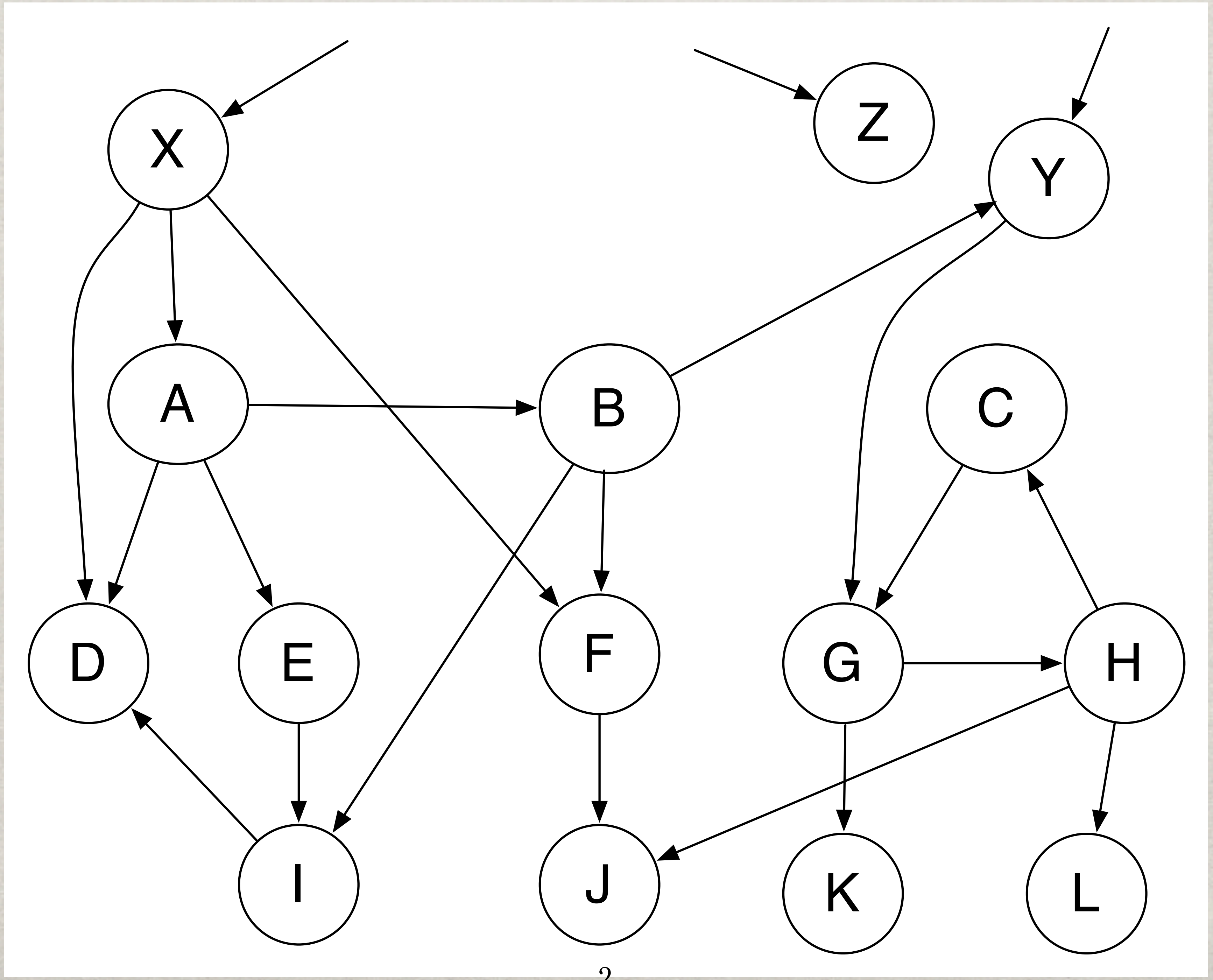
Problems and Challenges when Building a Manager for Unused Objects

Mariano Martinez Peck
marianopeck@gmail.com
http://marianopeck.wordpress.com/

The context

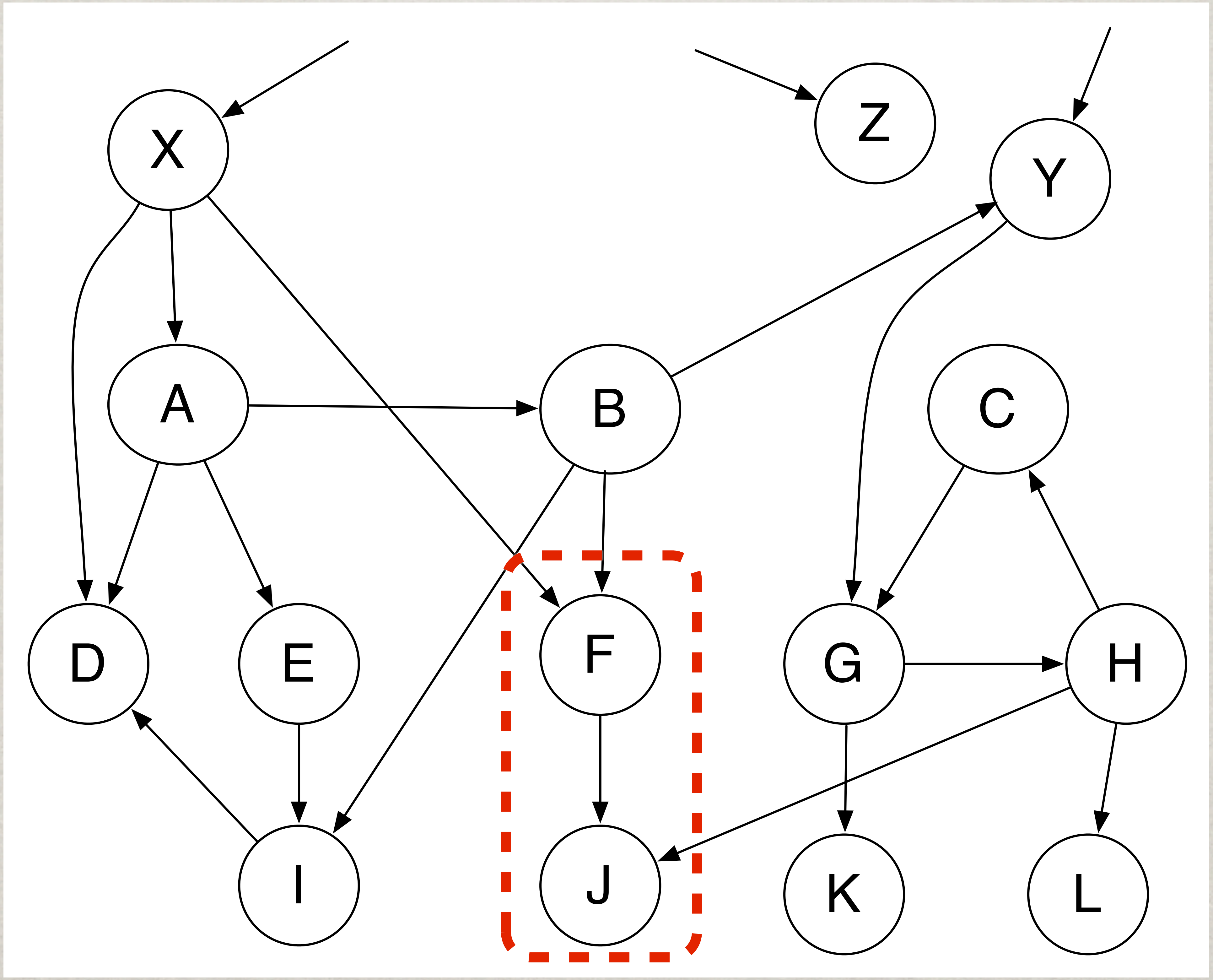
In OOP primary memory is represented by an object graph



What is an used object?

An object that receives a message or that is directly used by the virtual machine during a specific period of time.

The Garbage Collectors only collects objects that nobody else points to. But…what happens with referenced yet unused objects?



Our proposal

Build an Unused Object Manager (UOM)

The paper...

- Describes problems we have found so far.
- Lists "non-working" alternatives.
- Shows the first steps of our alternative.

UOM in a nutshell

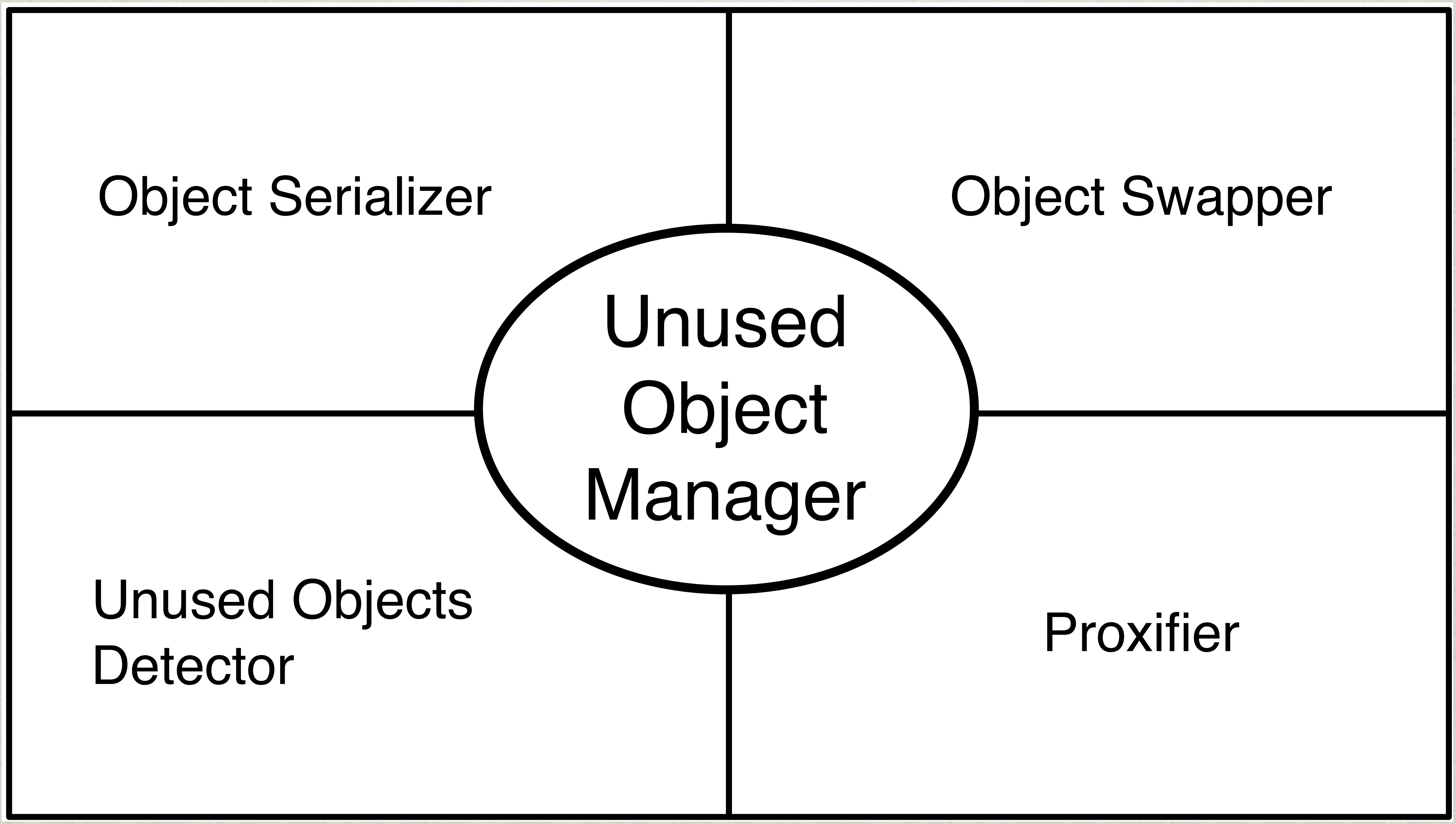
Swapping out:

1. Detect unused objects to swap out.
2. Replace some objects with proxies.
3. Serialize the original object and write it to disk.

Swapping in:

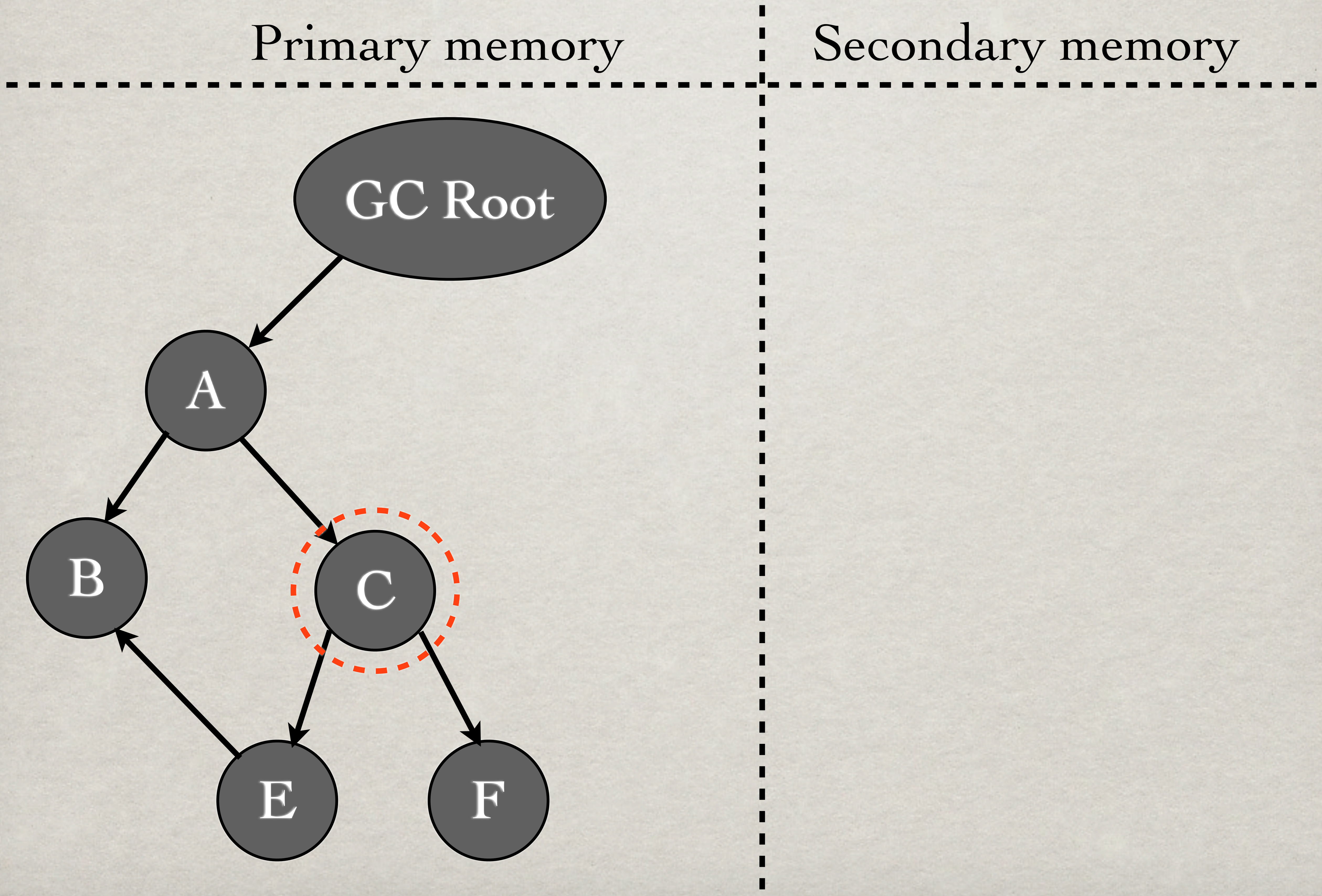
1. When a proxy intercepts a message, materialize object from disk.
2. Replace proxies with loaded objects.

