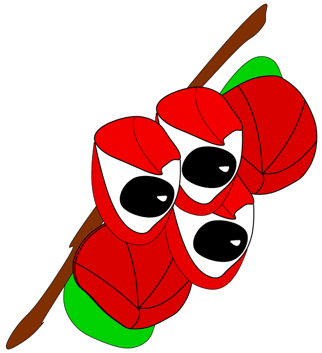


Guaraná

Reflexão Computacional e Aplicações

http://www.ic.unicamp.br/~oliva/guarana/

Alexandre Oliva

aoliva@redhat.com

Red Hat, Inc

Luiz Eduardo Buzato

buzato@ic.unicamp.br

Instituto de Computação, Unicamp

Roteiro

- Motivação
- Reflexão Computacional
- **Guaraná**
- **MOLDS**
- Presente e Futuro

Motivação

- Reuso de *software*
- Adaptação
  - Engenharia reversa
- Compreensão
  - Documentação, tutorias, exemplos
  - Observação
- Ferramentas de introspecção
  - *Java Core Reflection*
  - Depuradores simbólicos
  - Reflexão Computacional

Java Core Reflection

- Informação estrutural
  - `Object.getClass()`
  - `Class.getFields()`
  - `Class.getMethods()`
- Manipulação reflexiva
  - `Field.get()`, `Field.set()`
  - `Method.invoke()`
- Controle de acesso
- Não expõe o comportamento

Depuradores simbólicos

- *Breakpoints*
  - Condições
- *Watchpoints*
  - Eficiência
- Instanciação de objetos
  - Conhecimento prévio

Reflexão Computacional

- Estrutural
- Comportamental

Alternativas

- Interpretadores
- Pré-processadores
- Interceptadores

Interpretadores

- Inerentemente reflexivos
- Implementação na própria linguagem
  - Quebra de circularidade
- Alterações ao interpretador
  - Dificuldade
    - recai no problema de entender o interpretador
  - Ineficiência
- Exemplo: Lisp

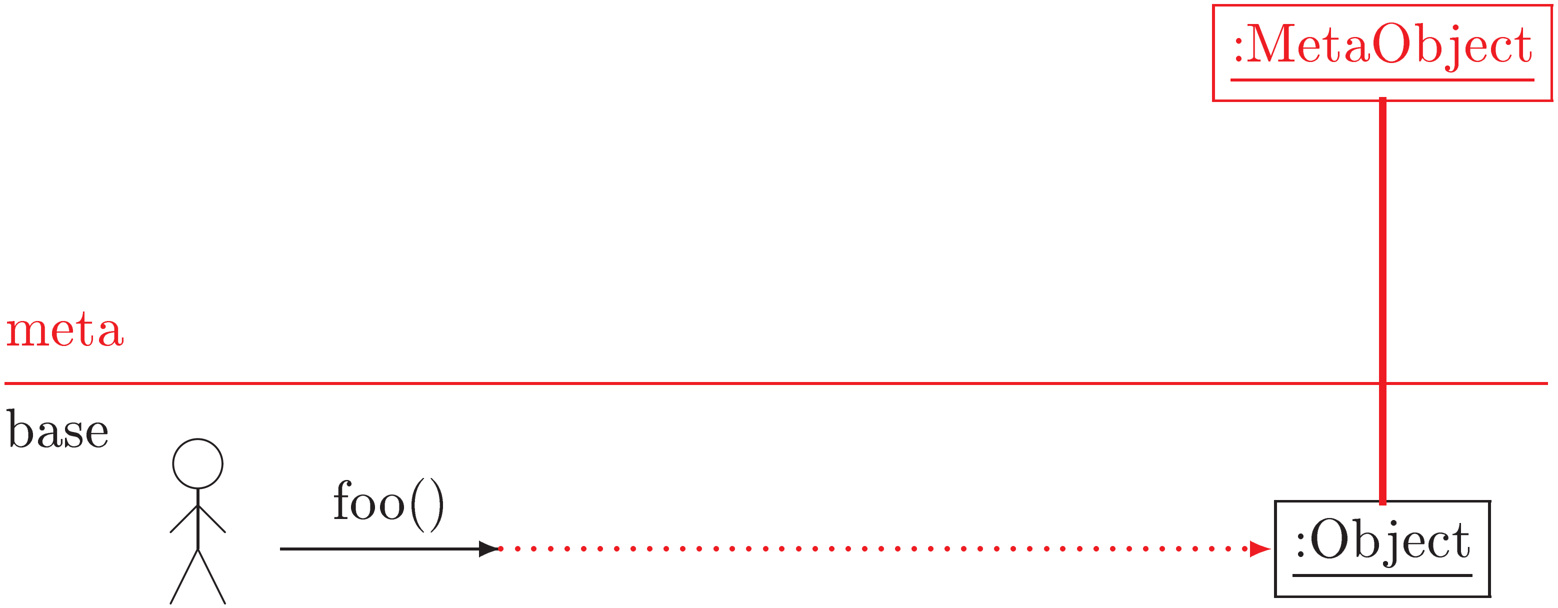
Pré-processadores reflexivos

- Alteração do código em tempo de compilação
- Alteração da linguagem
- Introdução de “ganchos”
  - Onde?
- Eficiência × Flexibilidade
- Decisões por classe
- Exemplos: Open C++, OpenJava, AspectJ

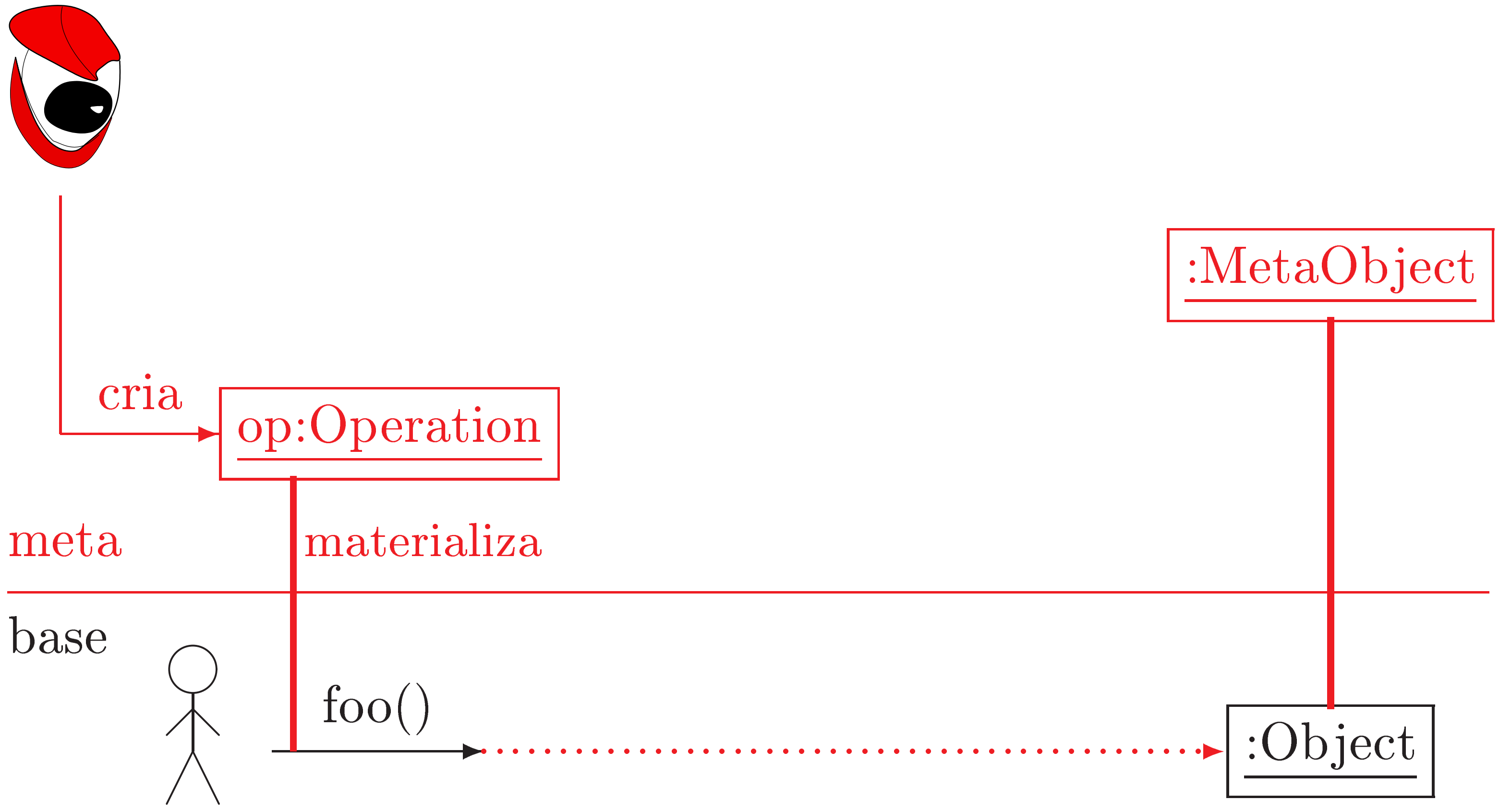
Interceptadores

- Tempo de execução
- Associações dinâmicas com objetos
- Compilador/Interpretador modificado
- Exemplos: Aperios, ABCL/R2, CodA, Moostrap, MetaXa, Dalang, CORBA *Portable Interceptors*, **Guaraná**

