Hygenic Macro Expansion

History of Programming Languages

Zane Shelby
zane@ccs.neu.edu

1 Hygenic Macro Expansion


1.1 Abstract

Macro expansion in current Lisp systems is naive with respect to block structure. Every macro function can cause the capture of free user identifiers and thus corrupt intended bindings. We propose a change to the expansion algorithm so that macros will only violate the binding discipline when it is explicitly intended.

1.2 Significance

Hygenic Macro Expansion was the first paper that presented an algorithm that accomplishes automatic renaming (alpha-conversion) of bound variables to satisfy the hygiene condition. Unfortunately, since the algorithm traverses each expression again after it has been converted it has quadratic runtime complexity on the number of expressions in the source.

2 Macros that Work

2.1 Abstract

This paper describes a modified form of Kohlbeckers algorithm for reliably hygienic (capture-free) macro expansion in block-structured languages, where macros are source-to-source transformations specified using a high-level pattern language. Unlike previous algorithms, the modified algorithm runs in linear instead of quadratic time, copies few constants, does not assume that syntactic keywords (e.g. if) are reserved words, and allows 10CSJ(scoped) macros to refer to lexical variables in a referentially transparent manner.

Syntactic closures have been advanced as an alternative to hygienic macro expansion. The problem with syntactic closures is that they are inherently low-level and therefore difficult to use correctly, especially when syntactic keywords are not reserved. It is impossible to construct a pattern-based, automatically hygienic macro system on top of syntactic closures because the pattern interpreter must be able to determine the syntactic role of an identifier (in order to close it in the correct syntactic environment) before macro expansion has made that role apparent.

Kohlbeckers algorithm maybe viewed as a book-keeping technique for deferring such decisions until macro expansion is locally complete. Building on that insight, this paper unifies and extends the competing paradigms of hygienic macro expansion and syntactic closures to obtain an algorithm that combines the benefits of both. Several prototypes of a complete macro system for Scheme have been based on the algorithm presented here.

2.2 Significance

Macros that Work presents an algorithm that satisfies the hygiene condition and does not have the quadratic complexity of the algorithm presented in Hygienic Macro Expansion as it marks only the identifiers introduced by macro transformations. This added performance comes at the cost of only being able to expand macros that have been defined in a high-level pattern-matching language.

3 Syntactic Abstraction in Scheme

3.1 Abstract

Naive program transformations can have surprising effects due to the interaction between introduced identifier references and previously existing identifier bindings, or between introduced bindings and previously existing references. These interactions can result in inadvertent binding, or capturing of identifiers. A further complication is that transformed programs may have little resemblance to original programs, making correlation of source and object code difficult. This article describes an efficient macro system that prevents inadvertent capturing while maintaining the correlation between source and object code. The macro system allows the programmer to define program transformations using an unrestricted, general-purpose language. Previous approaches to the capturing problem have been inadequate, overly restrictive, or inefficient, and the problem of source-object correlation has been largely unaddressed. The macro system is based on a new algorithm for implementing syntactic transformations and a new representation for syntactic expressions.

3.2 Significance

The expansion algorithm presented in Syntactic Abstraction in Scheme has similar performance to the algorithm presented in Macros that Work, but sheds the requirement that macros it expands be defined in a high-level description language. Additionally it facilitates correlation between expressions in the source language and the target language.