UOM subsystems



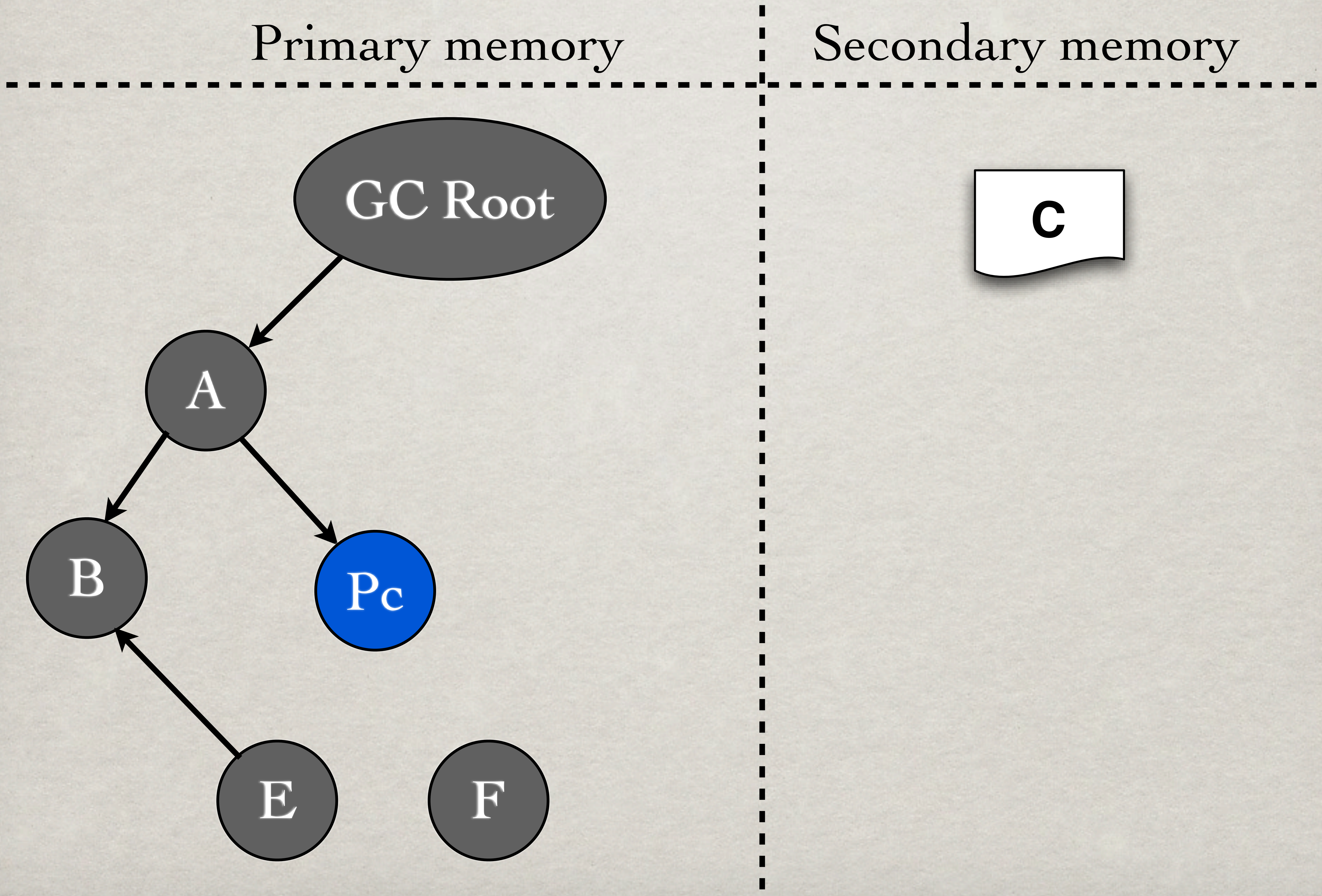
Basic Swapping Issues

Swapping unit

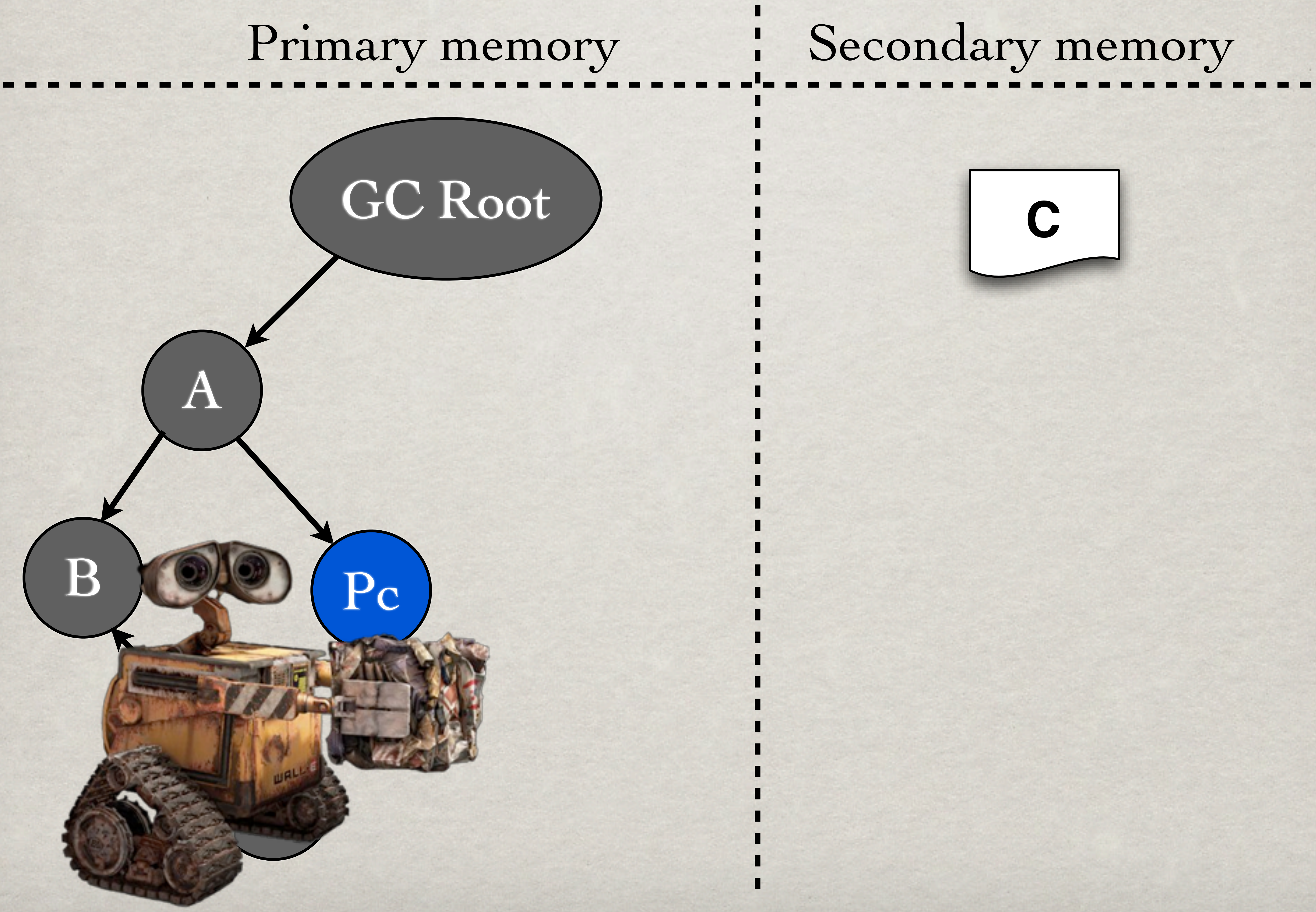


Friday, November 4, 2011

Swapping unit



SWAPPING UNIT



Lesson

To be efficient, we need to group objects and replace several objects with one or a few proxies.

Not everything can be swapped out

- Special objects: specialObjectsArray, nil, true, false.
- Special classes: ProtoObject, Object, Array, Symbol, BlockClosure, CompiledMethod, MethodDictionary, SmallInteger, etc.
- Objects/Classes needed to swap in/out.

Lesson

We need a way to tag system classes and objects that we shouldn't swap.

Proxies and Memory

Methods not intercepted 1/2

- If we use #doesNotUnderstand: , then all messages understood are not intercepted.
- Compiler optimizations: #ifTrue:, #ifNil:, #whileFalse:, #to:do:, etc (not so many at the end)

Methods not intercepted 2/2

Special bytecode: #class

```
(anObject class = User)
     ifTrue: [ self doSomething]
     ifFalse: [self doSomethingDifferent]
```

Special bytecode: #== is not a problem because there is a #become: between the proxy and the target.

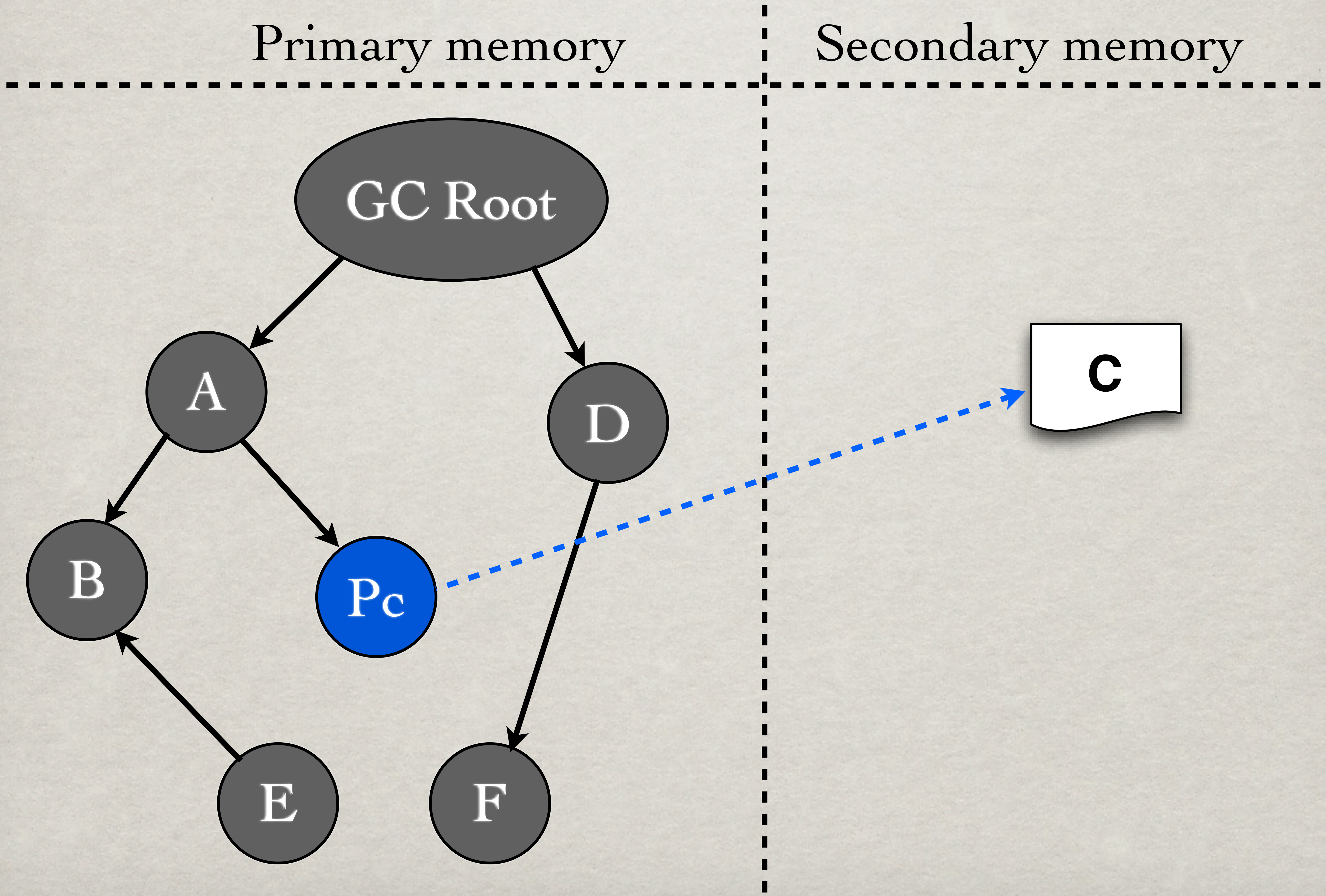
```
(anObject  == anotherObject)
     ifTrue: [ self doSomething]
     ifFalse: [self doSomethingDifferent]
```

