A CLOS Version of the CLtL2 Environment Protocol

Robert Strandh
University of Bordeaux
351, Cours de la Libération
Talence, France
robert.strandh@u-bordeaux1.fr

ABSTRACT
The concept of an environment is mentioned in many places in the Common Lisp standard, but the nature of the object is not specified. For the purpose of this paper, an environment is a mapping from names to meanings.

In this paper, we propose a replacement for the environment protocol documented in the book Common Lisp the Language, second edition. Rather than returning multiple values as the functions in that protocol do, the protocol suggested in this paper is designed so that functions return instances of standard classes. Accessor functions on those instances supply the information needed by a compiler or any other code walker application.

The advantage of our approach is that a protocol based on generic functions and standard classes is easier to extend in backward-compatible ways than the previous protocol, so that implementations can suggest additional functionality on these objects. Furthermore, CLOS features such as auxiliary methods can be used on these objects, making it possible to extend or override functionality provided by the protocol, for implementation-specific purposes.

Categories and Subject Descriptors
D.3.4 [Programming Languages]: Processors—Code generation, Run-time environments

General Terms
Algorithms, Languages

Keywords
CLOS, Common Lisp, Environment

1. INTRODUCTION
The Common Lisp standard contains many references to environments. Most of these references concern lexical environments at compile time, because they are needed in order to process forms in non-null lexical environments. The standard does not specify the nature of these objects, though in CLtL2 [1] there is a suggested protocol that is sometimes supplied by existing Common Lisp implementations.

The protocol documented in CLtL2 has several problems. Functions in the protocol return multiple values, which makes it hard to extend. Furthermore, the protocol is incomplete. A typical compiler needs more information that the protocol provides, making implementation-specific extensions obligatory for the protocol to be useful. For that reason, although existing Common Lisp implementations often provide this functionality, it is not what the native compiler of the implementation actually uses.

In this paper, we propose a modern alternative protocol based on CLOS. Rather than returning multiple values, our protocol functions return instances of standard classes. Accessors for those instances can be used by compilers and other code walker applications in order to obtain the information needed for the task to be accomplished.

2. PREVIOUS WORK
2.1 Common Lisp the Language, second edition
2.1.1 Environment query
Section 8.5 of “Common Lisp, the Language, second edition” describes a set of functions for obtaining information from environment objects and some other operators related to environments.

For environment query, it defines three functions. We describe them briefly here.

The function variable-information takes a symbol and an optional environment object as arguments. It returns three values. The first value indicates the type of the binding (lexical variable, special variable, symbol macro, constant variable) or nil if there is no binding or definition in the environment for that symbol The second value is a Boolean, indicating whether the binding is local or global. The third argument is an association list containing declarations that apply to the binding.

The function function-information takes a function name and an optional environment as arguments. Again, three values are returned. The first value indicates the type of the
binding (function, macro, special operator\(^1\)) or \texttt{nil} if there is no binding or definition in the environment for that function name. As before, the second value indicates whether the definition is local or global, and the third value is an association list of declarations that apply.

The function \texttt{declaration-information} takes a declaration name and an optional environment as arguments. The declaration name can be the symbol \texttt{optimize}, the symbol \texttt{declaration}, or some implementation-defined declaration names. It returns a single value that contains information related to the corresponding declaration name.

To begin with, it is clear that this set of functions is insufficient to process all Common Lisp code, because no mechanism is described for querying the environment for information related to \textit{blocks} and \textit{go tags}. Functions for this purpose are provided as extensions by Allegro Common Lisp as described in Section 2.6.

### 2.1.2 Environment augmentation

For augmenting an environment, i.e., creating a new, augmented, environment from an existing one, the same section describes the function \texttt{augment-environment}.

### 2.2 SBCL

#### 2.2.1 Native

#### 2.2.2 CLtL2

The distribution of SBCL contain a contribution that supplies some of the functionality described in the book CLtL2 but that was not included in the Common Lisp standard. Part of this contribution is an implementation of the environment protocol of CLtL2.

### 2.3 CCL

### 2.4 CMUCL

### 2.5 ECL

### 2.6 Allegro

#### 2.6.1 Native environments

#### 2.6.2 Support for CLtL2 protocol

The documentation for Allegro Common Lisp contains a separate document describing their protocol for environments in the spirit of CLtL2.\(^2\) We summarize the differences between the Allegro implementation and the CLtL2 protocol here.

- Information about blocks and tags have been added in the form of two new functions \texttt{block-information} and \texttt{tag-information}.
- (say more)

### 2.7 LispWorks

### 2.8 CLtL2 compatibility system

\texttt{cl-environments}

\(^1\)The term used in the book is \textit{special form}, but the terminology has been improved since then

\(^2\)https://franz.com/support/documentation/current/doc/environments.htm

---

3. OUR TECHNIQUE

4. BENEFITS OF OUR METHOD

5. CONCLUSIONS AND FUTURE WORK

6. REFERENCES