Method Combinators

ELS 2018

Didier Verna
EPITA / LRDE

didier@lrde.epita.fr
Introduction

- CLOS improvements over mainstream object systems
  - Multiple dispatch
    *Increased SOC: polymorphism / inheritance*
  - MOP
    *Homogeneous behavioral reflection*
  - Method combinations
    *Increased SOC: methods / dispatch*

- Standardization drawbacks
  - Method combinations underspecified
    *Considered not mature enough*
  - MOP only a later addition
    *Unclear or contradictory protocols*
Method Combinations Issues

The Case of SBCL

Method Combinators

Combined Generic Functions

Alternative Combinators

Performance
Method Combinations Issues

The Case of SBCL

Method Combinators

Combined Generic Functions

Alternative Combinators

Performance
Orthogonality

Short combination example

(defun details (human)
  (:method-combination append :most-specific-last)
  (:method append ((human human)) ...)
  (:method append ((employee employee)) ...))

Problems

- Method qualification required
  Combination change impractical
- Except for the option
  Inconsistent
- No :before or :after methods
  No good reason

Workaround: long method combinations
Introduction

Issues

SBCL

Combinators

CGFs

Alt. MCs

Perfs

Conclusion

Structure

System classes

\textit{method-combination}

\textless\text{unspecified}\textgreater

Abstract

- Portable specialization impossible
  \textit{At least one implementation-specific (sub)class}

- Unclear nature (classes vs. instances)
  \textit{Mix of define / call-time parametrization}
Lookup (MOP)

`find-method-combination gf name options`

- “called to determine the combination object used by a generic function”
  - What are name and options for?
  - Error behavior?
  - There already is generic-function-method-combination
Generic function invocation protocol (MOP)

compute-effective-method gf combination methods

- What is combination for?
- Caching policy unspecified
  
  Contrary to applicable methods
Method Combinations Issues

The Case of SBCL

Method Combinators

Combined Generic Functions

Alternative Combinators

Performance
Method combination classes hierarchy

- **standard-method-combination**
  - type-name
  - options
- **short-method-combination**
  - operator
  - identity-with-one-argument
- **long-method-combination**
  - function
  - args-lambda-list

- options: use-time (:method-combination options)
- Below: define-time
Short Method Combinations

Creation

```
(define-method-combination name option*)
```

```
(find-method-combination gf (eql name) options)
```

- No global namespace
  - One method combination object per generic function
  - Redefinitions don’t affect existing generic functions

- `find-method-combination` ≠ the expected or the specified
Long method combination functions

*long-method-combination-functions*

- long-method-combination
  - function
  - args-lambda-list

- Similar behavior, one additional oddity
  - Local method combination objects
  - Global method combination functions
Long Method Combinations (cont.)

**Code**

```lisp
(define-method-combination my-progn ()
  ((primary () :order :most-specific-first :required t))
  `(progn ,@(mapcar (lambda (method)
               `(call-method ,method))
             primary)))

(defun test (i)
  (:method-combination my-progn)
  (:method ((i number)) (print 'number))
  (:method ((i fixnum)) (print 'fixnum)))
```

**REPL**

```
CL-USER> (test 1)
FIXNUM
NUMBER
```
Code

(define-method-combination my-progn ()
  ((primary () :order :most-specific-last :required t))
  `(progn ,@(mapcar (lambda (method)
        `(call-method ,method)
        primary)))))

REPL

CL-USER> (test 1)
FIXNUM
NUMBER
Code

(defmethod test ((i float)) (print 'float))

REPL

CL-USER> (test 1.5)  CL-USER> (test 1)
NUMBER             FIXNUM
FLOAT              NUMBER
Overview

Classes

- short-method-combination
- long-method-combination
- method-combinator-mixin
- short-method-combinator
- long-method-combinator

- Stored in a global hash table
- [setf] find-method-combinator
In 3 layers

- define-[short|long]-method-combinator
- ensure-[short|long]-method-combinator
- ensure-[short|long]-method-combinator-using-class
In 4 steps

1. Define a regular combination
   (find-method-combination)
2. Retrieve it
3. Make it combinator
   (change-class)
4. Store it
   (setf find-method-combinator)

Note: regular combination injection
Method Combinations Issues

The Case of SBCL

Method Combinators

Combined Generic Functions

Alternative Combinators

Performance
Overview

Classes

- standard-generic-function
- funcallable-standard-class
- combined-generic-function
- functions

Wrappers

(defcombined cgf (args...)
  (:method-combinator mc)
  ...)

Method Combinators / ELS 2018 – Didier Verna
Method Combinator Management

- **Initialization**

  ```lisp
  (defmethod find-method-combination (cgf-class-prototype ...) 
   (find-method-combinator ...))
  ```

- **Sanitation**

  ```lisp
  (defmethod find-method-combination (cgf ...) 
   (method-combinator cfg) #|or mismatch error|#)
  ```

- **Updating**

  ```lisp
  (change-method-combinator cgf method-combinator)
  ```
Client Maintenance

- **Client registration:**
  - ([re]initialize-instance cgf ...)

- **Client updating:**
  - (reinitialize-instance-instance mc ...)
  - (u-i-f-d-c mc ...)

**New protocol**

```
method-combinator-mixin
  clients
```

- `make-clients-obsolete`
  - `update-combined-generic-function-for-redefined-method-combinator`

Method Combinators / ELS 2018 – Didier Verna
Plan

Method Combinations Issues

The Case of SBCL

Method Combinators

Combined Generic Functions

Alternative Combinators

Performance
Overview

- **Idea:** generic functions / combinators complete decoupling
- **Use:** ≠ logical method combinations, selected methods etc.
- **Note:** already possible, but extremely costly
  - 2 calls to reinitialize-instance

Protocols

```lisp
(call-with-combinator (find-method-combinator 'combinator)
  #'func arg1 arg2 ...)

(call/cb combinator func arg1 arg2 ...)

#!combinator(func arg1 arg2 ...)
```
Optimization

- Discriminating functions caches
- Client maintenance aware of them
- Cost
  - First alternative call: as before
  - Next: 1 or 2 hashtable lookups

**Warning:** discriminating functions must close over all caches!
Plan

Method Combinations Issues

The Case of SBCL

Method Combinators

Combined Generic Functions

Alternative Combinators

Performance
Performance

- Numeric (10e8 iterations)
  - std
  - std
  - alt. :+
  - alt. :+

- I/O (10e7 iterations)
  - std
  - std
  - alt. :progn
  - alt. :progn
**Conclusion**

- Method combinations are powerful yet underspecified
- Method combinators improve their consistency
- Code available on GitHub

**Perspectives**

- Refine / properly package implementation
- Port to other compilers
- Experiment with “floating” floating methods