Lessons

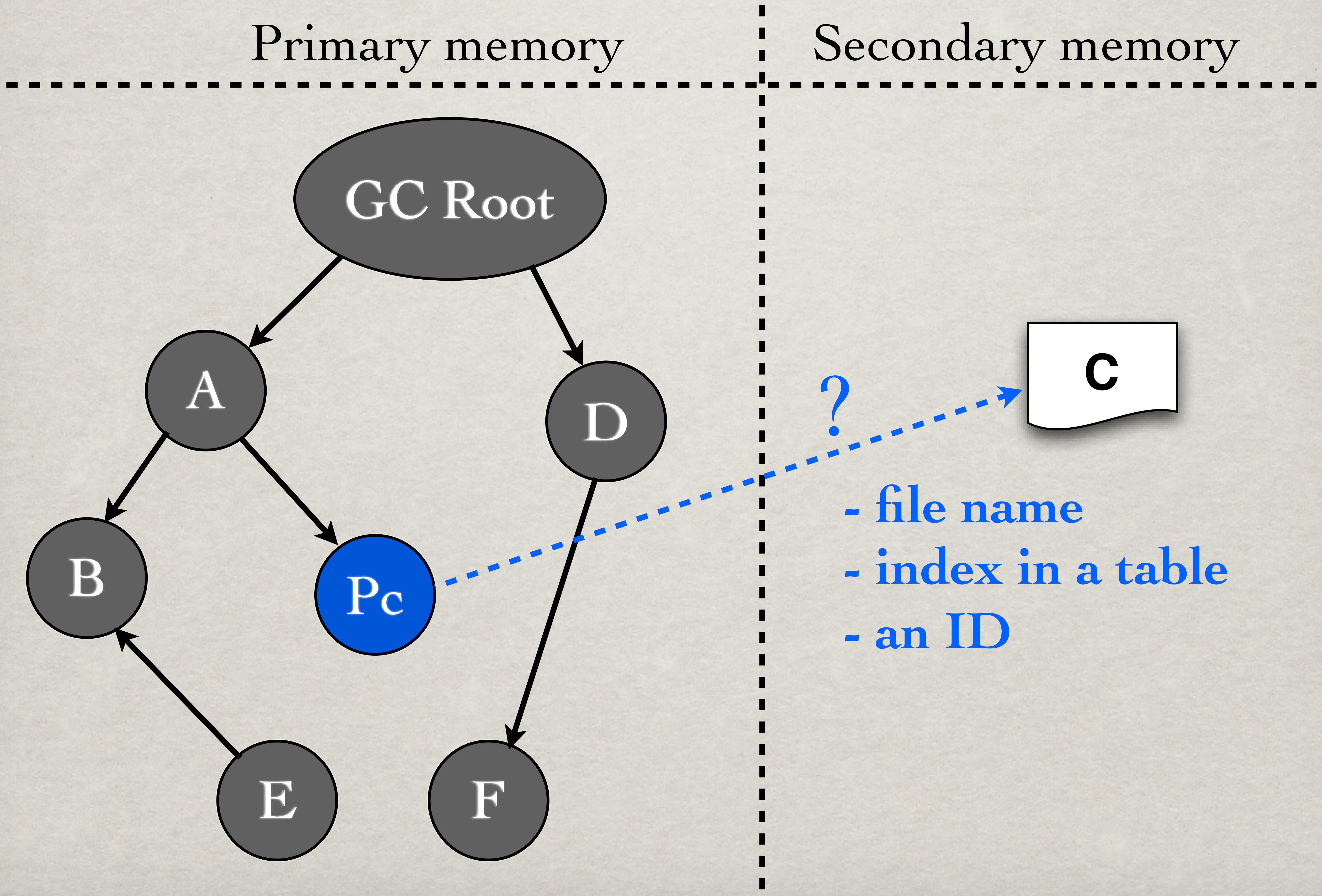
We should not replace instances of True, False, BlockCloure and SmallInteger with proxies.

Some optimizations such as #class should be disabled.

POINTER SWIZZLING

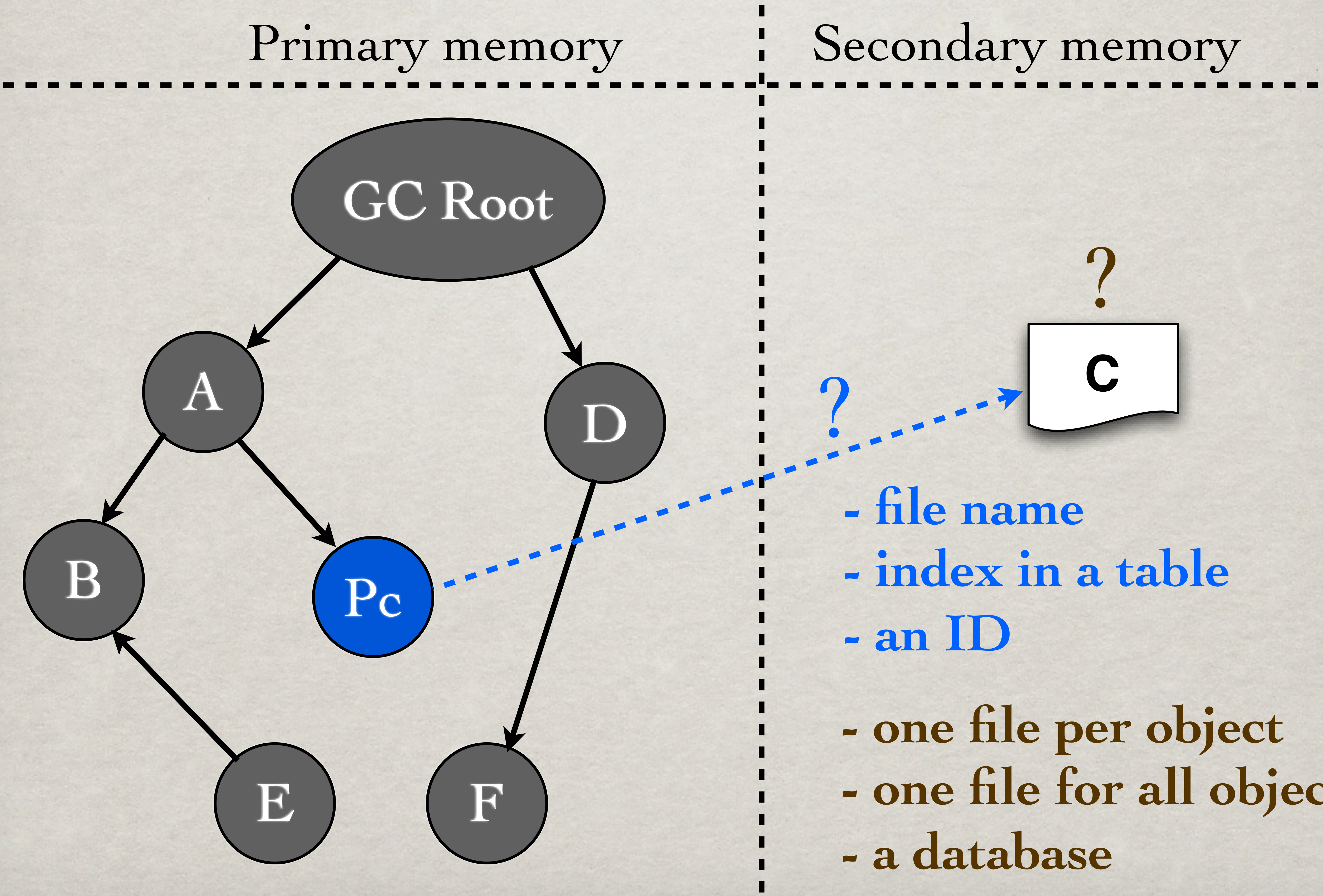


POINTER SWIZZLING



- file name
- index in a table
- an ID

POINTER SWIZZLING



- file name
- index in a table
- an ID

- one file per object
- one file for all objects
- a database

Lessons

We need to map memory addresses of primary memory to addresses in secondary memory.

We need to define how objects are stored in secondary memory.

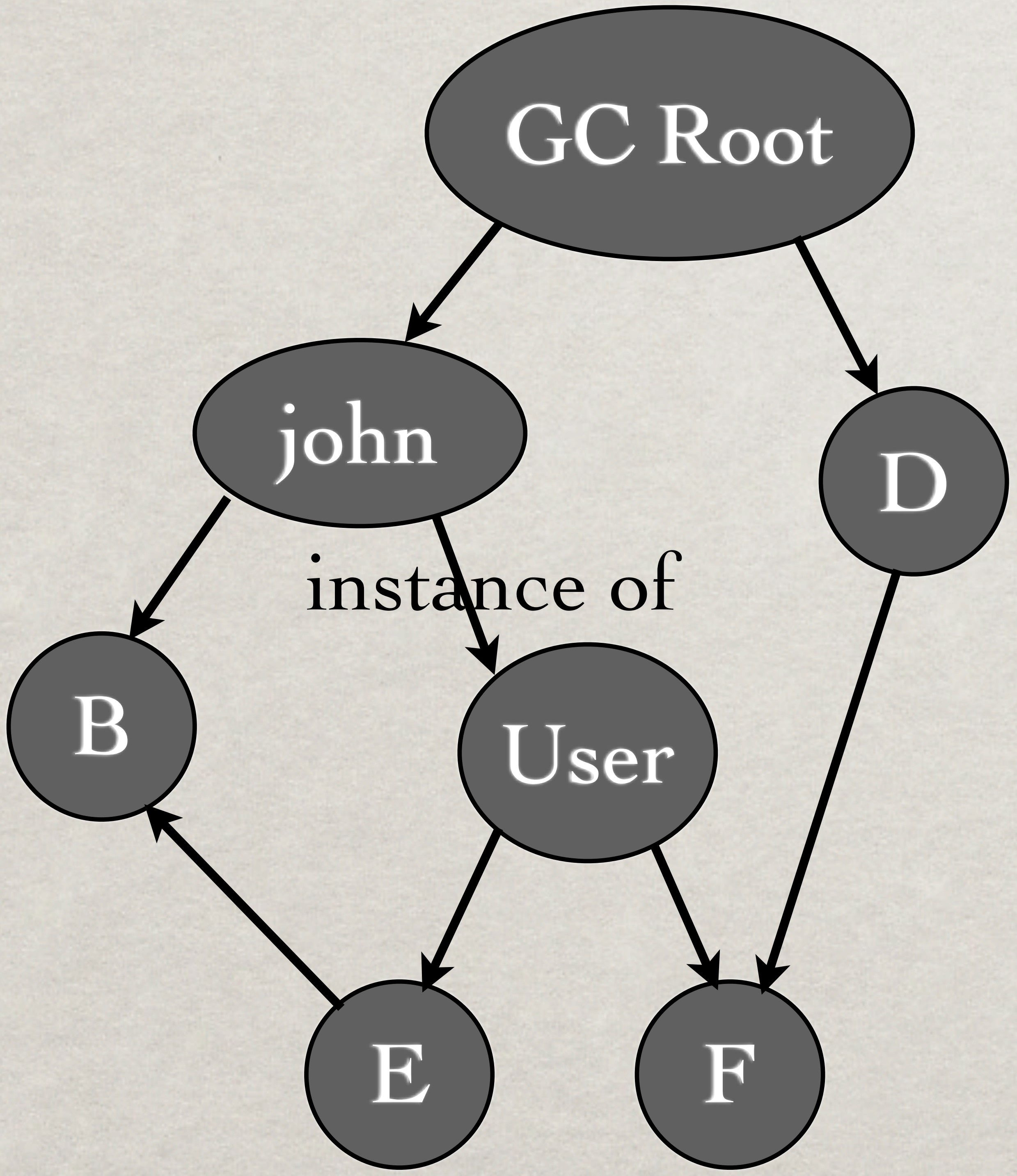
Small proxies

- Proxies as regular objects.
  - Store the minimal possible state.
- Proxies as immediate objects.
  - We need space in the memory address.

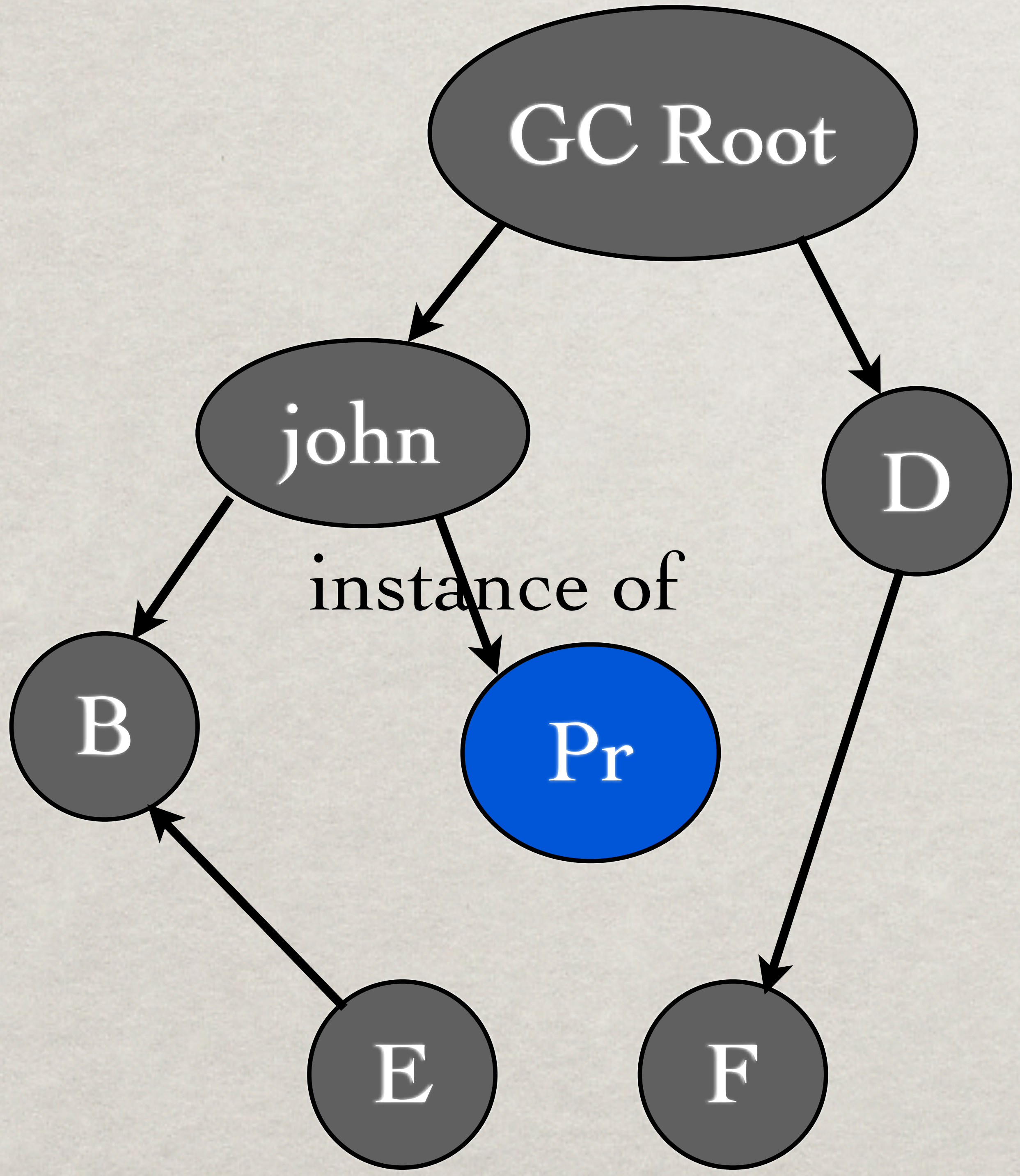
Lesson

We have to make proxies use as little memory as possible.

