SPICE LISP FLAVORS

1. Introduction

Spice Lisp Flavors was written based on a textual description of version 5.0 Zetalisp. Various modifications have been made to convert Flavors into Common Lisp, but hopefully code that does not depend on other features of Zetalisp (such as the window system) will work in Slisp Flavors.

Basic Operation

Spice Lisp Flavors was deisgned to never lose state. Any change that affects other flavors first propogates notice of the change to all the affected flavors, the "dirty" flavors. The change is them made, and then a general "cleanup" procedure is invoked that updates all the dirty flavors.

This has a few important implications. One useful result is that it's quite simple to make efficient changes to large Flavors structures; all you have to do is inhibit cleanup until the changes are finished. However, various operations (such as using compiler-compile-flavors) use the cleanup machinery in order to function, and so must clean up before continuing.

Slisp Flavors also follows version 5.0 in making recompilation due to wrapper redefinition automatic. It does this generally by storing the sxhash of the wrapper code, and comparing this to the sxhash of the new defwrapper. (Actually, in interpreted interactions the code is compared with equal since that's faster, but wrapper code loaded from compiled files only includes the sxhash). This will probably detect a change in virtually all of the cases; however, it is conceivable that the user might enter a different wrapper that hashes to the same value. In this case, he will probably have to detect this himself, and use recompile-flavor to recalculate the combined method.

2. Additions to Slisp Flavors

Wrappers and whoppers, in addition to normal methods, can have any type. However, you must make your own method combinations in order to make use of this.

undefined-flavor-names [Variable]
: A list of referred-to but not yet defflavored flavors.

flavor-compile-methods [Variable]
: If this is T, newly calculated combined methods are automatically compiled. Our machine and compiler just isn't as fast as for a 3600.

dirty-flavors [Variable]
: Holds an array of the dirty flavors. Please don't alter.

cleanup-all-flavors [Function]
: If you interrupt a change in progress, you can use this to finish it.

without-cleaning-flavors &rest forms [Function]
: This executes the forms, but recompilation and updating of flavors that the execution of the forms may cause is done after executing them all. This is useful if you want to change a bunch of flavors that affect the same combined methods.

continue-whopper-all [Macro]
: This form, when executed in a whopper definition, causes the combined method to be continued with all the arguments received

by the whopper. This form exists because Common Lisp doesn't have stack &rest args, and we might save a lot of consing here.

3. Differences between Symbolics and Slisp Flavors

- There is no variant form of defmethod (that takes a function name). This could be done easily, but it would probably do an extra function call.

- Mixtures, and flavor-default-init-putprop, etc. are not yet implemented.

- Instances cannot be funcalled. If they are, it should probably be object-specific; i.e., funcalling an object sends a 'funcall message to the object with the arguments.

- The following things will probably never be implemented: instantiate-flavor, defun-method, defselect-method declare-flavor-instance-variables, locate-in-instance, describe-flavor forms; *flavor-compilations* and *flavor-compile-trace* specials; and :special-instance-variables and :default-handler flavor options. [To make a default handler, handle :unclaimed-message.]

- In the Symbolics implementation, any method having a type not used by the method combination defined for that method signals an error. This is currently not done in Slisp Flavors.

Continue-whopper &rest args [Macro]

Lexpr-continue-whopper &rest args [Macro]

These are macros in Slisp Flavors, so that the continuation can be called with extra implementation-dependent arguments.

recompile-flavor message &optional (do-dependents t) [Function]

Message can be either nil for all messages, a single message name, or a list of message names. Note that the arguments are different.

compile-flavor flavor [Function]

For use in the interpreter only. To do the right thing in compiled files, use compiler-compile-flavors, below. These are two forms because I couldn't think of a way to tell whether one was in the compiler or not.

compiler-compile-flavors &rest flavors [Macro]

This is different in a couple of ways from compile-flavor of all of the named flavors. Compile-flavor doesn't recompile anything it can find in the current environment; compiler-compile-flavors does, so that it will surely be there in the loadtime environment. Compiler-compile-flavors therefore also makes sure that each combined methods calculated is only calculated once.

Index

Table of Contents