <[usenet@xxxxxxxxxxxxxxxx](mailto:usenet@xxxxxxxxxxxxxxxx)>
  - *Date:* Fri, 19 Aug 2005 19:06:08 +0100
- 

Brian Downing wrote:

> In article <m2iry1q2j6.fsf@xxxxxxxxxxxxxxxx>,  
>> I'm not sure it's been established that Mathematica has macros that  
>> are similar in power to Lisp's macros. If you want to see what an  
>> honest attempt to combine infix operators with Lisp-style macros looks  
>> like you should check out Dylan or (more recently) the Java Syntactic  
>> Extender[1]. However to understand the comparison you'll, of course,  
>> have to learn something about Common Lisp macros. And if you really  
>> want to go to town, look into Scheme's various macro systems which  
>> offer a slightly different take on the problem.

Can you give me a reference explaining the difference between Lisp and Scheme here? I didn't realise this existed...

> To Mathematica's credit its syntax always boils down to it's own  
> funny-looking sexprs:  
>  
> (2 + {3, 4, 5}) ~Frob~ x is really Frob[Plus[2, List[3, 4, 5]], x]

Yes, exactly.

> So given that you can programmatically manipulate Mathematica symbolic  
> expressions, it's probably not a stretch to say that in theory you can  
> generate arbitrary code like you can with CL macros.

Absolutely. In this respect, I do not think you can get any more general than Mathematica. The main disadvantage is the performance cost of this generality. On the other hand, I believe you can replace the lexer (reader?) in Lisp, which you cannot do in Mathematica.

> The problem, as mentioned in another branch of the thread here, is that  
> the Mathematica syntax is incredibly complicated to support some of this  
> flexibility, and the Mathematica evaluator is so /incredibly/ complex as  
> to basically prohibit understanding of what's going to happen when  
> things are evaluated, at least for me.

Not really. It took me 4 days to write a mini-Mathematica implementation in OCaml. Once you are familiar with Mathematica, it is fairly obvious what's going on "under the hood".

Re: Very poor Lisp performance

> There's certainly no concept of "macroexpansion time" and "run time" like  
> there is in CL,

So you cannot use macros to generate code to generate macros to generate  
code in Lisp?

> and there's no simple QUOTE operator

What does the quote operator do?

> – things keep evaluating until they stop changing  
> unless wrapped in special Hold forms that have to stick with it to keep  
> it from evaluating in the future.

Yes. I thought that was a direct equivalent of QUOTE.

—

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

.

---

• *Follow-Ups:*

- ◆ **Re: Very poor Lisp performance**  
    ◇ From: jayessay
- ◆ **Re: Very poor Lisp performance**  
    ◇ From: Ulrich Hobelmann
- ◆ **Re: Very poor Lisp performance**  
    ◇ From: Jens Axel Søggaard
- ◆ **Re: Very poor Lisp performance**  
    ◇ From: Jens Axel Søggaard
- ◆ **Re: Very poor Lisp performance**  
    ◇ From: Peter Seibel

• *References:*

- ◆ **Very poor Lisp performance**  
    ◇ From: Jon Harrop
- ◆ **Re: Very poor Lisp performance**  
    ◇ From: Ulrich Hobelmann
- ◆ **Re: Very poor Lisp performance**  
    ◇ From: Jon Harrop
- ◆ **Re: Very poor Lisp performance**  
    ◇ From: Peter Seibel
- ◆ **Re: Very poor Lisp performance**  
    ◇ From: Brian Downing

- Prev by Date: **Re: ADA and my own business**
- Next by Date: **Re: Very poor Lisp performance**
- Previous by thread: **Re: Very poor Lisp performance**

Re: Very poor Lisp performance

- Next by thread: ***Re: Very poor Lisp performance***
- Index(es):
  - ◆ ***Date***
  - ◆ ***Thread***