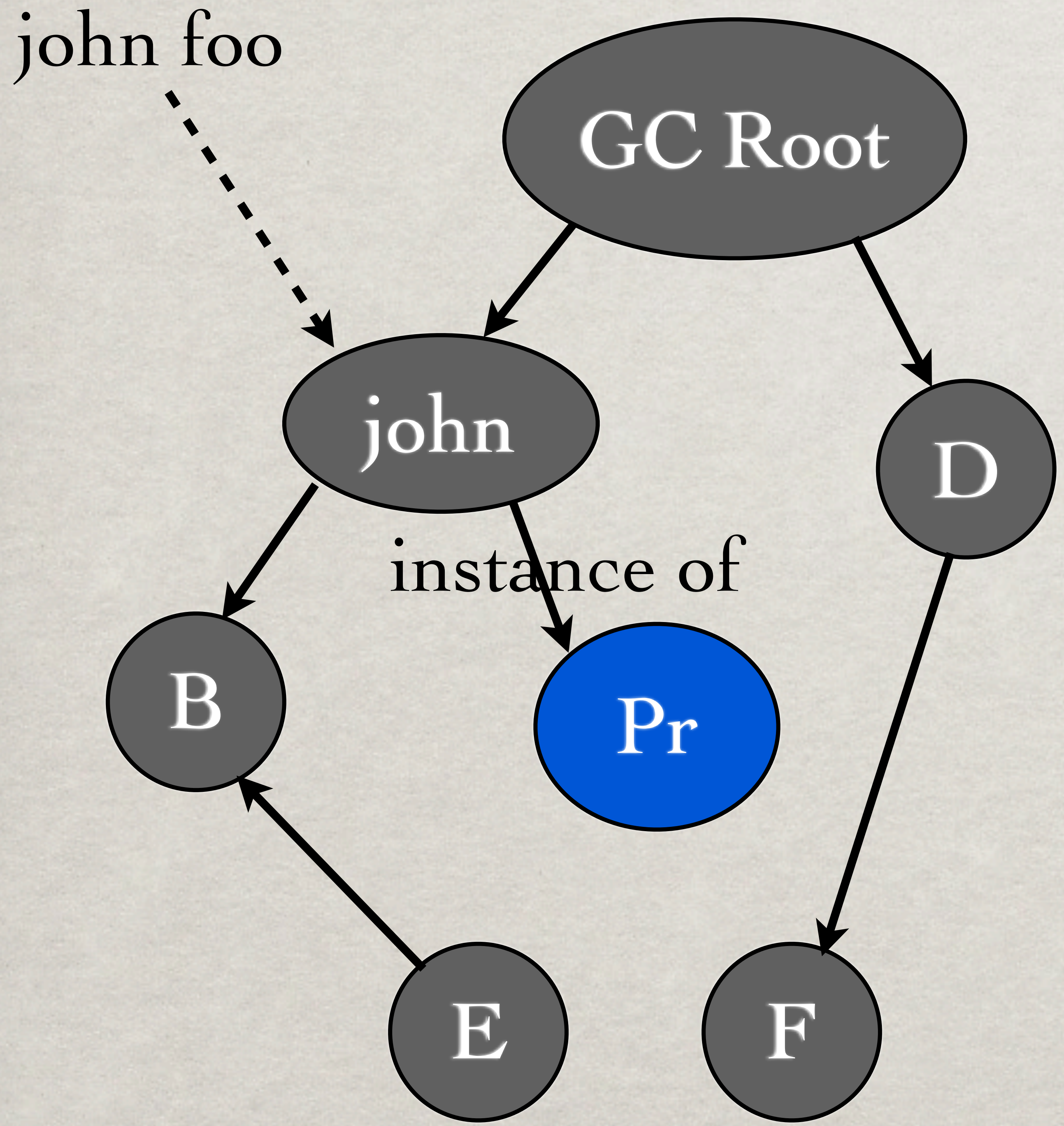
Special proxies



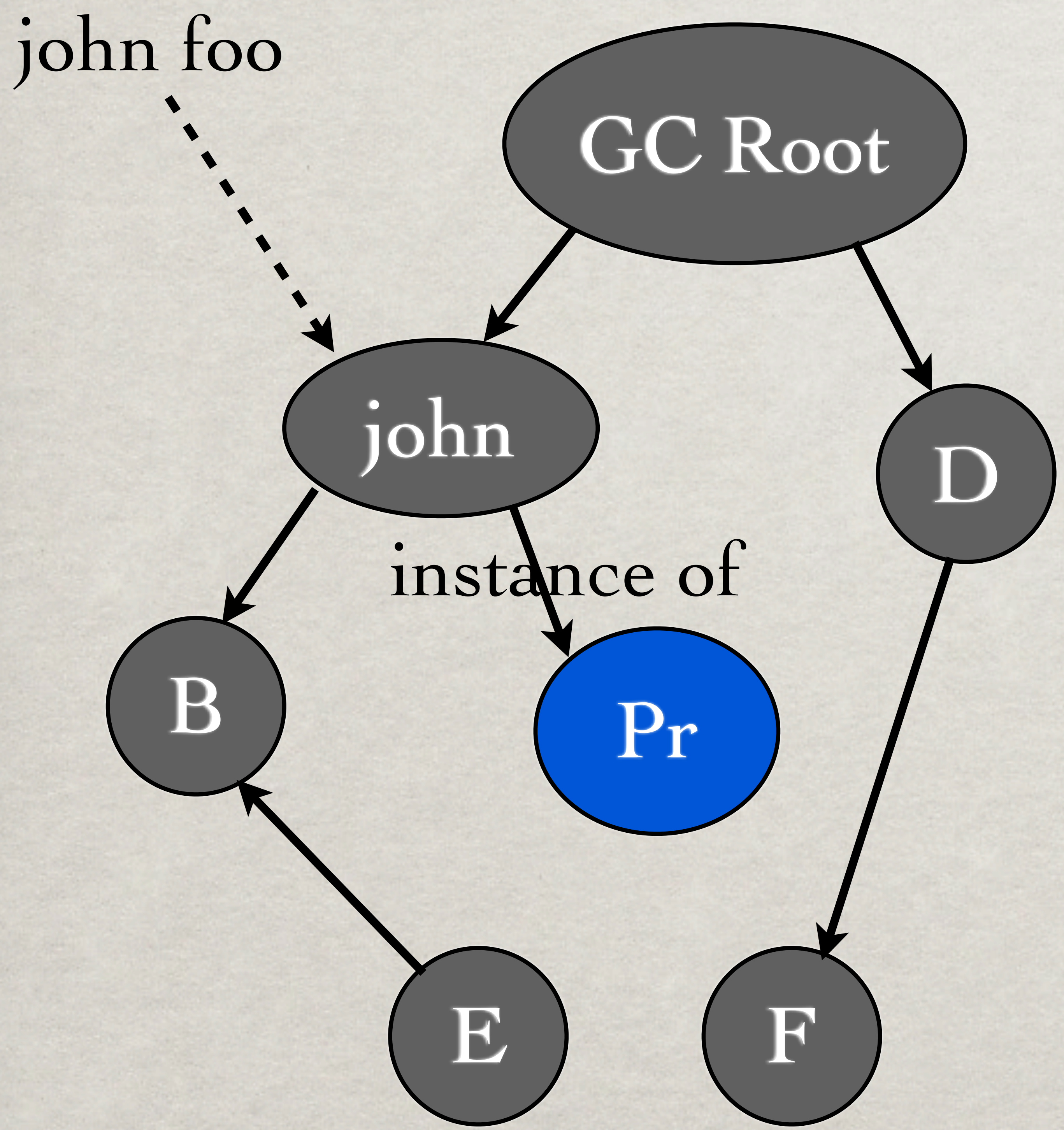
Special proxies



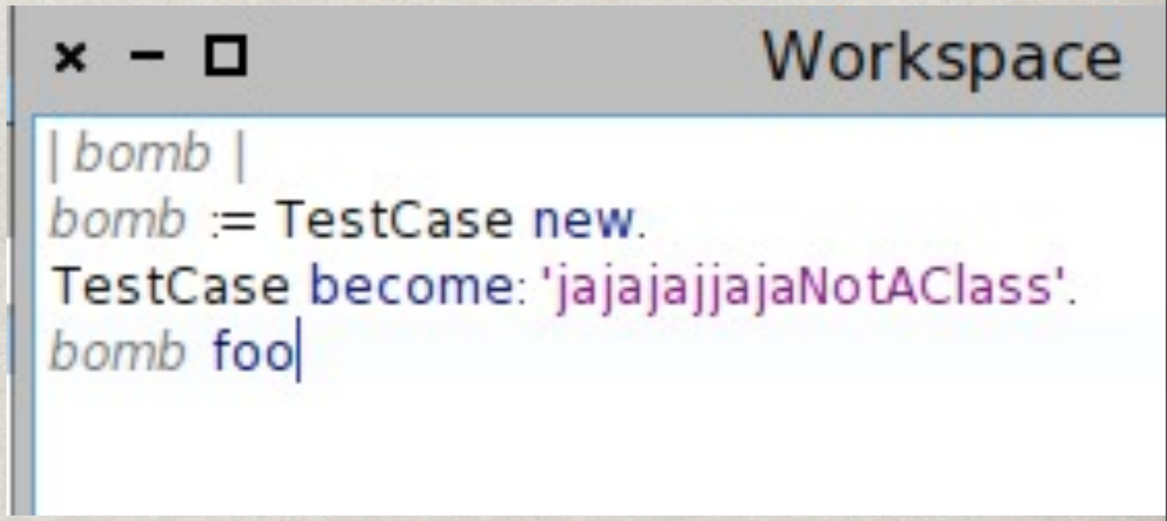
Special proxies



Special proxies



VM Crash!



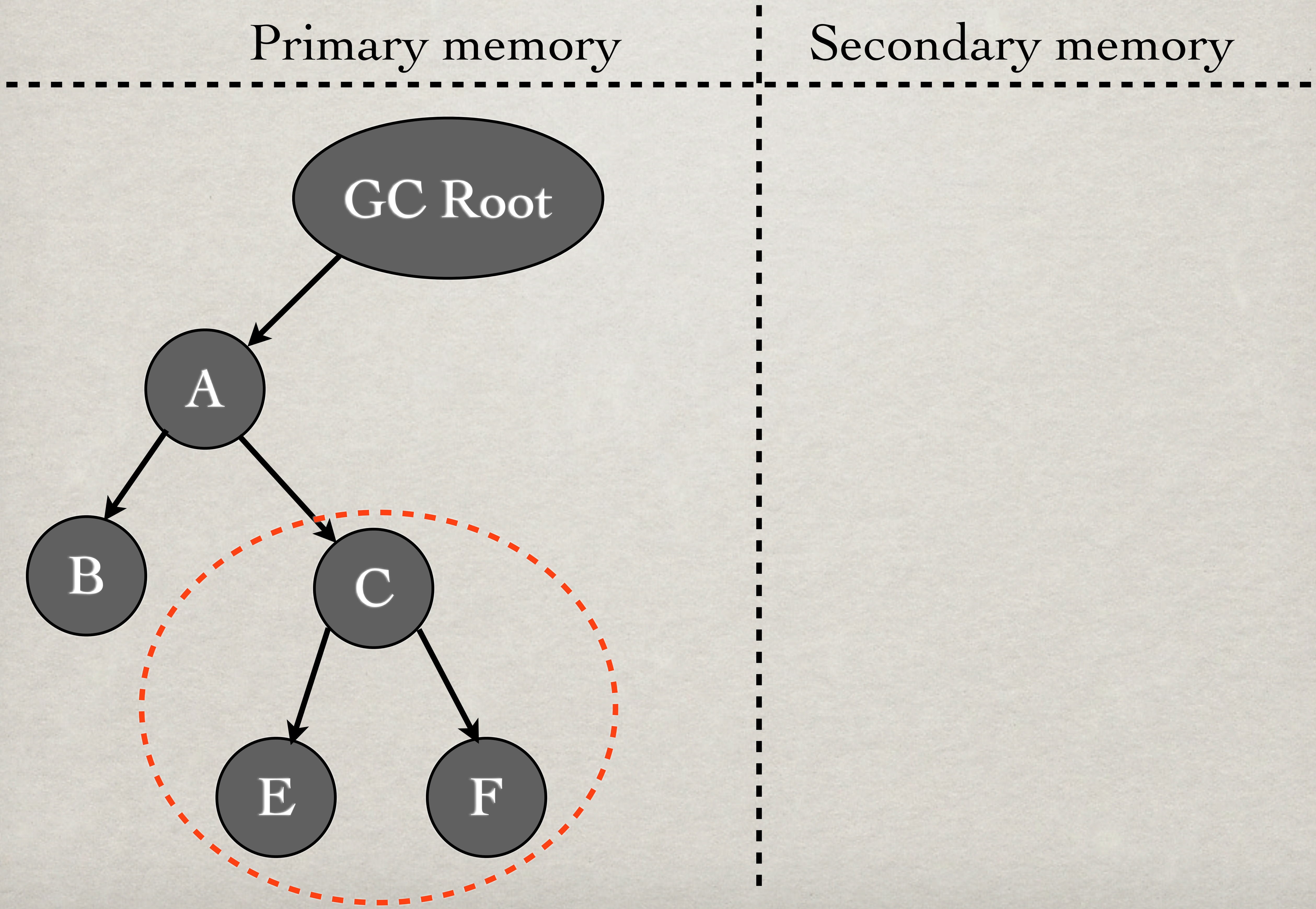
```
| bomb |
bomb := TestCase new.
TestCase become: 'jajajajjajaNotAClass'.
bomb foo
```

