

Re: Static/Strong/Implicit Typing

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2004-01/1723.html>

From: Rahul Jain (*rjain_at_nyct.net*)

Date: 01/27/04

Date: Tue, 27 Jan 2004 00:16:26 -0500

"sajiimori" <myusername@hotmail.com> writes:

>> *Then I can't use the sequence functions of Common Lisp that
>> accept both lists and vectors. I can't use the nice flexible
>> arithmetic functions.*
>
> *You could if the functions had multiple versions, and the compiler inserted
> calls to the appropriate ones (since it has all the static type information
> it needs to make that determination).*

Oh... does it? What if the object being operated on is chosen by the user with a mouse? Should we forbid them from using a vector because the code wanted a static declaration and the programmer chose to declare it to be a list?

>> *(It would probably be necessary to forbid
>> user-defined types appearing here, but maybe that's not
>> so terrible.)*
>
> *Don't user-defined types have to be based on built-in types anyway?*

They have to be subclasses of built-in types that are there specifically for being subclassed by user-defined types. They still break the concept of knowing the exact type of all possible bindings of a variable at compile time and then comparing them to the types that cause something else to break. Of course, type errors can't cause lisp programs to break, but that's something you refuse to accept, or so it seems.

>> *Erik Naggum observed that you can *already* consider CL
>> to be strongly, statically, implicitly typed, with everything
>> having type T.*
>
> *Still sounds like a joke. Implementing such a type system requires exactly
> 0 lines of code. Its requirements are necessarily met. It's like a type
> system that requires that objects are not of negative size.*

That's because lisp doesn't draw a distinction between static and

comp.lang.lisp: Re: Static/Strong/Implicit Typing

dynamic typing. You already have both. You can declare the types of bindings to be as narrow as you'd like and you can declare the safety of the compiled code to be what you like.

>> *So if you want to ask questions*
>> *about a "statically typed Lisp" then you ought to be more*
>> *specific about just how fine-grained you want that partition*
>> *to be, and why.*
>
> *Akin to Haskell. And...umm...because I feel like it? =)*

So you don't want anyone else to give you any serious, thoughtful help because your goals are a secret. Got it.

--
Rahul Jain
rjain@nyct.net
Professional Software Developer, Amateur Quantum Mechanicist