Object graphs swapping

Mariano Martinez Peck

marianopeck@gmail.com

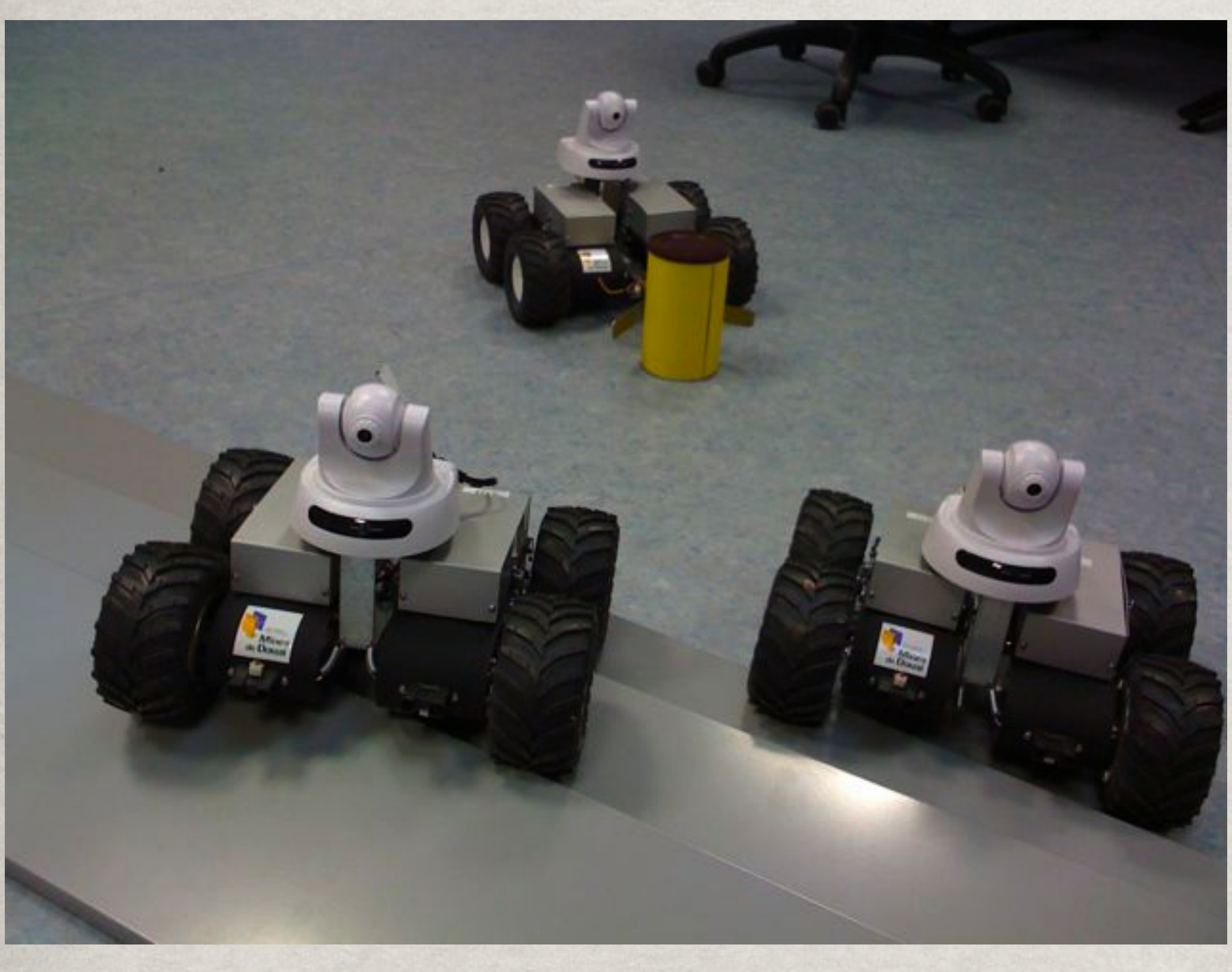
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE INRIA

RMod

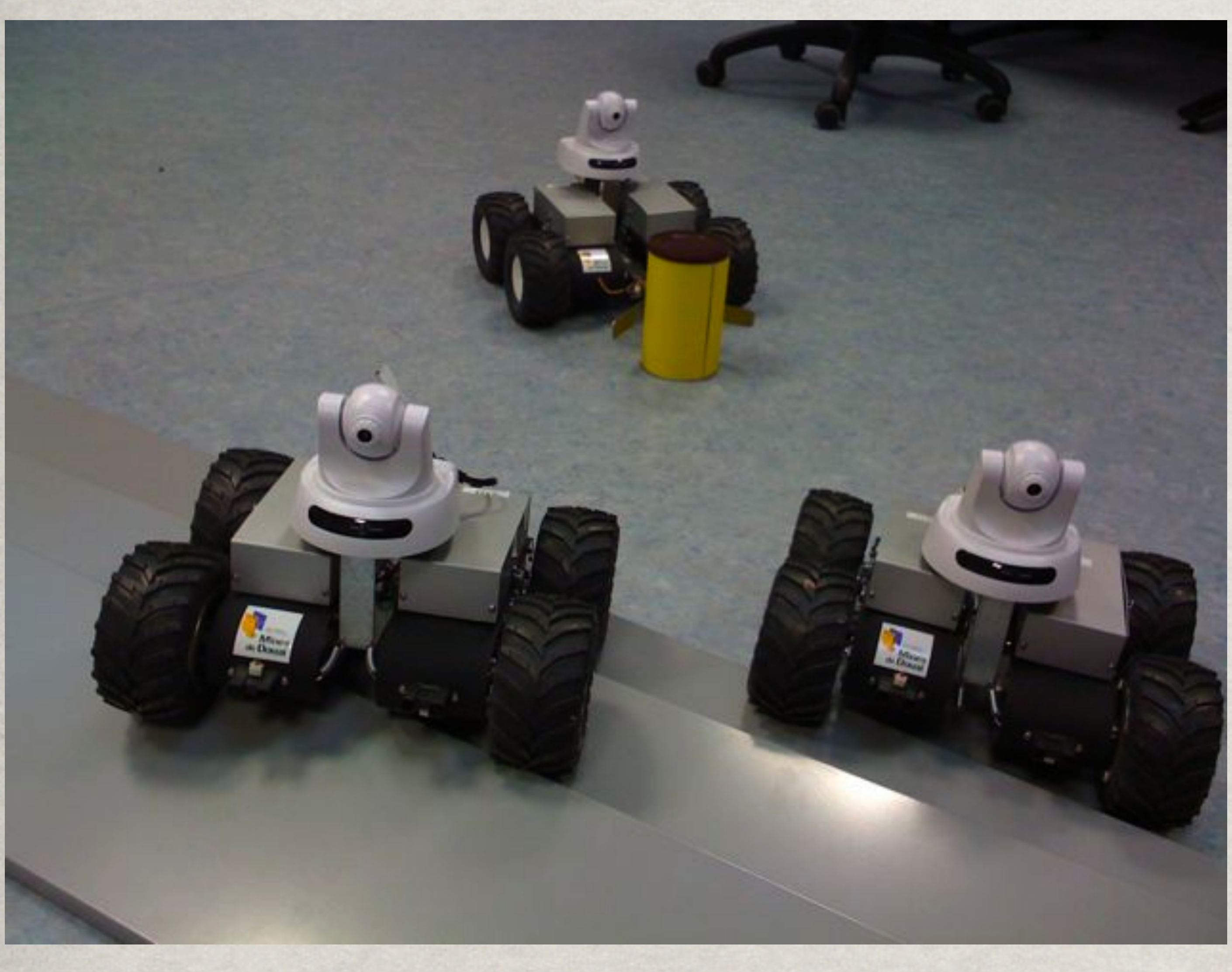
Ecole d'Ingénieurs Centre de Recherche Mines de Douai LILLE EURORÉGION