Lesson

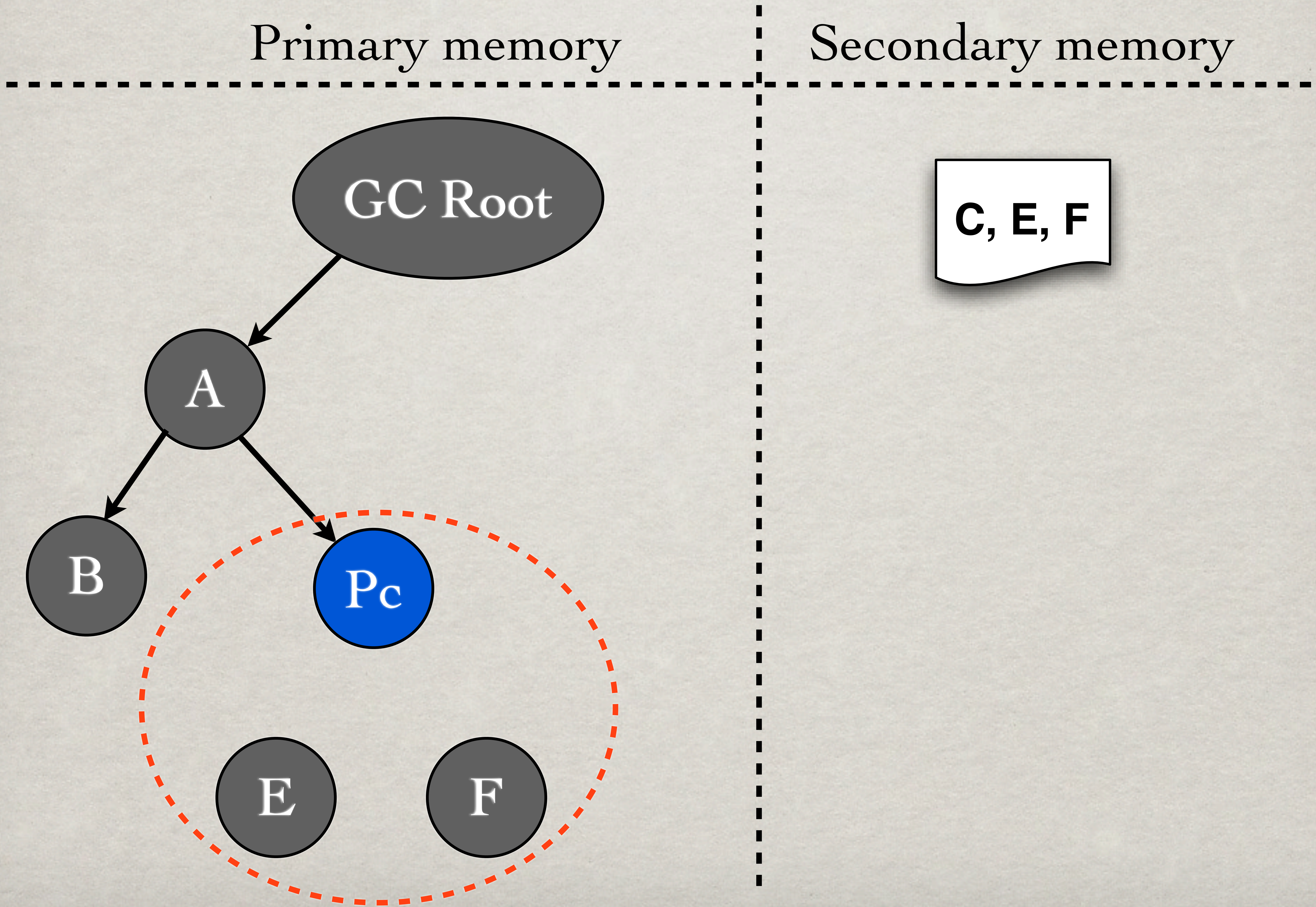
We need special proxies for those classes that the VM expects to have certain shape.

Graphs and Shared Objects

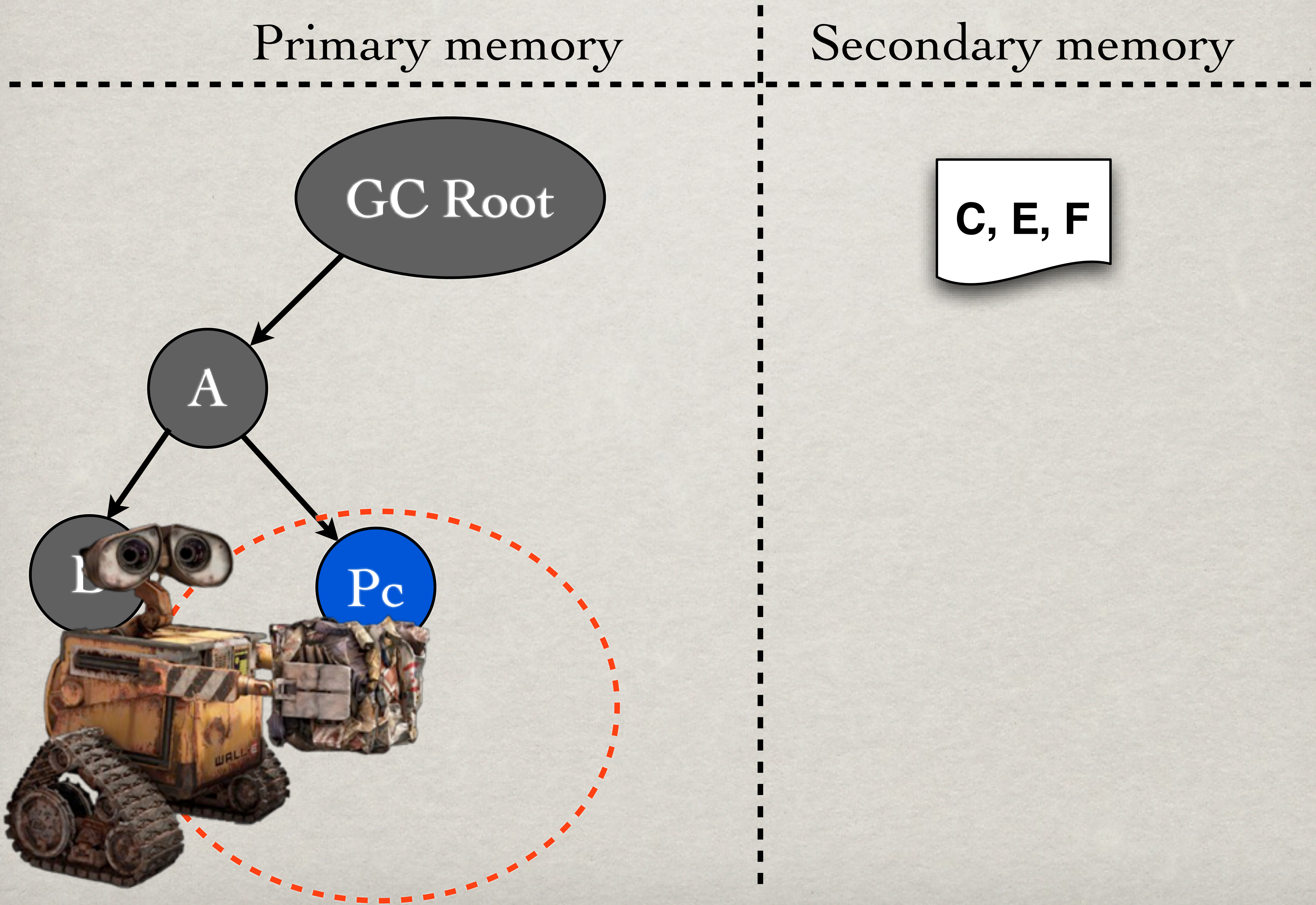
How to group them



How to group them



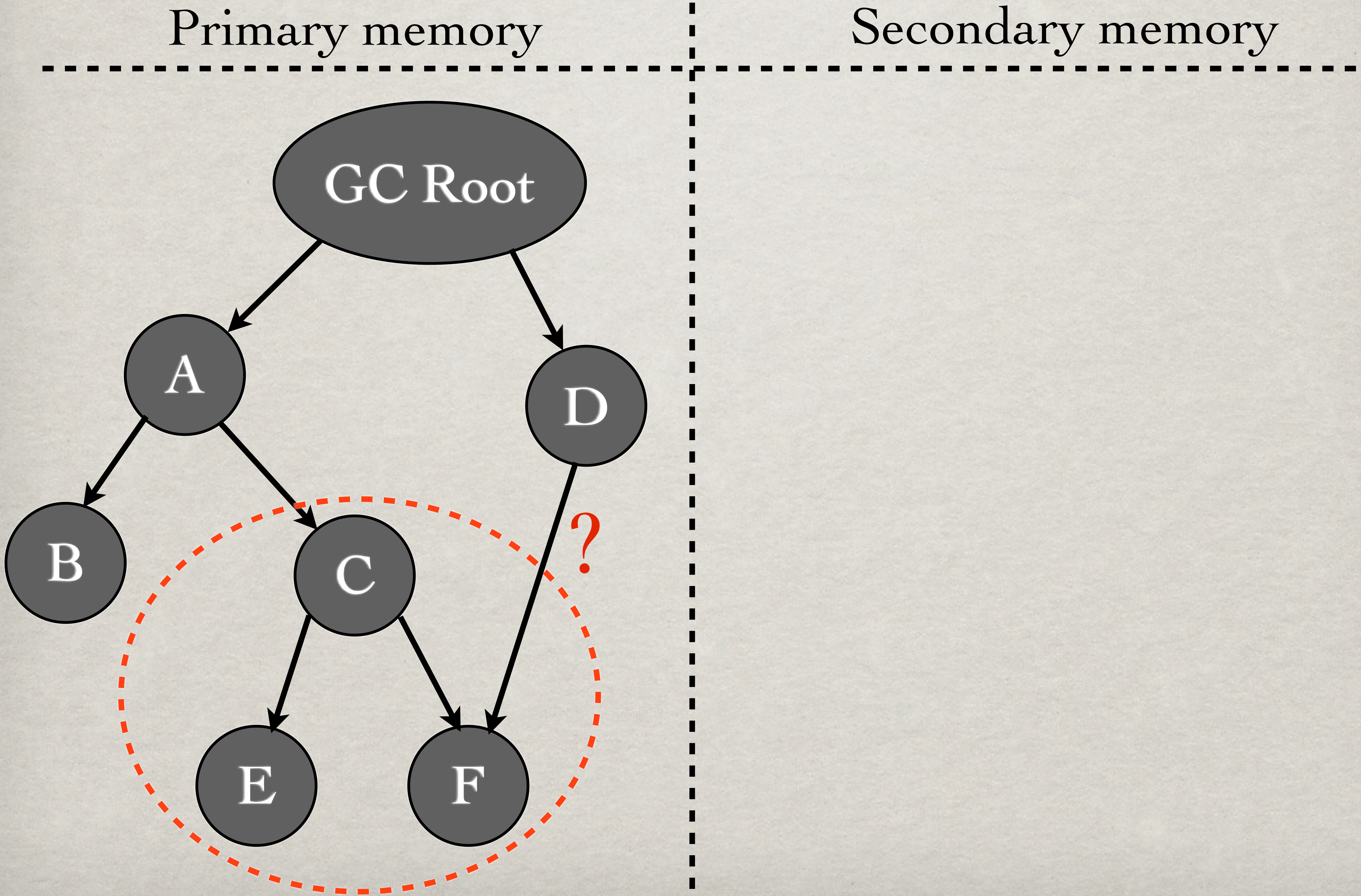
How to group them



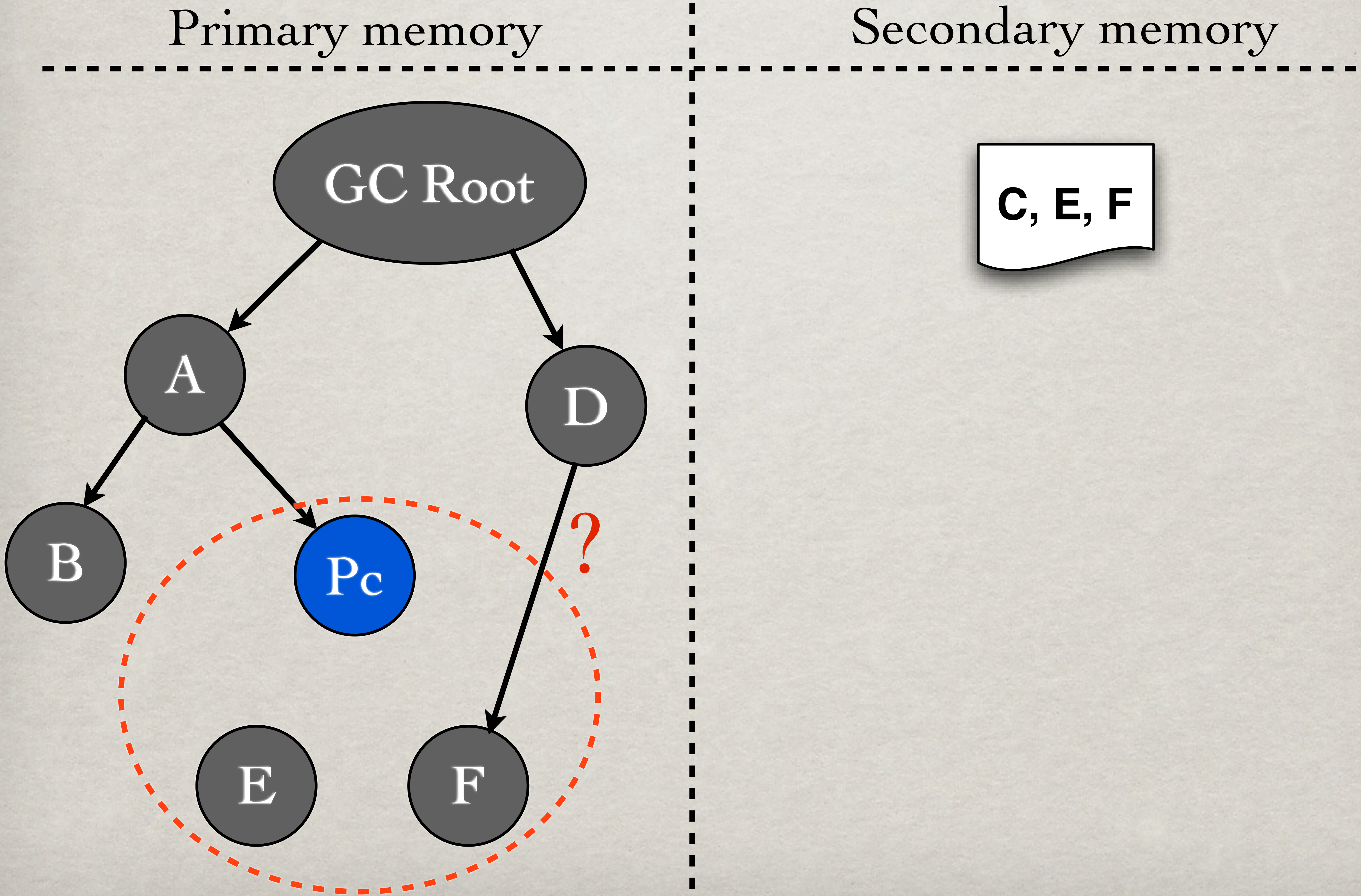
Lesson

Grouping unused objects in graphs allows us to use less proxies. In addition, objects inside a graph may be used all together or not used at all.

