An Update on the Method Combinations MOP

ELS 2026, Kraków, Poland

Didier Verna
<didier@didierverna.net>

May 11


didierverna.net


@didierverna


didier.verna


in/didierverna

Build 2026-05-13 13:09:23+02:00

Plan

Introduction

method-combination

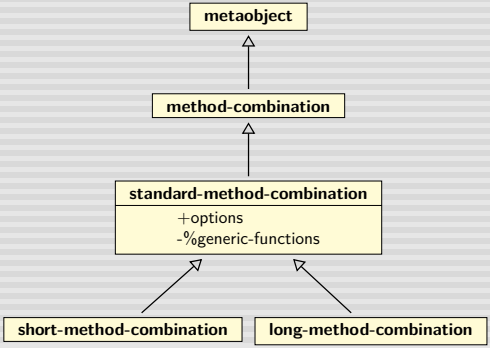
`define-method-combination` *(macro)*

two forms (short / long) with options

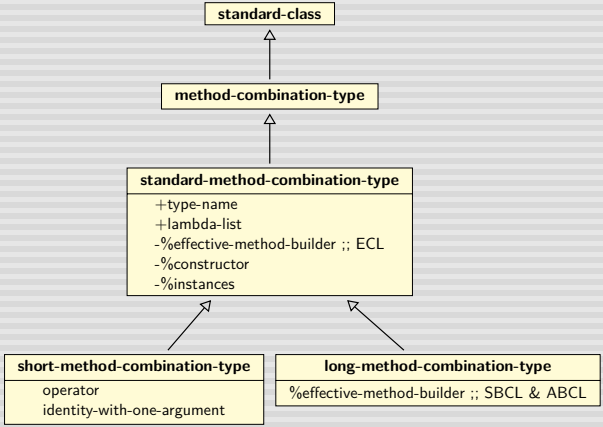
Plan

Structure

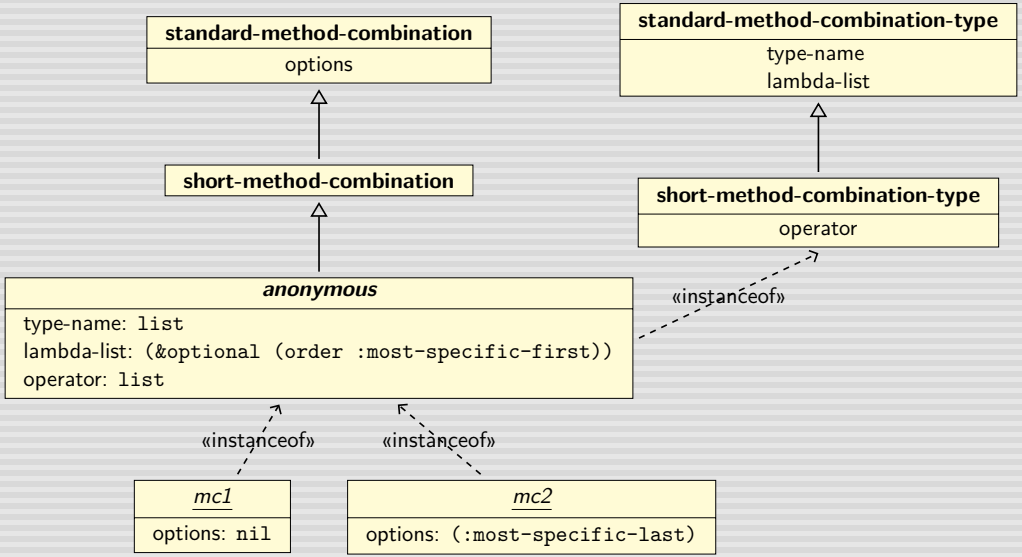
1. A hierarchy



2. A meta-hierarchy



Example: the `list` (short) method combination



Usage

define-method-combination

```
;; short form
:method-combination-class name
:method-combination-type-class name
:method-combination-type-class (name initargs*)

;; long form
(:method-combination-class name)
(:method-combination-type-class name initargs*)
```

- Appropriate defaults + validation protocol

Access

```
find-method-combination-type (name &optional (errorp t))
"Find a NAMEd method combination type.
If ERRORP (the default), signal an error if no such method combination
type is found. Otherwise, return NIL."
```