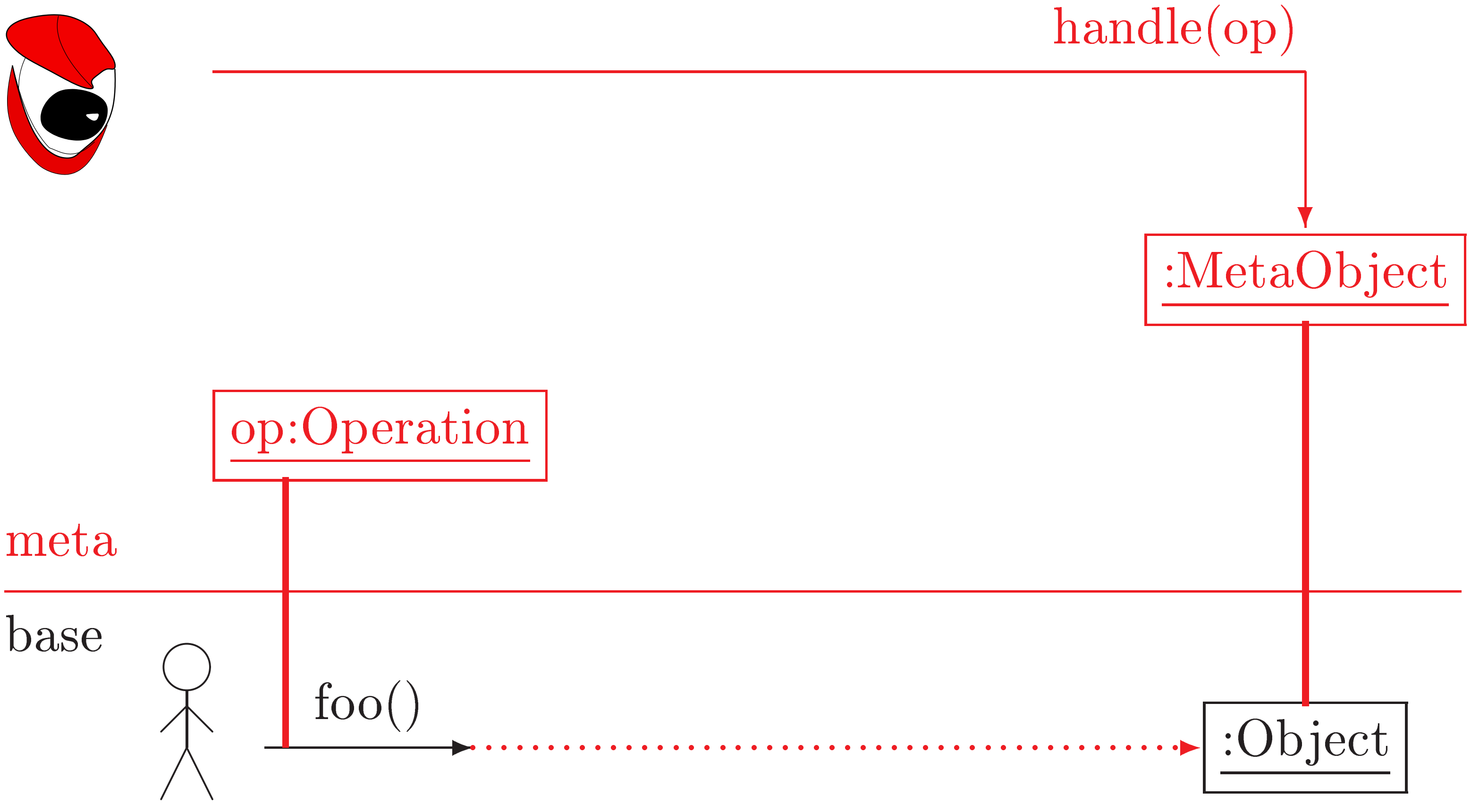
Invocação do método `foo()`



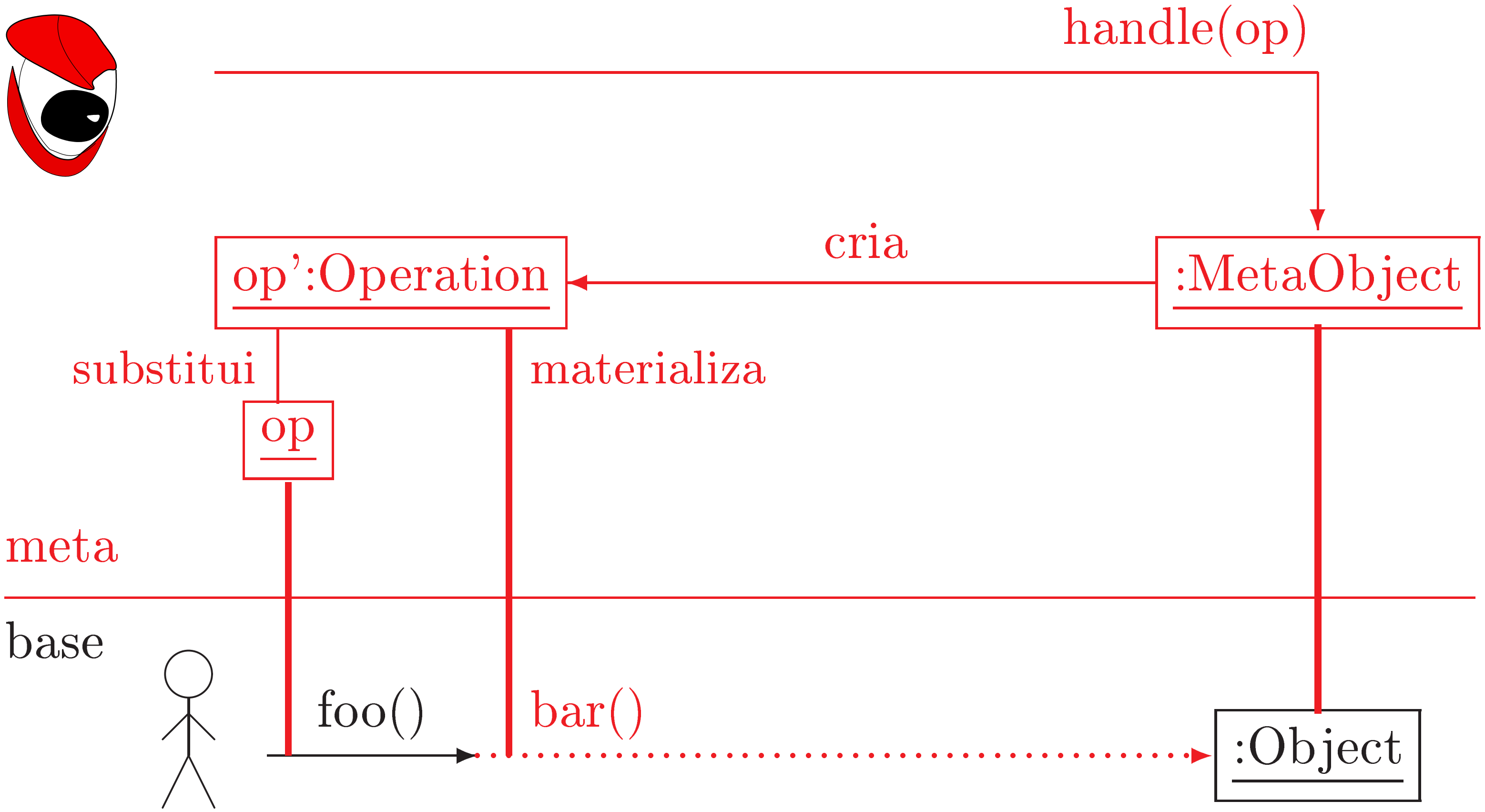
Materialização da operação



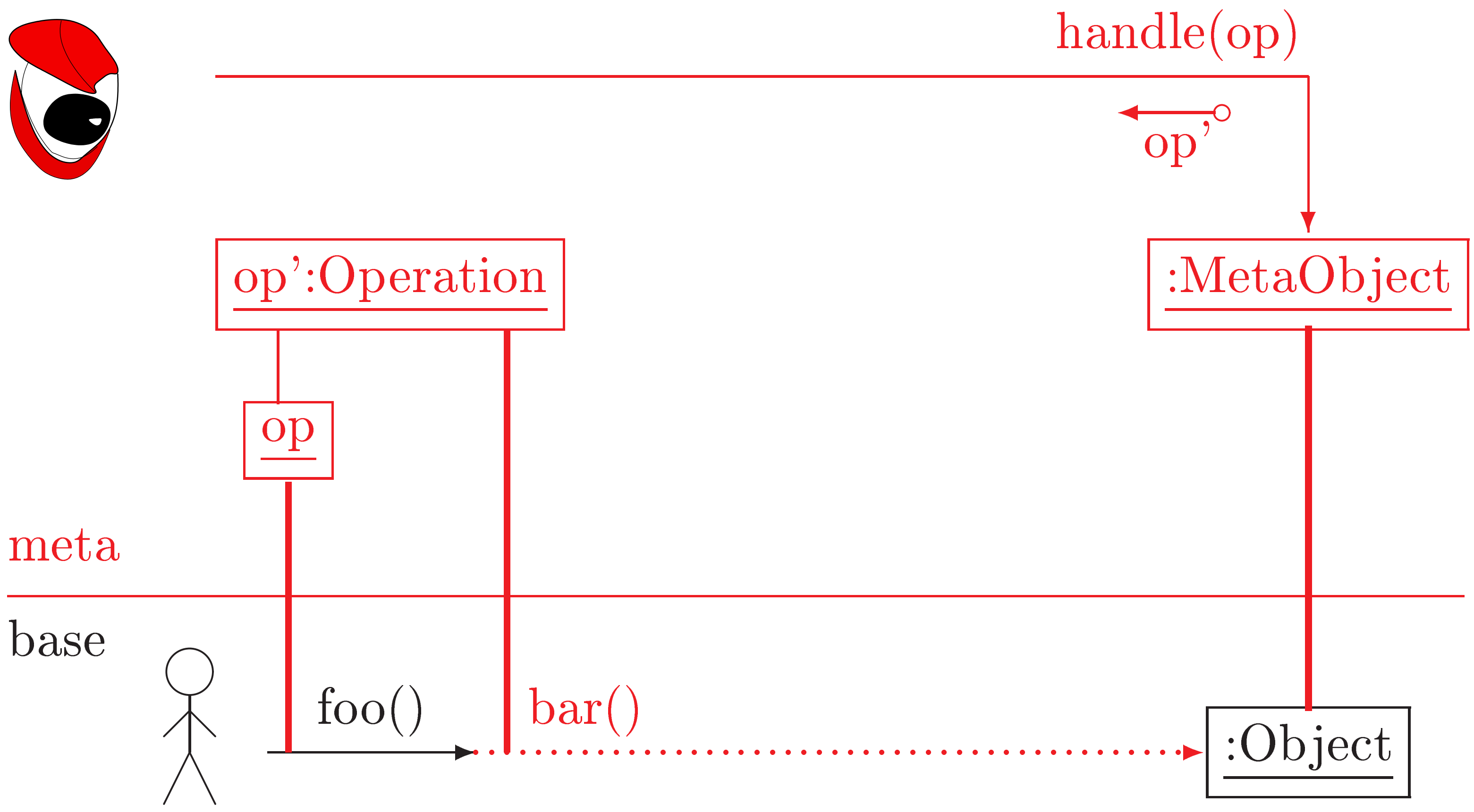
Apresentação para o meta-objeto



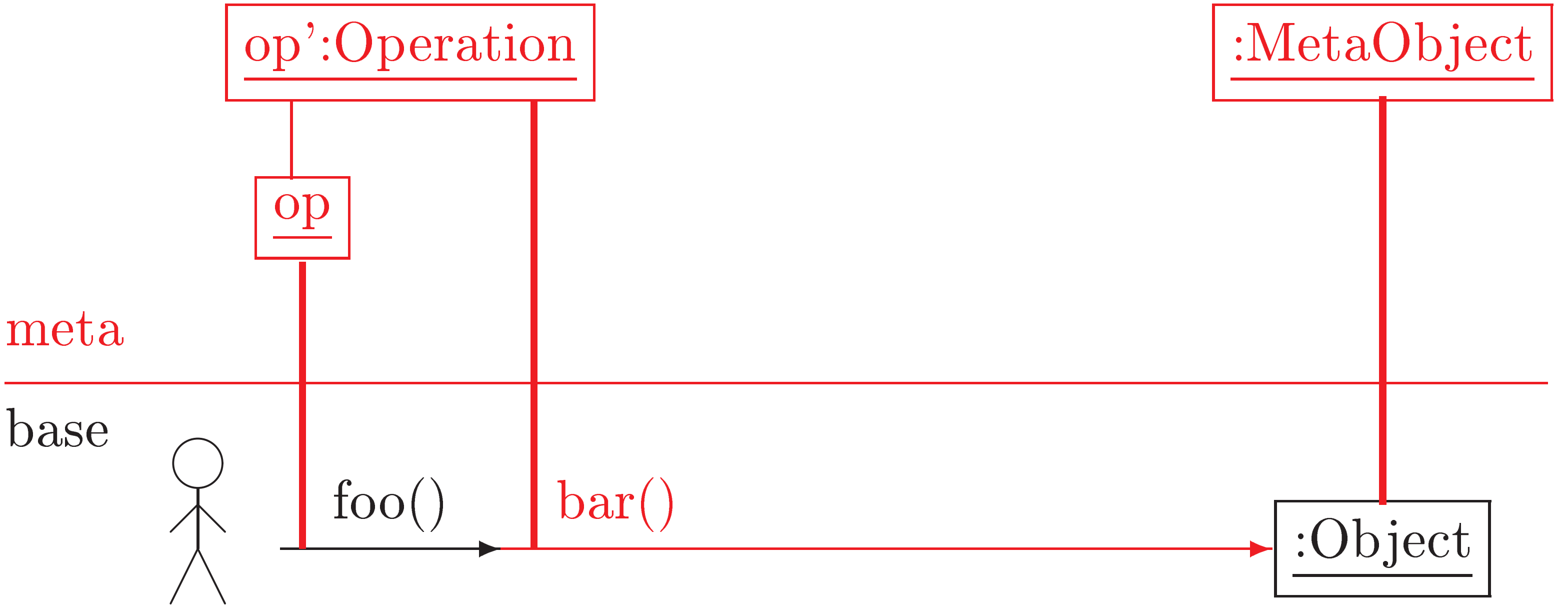
