

Package Locks: Marking Abstraction Boundaries

Nikodemus Siivola*
Department of Computer Science
Helsinki University of Technology
PL 5400, 02015 TKK, Finland

Christophe Rhodes†
Department of Computing
Goldsmiths College
University of London
London SE14 6NW

June 10, 2005

Abstract

Package locks are a class of vendor extensions supported by several Common Lisp implementations. We examine their rationale and features in the context of our own work in designing and implementing package locking for SBCL. We also point out possible directions for future work, both practical and theoretical, and opportunities for harmonization between implementations.

1 Outline

We discuss the concept of package locks in Common Lisp, in the context of our recent implementation for SBCL. We examine the rationale for overrideable per-package namespace protection, and discuss how this leads to a set of desired properties for these locks. These properties, briefly, are: that they protect against undefined consequences in interactions involving the COMMON-LISP package and make similar protection available to user packages; that they have negligible runtime cost in normal operation; that both lexical and dynamic control be available to the implementor and client code; that development style and available idioms in locked packages are not constrained; and that lock violations do not lead to multiple unnecessary debugger invocations.

We examine some previous implementations of package locks, discussing where, as applicable, these implementations do not meet these desired criteria, before presenting our own implementation, which meets our requirements and is sufficiently well-specified, we believe, that it could act as a starting point for future work.

We conclude by examining a subset of our specification that covers what we find the most salient features found in other implementations while relaxing only the level of protection from our stated goal, and note the possibility of further extensions such as the ability to protect at a finer granularity than the package level. Finally, we allude to the scope for related theoretical work regarding protection of abstraction boundaries in dynamic languages and environments.

*nikodemus@random-state.net

†c.rhodes@gold.ac.uk