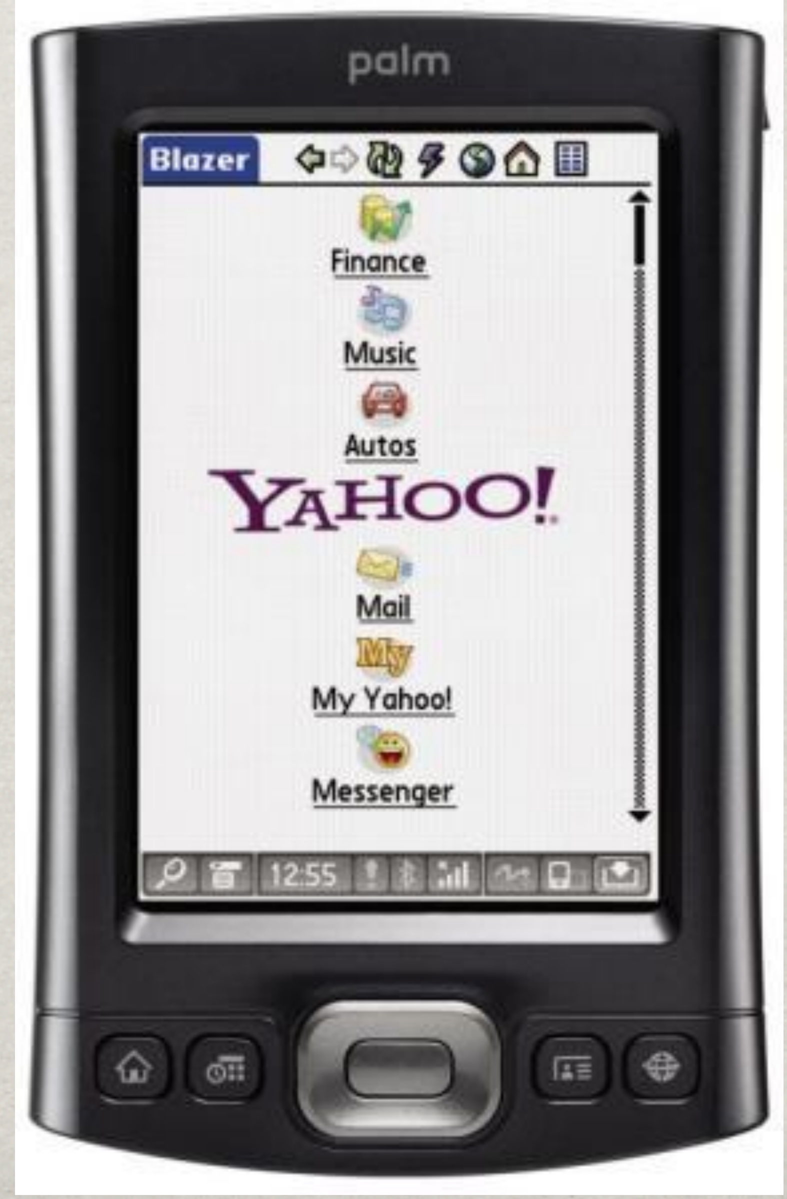
The context

The context

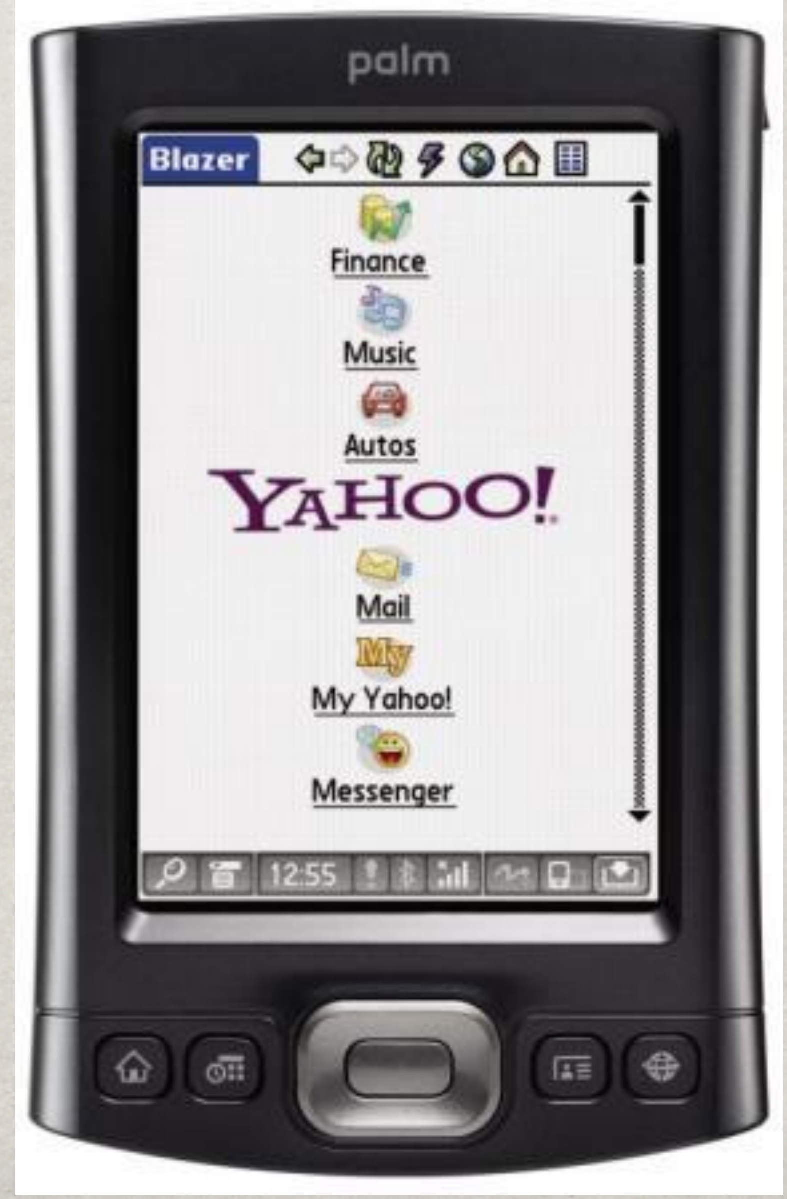


The context




palm
Blazer
Finance
Music
Autos
YAHOO!
Mail
My Yahoo!
Messenger
12:55

The context


palm
Blazer
Finance
Music
Autos
YAHOO!
Mail
My Yahoo!
Messenger
12:55

The context

The context

Problem

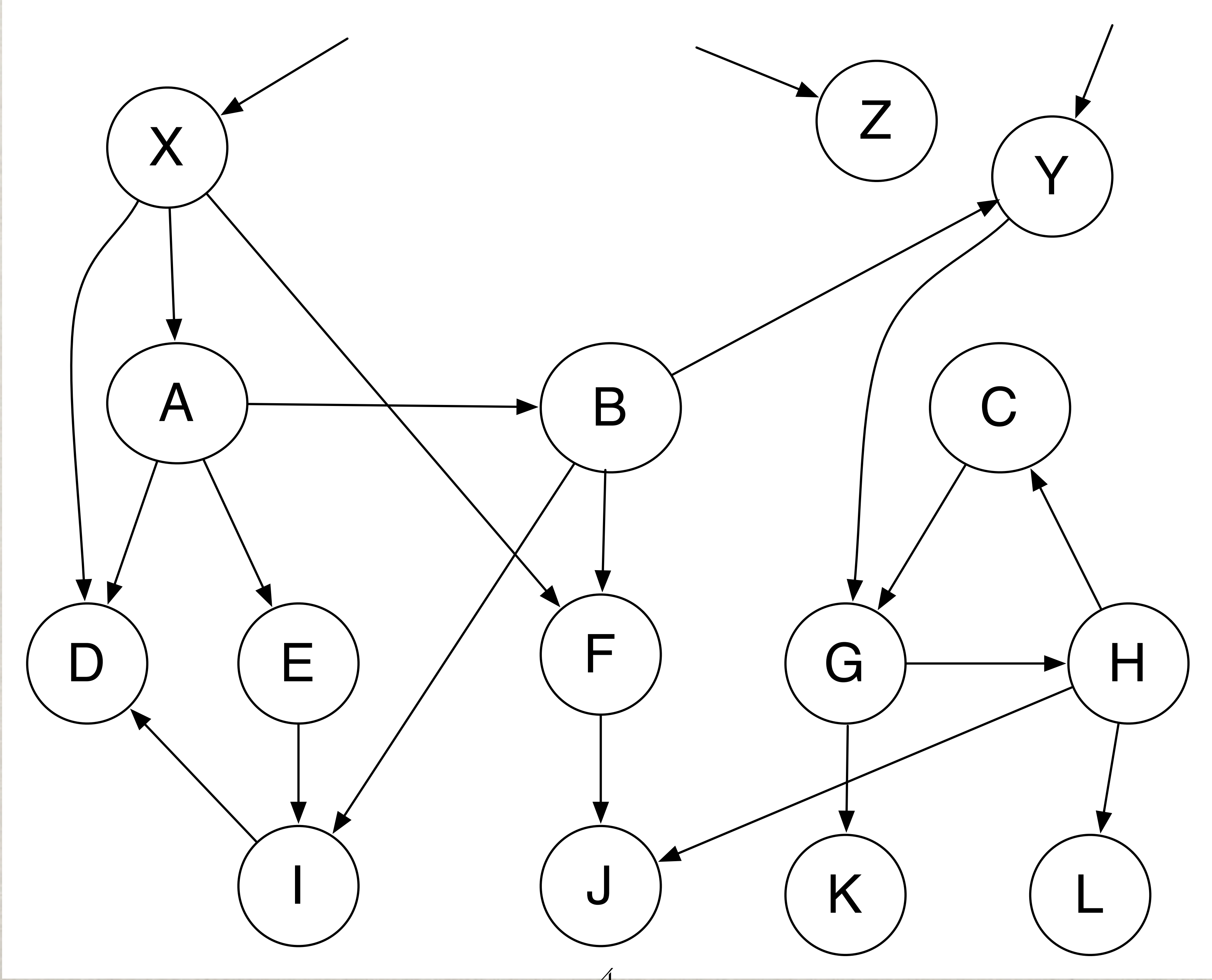
- Use more memory than needed.

- Make OOP languages unsuitable for memory limited devices.

