

Re: Minimal keywords needed for constructing a full CL system

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2006-07/msg00842.html>

- *From:* rpw3@xxxxxxxx (Rob Warnock)
 - *Date:* Sun, 16 Jul 2006 21:27:10 -0500
-

Lars Brinkhoff <lars.spam@xxxxxxxx> wrote:

```
+-----
| rpw3@xxxxxxxx (Rob Warnock) writes:
| > So I'd like to suggest that a starting set of semantics for a
| > "minimal CL" include at least the following:
|
| I've implemented much of CL in Emacs Lisp, which has a much smaller
| set of primitives than CL.
+-----
```

Very interesting indeed! Do you think your "much of CL" meets the definition of a "subset of CL" given in CLHS 1.7 "Language Subsets"? I'm particularly concerned about the "equivalent semantics" clause and the "no extralingual pre-processing, and no special compatibility packages" clauses. If you hid/disabled/undocumented the pieces of your "much of CL" which don't meet the strict "subset" definition, would you still have a generally-useful language?

[Note: One may still use non-compatible/non-equivalent features inside the *implementation* of a "subset CL", as long as they are not documented/exported to the "user", so that any "valid program" that is written in the "subset CL" does not explicitly depend on those features and thus will run "with equivalent semantics" on any conformant full ANSI CL.]

```
+-----
| You could even subset Emacs Lisp and use that as an implementation
| substrate, which would get us a lot closer to a "minimal" subset.
+-----
```

Right, but consider the above comments on the relationship between the implementation substrate and the "subset CL" which it hosts.

I may differ from others in this regard, but personally I am not so much interested in "just another small Lisp" [there are a plethora of those!] as I am in a "reasonably-shaped" [no "sharp corners", though huge "missing areas" are o.k. as long as the edges are "clean"] language which meets all of the following criteria:

Re: Minimal keywords needed for constructing a full CL system

1. Is substantially "smaller" than full ANSI CL by at least *some* "obvious"/noncontroversial metric(s). [Else what's the point?!? Just use full CL. There are plenty of those already.]
2. Is "large enough" to be generally useful for at least *some* area(s) of interest -- say, "scripting", one-off programs, introductory tutorials/teaching, etc., or as an "extension language" [yet another ill-defined term, sorry] for embedding in other programs.
3. Meets the strict "letter of the law", CLHS 1.7 "Language Subsets", even if to do so one must bend the spirit a bit, e.g., hide/undocument/unexport portions of the implementation substrate so that they are *not* a part of the "valid programs" which are being considered to be written in the "subset".

The CLHS "Issue SUBSETTING-POSITION Writeup" [linked from CLHS 1.7] makes very interesting reading, especially the discussion of the then-current practice in other languages, particularly these bits:

Cobol had multiple levels of subsets. However, the only two levels that were important were the minimum subset and the full language. The middle levels were seldom used other than transient points to the full language.

...

At one point, there was an ANSI standard for "Minimal Basic". It was too minimal.

That's probably my real interest in this topic. Phrased another way, *is* there any (at least one) subset of CL [by the CLHS 1.7 definition] that is neither "too minimal" nor too close to the full language? It is not yet clear to me that there is... or isn't.

+-----

| > - At least one of the set of non-local control transfer primitives
| > sufficient to implement the rest, i.e. GO, RETURN-FROM, THROW, and
| > their establishing forms TAGBODY, BLOCK, & CATCH.

|

| Right. Emacs Lisp only has catch and throw.

+-----

Right, but as Baker's paper shows, any of the three pairs is "primitive" with respect to the other two. In my "QDL" (attempt at a subset), I'm probably going to use TAGBODY/GO as the "primitive" one, if only because I want those forms with implicit TAGBODYs [DO, DO*, DO{LIST,TIMES}, etc.] to be efficient.

+-----

| > - The CL exception system, minimally [Kent might suggest a better set]
| > MAKE-CONDITION, SIGNAL, HANDLER-BIND, RESTART-BIND, FIND-RESTART, and

Re: Minimal keywords needed for constructing a full CL system

| > INVOKE-RESTART.

|

| None of these are primitive. The binding forms can just add some
| information to a dynamic variable. SIGNAL and FIND/INVOKE-RESTART
| search that information and calls a closure. MAKE-CONDITION can be a
| thin layer on top of MAKE-INSTANCE. (And all of CLOS can be
| implemented from more primitive constructs.)

+-----

O.k., fine. And as Kent has pointed out elsewhere in this thread:

...in an error system...really none of the functionality needs
to be primitive. This was already possible to write in portable CL.
What was not portable to write was the "fact" that it gets called.
That is, what makes an error system interesting is not its
"capability" but that it is the chosen entry point for doing what
it does. All of what it does is non-primitive, and yet it is not
"the error system" it is just "a toy" unless it's the function the
system calls when it detects an error.

