



Introduction



Issues



SBCL



Combinators



CGFs



Alt. MCs



Perfs



Conclusion

Method Combinators

~ ELS 2018 ~

Didier Verna
EPITA / LRDE

didier@lrde.epita.fr



[lrde/~didier](#)



[@didierverna](#)



[didier.verna](#)



[google+](#)



[in/didierverna](#)



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





Plan



Method Combinations Issues

The Case of SBCL

Method Combinators

Combined Generic Functions

Alternative Combinators

Performance





Plan



Method Combinations Issues

The Case of SBCL

Method Combinators

Combined Generic Functions

Alternative Combinators

Performance



Orthogonality

Short combination example

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

► Problems

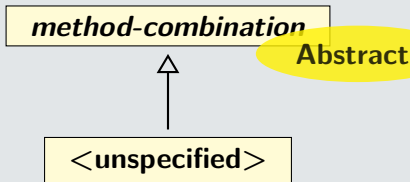
- Method qualification required
Combination change impractical
- Except for the option
Inconsistent
- No `:before` or `:after` methods
No good reason

► Workaround: long method combinations



Structure

System classes



- ▶ Portable specialization impossible
At least one implementation-specific (sub)class
- ▶ Unclear nature (classes vs. instances)
Mix of define / call-time parametrization



Protocols

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



Protocols (cont.)

Generic function invocation protocol (MOP)

```
compute-effective-method gf combination methods
```

- ▶ What is combination for?
- ▶ Caching policy unspecified
Contrary to applicable methods





Plan



Method Combinations Issues

The Case of SBCL

Method Combinators

Combined Generic Functions

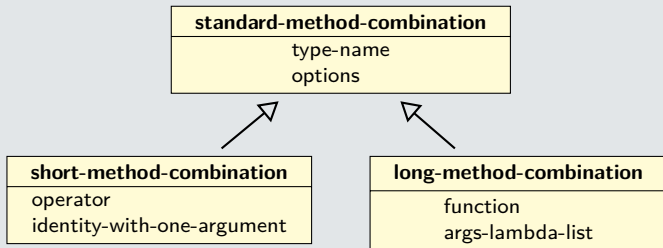
Alternative Combinators

Performance



Classes

Method combination classes hierarchy



- ▶ `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)
```

```
short combination object
```

- ▶ **No global namespace**
 - ▶ One method combination object per generic function
 - ▶ Redefinitions don't affect existing generic functions
- ▶ `find-method-combination` \neq the expected or the specified



Long Method Combinations

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

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

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

REPL

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



Long Method Combinations (cont.)

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
```



Long Method Combinations (cont.)

Code

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

REPL

```
CL-USER> (test 1.5)
```

```
NUMBER
```

```
FLOAT
```

```
CL-USER> (test 1)
```

```
FIXNUM
```

```
NUMBER
```





Plan



Method Combinations Issues

The Case of SBCL

Method Combinators

Combined Generic Functions

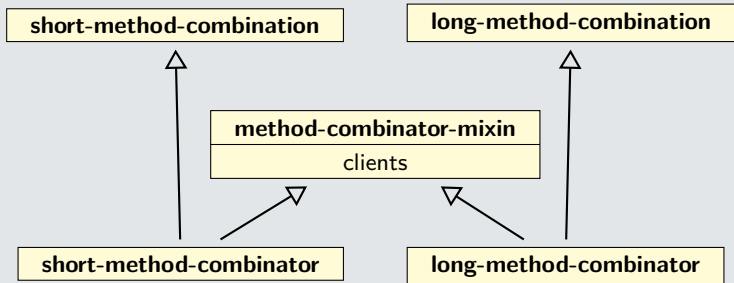
Alternative Combinators

Performance



Overview

Classes



- ▶ Stored in a global hash table
- ▶ `[setf] find-method-combinator`



Protocols

In 3 layers

`define-[short|long]-method-combinator`



`ensure-[short|long]-method-combinator`



`ensure-[short|long]-method-combinator-using-class`

Implementation (layer 3)

In 4 steps

define a regular combination

(find-method ↓ combination)

retrieve it

(change ↓ class)

make it combinator

(setf find-method-combinator ↓)

store it

- Note: regular combination injection





Plan



Method Combinations Issues

The Case of SBCL

Method Combinators

Combined Generic Functions

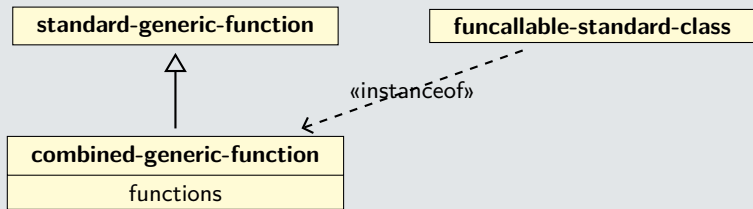
Alternative Combinators

Performance



Overview

Classes



Wrappers

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



Method Combinator Management

► Initialization

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

► Sanitation

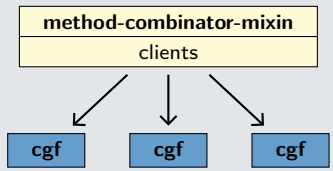
```
(defmethod find-method-combination (cgf ...)  
  (method-combinator cfg) #/or mismatch error/#)
```

► Updating

```
(change-method-combinator cgf method-combinator)
```



Client Maintenance



- ▶ Client registration:
([re]initialize-instance cgf ...)
- ▶ Client updating:
(reinitialize-instance mc ...)
(u-i-f-d-c mc ...)

New protocol

make-clients-obsolete



update-combined-generic-function-for-redefined-method-combinator





Plan



Method Combinations Issues

The Case of SBCL

Method Combinators

Combined Generic Functions

Alternative Combinators

Performance



Overview

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

Protocols

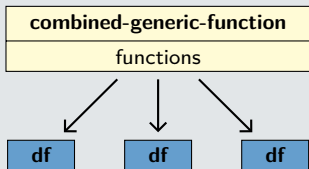
```
(call-with-combinator (find-method-combinator 'combinator)
  #'func arg1 arg2 ...)
```

```
(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

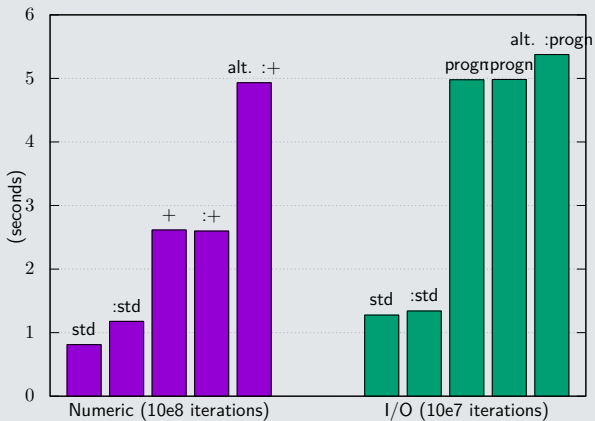
Combined Generic Functions

Alternative Combinators

Performance



Performance



Conclusion / Perspectives

▶ 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

