

Re: SBCL is now faster than Java, as fast as Ocaml, and getting better

Re: SBCL is now faster than Java, as fast as Ocaml, and getting better

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2008-07/msg00522.html>

- *From:* Marco Antoniotti <marcoxa@xxxxxxxx>
 - *Date:* Mon, 14 Jul 2008 00:34:15 -0700 (PDT)
-

On Jul 13, 2:42 am, Jon Harrop <j...@xxxxxxxxxxxxxxxxxxxx> wrote:

Marco Antoniotti wrote:

On Jul 12, 11:05 pm, Jon Harrop <j...@xxxxxxxxxxxxxxxxxxxx> wrote:

Mathematica is similar to Lisp in many ways but it does use vectors entirely rather than cons cells. I think the Lisp community have a huge amount to learn from Mathematica because it addressed so many of Lisp's deficiencies.

So, Mathematica is no more Conses deep down. As per Lisp deficiencies w.r.t. Mathematica I think it goes both ways.

What do you think Mathematica can learn from Lisp?

It already did. And it forgot a few things in the process. But you are overestimating the "language merits" of Mathematica as far as its diffusion is concerned. Mathematica greatest contribution is the notion of IDE for mathematics. As another example, look at Matlab. It is an even more diffused system simply because they did one thing very, very well; incidentally the "language implications" of Simulink go beyond our discussion of Lisp, MLish and pattern matching. But this is for another thread.

In general I defer my opinions about Mathematica to people who know better, like Fateman.

Re: SBCL is now faster than Java, as fast as Ocaml, and getting better

Fateman is just another grumpy old Lisper who tried to plagiarise Wolfram's work, got in trouble with the law for it and then spent the best years of his life failing to get people to use his free software whilst publishing a series of petty jabs at Mathematica. His knowledge is now decades out of date.

Fateman is also the father of a rock singer! Difficult to beat that! :) ...and an emeritus professor at Berkeley (last I checked, the aroma that was coming from Telegraph Street did help the creativity of people hanging around Soda Hall :) or maybe it was the coffee from Nefeli's :)).

I don't know what you mean by "a language on its own" but F# is being productized by Microsoft and placed along C# and VB as their next generation language for .NET.

I mean that it is not an extension of Java or C++ or C#.

Yes, of course.

So OCaml is actually every bit as extensible as Lisp and, of course, many people have published extensions for it. For example, Martin Jambon has extended OCaml's ordinary pattern matching to also support regular expression matching, by writing a `camlp4` syntax extension:

<http://martin.jambon.free.fr/micmatch-howto.html>

Nope. It still does not count. Of course you can do very neat things with `camlp4`, but it just isn't the same as the macro system plus the code-is-data of (any) Lisp.

Re: SBCL is now faster than Java, as fast as Ocaml, and getting better

Perhaps we can get to the root of your misconception: what exactly do you mean by "just isn't the same" given that they have the same capabilities?

camlp4 is a pre-processor and its "language" is not Ocaml; similar yes, but no cigar. You invoke it on the command line (e.g., with the `-pp` option to `ocamlc`). Plus it seems to rely on deep knowledge of the Ocaml internal AST implementation. I am not saying that you cannot use camlp4 to do very fancy things. After all it is (it **has** to be!) a full blown parser. camlp4 is a good thing and quite a feat; plus, I do think that some of the best language and compiler people nowadays are working on Ocaml and friends. Yet, it just is not the same as DEFMACRO. camlp4 is more like using SQL embedded in C/C++, while in CLSQL you do

```
(select [emplid] [last-name] :from [employee] :where [= [emplid]
42])
```

Full blown CL!

To continue, I have not seen a similar facility in F# (I mean camlp4-like: again, I may be wrong).

CL and S-exprs have a much more organic loon'n'feel to it. Either you get it or you don't (if you don't, you are not alone :). To this end, we should consider the following quote from somebody who knows what is talking about.

"Different languages choose different sweet spots"

I do not know F# dom, but I would be surprised if it were different.

Put it this way. Get me away from actual work and saturday afternoons writing on CLL :) and I will get you an extended CL written **in** CL that did pattern matching (avec pattern compilation).

You are grossly underestimating the effort required to Greenspun modern language features.

Re: SBCL is now faster than Java, as fast as Ocaml, and getting better

I specifically did not talk about writing a full blown type-inferencer for CL.

Neither was I.

That would definitively prove the fundamental theorem on programming languages :) Yet, doing pattern compilation in CL is simple.

You are grossly underestimating the effort required to Greenspun a modern pattern matcher.