Criação de uma nova operação



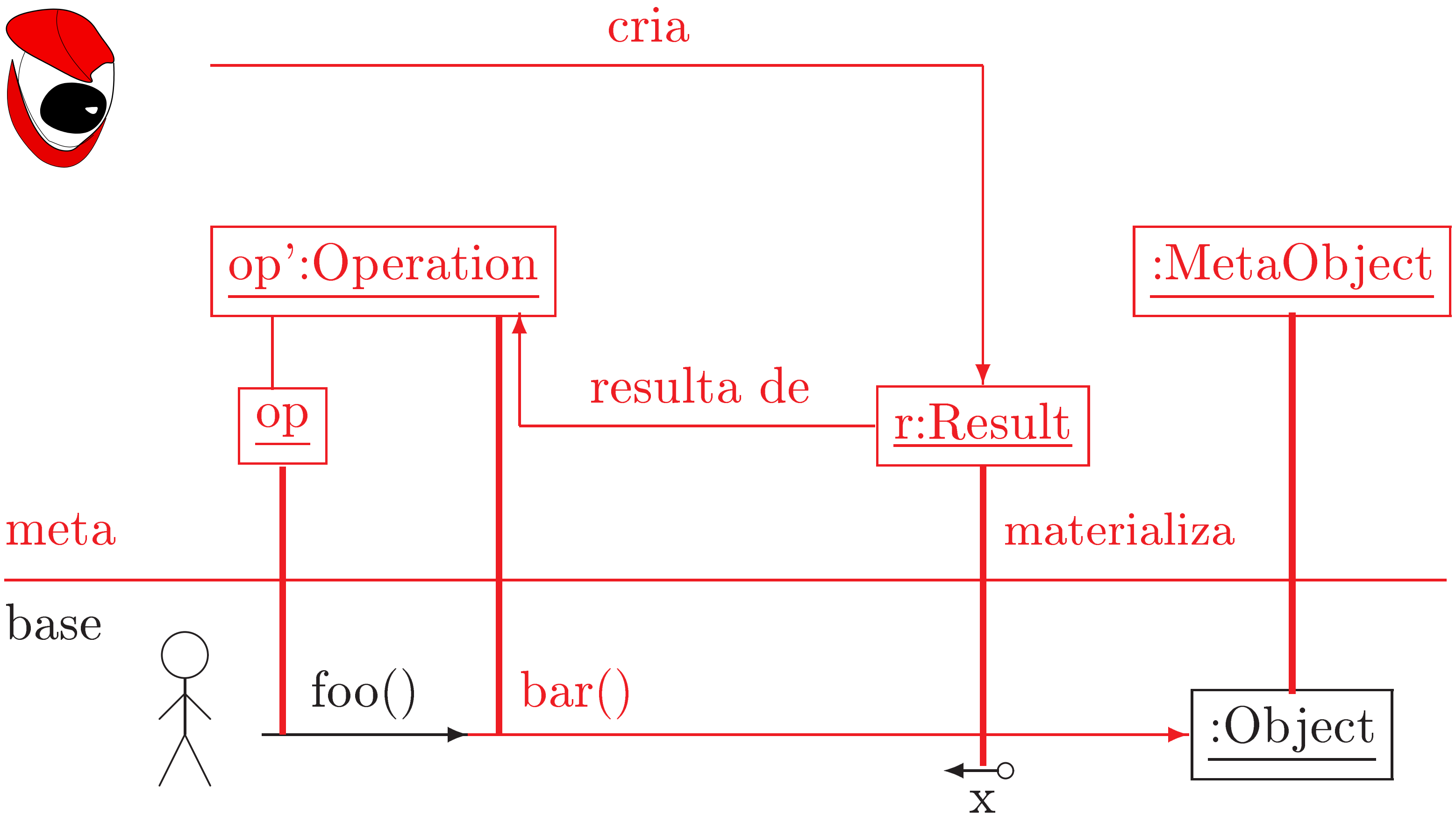
Pedido de substituição da operação



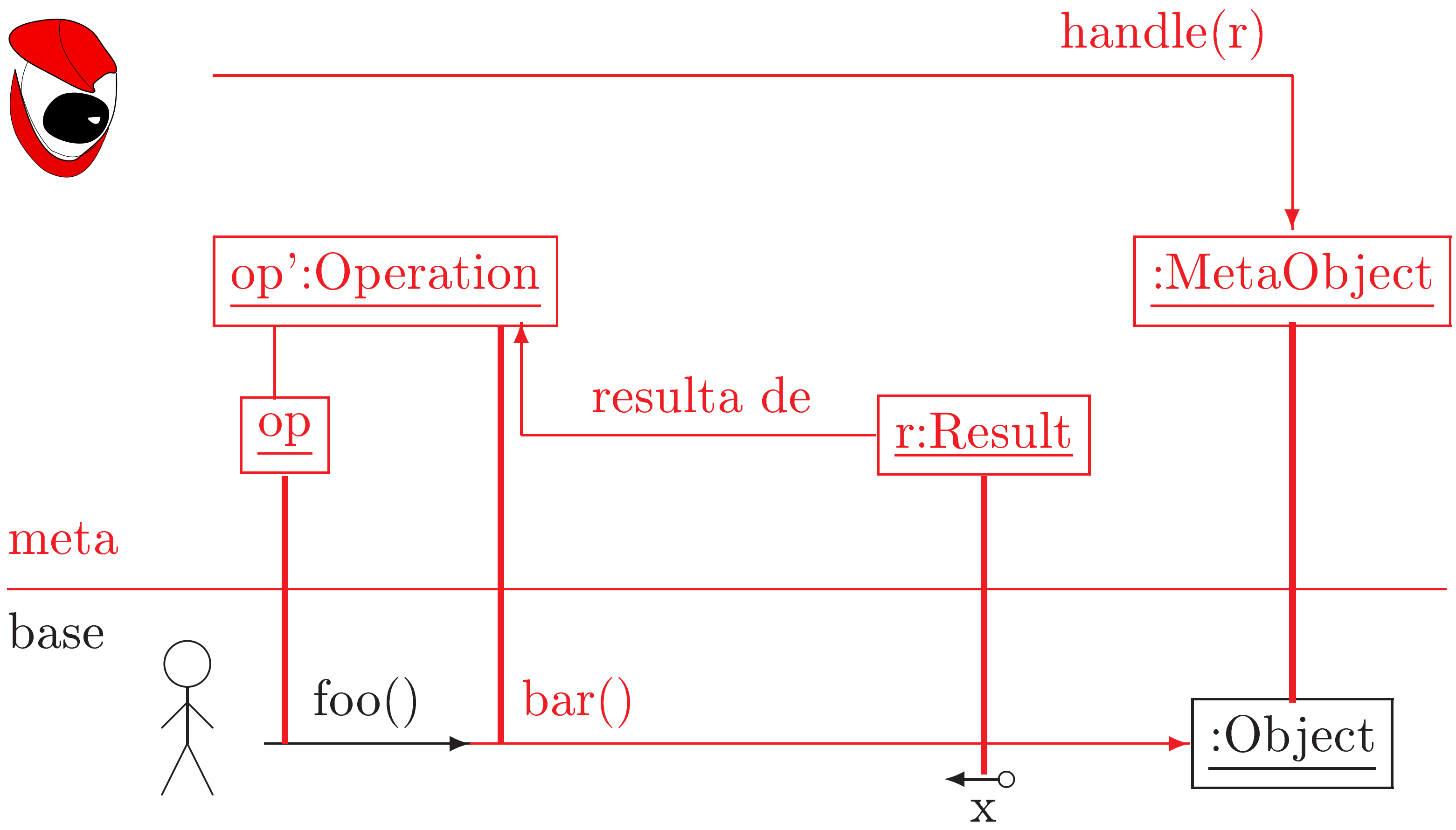
Execução da operação



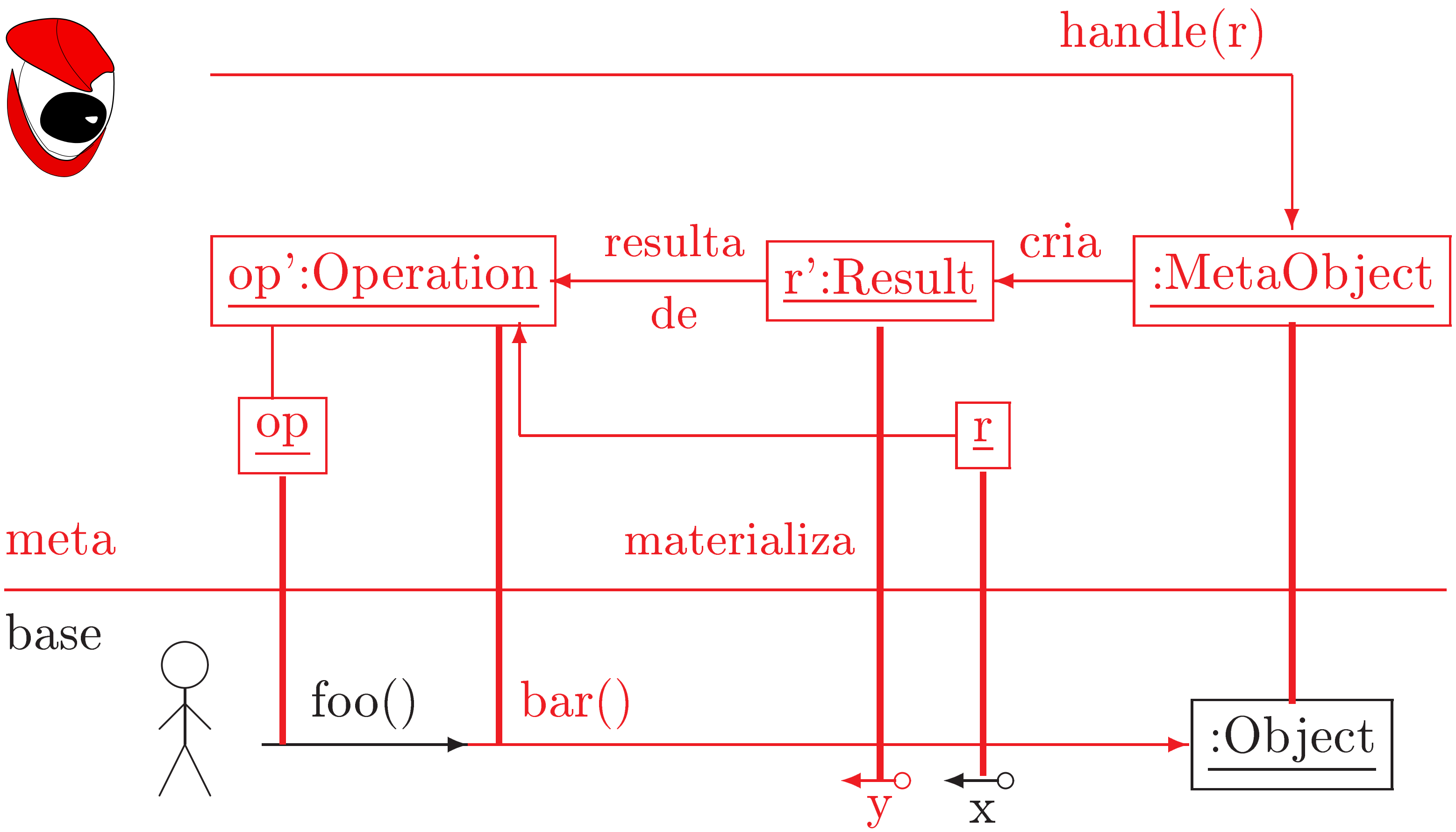
Materialização do resultado