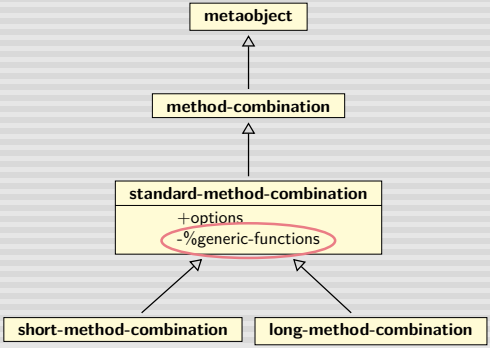
```
find-method-combination-instance (name &optional options (errorp t))
"Find a method combination object of type NAME, for OPTIONS. If ERRORP
(the default), signal an error if no NAMEd method combination type is
found. Otherwise, return NIL.

Note that when a NAMEd method combination type exists, asking for a new
set of (conformant) OPTIONS will always instantiate the combination
again, regardless of the value of ERRORP."
```

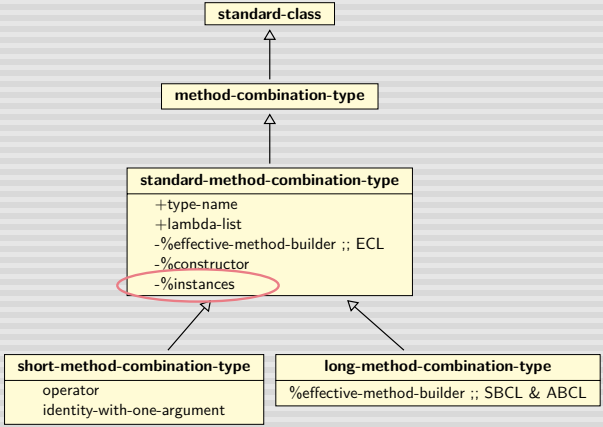
- A better `find-method-combination`

Consistency

1. A hierarchy



2. A meta-hierarchy



Consistency

- A generic function changing its method combination
  GF caches updated via the standard (re)initialization protocols
- A method combination being redefined
  MC instances cache updated via `change-class` $\longrightarrow$ `update-instance-for-different-class`
  $\longrightarrow$ `(update-generic-function-for-redefined-method-combination gf old-mc new-mc)`

Plan

Prior status

- **ABCL**
  - Closette-based CLOS/MOP (similar to PCL ⇒ pre-2018 SBCL)
  - 3 classes: `[short-|long-]method-combination`
  - Mixture of define-time (*e.g.* `operator`) and use-time (*e.g.* `options`) slots
  - Standard method combination = instance of `method-combination` ⇒ non-conformant
  - `find-method-combination` recreates new instances every time ⇒ inconsistent
- **ECL**
  - *Not* PCL-based, declarative CLOS/MOP class hierarchy
  - Only one class: `method-combination`
  - `define-method-combination` *does not* create instances

    Only creates an effective method builder function
  - No distinction between short and long method combinations

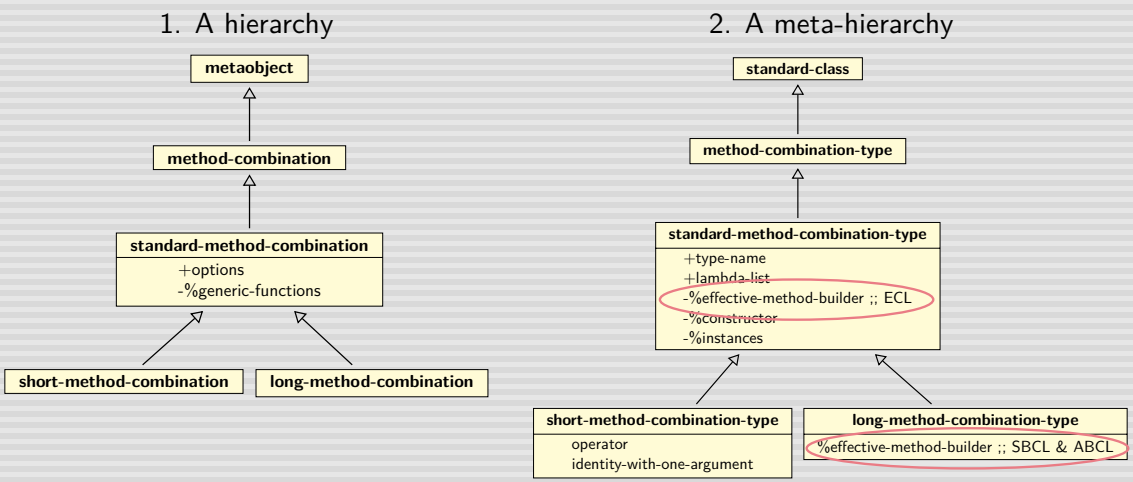
    Each short method combination gets its own effective method builder function
  - `find-method-combination` reinstantiates `method-combination` every time