- **Existence of unused but referenced objects.**

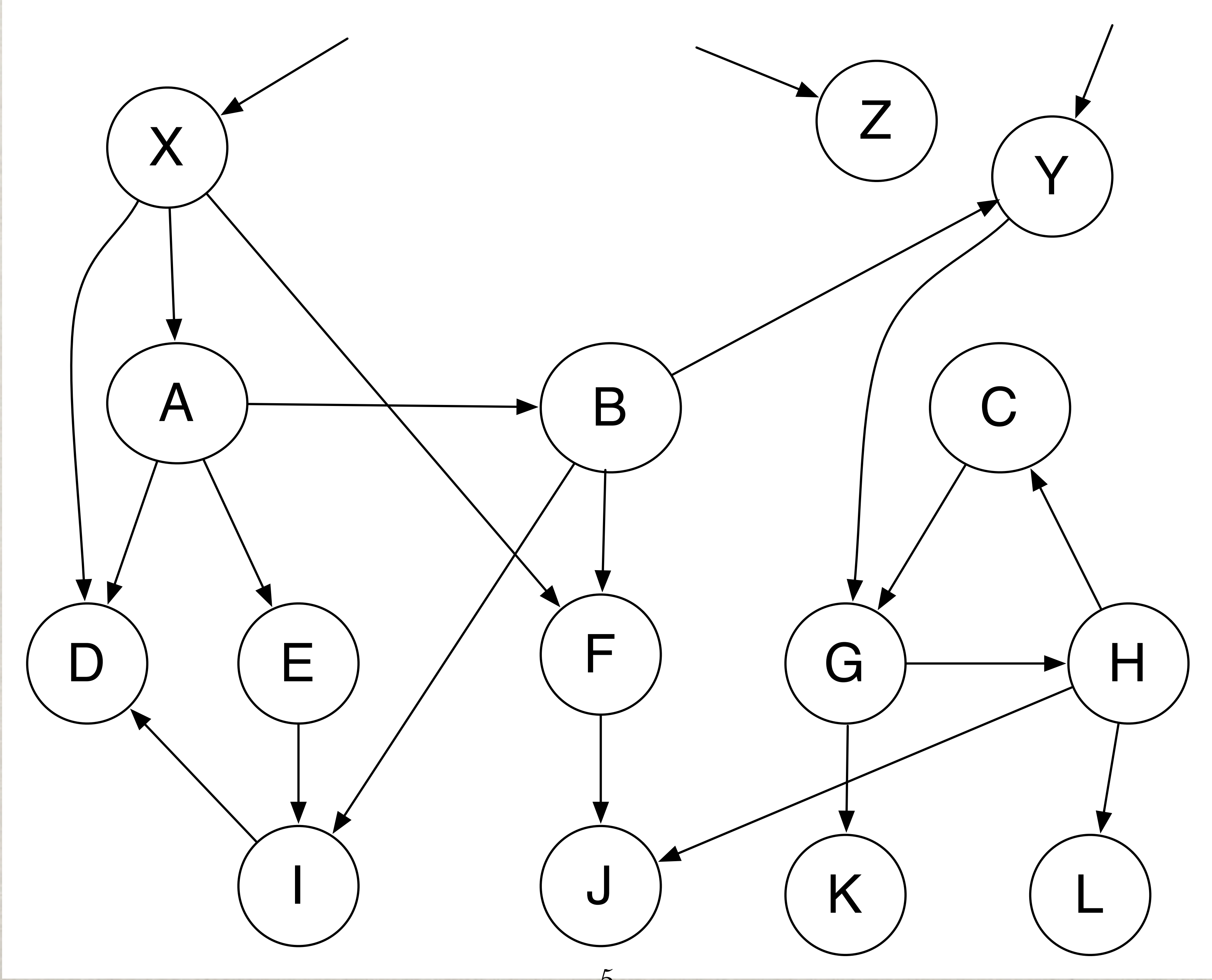
The context

In OOP primary memory is represented by an object graph



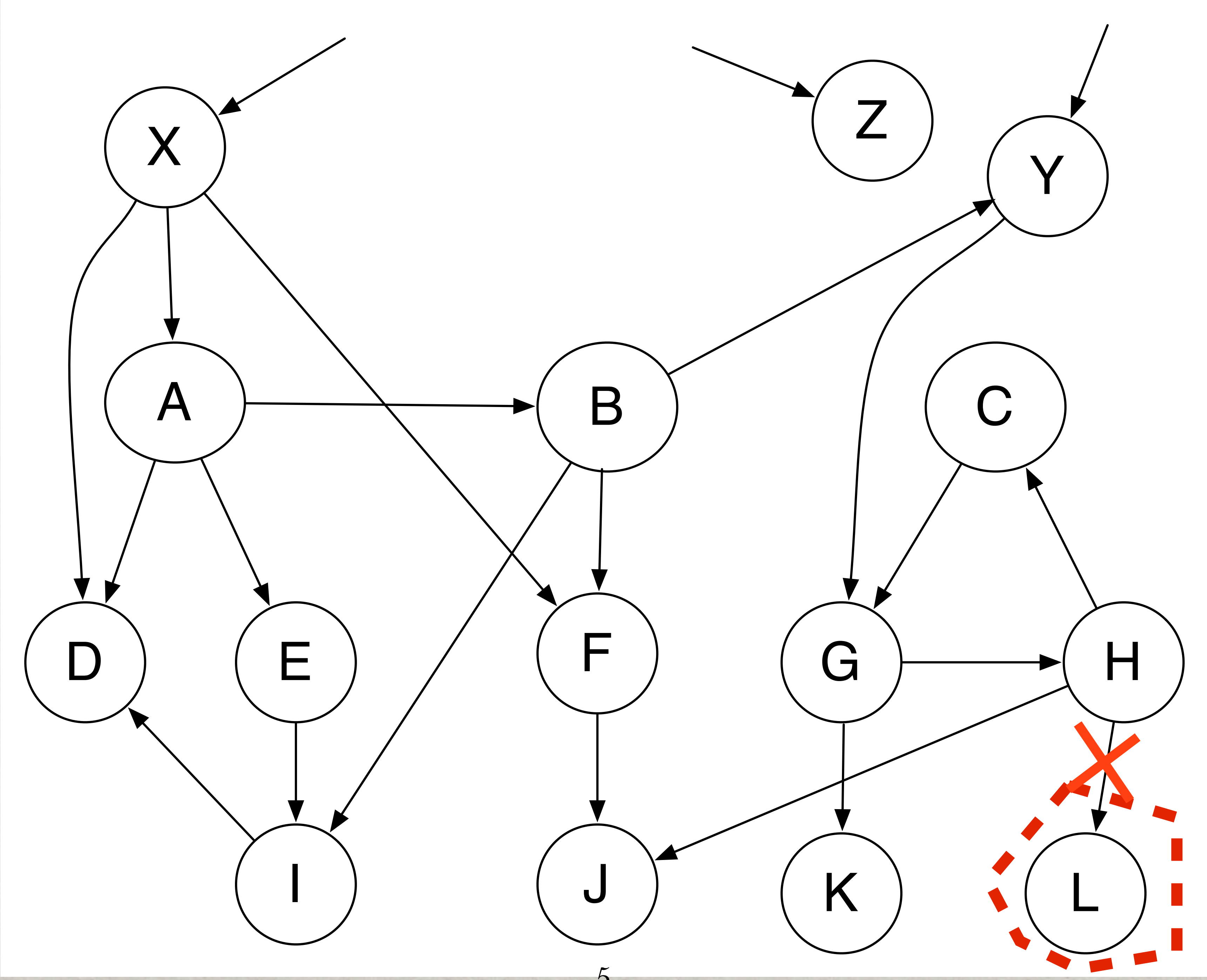
Garbage Collector

Only collects objects that nobody else points to.



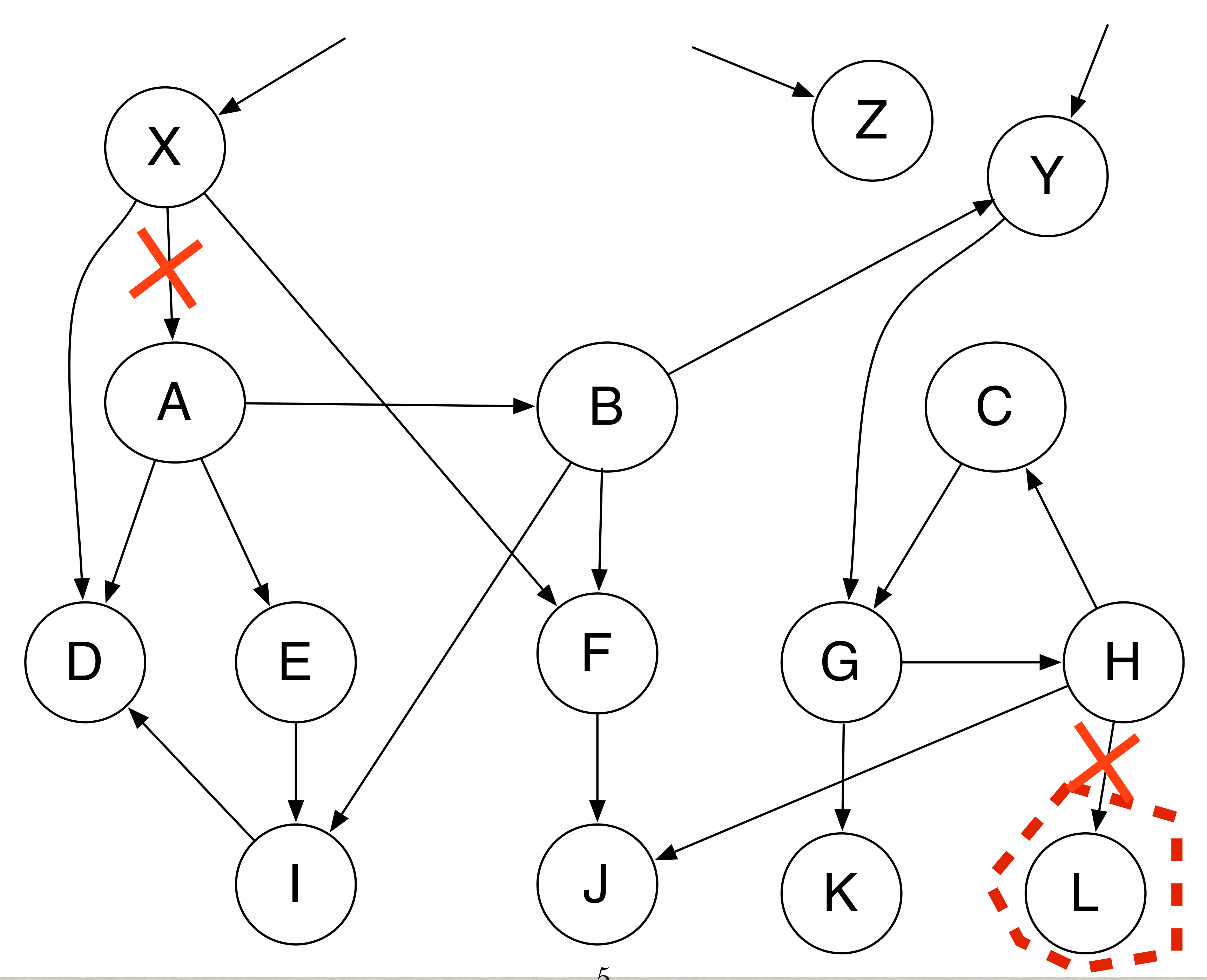
Garbage Collector

Only collects objects that nobody else points to.



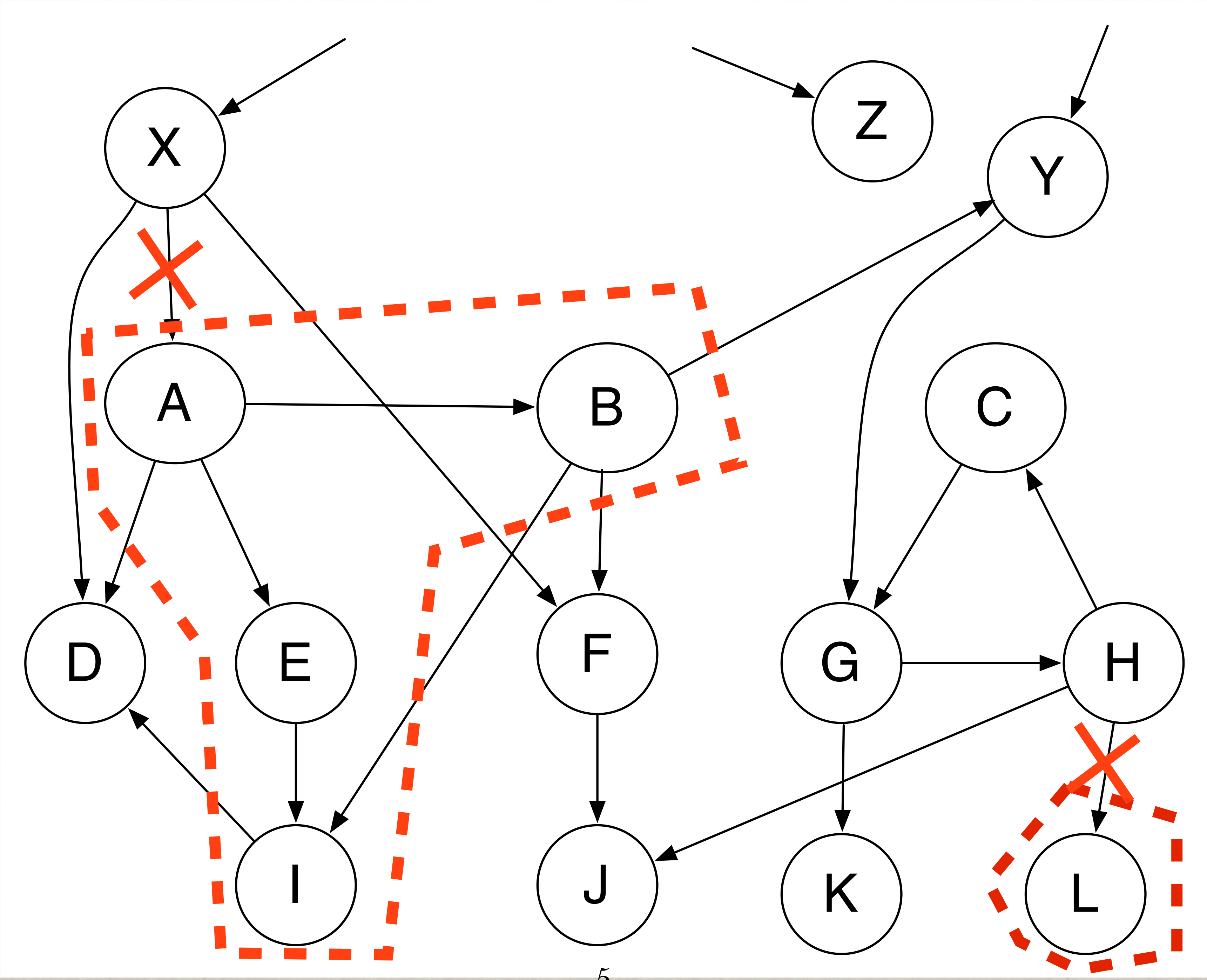
Garbage Collector

Only collects objects that nobody else points to.