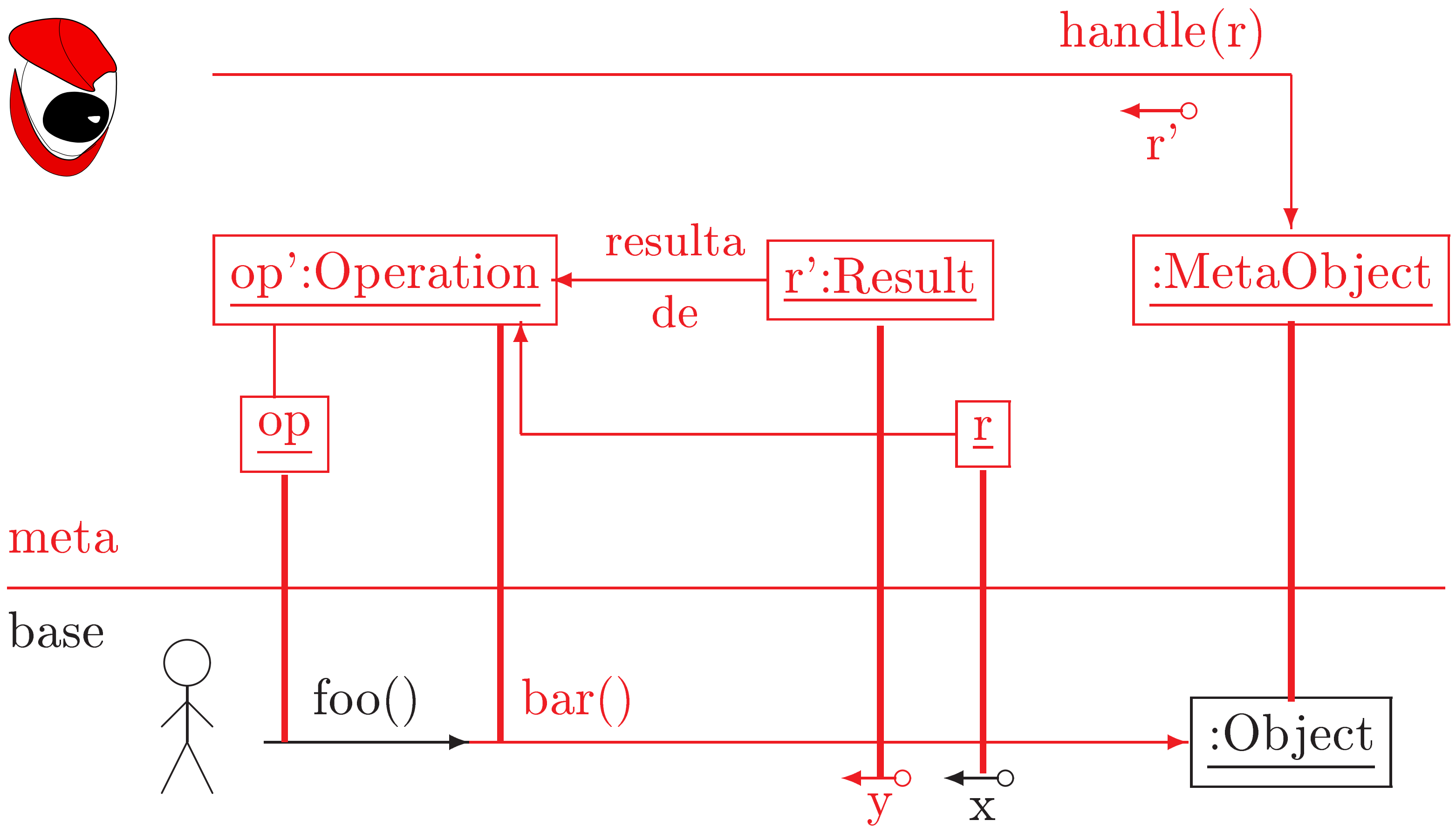
Apresentação do resultado



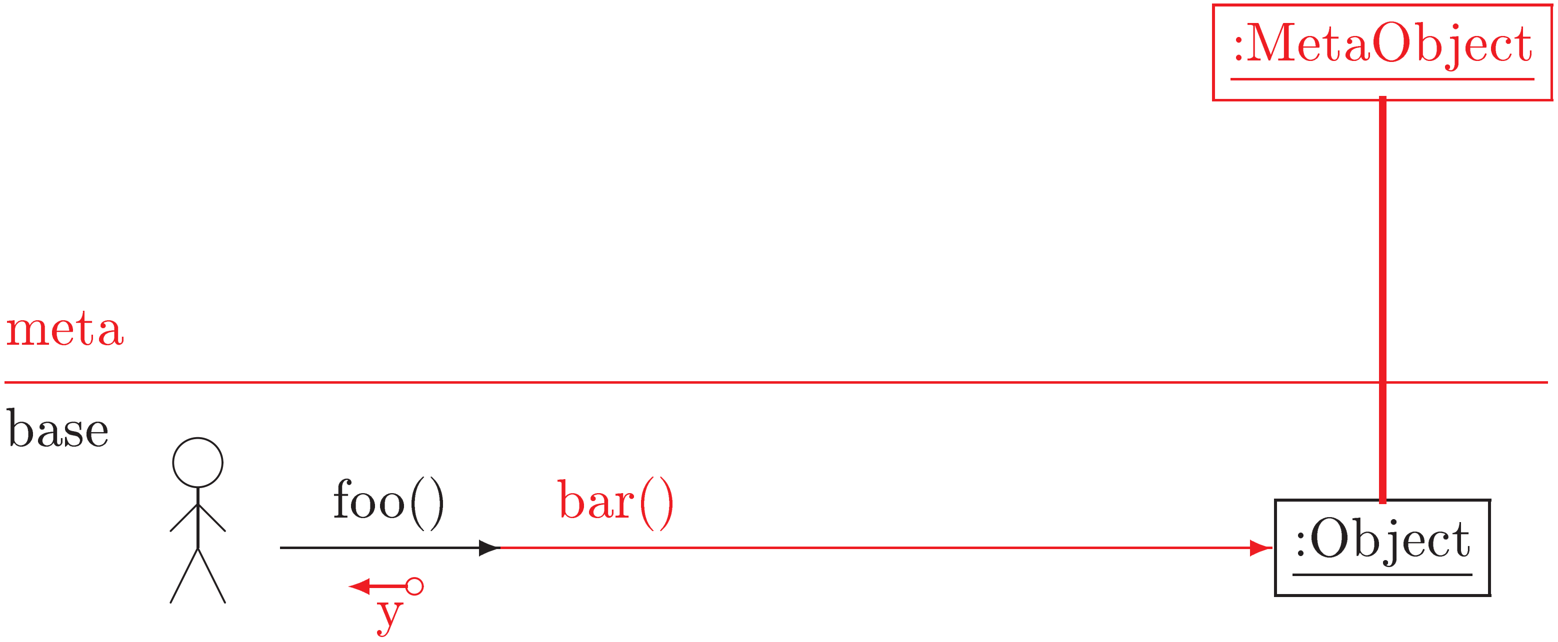
Criação de um novo resultado



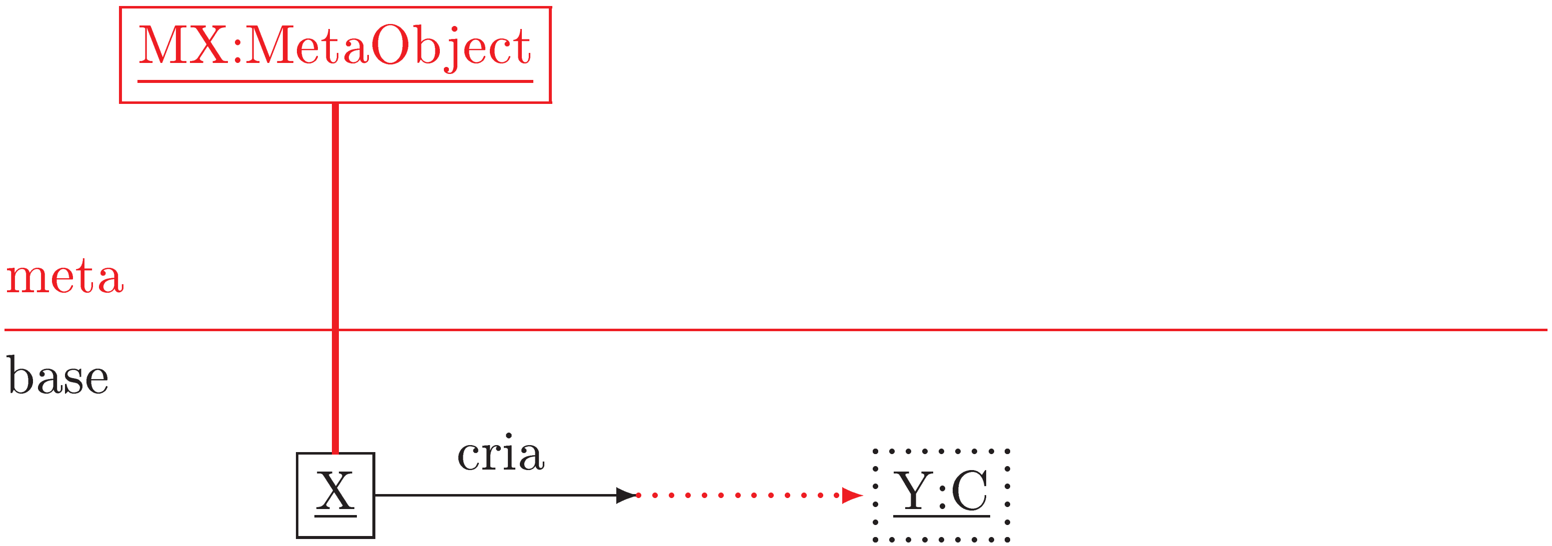
Substituição do resultado



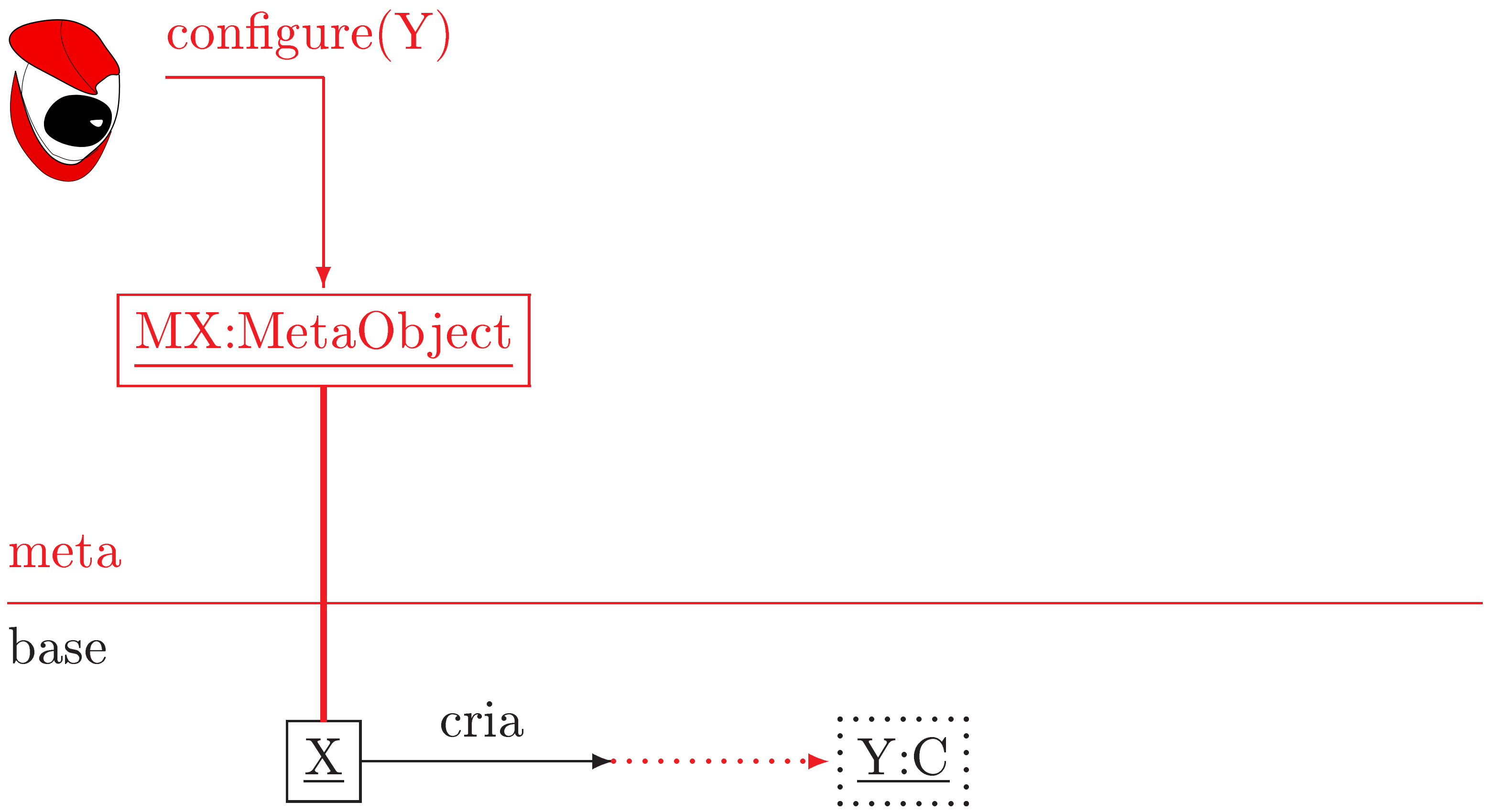
Retorno do resultado substituído



