Goldsmiths UNIVERSITY OF LONDON

Package Locks

Marking Abstraction Boundaries

Nikodemus Siivola
Helsinki University of Technology
Helsinki, Finland

Christophe Rhodes
Goldsmiths College
University of London

Goldsmiths UNIVERSITY OF LONDON

Outline

- ANSI CL 11.1.2.1.2
  - “Thou shalt not”
- Implementation constraints
  - “We shall not allow”
  - “We will not claim to prevent”
- Conclusions
  - “Demo”

Goldsmiths UNIVERSITY OF LONDON

ANSI CL 11.1.2.1.2

- Thou shalt not cause an external symbol of CL
  - to be bound (lexically or dynamically)
  - to be bound as a function
  - to be bound as a macro or compiler-macro
  - to name a type specifier
  - to name a structure
  - to be a declaration
  - to be a symbol macro
  - to alter its home package
  - to be traced

Goldsmiths UNIVERSITY OF LONDON

11.1.2.1.2 (cont...)

- ...
  - to be declared or proclaimed special
  - to have type or ftype declared or proclaimed
  - to be removed from the CL package
  - to have a setf expander defined
  - to undefine or bind a setf function name
  - to name a method combination
  - to be passed to (setf find-class)
  - to be bound as a restart name or catch tag
  - (methods with d.i.s on generic functions)

Goldsmiths UNIVERSITY OF LONDON

11.1.2.1.2 rationale

- Can cause catastrophic problems for implementation
  - assumptions of static base
- Violations non-local
  - almost all software uses CL package
  - colliding functionality

Goldsmiths UNIVERSITY OF LONDON

SBCL philosophy

- Prevent users from writing accidentally unportable programs
  - or: annoy users as much as possible
  - should be able to alert user to portability problems
    - but in any case, depend critically on some of 11.1.2.1.2 not to be violated
      - e.g. STRING (Maxima special variable)
  - leads to package lock concept

Goldsmiths UNIVERSITY OF LONDON

package locks

- Design criteria
  - Protect the user as much as possible from unintentional violations of 11.1.2.1.2
  - Allow conforming code to run unmodified
  - Negligible performance penalty for conforming code
  - Straightforward debugger interface for manipulation of violations
  - Generalize to packages other than CL where sensible

Goldsmiths UNIVERSITY OF LONDON

User protection

- protect user from unintentional violations of ANSI CL 11.1.2.1.2
  - formally unportable code
  - leads to bugs that can be difficult to diagnose
    - special STRING leads to compiler failure
- two forms of violations
  - operations on symbols
  - operations on packages

Goldsmiths UNIVERSITY OF LONDON

Conforming code

- Correct, conforming code should be unaffected
  - default state of non-implementation packages is unlocked
  - (state of implementation-specific packages can be locked)
  - exception: CL package is locked against interning, where (intern “FOO” “CL”) is formally conforming

Goldsmiths UNIVERSITY OF LONDON

Performance

- Correct code should not have worse performance
  - compile-time checking (where possible)
  - lock consistency requirement
    - “undefined” if compile-time locks not the same as run-time locks
    - get-out-of-jail-free clause
    - no load-time errors for interning symbols into packages unlocked at compile-time

Goldsmiths UNIVERSITY OF LONDON

Debugger interface

- One condition per locked package per operation

```
(defclass foo:point ()
  ((x :accessor bar:x)
   (y :accessor bar:y)))
```

- if FOO and BAR are locked, leads to exactly two lock violation conditions

Goldsmiths UNIVERSITY OF LONDON

Generalization to other packages

- DEFPACKAGE :lock keyword
- declarations disable-package-locks and enable-package-locks

```
(defmacro with-foo (&body body)
  `(locally (declare (disable-package-locks foo))
     (flet ((foo (x) x))
       (declare (enable-package-locks foo))
       ,@body)))
```

Goldsmiths UNIVERSITY OF LONDON

Similar concepts

- Allegro CL
- CLISP
- Others?

Goldsmiths UNIVERSITY OF LONDON

Conclusions

- Value in decreasing free-for-all
  - implementation assumptions
  - prevent library / application collisions
- Implementation virtues
  - does everything we asked for
  - no complaints!
- “Demo”...