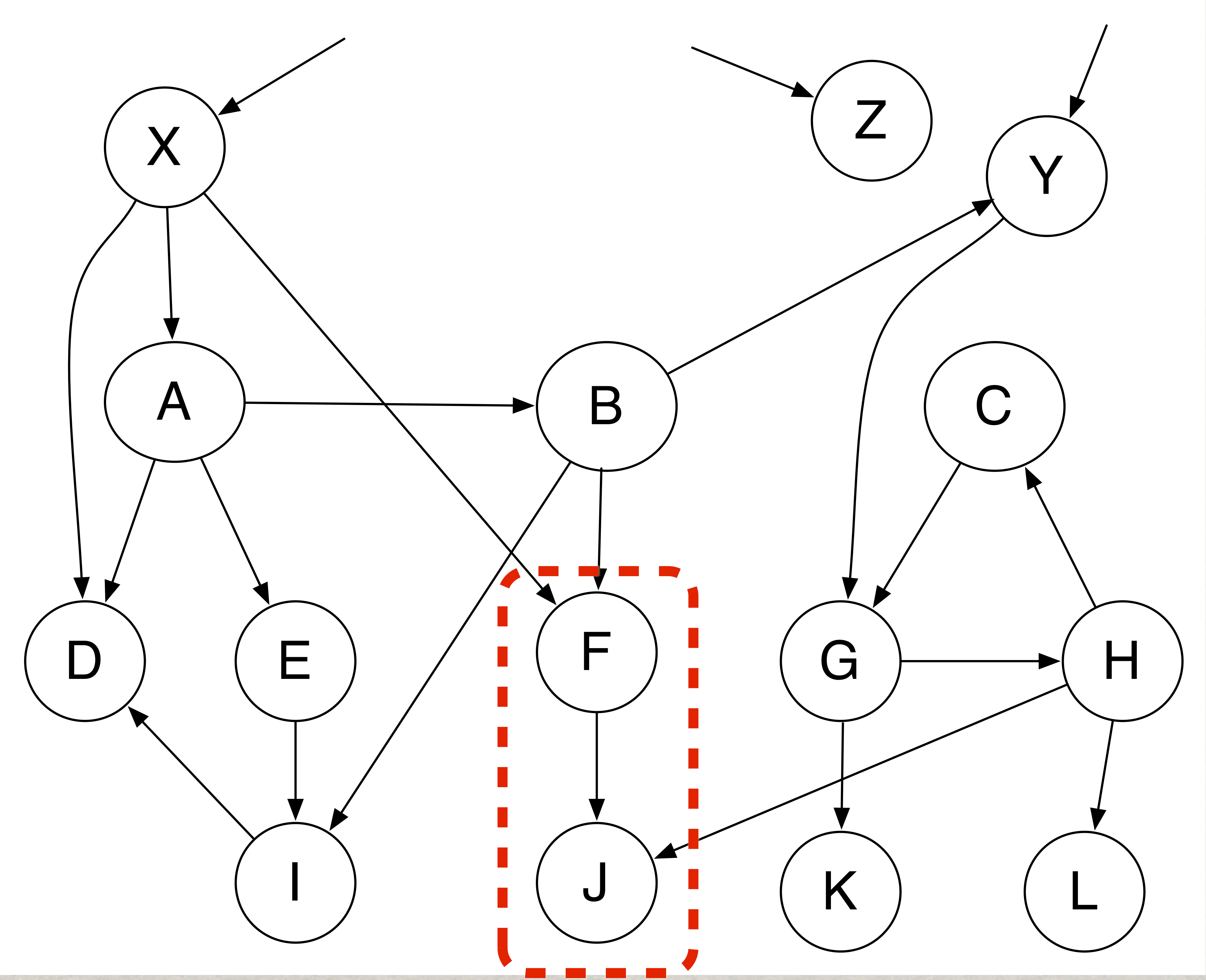


Garbage Collector

Only collects objects that nobody else points to.



But...what happens with referenced yet unused objects?



Idea

- Swap out (not remove) unused objects to disk.
- Automatically load them back when needed.

Related work

- Large object oriented memory (LOOM).
- Melt - Supporting memory leaks.
- ImageSegments.

But...no one solves all the problems

Main challenges

- Not to use more memory than the one released by swapping.
- Low overhead penalty.
- Group objects in an smart way.

Key aspects

- Mark and trace unused/used objects at runtime.
- The usage of proxies.
- Group unused objects (**subgraphs**).

Why we need to group objects?

- Because if we replace each object by a proxy, we release little memory.
- We want to replace a whole group by one or a few proxies.

Why subgraphs?

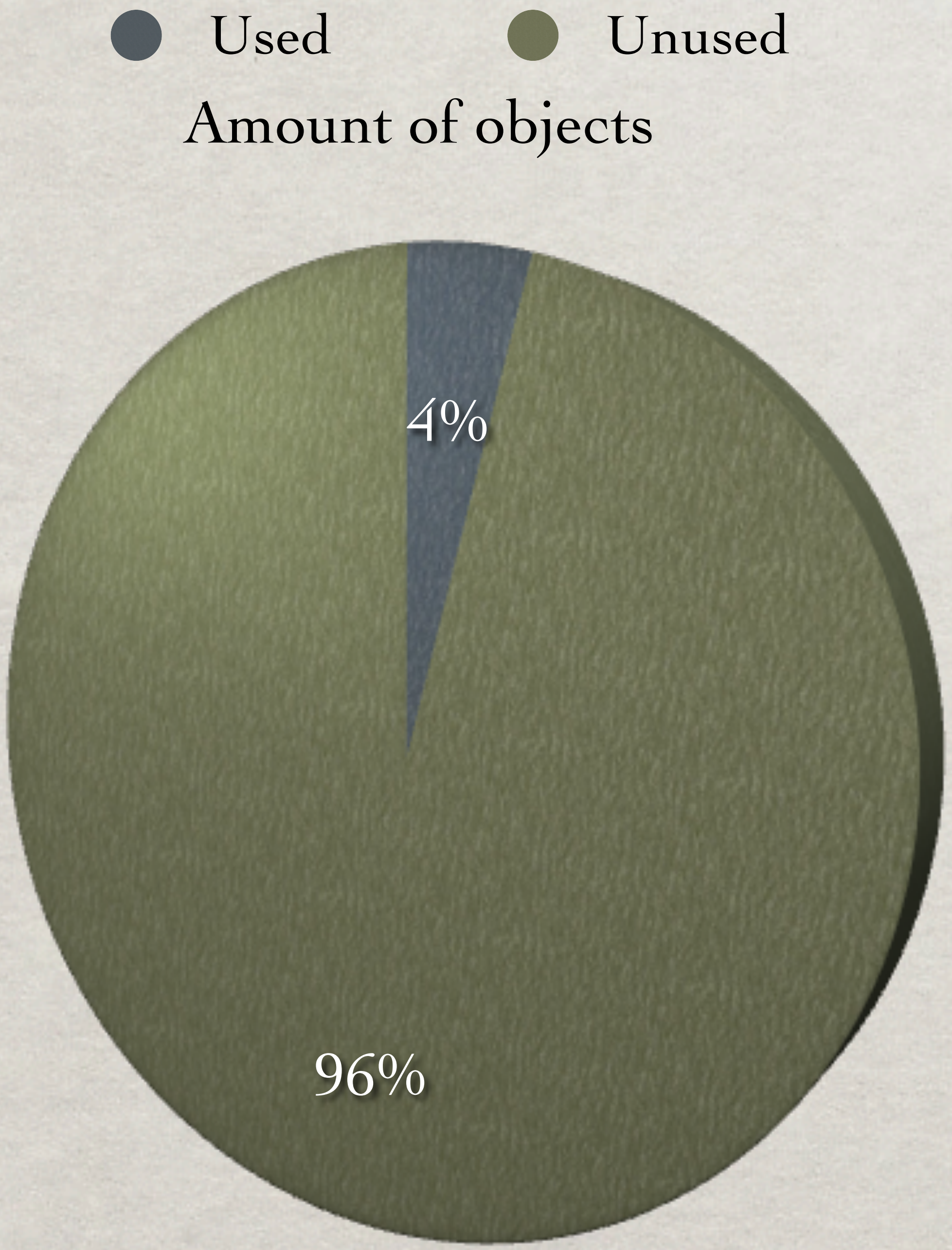
- Group of objects that are used (or not used) together.

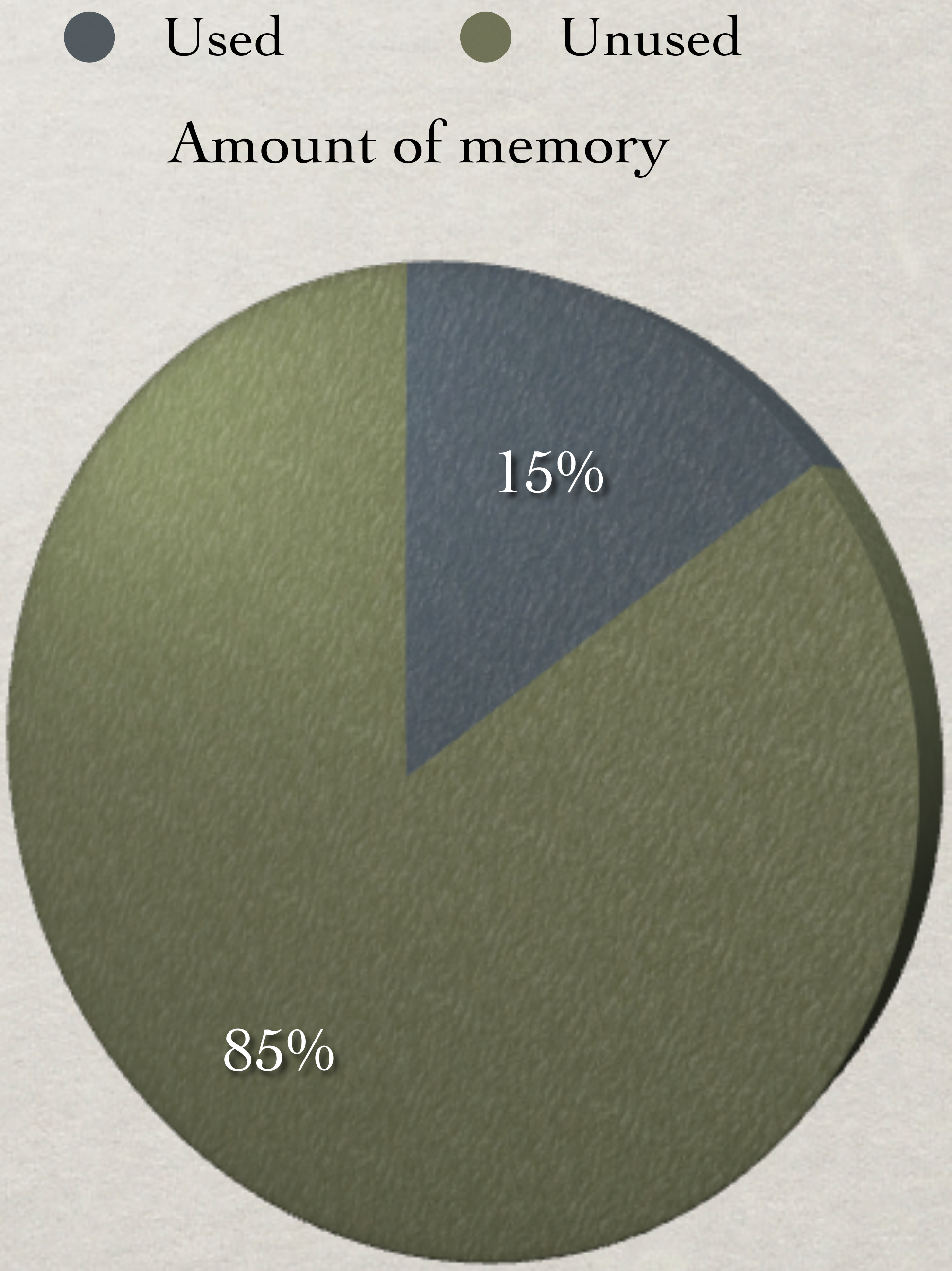
- We need few proxies (for the roots) for several objects.