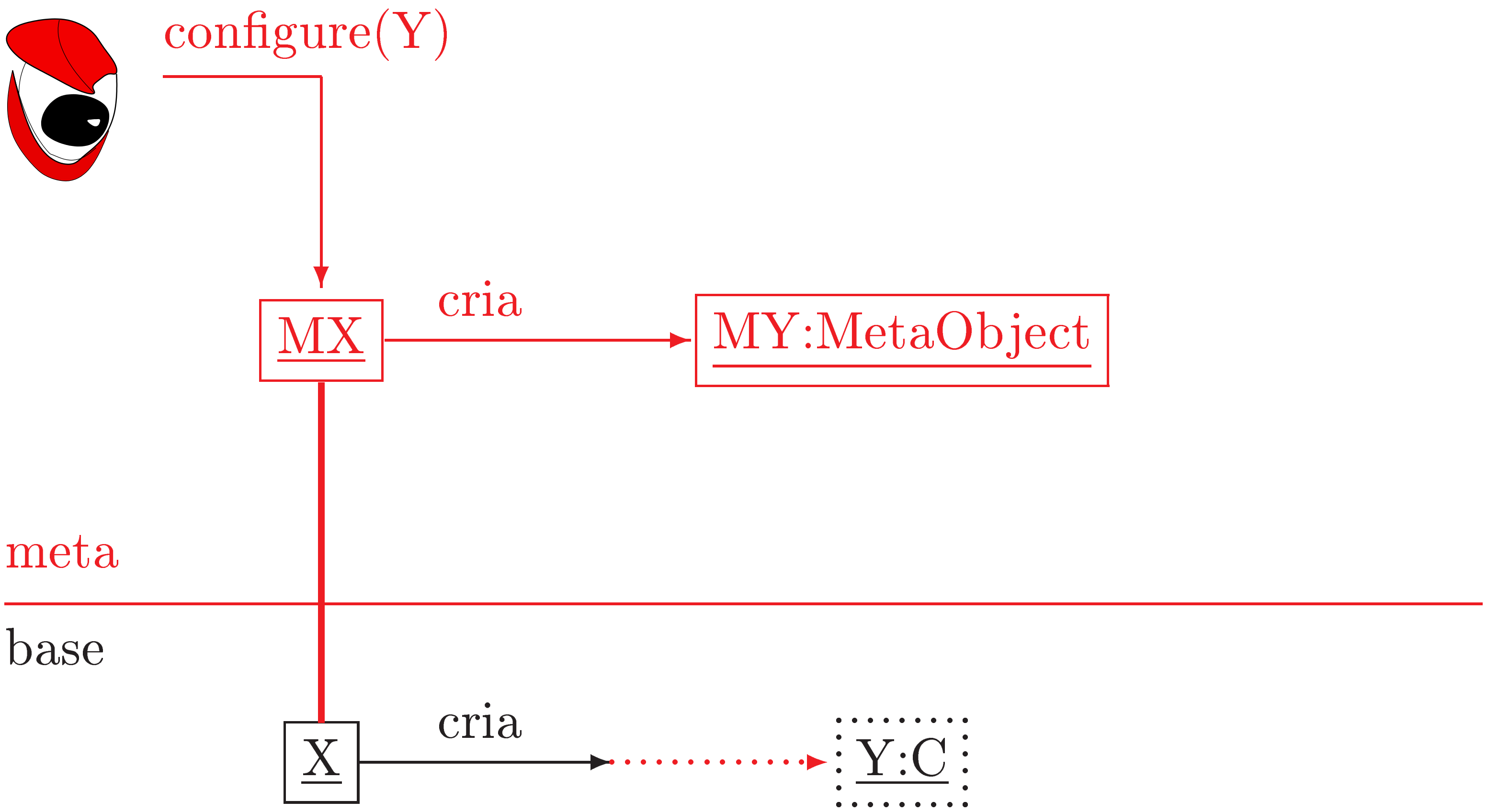
Criação de um novo objeto



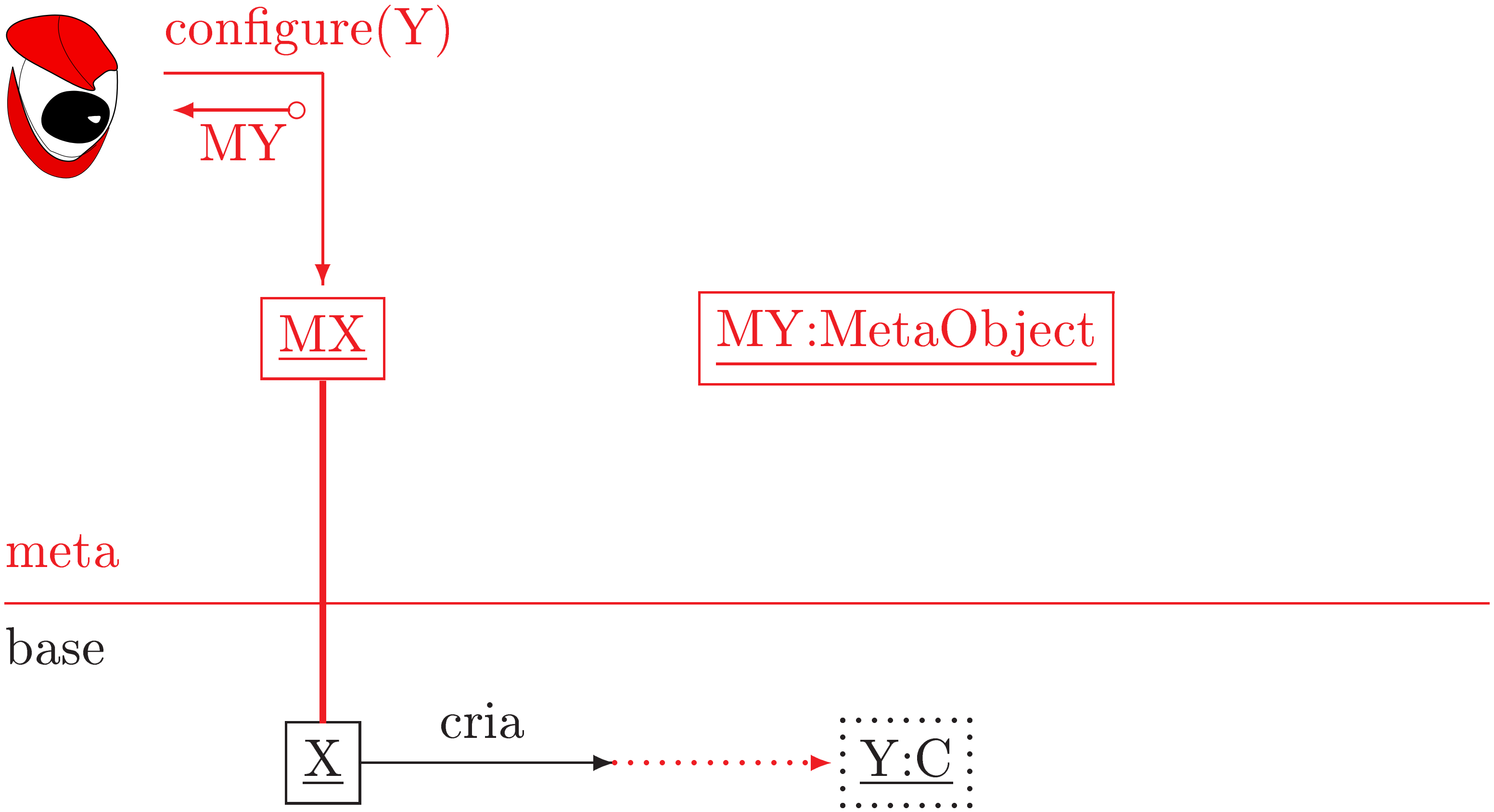
Pedido de configuração



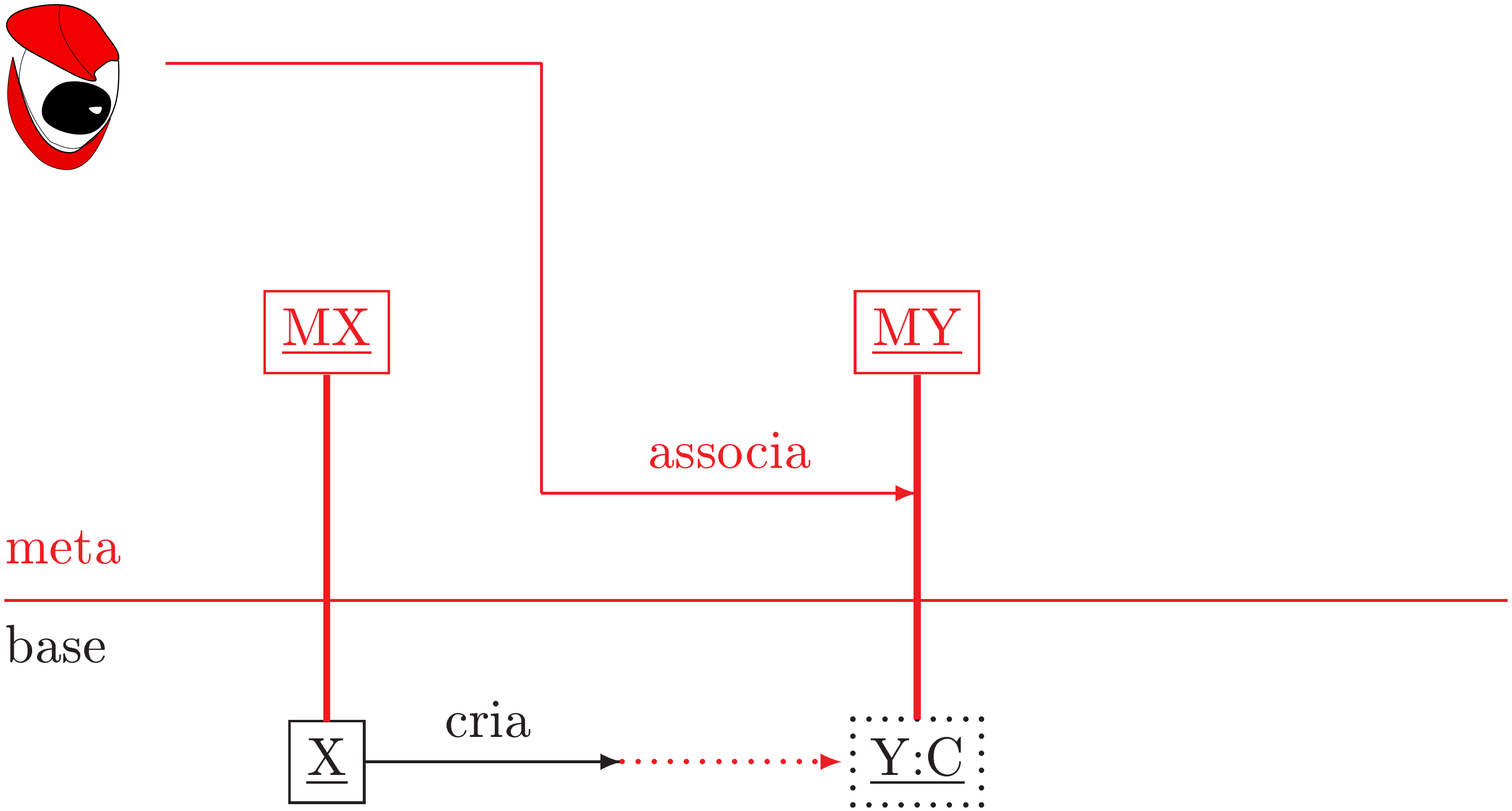
Criação de MY



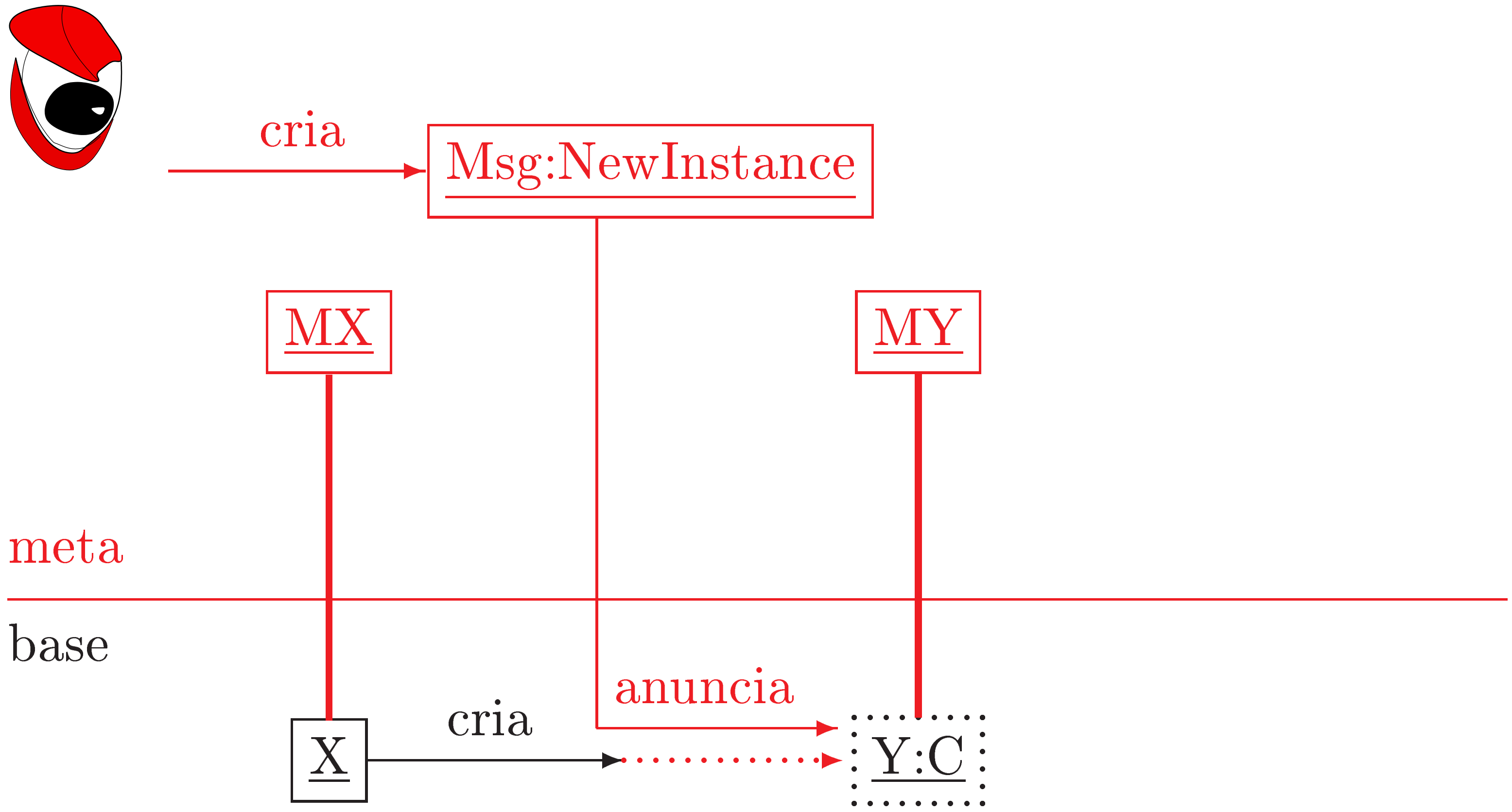
Apresentação de MY para o núcleo