Extending CL is simple.

Not if the extension is inherently complicated, as pattern matching is.

One of the problems I have with your posts is that you keep confusing pattern matching and type inferencing.

I have said nothing of type inference. I am telling you that implementing a modern pattern matcher is far harder than you apparently think.

And I think, you are grossly underestimating the easiness with which you do these things in CL. :)

On my part I also think you think it is so difficult because you are mixing up type inferencing with pattern matching. But that it is me... I may be wrong.

These are separate issues, and I am staying clear of even hinting that I am going to write a type inferencing engine for CL.

This never had anything to do with type inference.

Every single statement you have made about OCaml's extensibility has been completely wrong. You appear to have read and misunderstood the Wikipedia page without actually trying to find how people have already

Re: SBCL is now faster than Java, as fast as Ocaml, and getting better

used camlp4
to do all of the things that you are claiming to be impossible.

I claim something very precise and very different from what you are attributing me.

You claims are anything but precise. Look:

I claim that the use of camlp4 is at a different level than using the CL macro system.

See. What is "at a different level" supposed to mean?

Like it or not that is the way it is.

Now you're asserting a meaningless statement as a fact.

Let's look at the following code snippets.

Snippet 1:

.....

```
(defgeneric iter (f container)
  (:method ((f symbol) (c t)) (iter (symbol-function f) c))
  (:method ((f function) (c list)) (map nil f c))
  (:method ((f function) (c vector)) (map nil f c))
  (:method ((f function) (c hash-table)) (maphash f c))
)
```

;;; 6 lines for ITER.

```
(defmacro foreach ((vs in-kwd container) &body forms)
  (assert (string-equal 'in in-kwd))
  `(iter (lambda ,(if (symbolp vs) (list vs) vs) ,@forms) ,container))
;;; 3 lines for FOREACH. You can make it 2: witness DOLIST.
```

.....

... just to be a little more sophisticated, let's make things a little more interesting (and rightly so, for the sake of the argument).
Let's substitute FOREACH with FOR-EACH.

Snippet 2:

Re: SBCL is now faster than Java, as fast as Ocaml, and getting better

```
.....  
(defmacro for-each (&whole for-form vs in-kwd container &body forms)  
(let ((lvar (if (symbolp vs) vs (gensym))))  
(cond ((string-equal 'in in-kwd) ; one at a time...  
  `(iter (lambda (,lvar)  
    ,@(if (symbolp vs) forms `((destructuring-  
bind ,vs ,lvar ,@forms))))  
  ,container))  
((and (string-equal 'over in-kwd) (null (assert (listp  
vs))))  
  `(iter (lambda ,vs ,@forms) ,container))  
(t (error "Invalid FOR-EACH syntax ~S." for-form))))  
;; 9 lines for FOR-EACH.
```

.....
The second version just separates the multiple lambda arg from the destructuring version. If we had "patterns", things would have to look different. You can use it as

```
(for-each x in '(1 2 3) (print x)  
(for-each (x y z) in '((1 2 3) (4 5 6)) (print (list z y x)))  
(for-each (k v) over some-hash-table (format t "~S -> ~S~%" k v))
```

The above was a half an hour hack. (I counted by thinking 10 minutes, doubled it and added 10 extra minutes for precision). Don't take it too seriously. Yet it "works".

After apples, let's now look at oranges.

http://www.ocaml-tutorial.org/camlp4_3.10/foreach_tutorial

Very nice, but I direct the reader at the full blown "for" example at the end, which, admittedly, leverages OCaml to do somewhat more than the CL code above. It definitely may have the look'n'feel of MLish languages, but it is different from OCaml (and different, as we learn at the top of the page, from version 3.9). And it requires an off-line call at the command line.

Finally it is 17 lines long (actually 36 if you count all the necessary supporting code). Now, which language is more verbose?

Again, you are proving my point about Lisps. You should get out there and learn more about modern languages by actually using them. Once you

Re: SBCL is now faster than Java, as fast as Ocaml, and getting better

know the facts you will be able to substantiate your arguments but, of course, if you knew the truth you would not be using Lisp...

If I remember correctly, I wrote a type inferencer in ML well before you appeared in C.L.L.; actually it was a translation of Miranda code. *And* I am playing with F# (and Haskell). I think I am looking around quite a bit; some things I like, other I like less. I don't program much anymore, and, BTW, most code I see these days is Java

Then I fully understand why you've stopped coding.

I stopped coding because I spend too much time on C.L.L. and elsewhere :)

In any case, enough of this. I'd better go back to coding. :)

Cheers

--

Marco

.