Experiments done

- Modify Smalltalk VM to mark and trace objects usage.
- Visualize objects and memory usage.
- Take statistics from different scenarios.

DEPLOYED WEB APPLICATION EXAMPLE


Amount of objects


Amount of memory

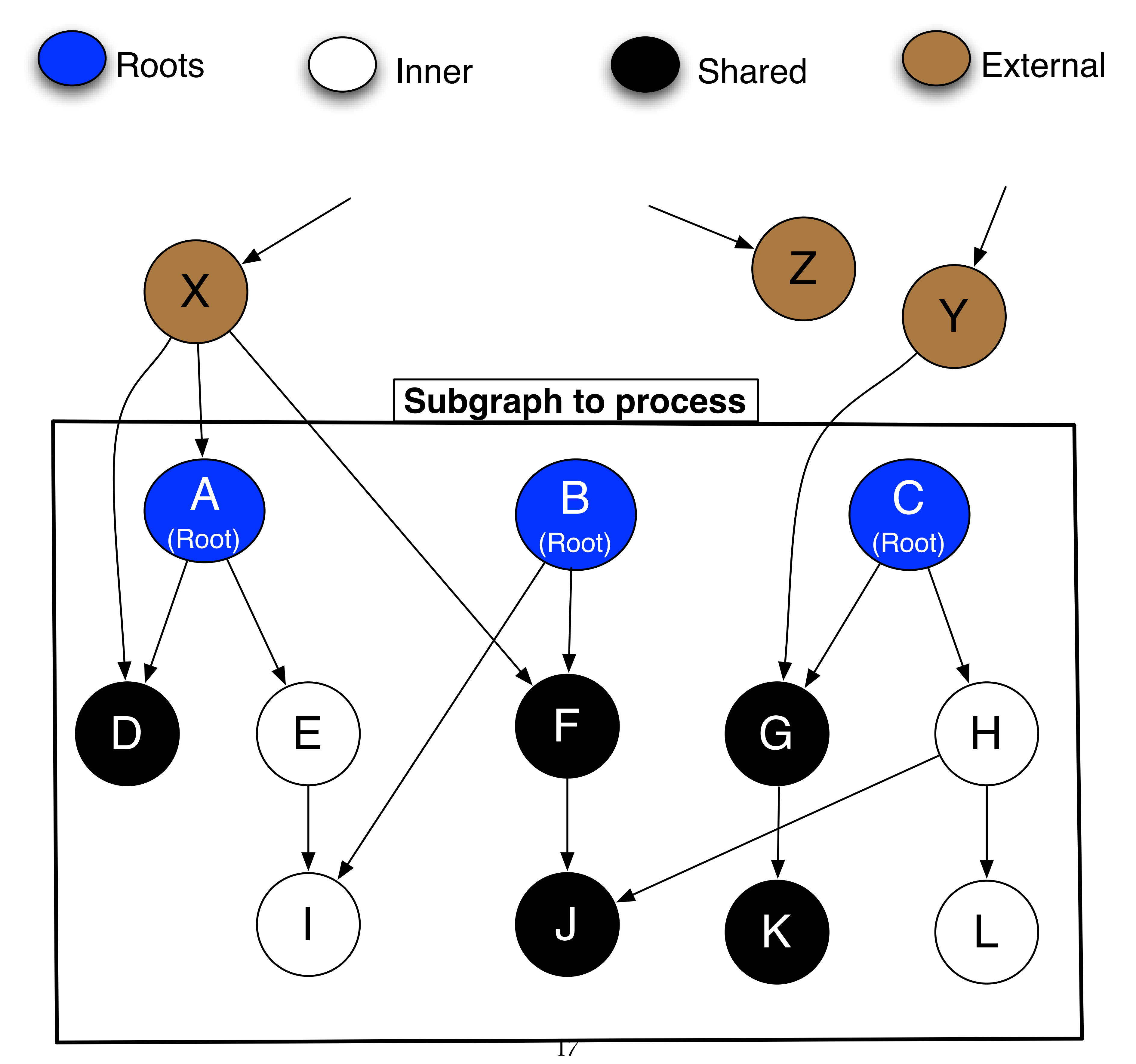
Swapping steps and challenges

1. Identify sets of objects and serialize them. Problems: cycles, **speed**, etc.

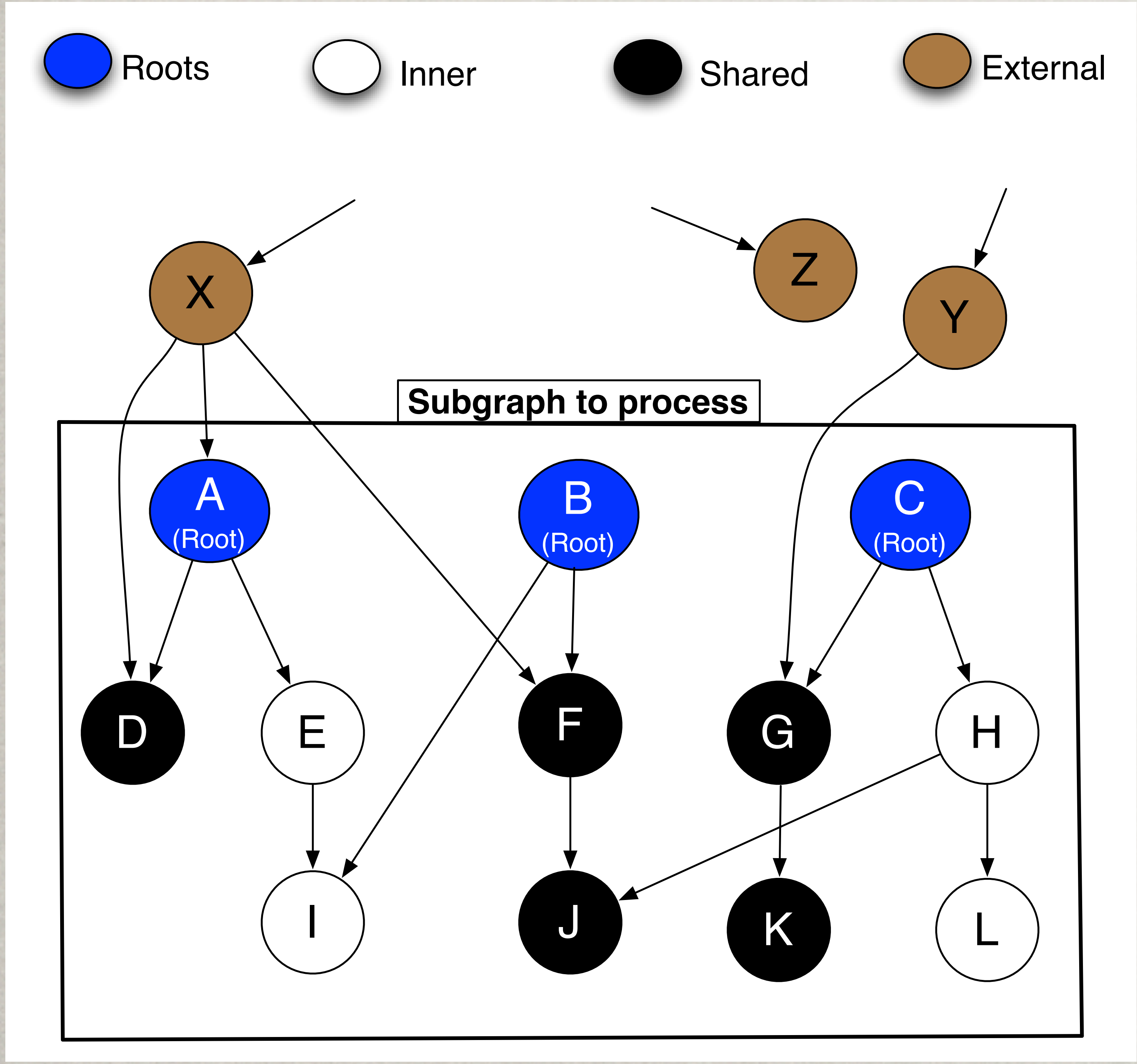
2. Write the serialized objects into a file. Problems: file format, encoding, **speed**, etc.

3. Load the objects from a file. Problems: class reshape, avoid duplicates, **speed**, etc.

Subgraphs


Roots
Inner
Shared
External
X
Z
Y
Subgraph to process
A
(Root)
B
(Root)
C
(Root)
D
E
F
G
H
I
J
K
L