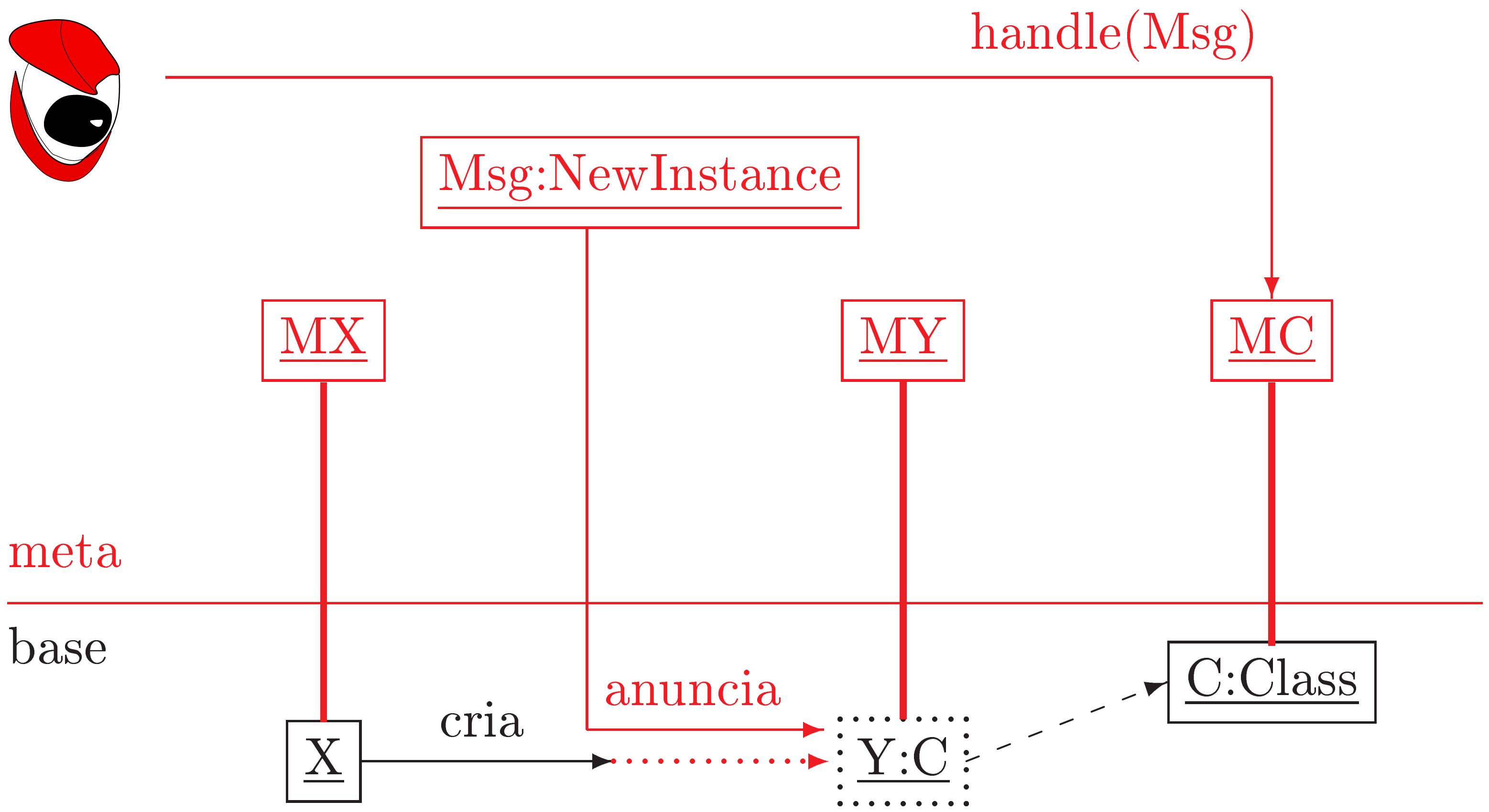
Estabelecimento da associação Y — MY



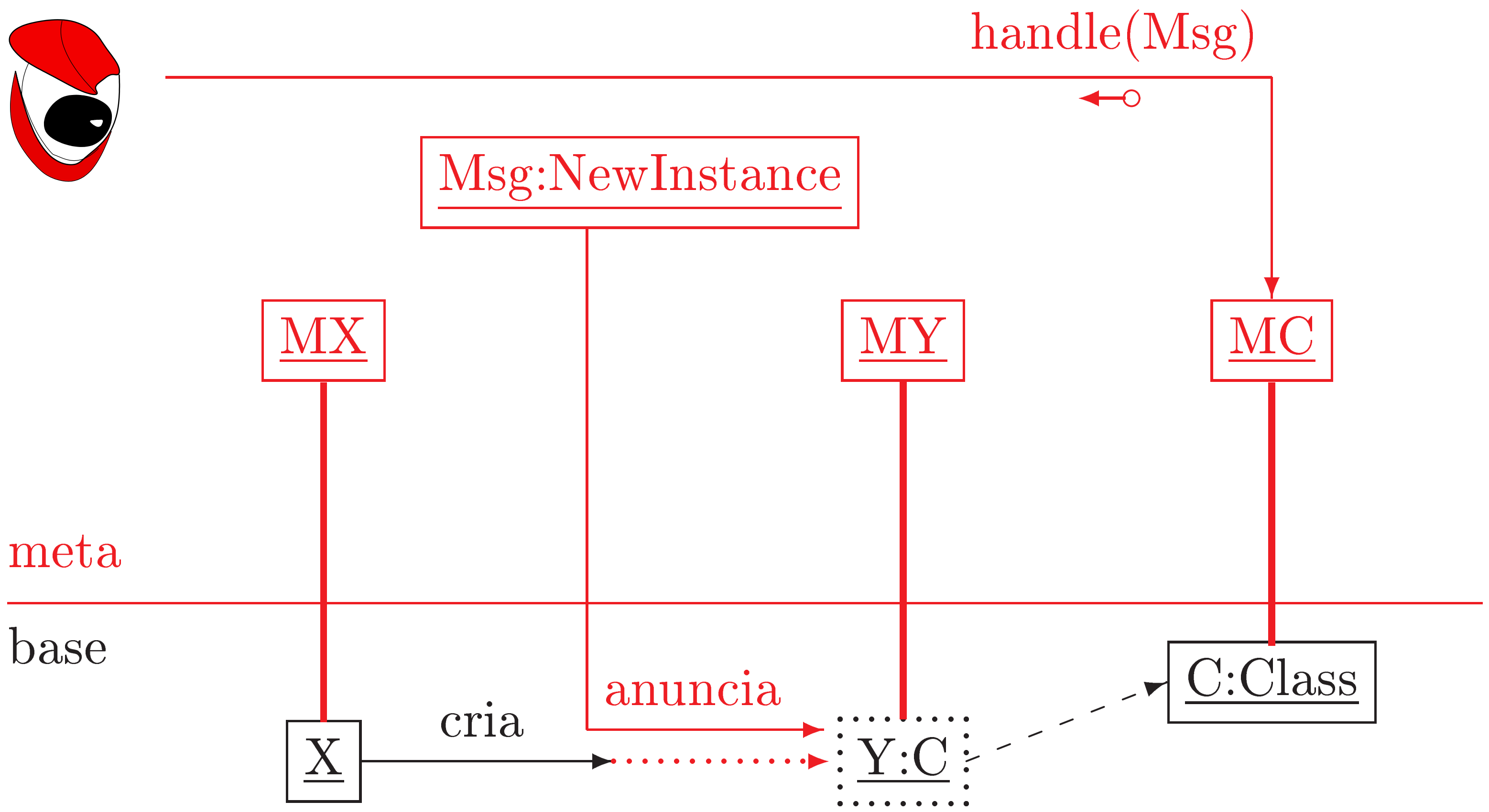
Criação de uma mensagem NewInstance



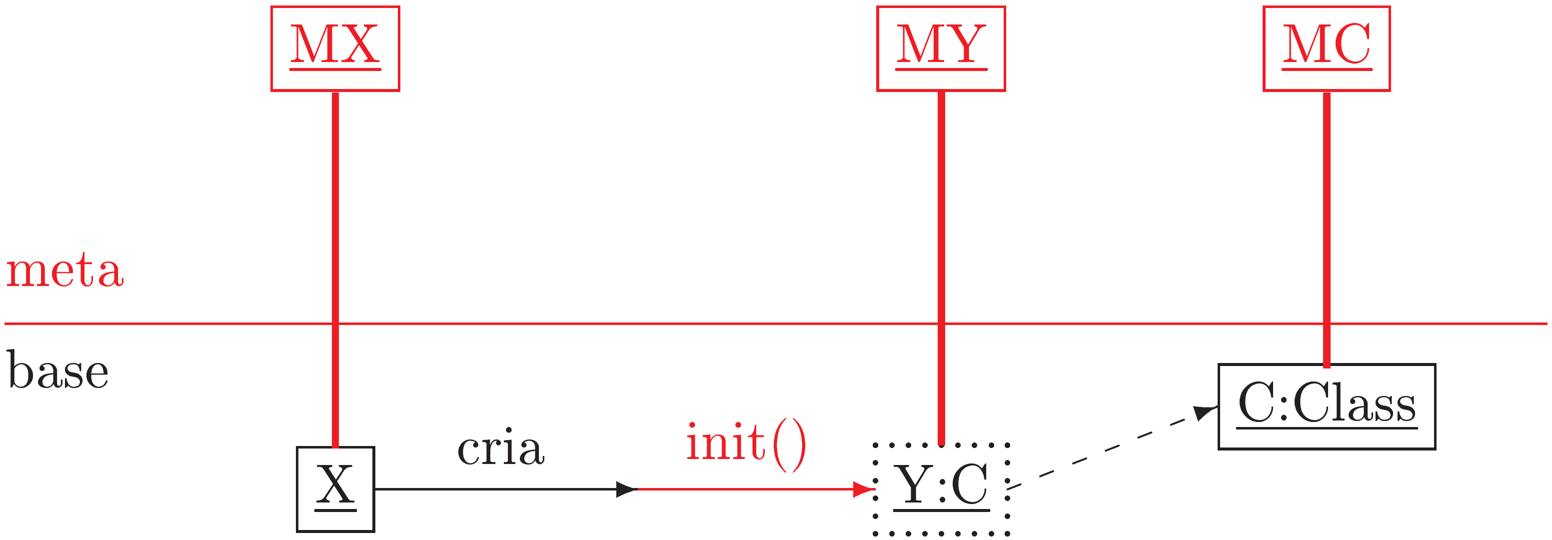
Apresentação para o meta-objeto da classe