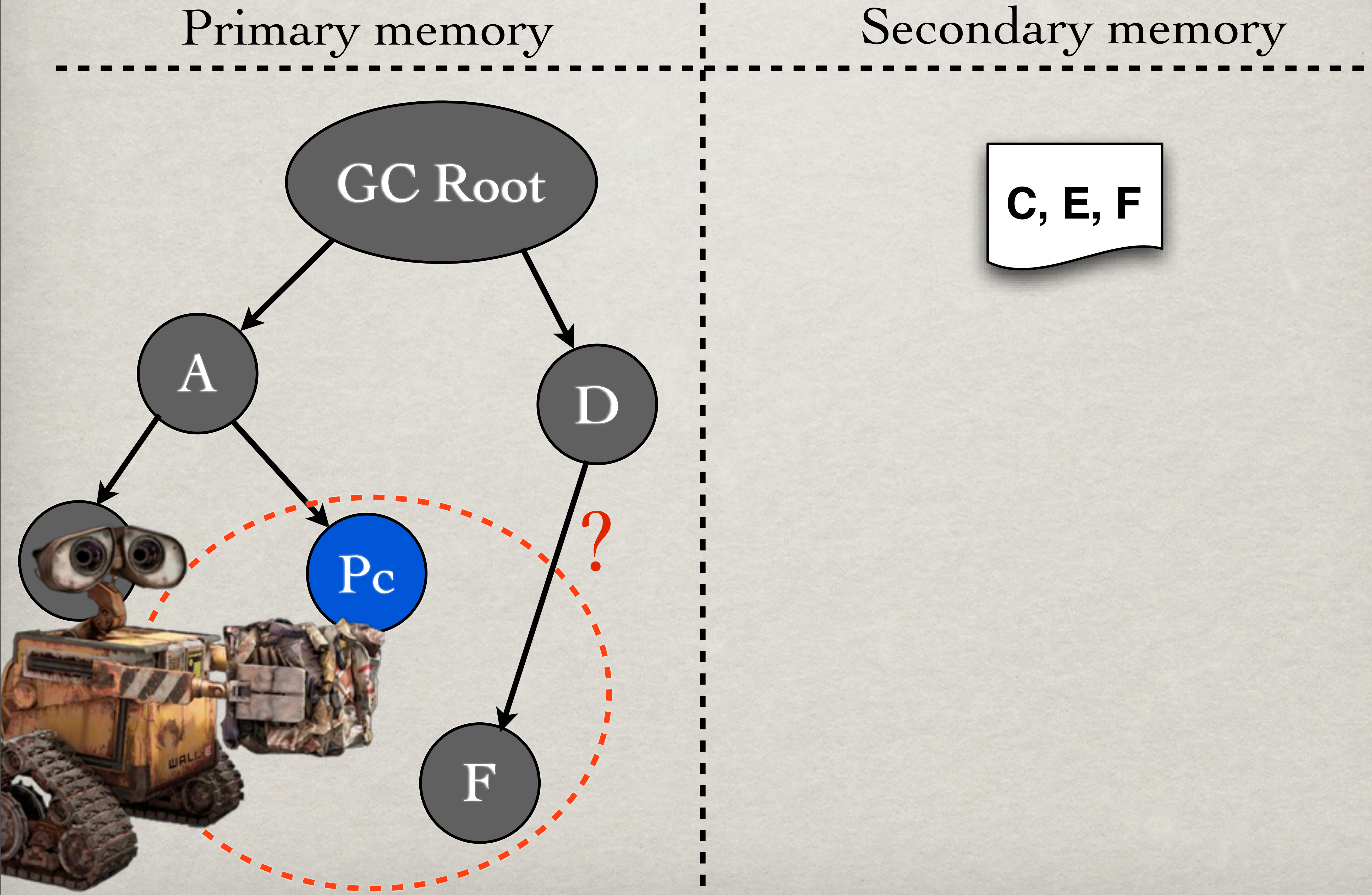
Shared objects



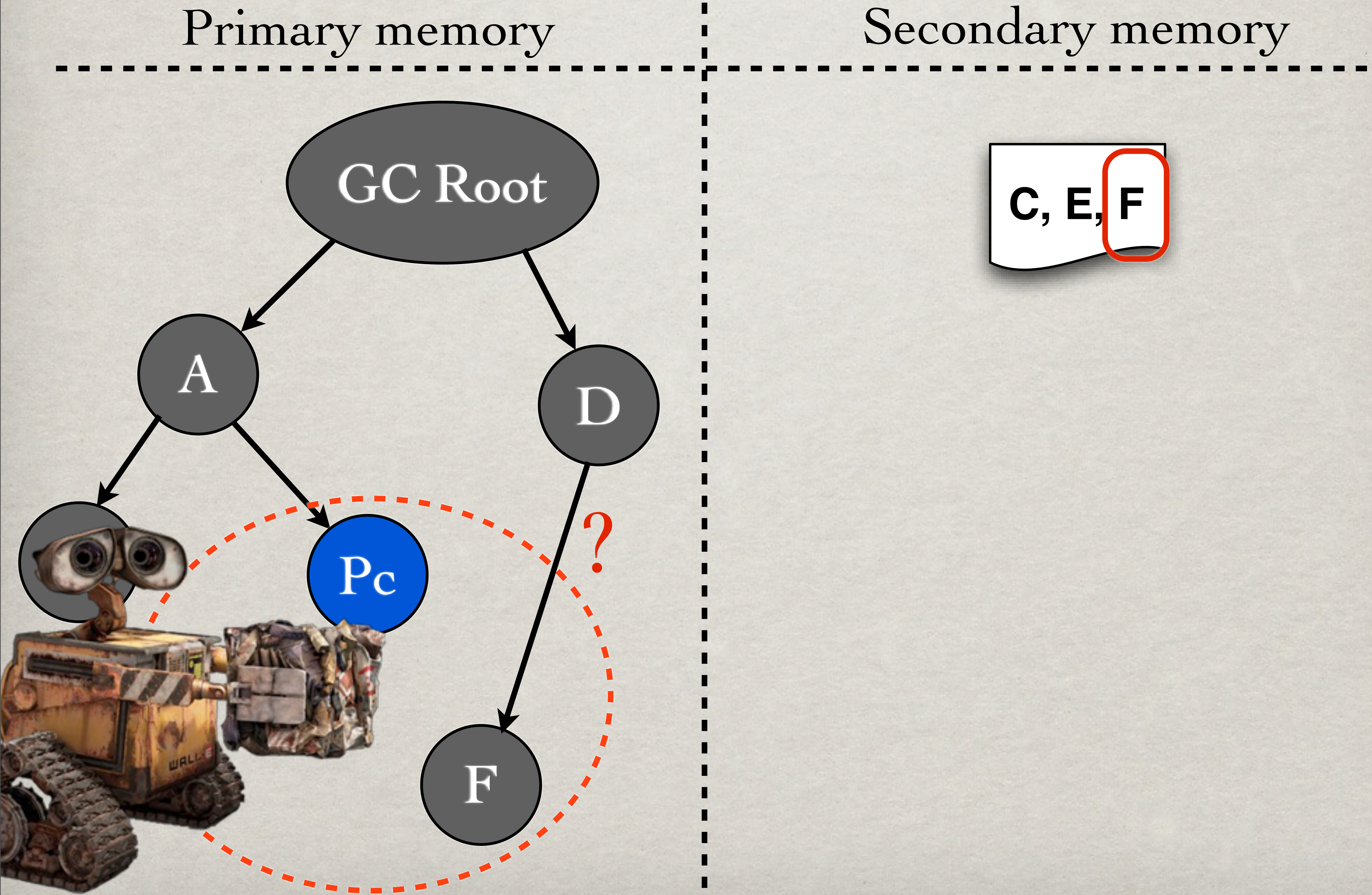
Shared objects



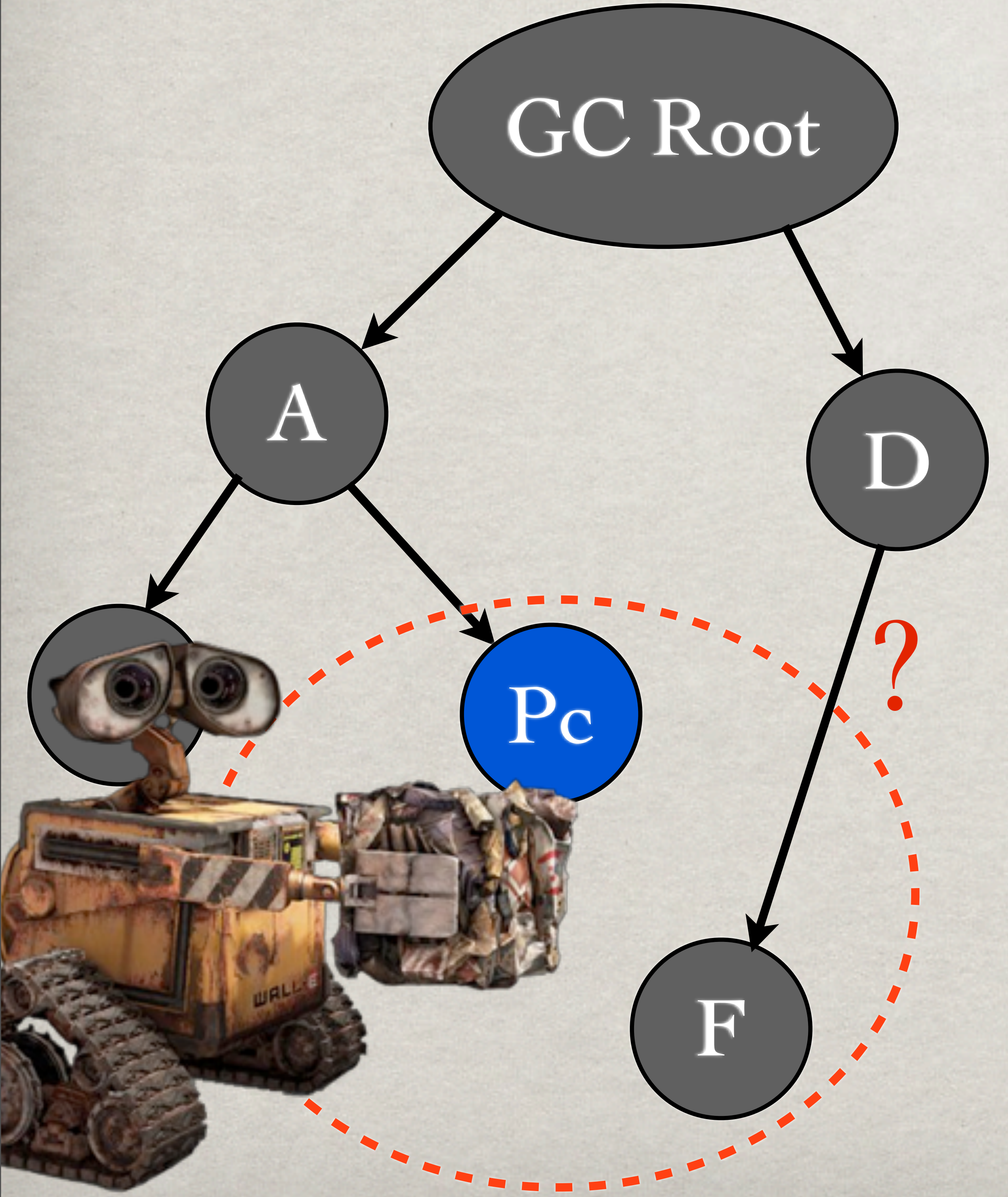
Shared objects



Shared objects



Shared objects



Secondary memory
C, E, F

- Should we swap shared objects or not?
- If not... proxies for them too?
- How can we detect shared objects?
- Much more...

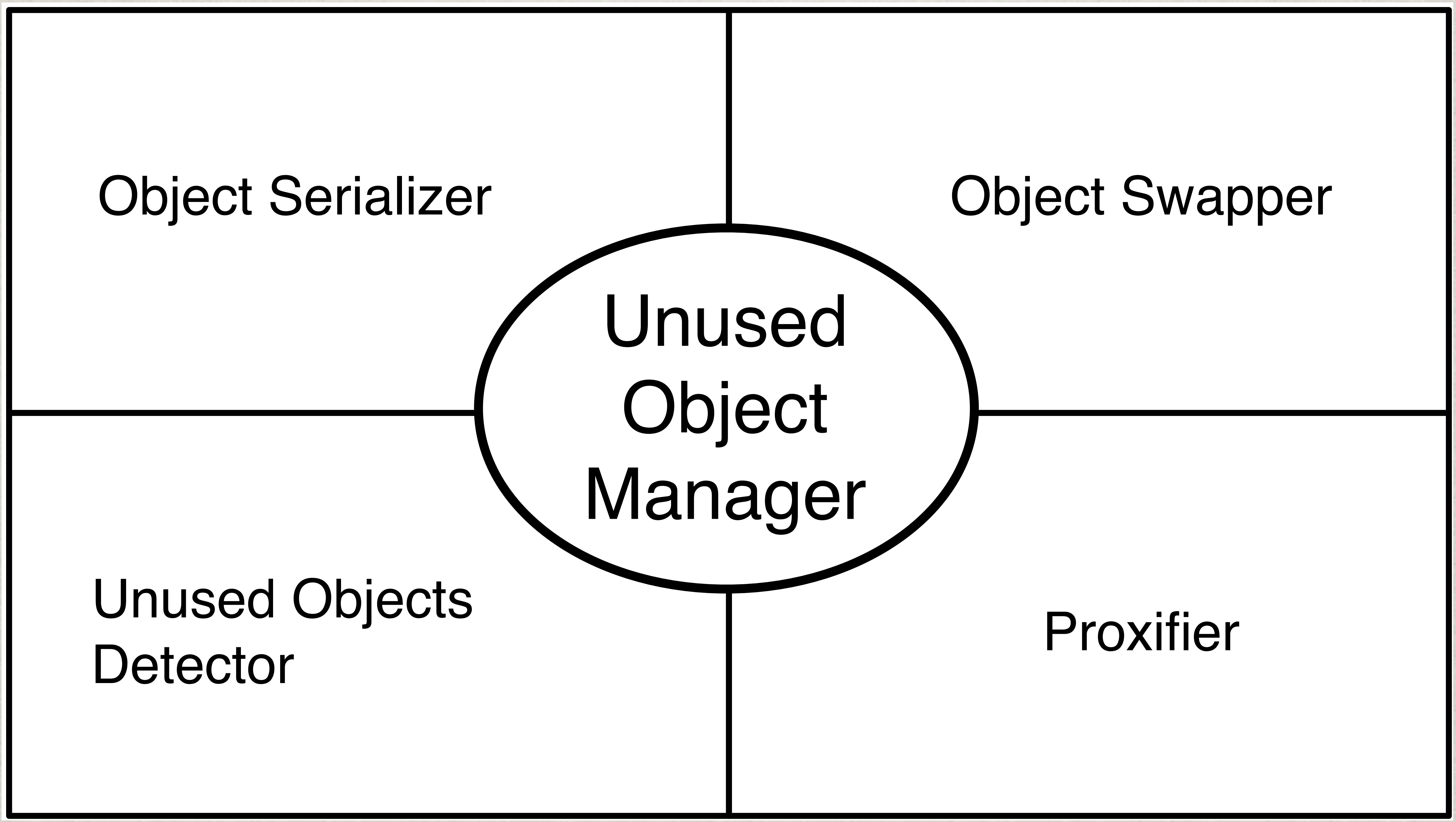
For more details, read the paper :)

Lesson

Correct and efficient handling of shared objects inside graphs is a really difficult task.