More problems



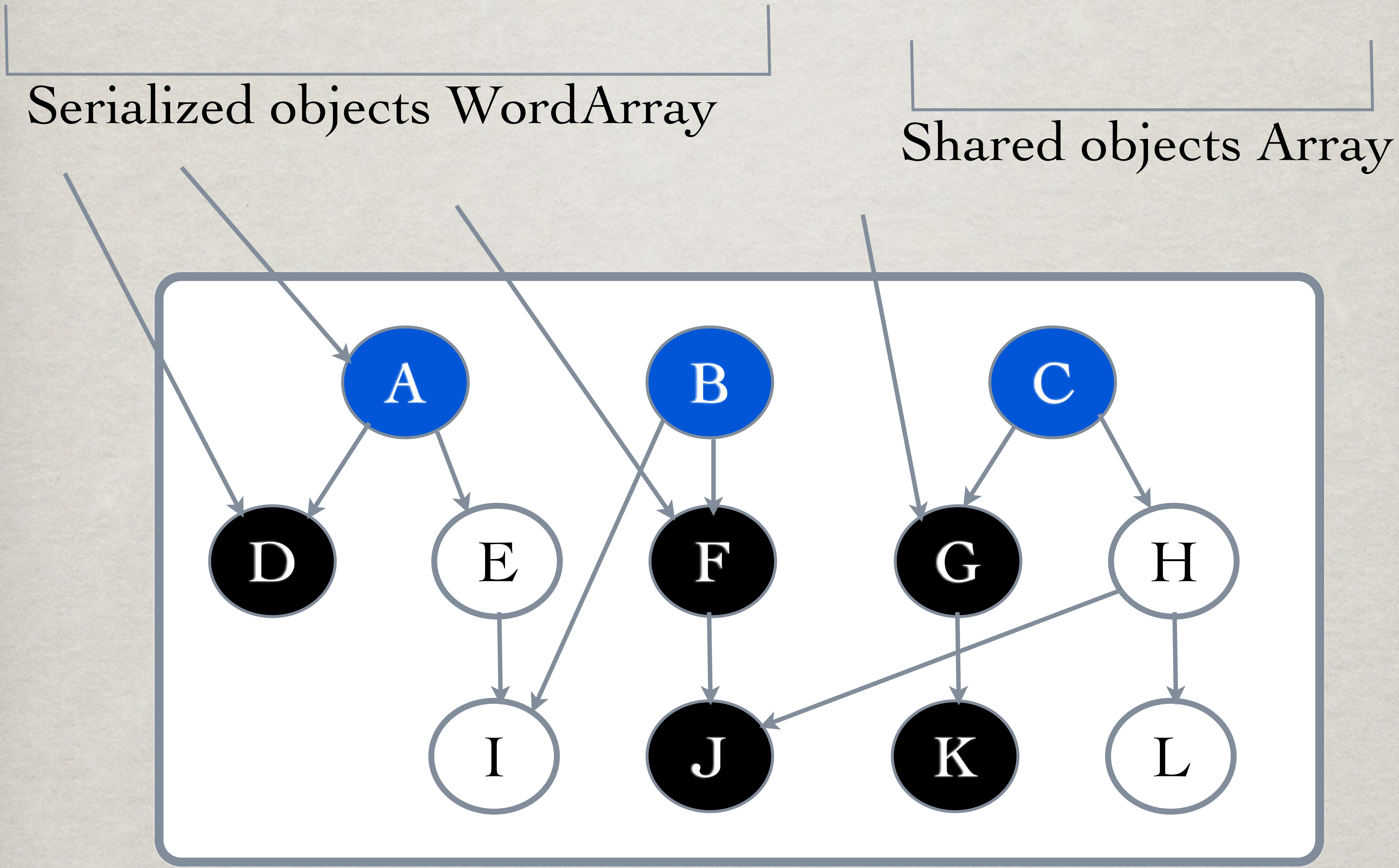
- Should shared objects be included or not?
- GC moves objects.
- Pointers update.
- Class changes.
- Recreate and reinitialize objects.
- Code executed after loading.

ImageSegment

ImageSegment basis

- Only write/swap roots and inner objects.
- Shared objects are NOT swapped.
- Keep an array in memory for the shared objects.
- Update object pointers to point to a relative address inside the arrays (offset).
- Roots are replaced by proxies.
- Uses GC facilities to detect shared objects.