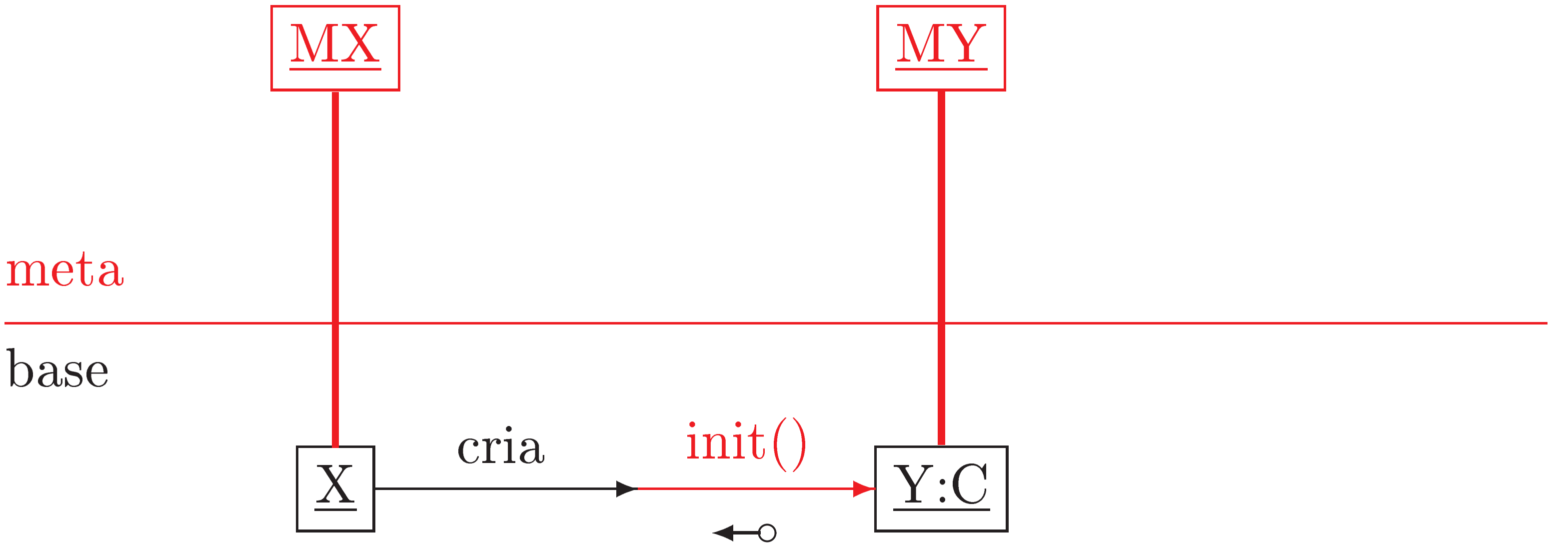
Fim do tratamento da mensagem



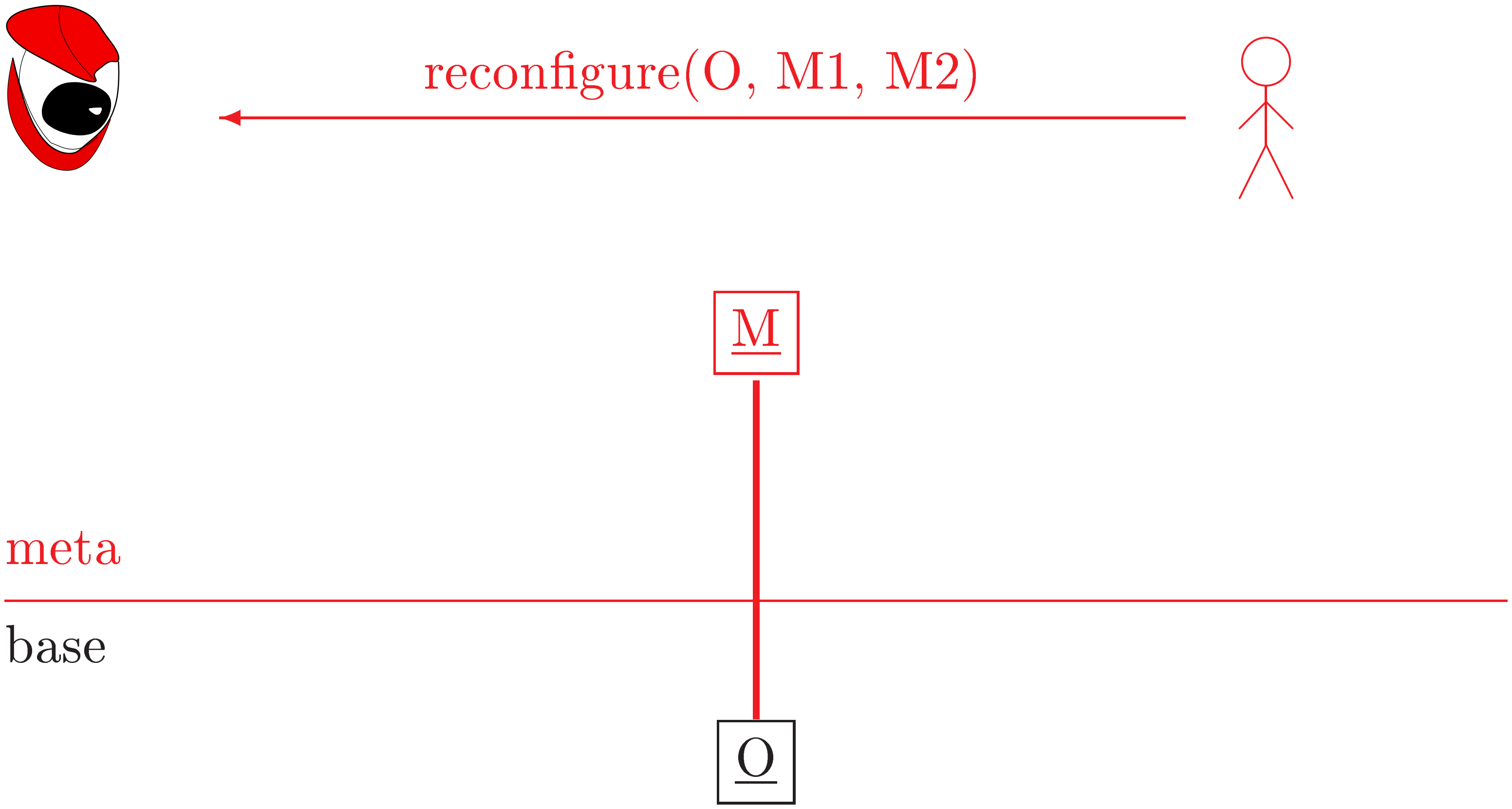
Chamada do construtor



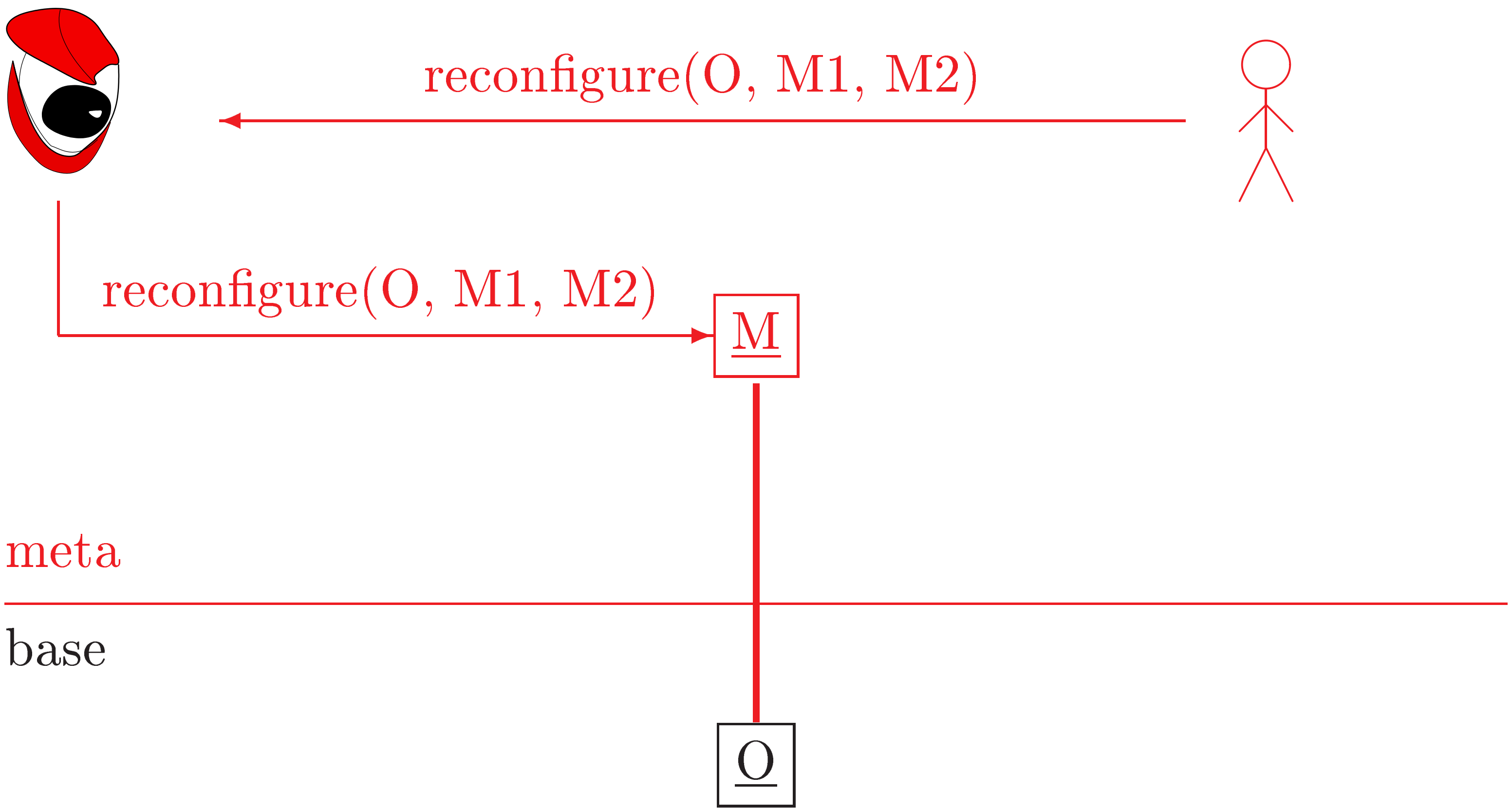
Criação do objeto concluída



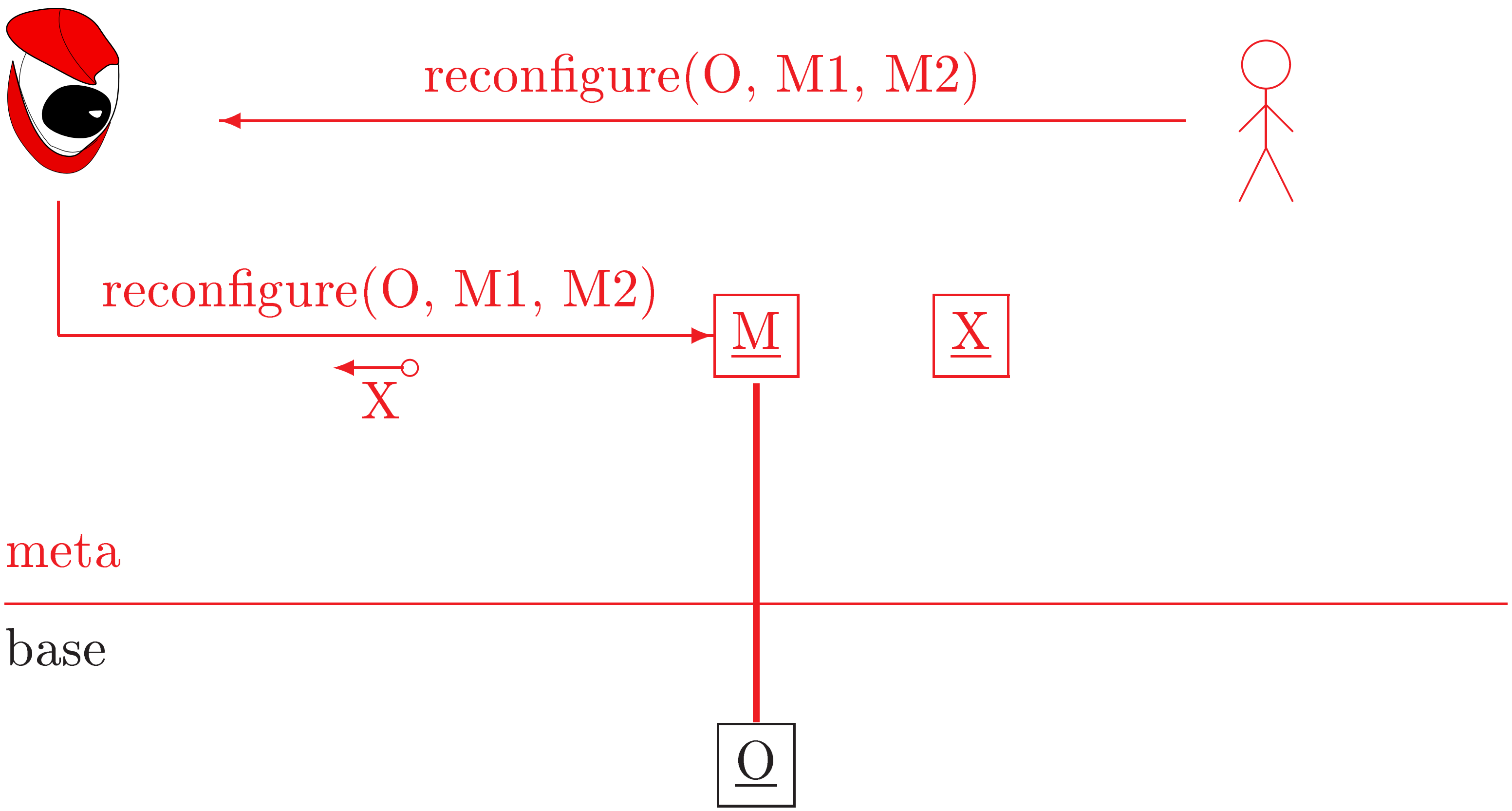
Reconfiguração



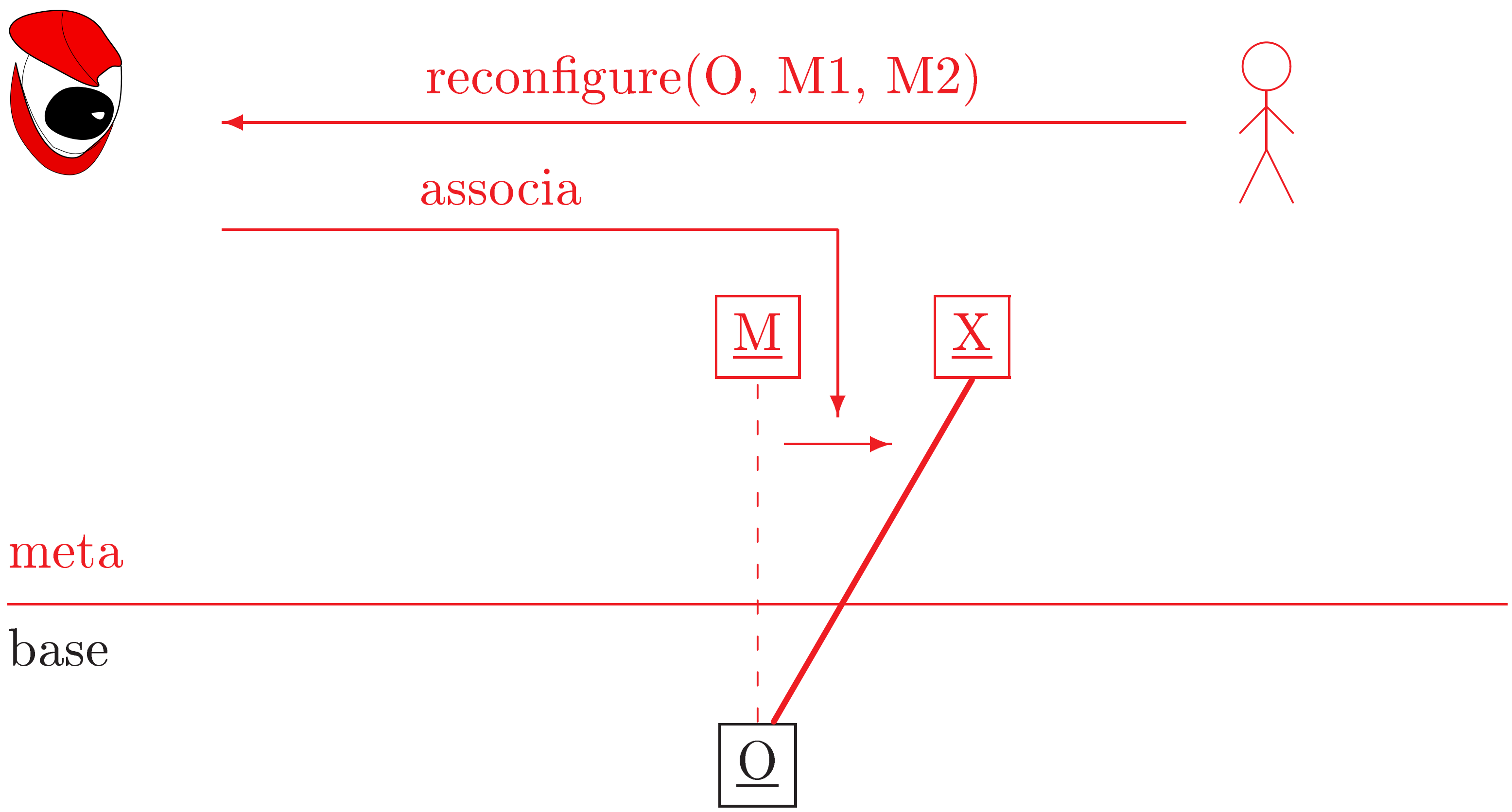
Apresentação do pedido para o meta-objeto



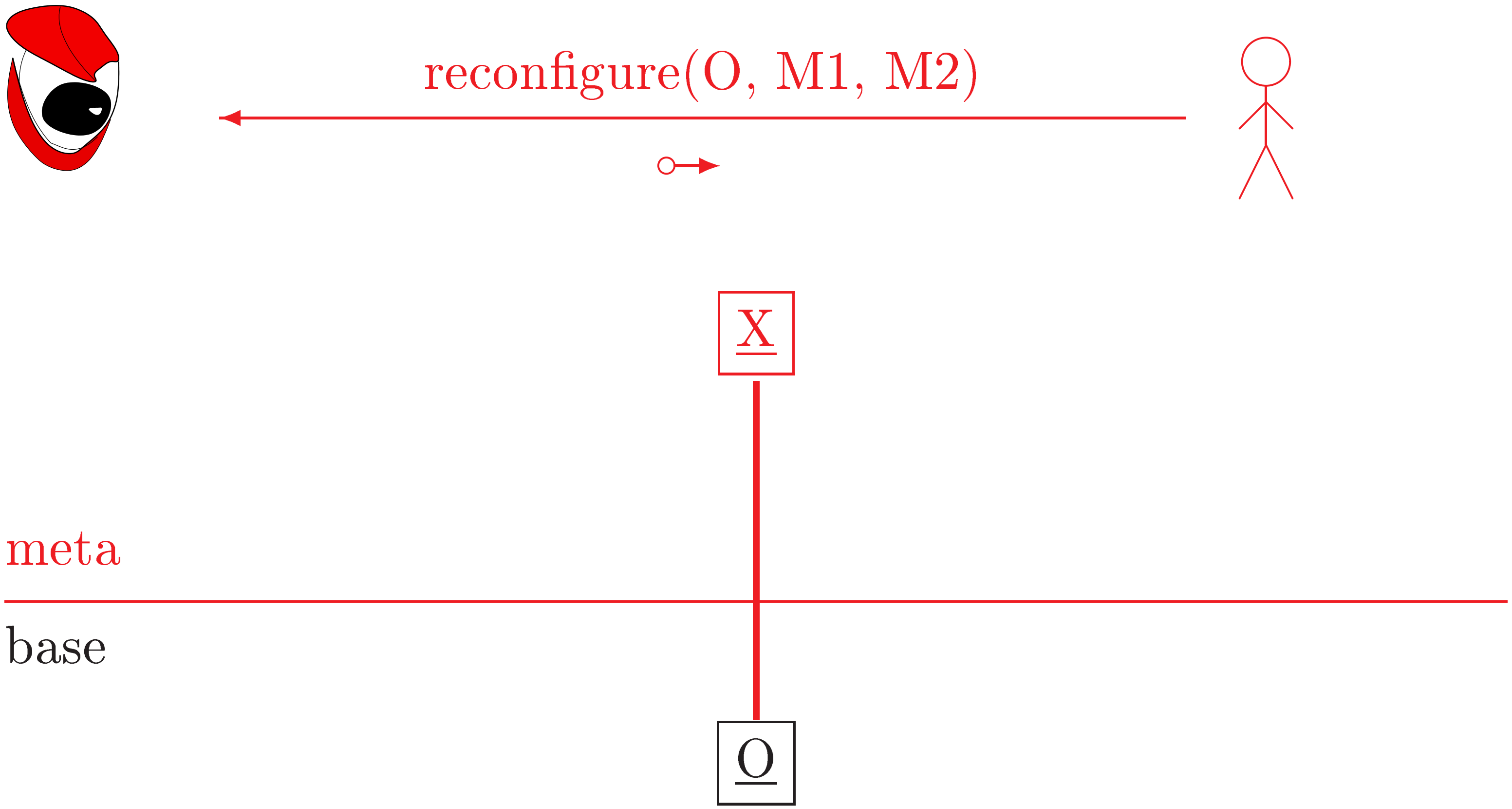
Um meta-objeto é retornado



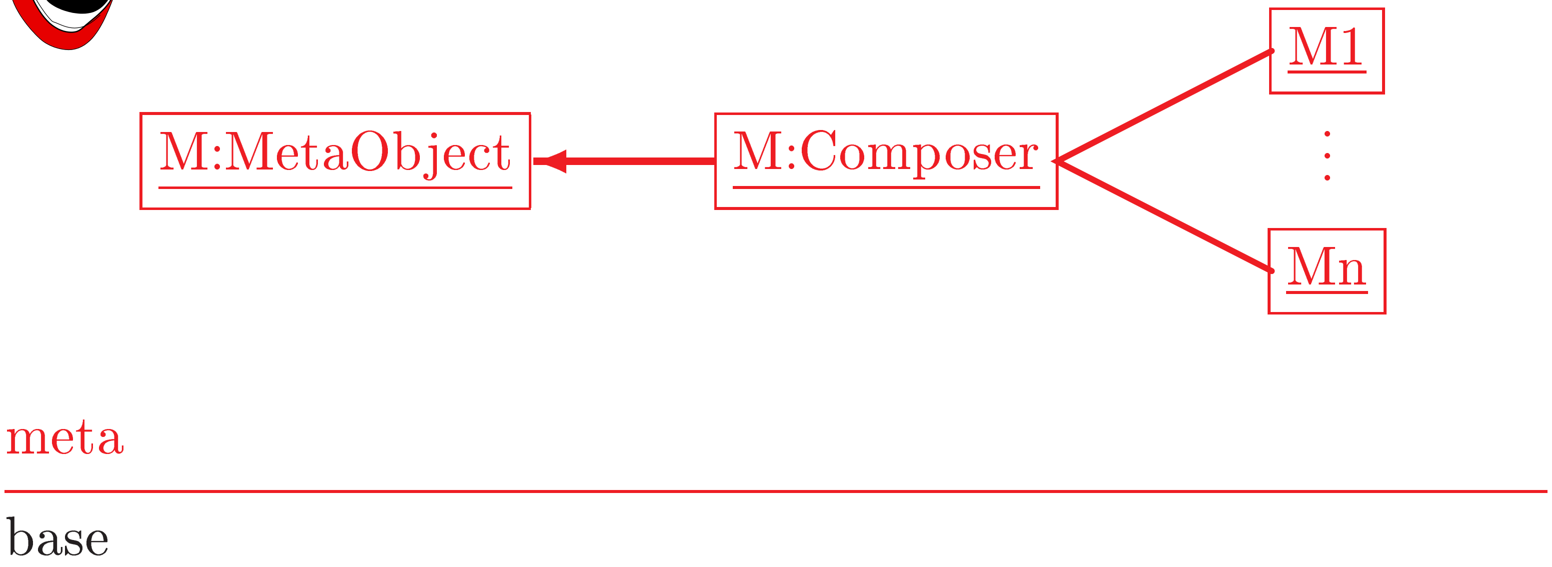
A reconfiguração é executada



Objeto reconfigurado



Composer



Guaraná: Sumário

- Operações e resultados
  - Invocação de métodos e construtores
  - Leitura e alteração de campos e elementos de arrays
  - Entrada e saída de blocos e métodos `synchronized`
  - Criação de operações
- Configuração de novos objetos
- Reconfiguração
- Mensagens
- *Proxies*

Exemplo

- Invocação de `foo.main(argv)`
- Leitura de `argv.length`
- `foo.class` cria `bar1 = new bar()`
- Invocação de `bar.<init>(0)`
- Atribuição `bar.i = 0`
- `bar.<init>` retorna
- `foo.main` retorna

Código do exemplo

```
public class foo {
   public static void main(String[] argv) {
      new bar(argv.length);
   }
}

class bar {
   int i;
   public bar(final int arg) {
      i = arg;
   }
}
```

Outras aplicações

- Teste
- Persistência
- Replicação
- Invocação remota
- Migração
- Monitoração
- Controle de concorrência

Meta-Object Library for Distributed Systems

- Transparência de localidade
- Migração e replicação automáticas
- Persistência por alcance
- Semântica transacional
- Intenso uso de composição
- Extensibilidade
- Contribuições são bem-vindas!

Presente

- Kaffe OpenVM 1.0.4
  - Interpretador e compilador JIT

Futuro

- Kaffe OpenVM 1.0.6
  - Novo compilador JIT
  - Otimização de locks
- *GNU Compiler for Java*
  - Compilação prévia
  - Integração com GDB e Kaffe OpenVM
- GNU C++?
- **MOLDS**

Mais informações

- `http://www.ic.unicamp.br/~oliva/guarana/`
- `mailto:guarana@egroups.com` (em inglês)
- `oliva@lsd.ic.unicamp.br` ou `aoliva@redhat.com`

?