Our first steps...

UOM subsystems

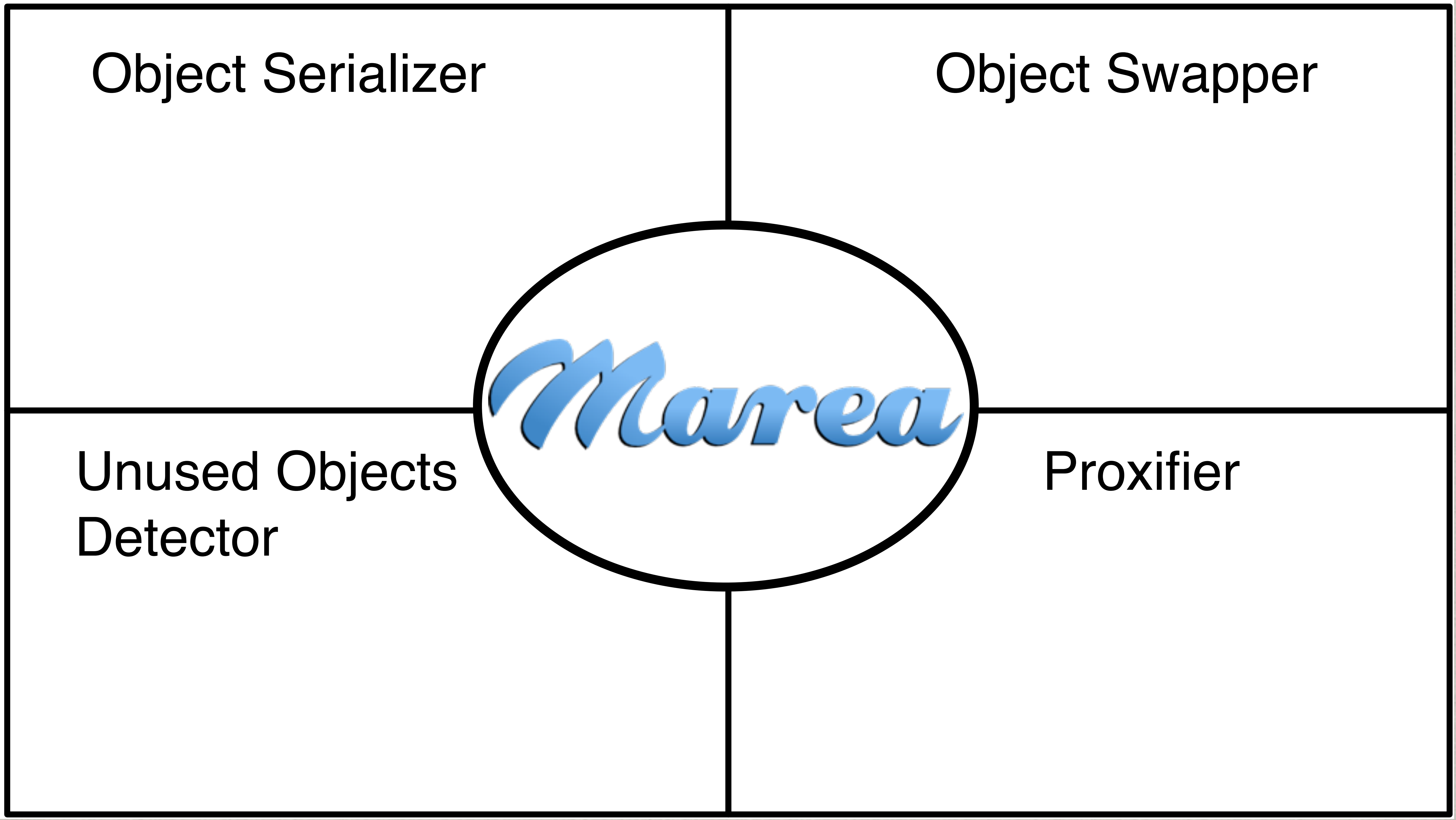


Marea

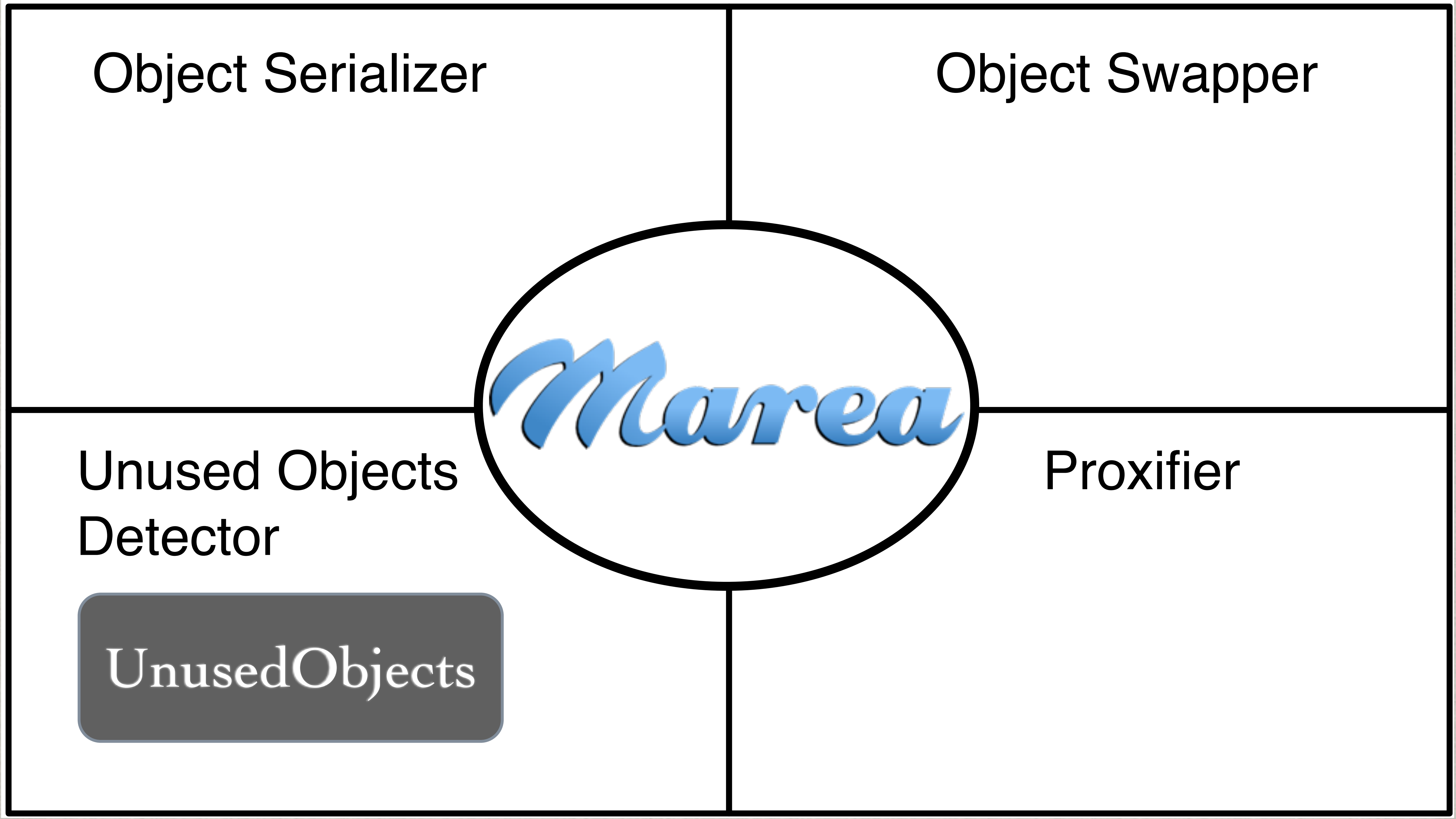
http://rmod.lille.inria.fr/web/pier/software/Marea