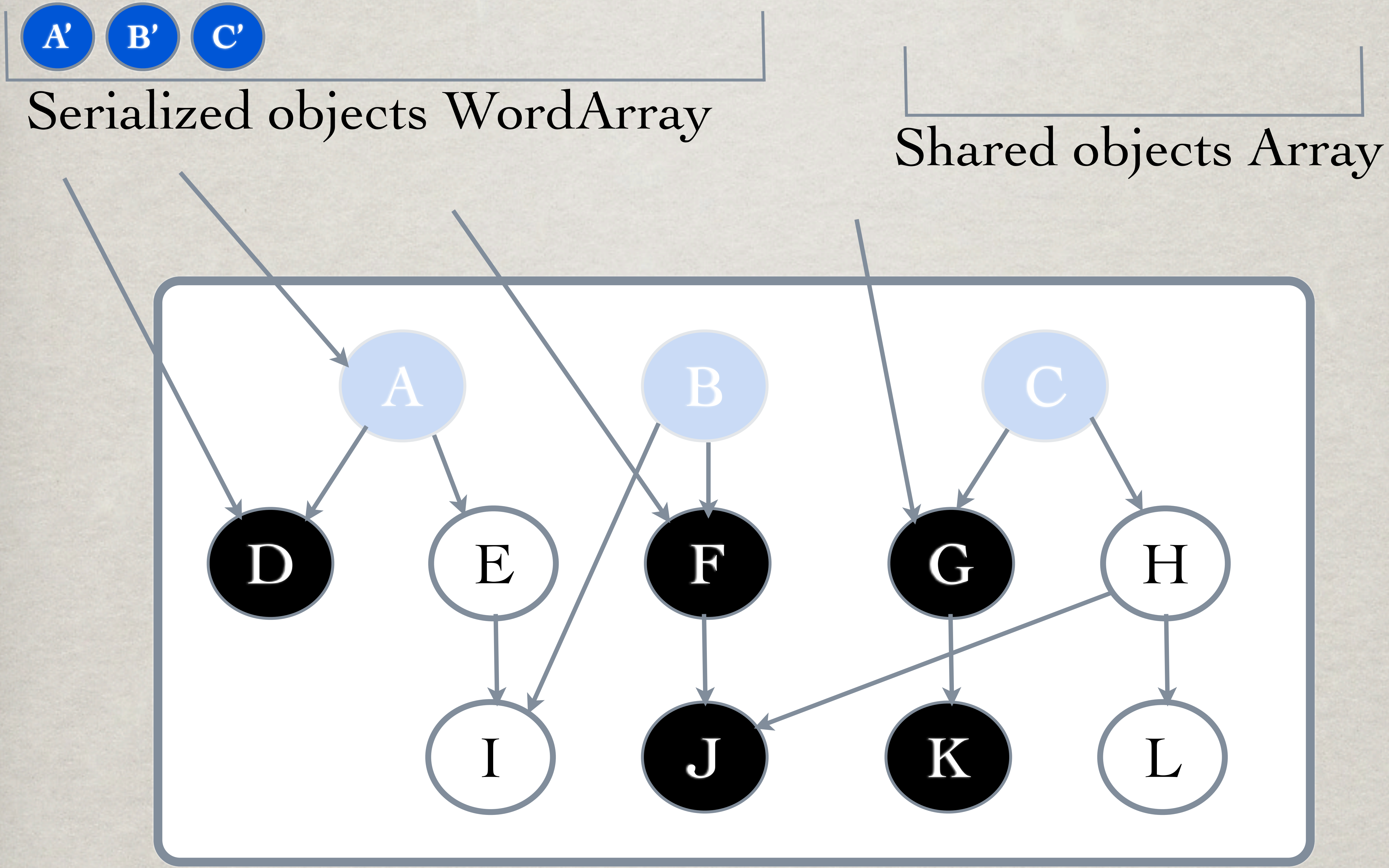
Subgraph traverse



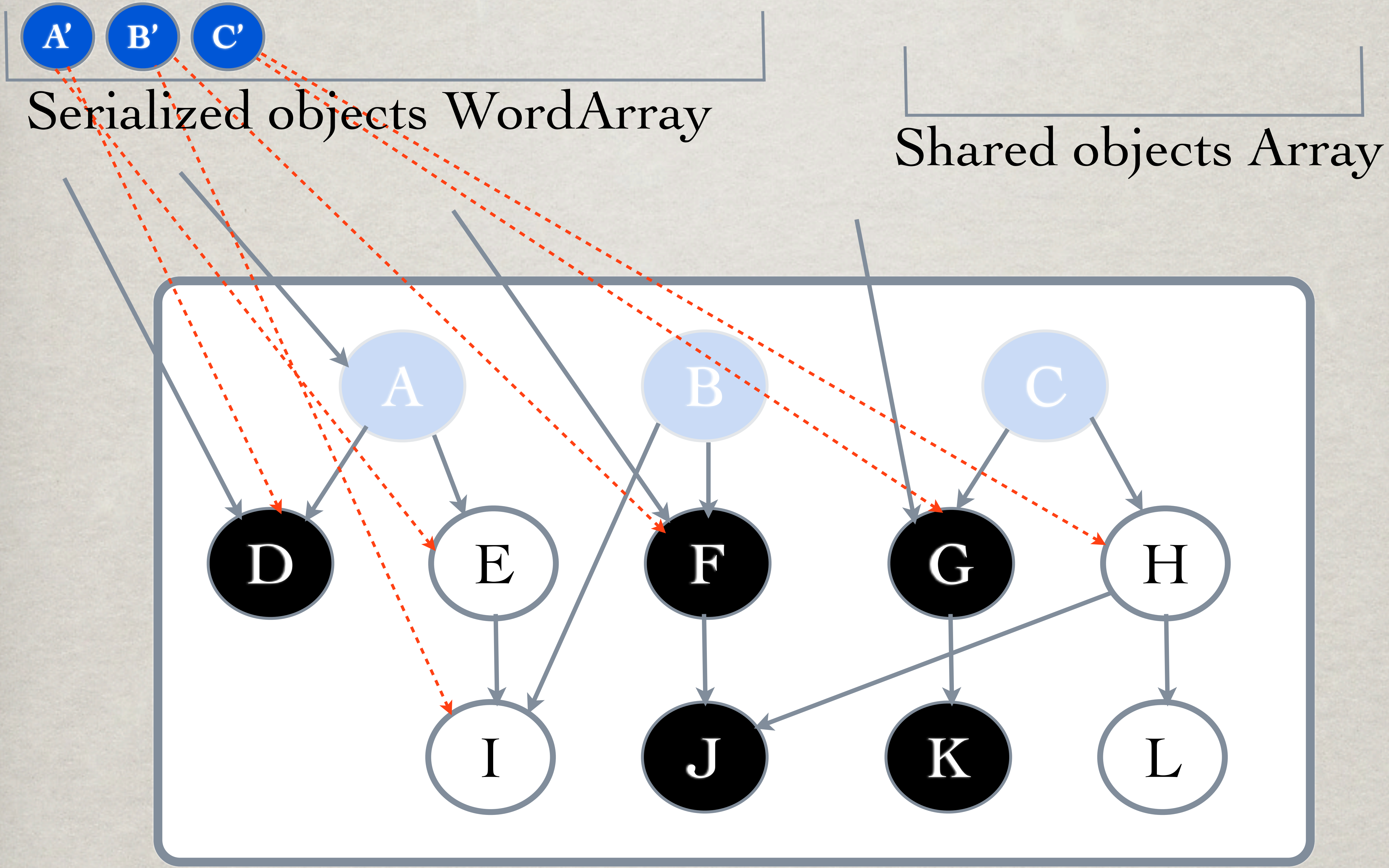
Serialized objects WordArray

Shared objects Array

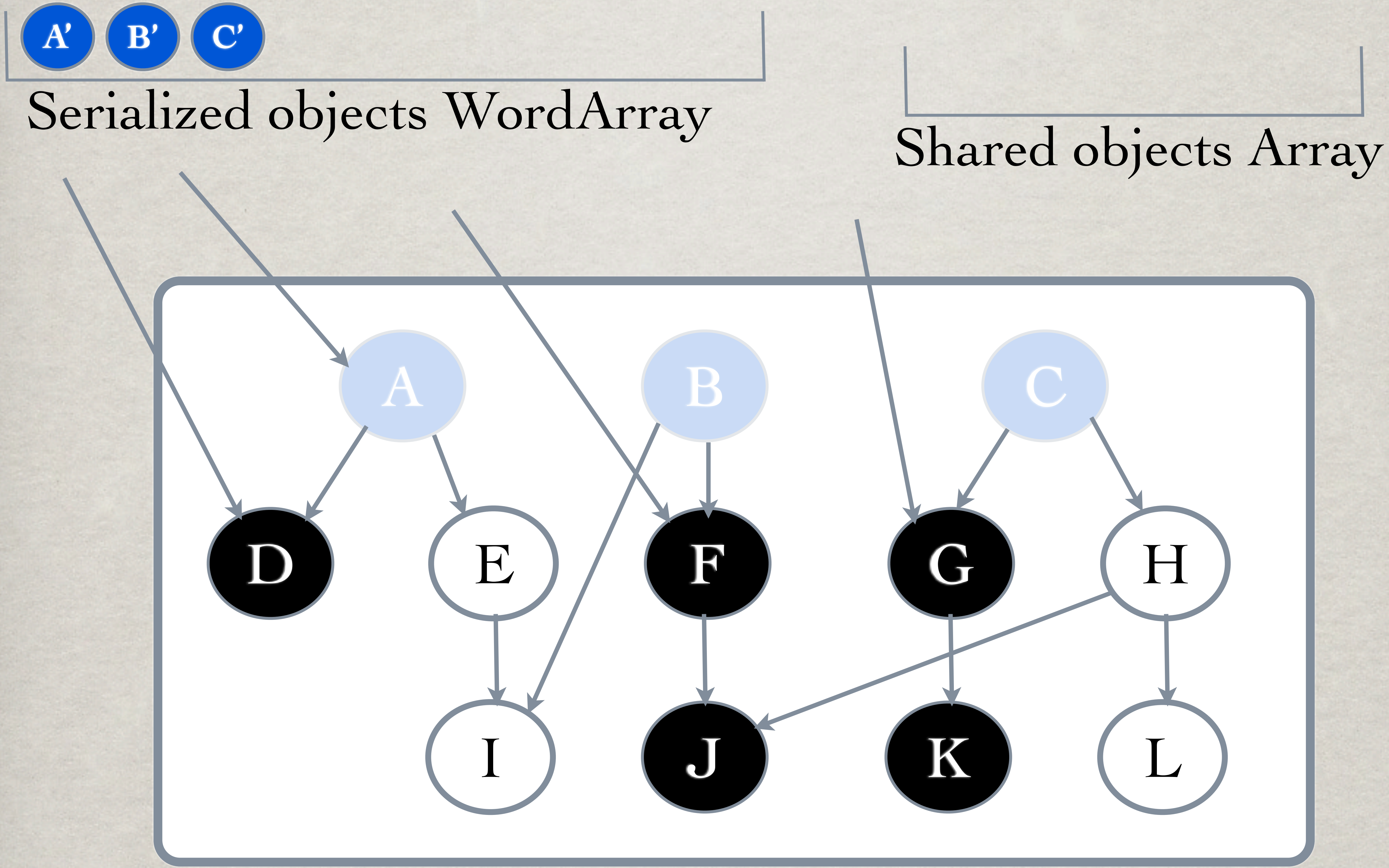
Subgraph traverse



Subgraph traverse



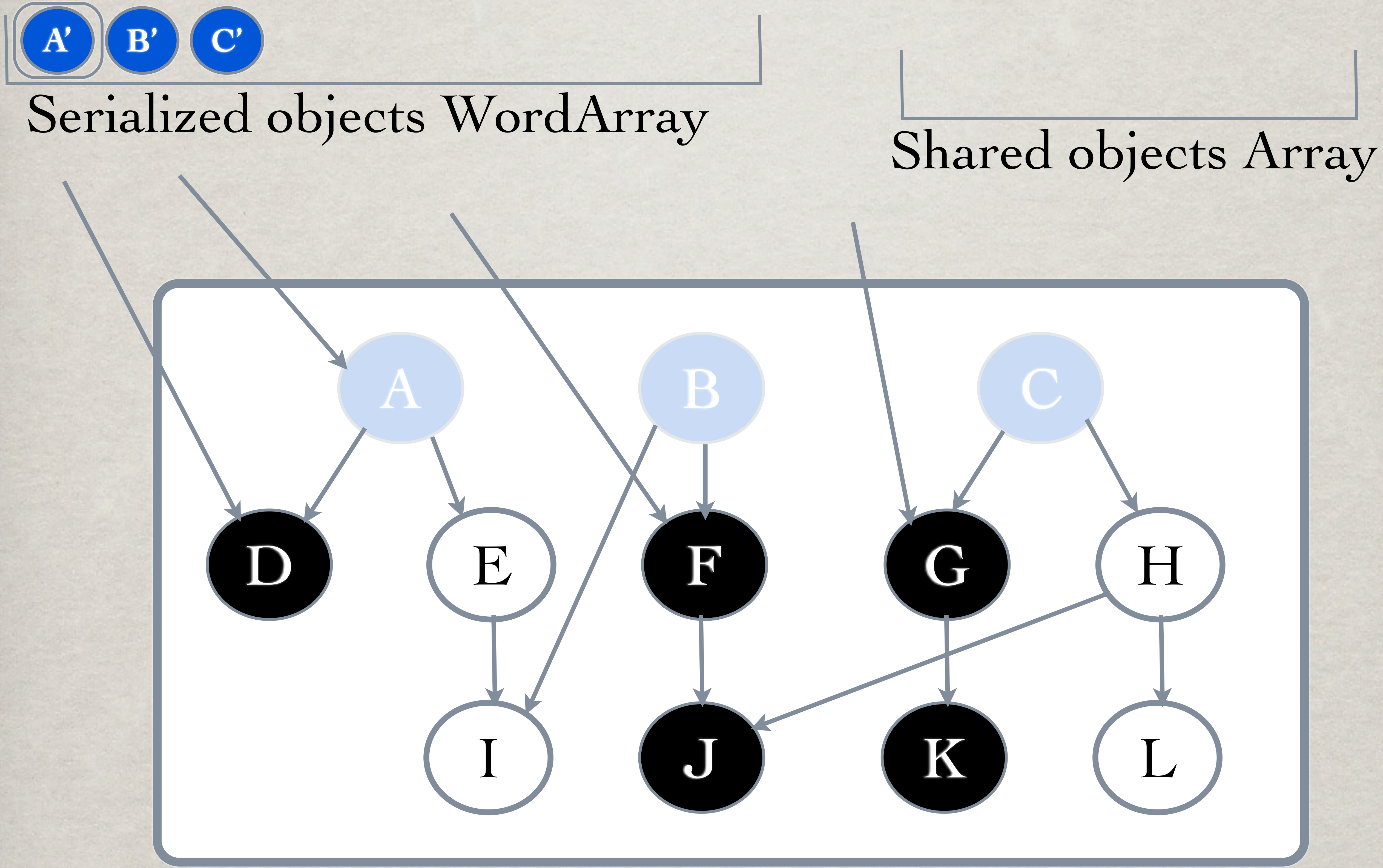
Subgraph traverse



Serialized objects WordArray

Shared objects Array

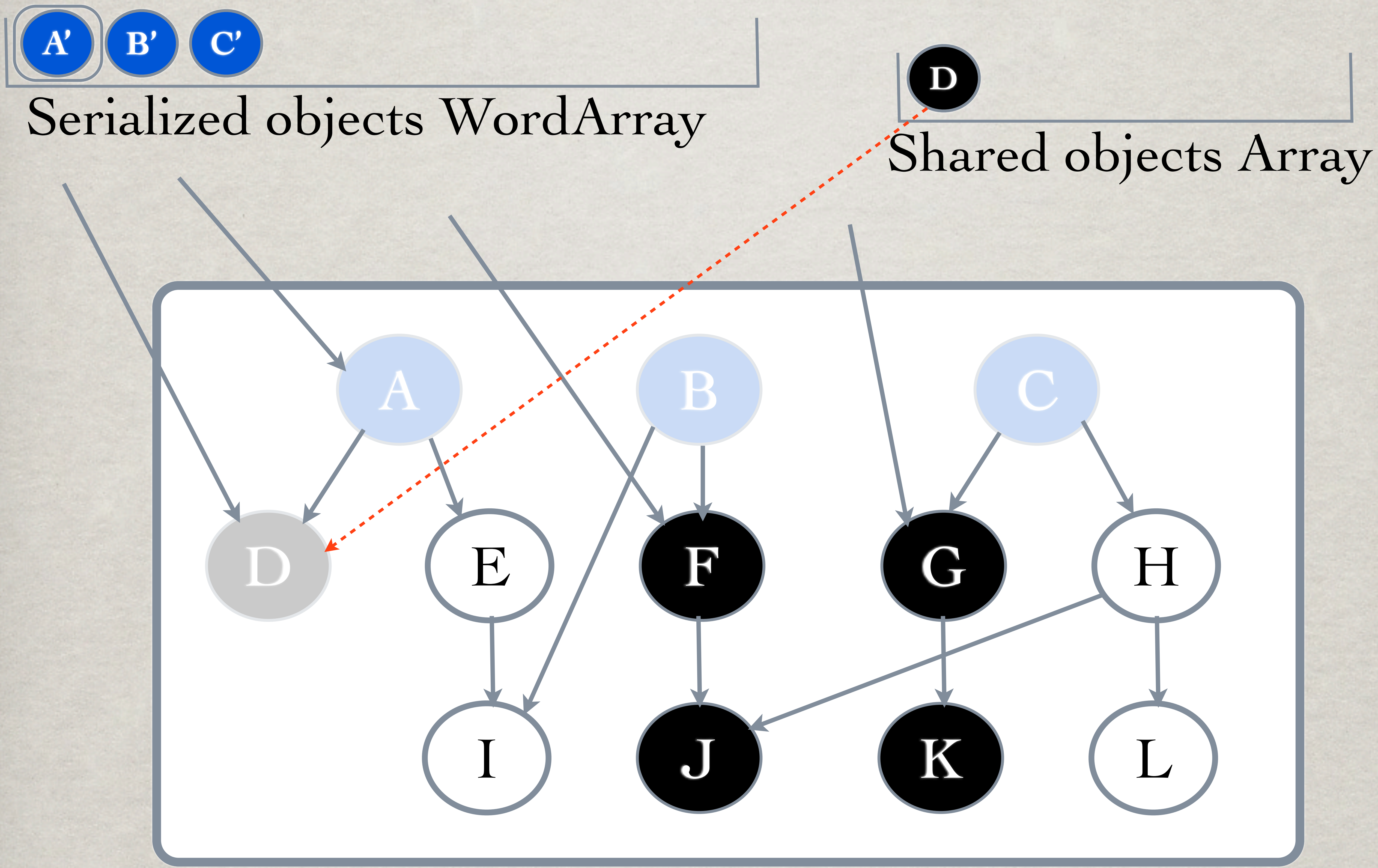
Subgraph traverse