    ⇒ non-conformant & inconsistent

Upgrade

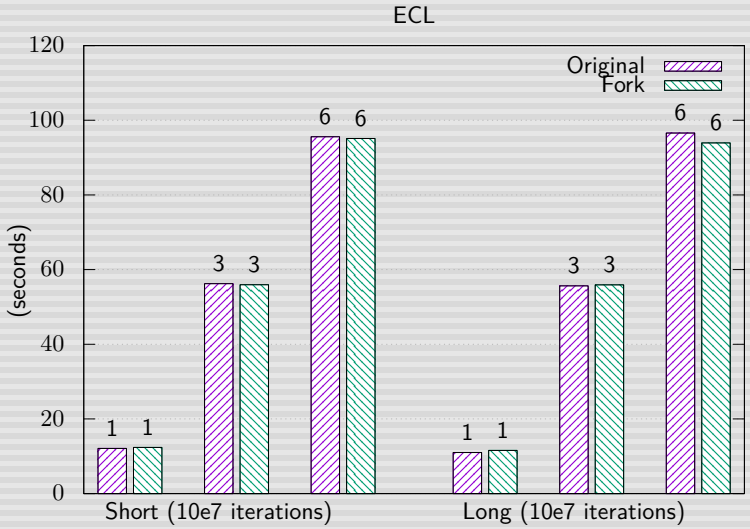
- **ABCL**
  - A single `early-method-combination` class during bootstrap
  - Full infrastructure injection afterwards
- **ECL**
  - No bootstrapping issues
  - Declarative CLOS/MOP class hierarchy extended

Upgrade

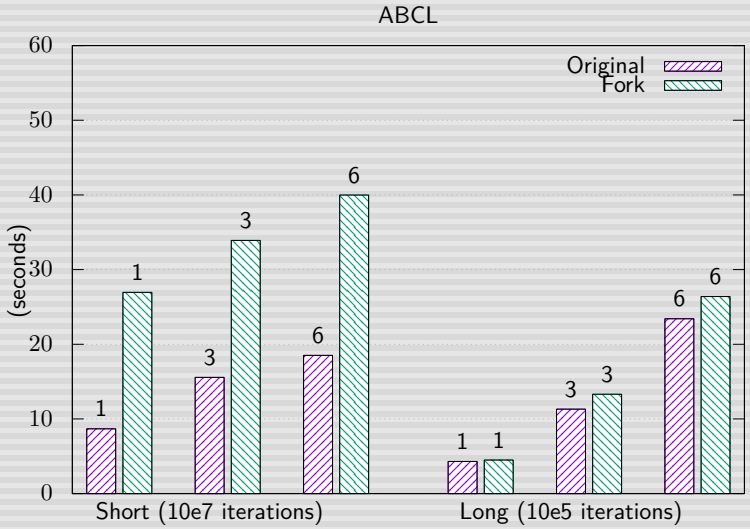


Plan

Performance (`compute-effective-method` calls)



Performance (`compute-effective-method` calls)



Plan

Specification

```
;; Both readers and SETF writers as follows.
(documentation symbol 'method-combination)
(documentation mc-object t)
(documentation mc-object 'method-combination)
```

- No required consistency

Prior Status

- **SBCL**
  - Documentation initially available the 3 ways
  - Subsequent divergence between `symbol` and `mc-object`
- **ABCL & ECL**
  - No methods on `mc-object`
  - Thus, no divergence, but not conforming

Proposition

```
(documentation symbol 'method-combination)
(documentation mc-object 'method-combination)
(documentation mc-type 'method-combination)
```

Access the documentation string attached to `symbol`, a method combination type name, where `symbol` = `(type-name mc-type)`, and `mc-type` = `(class-of mc-object)`.

```
(documentation mc-object t)
(documentation mc-type t)
```

Access the documentation string of `mc-type`, where `mc-type` = `(class-of mc-object)`.

- Potential use: distinguish usage doc from implementation doc

Plan

Related Work: SICL

- `mc-object` = instance of `standard-method-combination`
  We have 3 classes instead
- `mc-type` = instance of `method-combination-template`
  We create new classes instead

Bottom line

```
(property (template mc-object)) ;; SICL: agregation
(property (class-of mc-object)) ;; this work: meta-level
```

Plan

Conclusion

- 3 working implementations
  - `https://github.com/didierverna/sbcl/tree/method-combination-types`
  - `https://github.com/didierverna/abcl/tree/method-combination-types`
  - `https://gitlab.com/didierverna/ecl/-/tree/method-combination-types?ref_type=heads`
- Testing library with 144 unit tests + 1 experimental feature
  - `https://github.com/didierverna/els2023-method-combinations`

- **TODO**
  - More implementations (CCL, CMU-CL)
  - Performance problem in ABCL
  - Ensure consistency between MC lambda lists and GF requests (SICL does that)