http://www.squeaksource.com/Marea.html

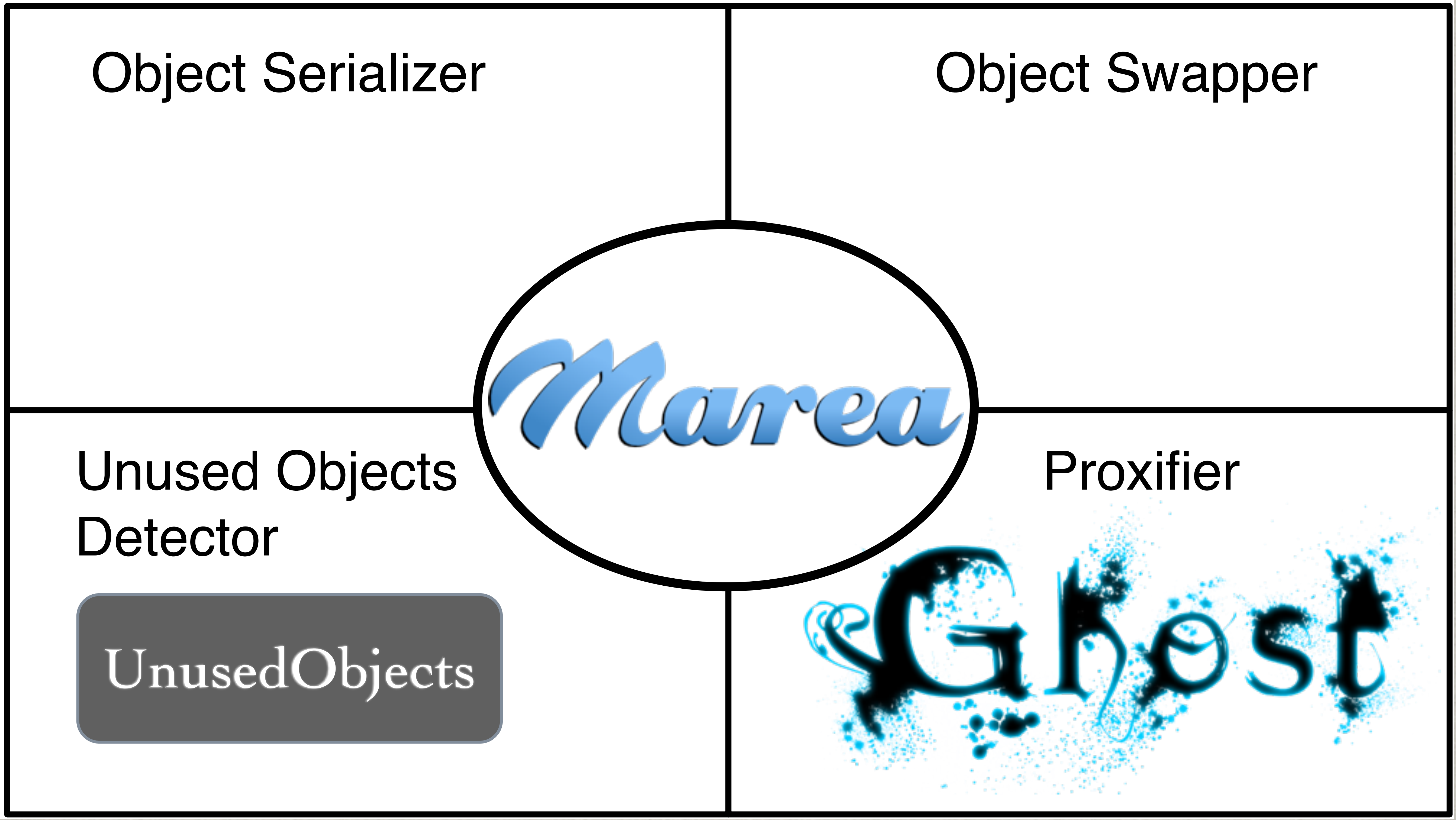
Marea subsystems



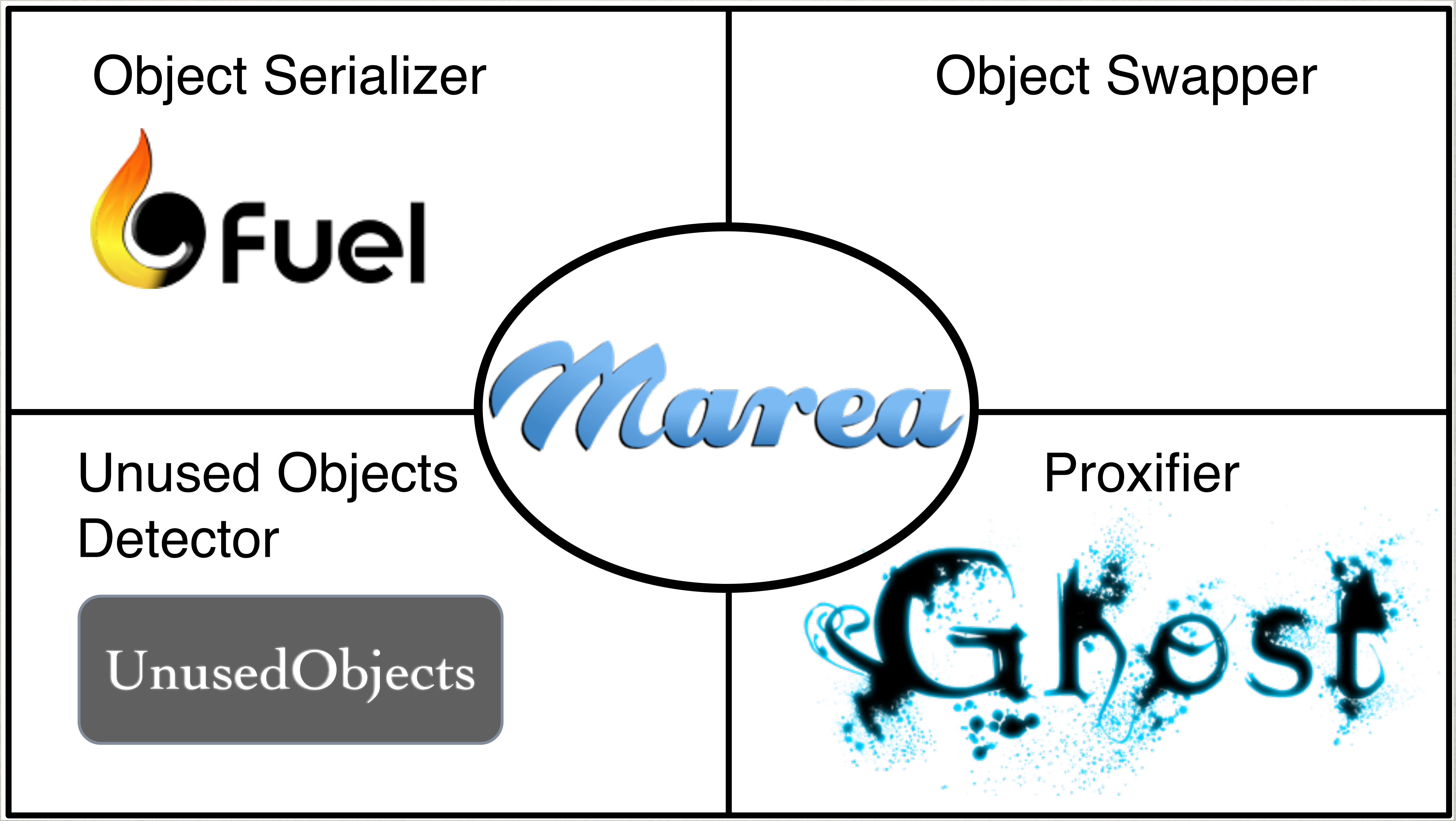
Marea subsystems



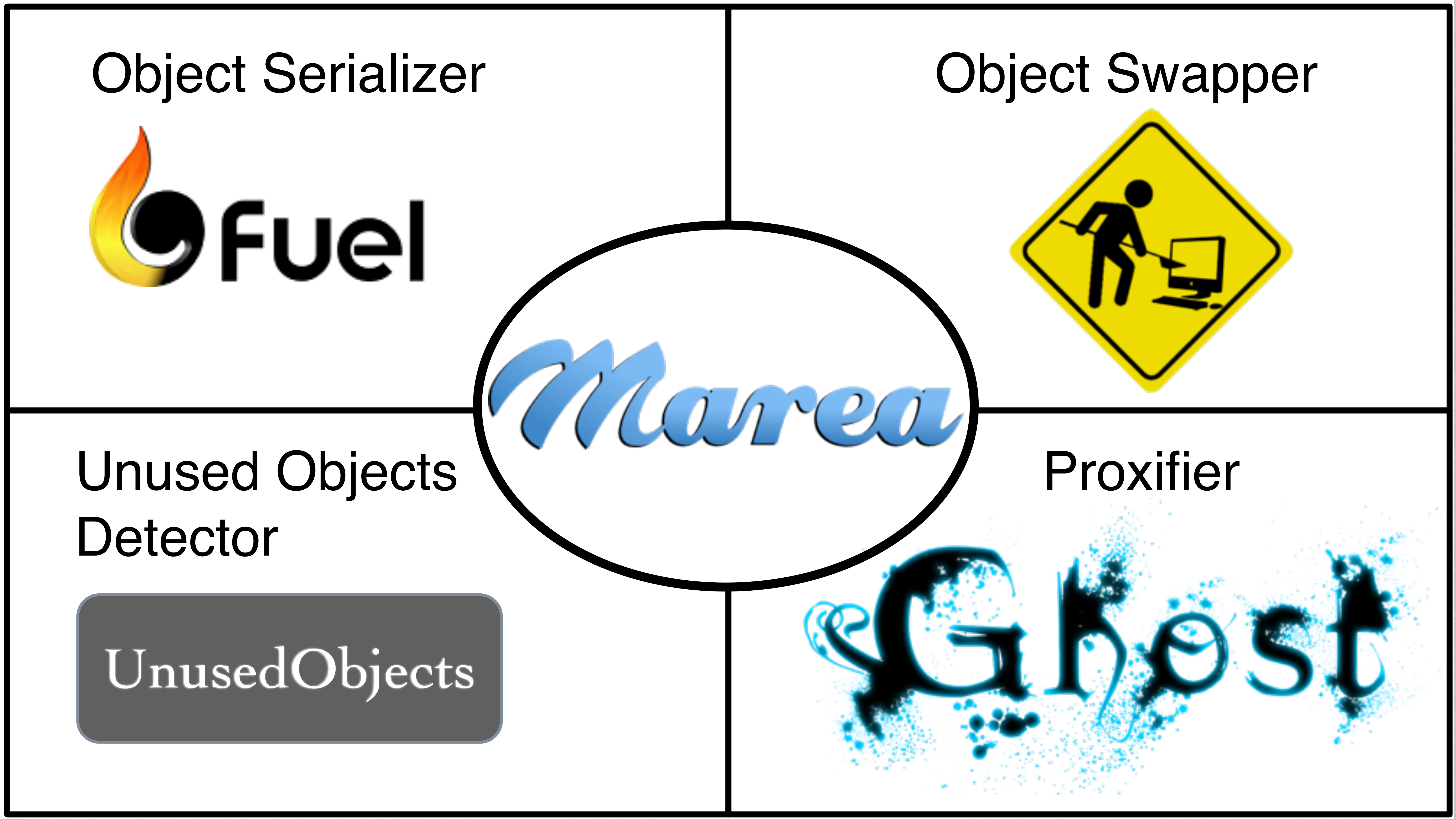
MAREA SUBSYSTEMS



Marea subsystems

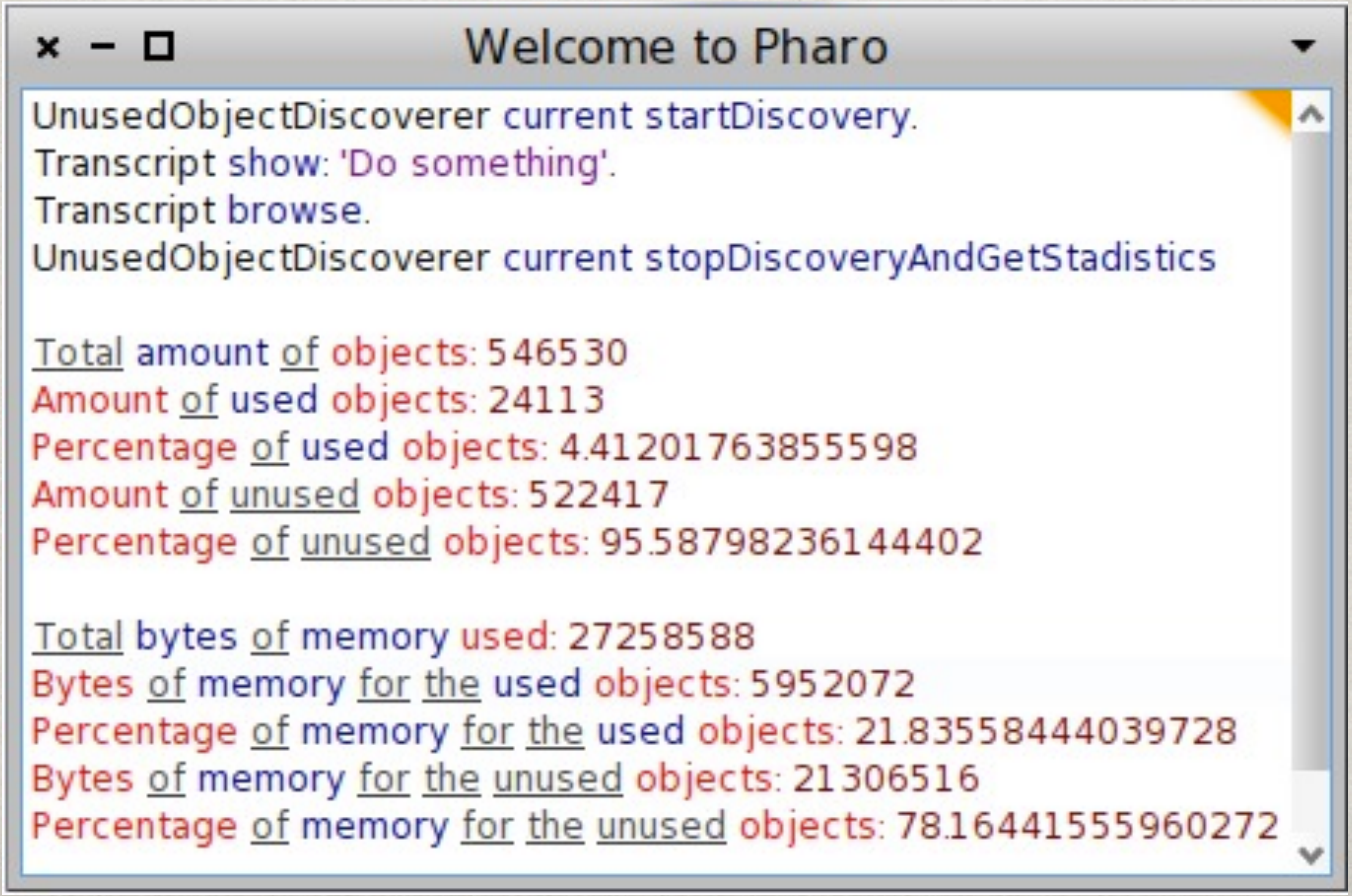


Marea subsystems



Unused Objects

- CogVM fork to mark objects when “used”.
- Image side code to set and get “usage bit”.
- Some other useful primitives.

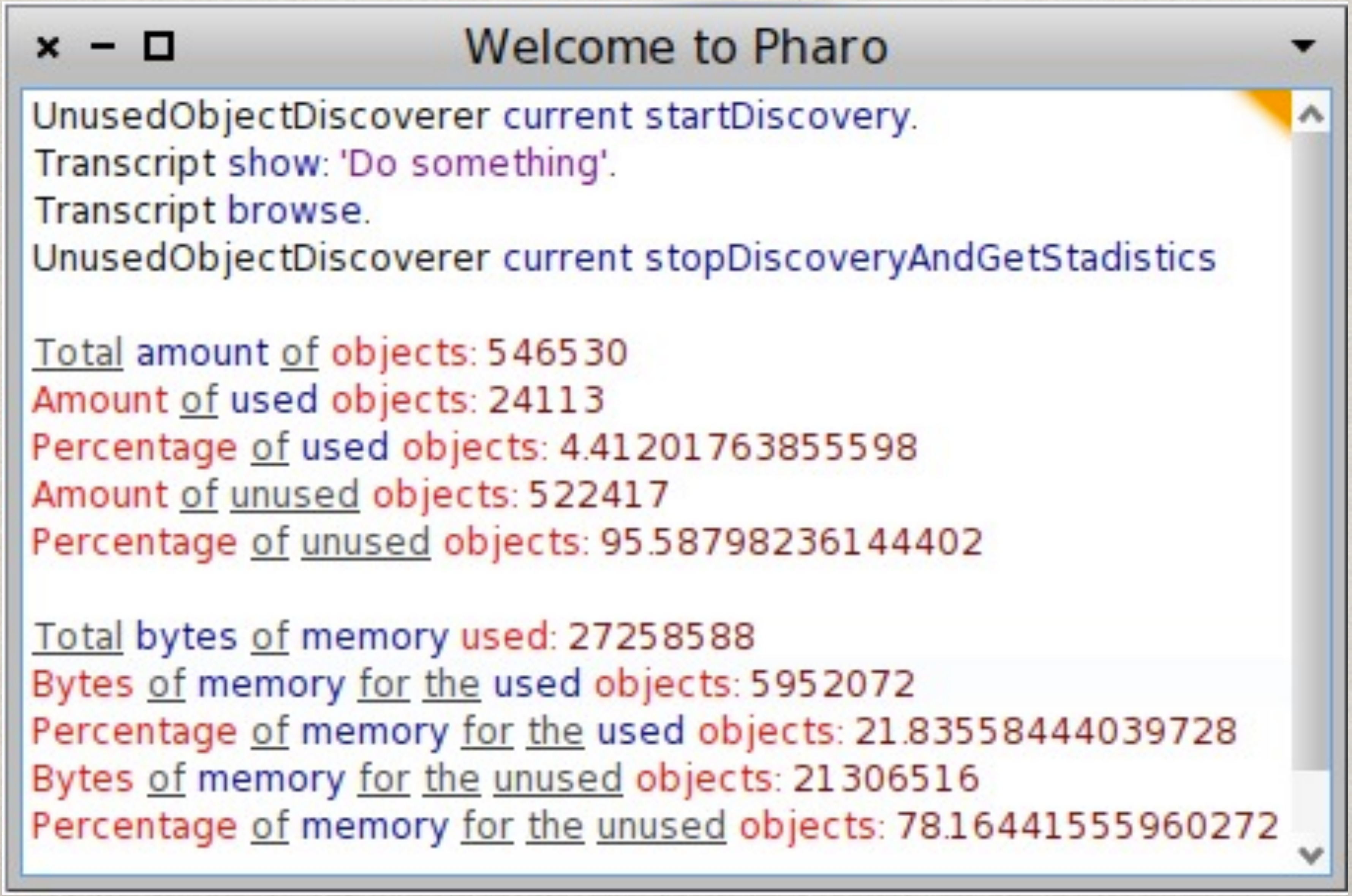


```
UnusedObjectDiscoverer current startDiscovery.
Transcript show: 'Do something'.
Transcript browse.
UnusedObjectDiscoverer current stopDiscoveryAndGetStadistics

Total amount of objects: 546530
Amount of used objects: 24113
Percentage of used objects: 4.41201763855598
Amount of unused objects: 522417
Percentage of unused objects: 95.58798236144402

Total bytes of memory used: 27258588
Bytes of memory for the used objects: 5952072
Percentage of memory for the used objects: 21.83558444039728
Bytes of memory for the unused objects: 21306516
Percentage of memory for the unused objects: 78.16441555960272
```

Unused Obj

Paper accepted Smalltalks 2010

- CogVM fork to mark objects when
- Image side code to set and get "usage bit".
- Some other useful primitives.



```
UnusedObjectDiscoverer current startDiscovery.
Transcript show: 'Do something'.
Transcript browse.
UnusedObjectDiscoverer current stopDiscoveryAndGetStadistics

Total amount of objects: 546530
Amount of used objects: 24113
Percentage of used objects: 4.41201763855598
Amount of unused objects: 522417
Percentage of unused objects: 95.58798236144402

Total bytes of memory used: 27258588
Bytes of memory for the used objects: 5952072
Percentage of memory for the used objects: 21.83558444039728
Bytes of memory for the unused objects: 21306516
Percentage of memory for the unused objects: 78.16441555960272
```

Ghost

- Do not use #doesNotUnderstand.
- Intercept “all” messages (except the optimized ones).
- Uniform (e.g, it can proxify classes and methods).
- Stratified.
- Small memory footprint.

Ghost

Paper accepted IWST 2011

- Do not use #doesNotUnderstand.
- Intercept "all" messages (except the optimized ones).
- Uniform (e.g, it can proxify classes and methods).
- Stratified.
- Small memory footprint.

Fuel

- Fast.
- Easy to adapt to my custom needs.
- Complete: can serialize almost any type of object.
- Well tested and benchmarked.

Paper accepted IWST 2011

- Fast.
- Easy to adapt to my custom needs.
- Complete: can serialize almost any type of object.
- Well tested and benchmarked.

Object Swapper

- Still in development.
- How to efficiently solve the problem of shared objects.
- Complete the process:
  - **Which** graphs to swap out.
  - **When** to swap out.

Thanks

http://rmod.lille.inria.fr/web/pier/software/Marea

Mariano Martinez Peck
marianopeck@gmail.com
http://marianopeck.wordpress.com/