So at least enough hooks have to be provided in the implementation
substrate that the implementation of the subset *will* find it
convenient -- nay, necessary -- to call the defined condition
system rather than either do nothing or do something idiosyncratic.
In particular, it needs to be possible to cleanly "call back into"
the CL subset during exception handling.

+-----

| > - The full numeric tower: Reals [float, rational, integer (fixnum
| > and bignum)] and Complex [pairs of those] and the usual related
| > functions.

|

| Emacs Lisp only has fixnums and floats. In my implementation, all
| other types of numbers are built from those. Of course, floats could
| be implemented with fixnums.

+-----

It's that sticky little "equivalent semantics" clause that worries me.
Right now, I could make the case [though it would be a stretch! ;-]]
that fixnum-only QDL *does* meet the clause, since it catches fixnum
overflow or non-integer division result and throws an error. One could
claim that a program that does that is not a "valid program", and thus
is not in the subset. The SUBSETTING-POSITION:NONE writeup allows that:

Some subsets are "dynamically" determinable, e.g, a subset might
signal an error if [an invalid argument].

...

Some "subsets" might be merely restrictive interpretations, e.g.,
a "run-time" implementation that [disallows some calls or exits on
BREAK].

Re: Minimal keywords needed for constructing a full CL system

But once you extend a fixnum-only subset to include floats, with automatic coercion to floats for non-fixnum results, then I think you can no longer claim to meet the "equivalent semantics" clause!!

And if Emacs Lisp is your implementation substrate, then the expression (/ 3 5) ==> 0.6 would fail to have equivalent semantics in full CL.

+-----
| > - Lists (of course!), symbols/packages, characters, vectors (including
| > strings), structures, and enough hooks to add fully-general arrays
| > later.
|
| It's enough to build up from conses, symbols, and vectors. Emacs Lisp
| has more data types, which is useful, but not necessary. I guess
| vectors could go if you don't need their O(1) properties.
+-----

For a "useful" [IMHO] scripting language, you do need O(1) access to strings, so vectors might as well stay. And as you point out, from vectors come structs, and from structs, CLOS.

+-----
| You don't need hash tables, but weak hash tables are very very handy,
| and I don't see how those could be implemented outside the core.
+-----

Just curious... How are weak hash tables more useful than generic weak pointers? [You need weak pointers for finalization, which I'll grant can be useful.]

+-----
| > - CLHS 3.2.2.2 "Minimal Compilation", so you can at least have macros
| > that only expand once. [If you're going to build up everything from
| > some minimal core, you need this to avoid horrible inefficiencies.]
|
| You do need minimal compilation, but the (minimal) compiler can be
| written in the core language. If you consider bootstrapping a solved
| problem, I don't think the compiler is a necessary primitive.
+-----

Well, if you allow the use of an external full CL for your bootstrapping [the way SBCL does, for example], then, yes, bootstrapping is a "solved problem". ;-) ;-) But I was kinda hoping to avoid requiring that.

Though standalone bootstrapping is a *lot* more work, and I'm no longer sure I have enough enthusiasm to do it that way. I'm particularly tempted by a suggestion from Carl Shapiro to "simply" build a run-time interpreter for CMUCL's byte-compiled code, which is a fairly simple stack machine. Then as long as the byte-compiled code only called other byte-code or "primitives" that were hand-coded, I could get a fairly high-quality compiler "for free" during the bootstrapping.

Re: Minimal keywords needed for constructing a full CL system

Re: Minimal keywords needed for constructing a full CL system

[Tempting, indeed...]

[Side note: It actually looks harder to read/parse CMUCL's FASL format than to interpret the byte-compiled codes contained *within* a FASL!! The FASL reader is its own separate stack machine, sort of.]

+-----
| > - CLHS 3.3.1 "Minimal Declaration Processing Requirements", which
| > [to me] also implies "doing the right thing" with dynamic and
| > lexical scoping, including special variables in LAMBDA lists.
|
| Declaration processing is just part of the compiler. You need one
| kind binding: lexical or dynamic. Or maybe none at all, just
| assignment to memory locations.
+-----

Well, mere "assignment to memory locations" doesn't provide the right semantics for LAMBDA, and if you have LAMBDA, then almost all the other binding constructs are just a hop, skip, & jump away.

+-----
| > - Places & DEFINE-SETF-EXPANDER & a basic set of SETF/INCF/etc.
|
| These are quite easy to write in the subset language.
+-----

Yup.

So anyway, I'm still curious as to whether any CL implementors or serious users have an opinion/belief/hunch about whether a "useful" subset of CL exists that meets CLHS 1.7...

-Rob

Rob Warnock <rpw3@xxxxxxxx>
627 26th Avenue <URL:<http://rpw3.org/>>
San Mateo, CA 94403 (650)572-2607

.