

Re: Unicode LISP??

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2004-09/0339.html>

From: Marcin 'Qrczak' Kowalczyk (qrczak_at_knm.org.pl)

Date: 09/04/04

Date: Sat, 04 Sep 2004 21:55:11 +0200

Ray Dillinger <bear@sonic.net> writes:

> *Okay... If you were designing a LISP, from the ground up, to be*
> *a fully unicode-aware language that "Does Unicode Right" what would*
> *you do?*

I'm not experienced with Common Lisp library, so it's hard to tell where it's incompatible with Unicode.

One thing that I noted previously: case mapping should be defined in terms of strings rather than characters.

Unfortunately this causes problems for the deeply hardwired case insensitiveness, because ignoring case is no longer such a simple thing. For example it would no longer be true that

(string= (string-downcase s) (string-downcase (string-upcase s)))
which fails for strings containing "ß", final small sigma, dotless i, apostrophe-n, long s, Greek iota under letter, ligatures like "fi" or other weird characters.

Unicode defines text mappings:

- upcasing
- downcasing
- titlecasing
- case folding

where case folding is the important one for case insensitive comparison. If two strings can be brought into the same string by other case operations, they case fold to the same. It's often the same as lowercasing, but it differs from it for various special cases like the above.

Neither lowercasing alone nor uppercasing alone is sufficient to fold all case differences. Lowercasing alone fails for characters like mentioned above. Uppercasing alone fails for capital I with dot above, Greek capital theta symbol and some compatibility variants of capital letters which don't have unique lowercase equivalents.

comp.lang.lisp: Re: Unicode LISP??

For me case sensitiveness in a programming language would be a good choice, but Lisp tradition is being case insensitive.

String representation is not obvious. Let's assume for now that from the programmer's point of view strings consist of code points.

If they are represented in UTF-8 or UTF-16, string indexing is not $O(1)$.

If they are represented in UTF-32, ASCII strings take 4 times more space than byte-packed ASCII would take.

If they are represented in UTF-32 or ISO-8859-1, depending on whether they contain some character above U+00FF, then strings may need to have their representation upgraded if they are updated in place.

Some languages don't have this problem by making strings immutable and using some other type for mutable strings (e.g. Python, Java, C#). It's fine for me, but again Lisp tradition is to have mutable strings.

Anyway, if they need to be upgraded, there are two wa