

Re: question about eval

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2005-03/0126.html>

bhaskara_at_gmail.com

Date: 03/03/05

Date: 2 Mar 2005 17:12:40 -0800

OK, but the point is that we may not know at compile time what the forms are. I want to be able to write things like

```
(defun print-a-number (M)
  (choose (loop for i below M collect '(pprint i))))
```

Is that just impossible in Lisp?

– Bhaskara

Peter Seibel wrote:

> *bhaskara@gmail.com* writes:

>

>> *Thanks everybody for all the responses. Here's the original problem I*

>> *was trying to solve. I'm writing an interpreter for a nondeterministic*

>> *language built on top of lisp that has a construct called choose.*

>> *choose L &rest ARGS takes a list l and some other arguments, picks an*

>> *index i between 0 and length(l)-1 (using reinforcement learning, but*

>> *that's another story), does some bookkeeping, and finally evaluates the*

>> *ith form in l. This last part requires a macro, call it select, that*

>> *takes a list l and an integer i and causes the ith form in list l to be*

>> *evaluated.*

>

> *I'm not sure why you want this extra macro. Anyway, here's a macro that implements a CHOOSE operator similar to what you want. It doesn't*

> *take any other arguments or do any bookkeeping since I didn't know what you needed but hopefully this will give you some ideas. The trick*

> *is that you need to be clear about what things you want to do at*

> *macroexpand time (most likely when you compile the code) and what you*

comp.lang.lisp: Re: question about eval

> *need to do at runtime. For instance this macro generates code that*
> *chooses, at runtime, which of several forms, all known at compile*
> *time, to execute.*
>
> *(defmacro choose (&body forms)*
> *(let ((tags (loop for f in forms collect (gensym))))*
> *(number (length forms))*
> *(block-name (gensym)))*
> *`(block ,block-name*
> *(tagbody*
> *(case (random ,number)*
> *,@(loop for n from 0 for tag in tags collect `(,n (go*
> *,tag))))*
> *,@(loop for tag in tags for form in forms*
> *nconc (list tag `(return-from ,block-name*
> *,form))))))*
>
> *This will let your users write:*
>
> *(choose*
> *(some-thing)*
> *(some-other-thing a b c)*
> *(biff-bam-boom))*
>
> *Each time that expression is run, one of the three forms will be*
> *executed. And code within a CHOOSE can refer to lexical bindings*
> *established in the surrounding lexical context.*
>
> *Anyway, I was messing around a bit more, and I came up with the*
> *following contorted attempt :*
>
>
> *(defmacro select-helper (l i)*
> *(elt (eval l) (eval i)))*
>
> *(defmacro select (l i)*
> *(let ((*l* (gensym)) (*i* (gensym)))*
> *`(progv '(*l* ,*i*) (list ,l ,i*
> *(select-helper ,*l* ,*i*))))*
>
> *Now this is really ugly, and I suspect that there's some case that*
it
> *fails on. But it works on the cases I tried. E.g.*
>
> *If that works, it's only by accident. Look at the definition of CHOOSE*
> *I gave you. Macro expand some uses of it and look at the code it*
> *generates. Or perhaps take a look at:*
>
> <http://www.gigamonkeys.com/book/macros-defining-your-own.html>>
>
> *-Peter*
>

comp.lang.lisp: Re: question about eval

> --

> *Peter Seibel*

peter@gigamonkeys.com

>

> *Lisp is the red pill. -- John Fraser, comp.lang.lisp*