Serialized objects WordArray

Shared objects Array

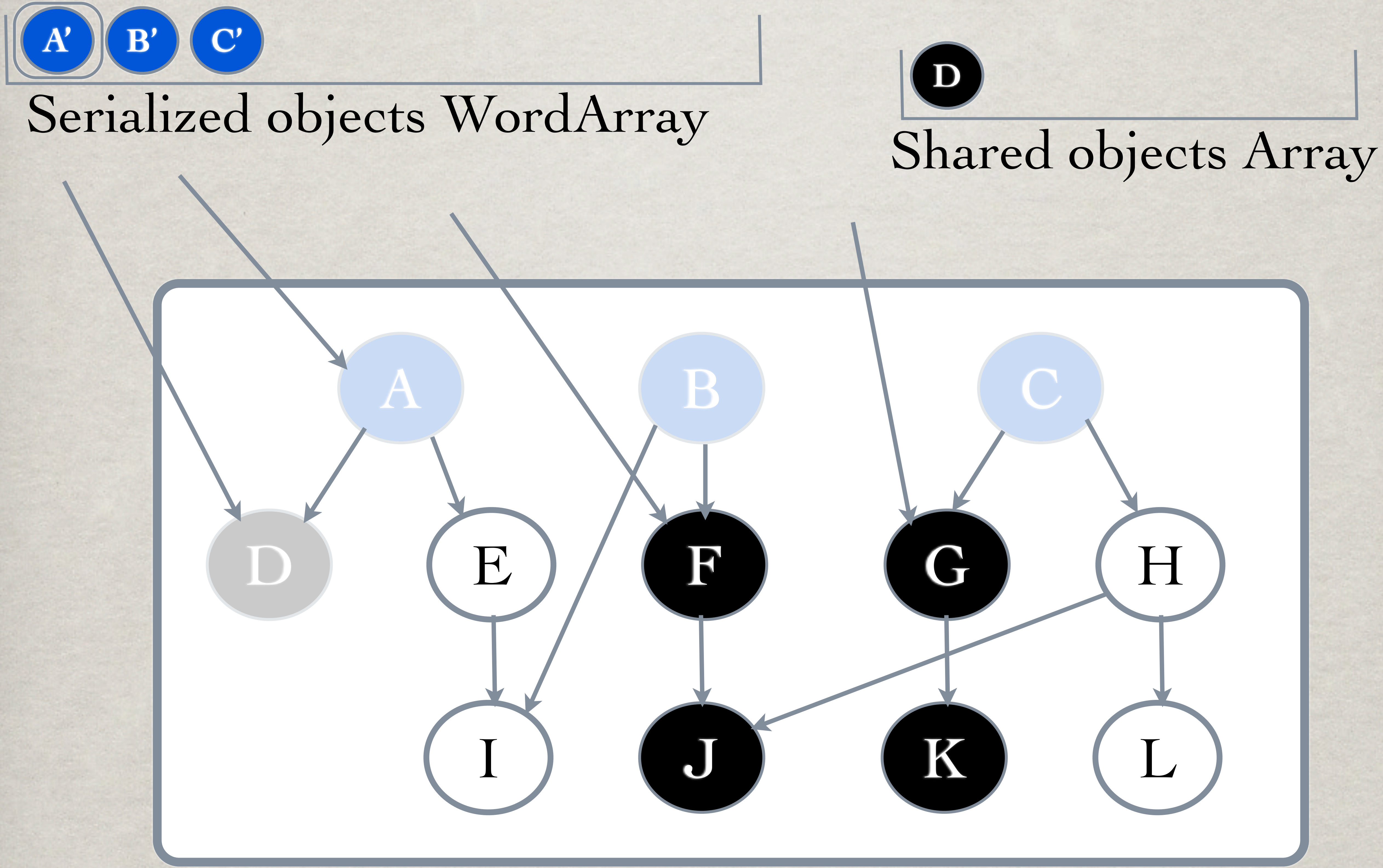
Subgraph traverse



Serialized objects WordArray

Shared objects Array

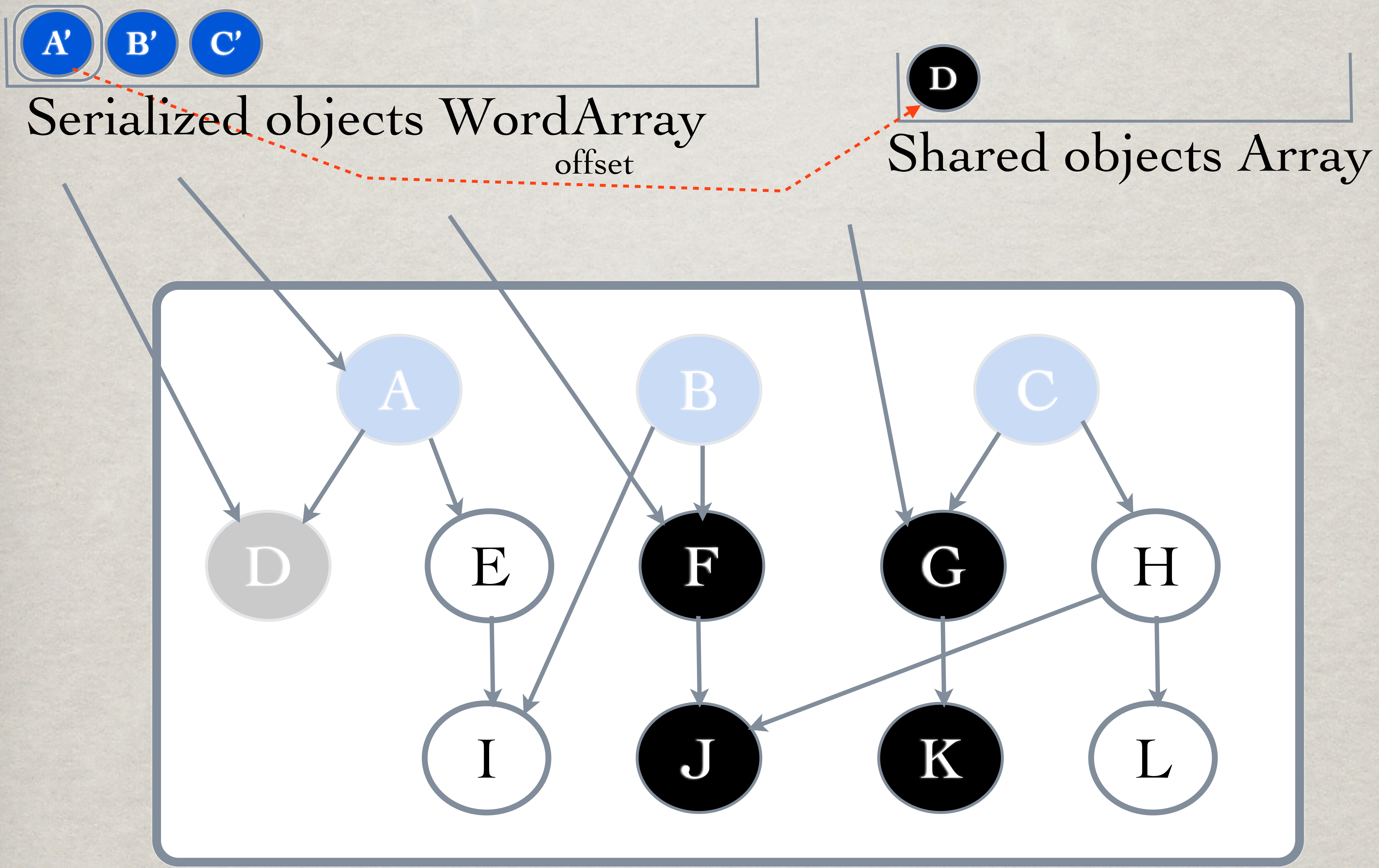
Subgraph traverse



Serialized objects WordArray

Shared objects Array

Subgraph traverse

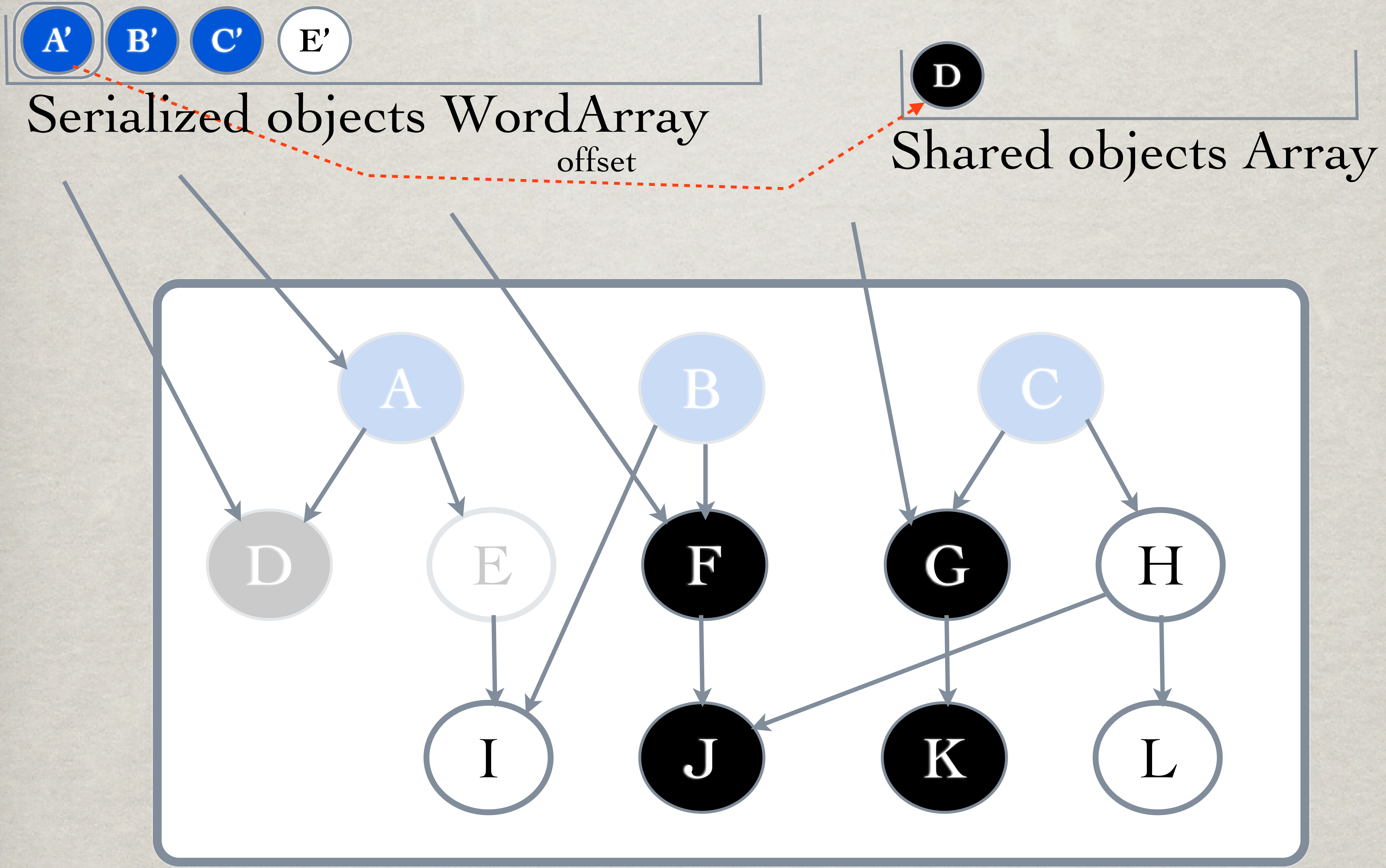


Serialized objects WordArray

offset

Shared objects Array

Subgraph traverse

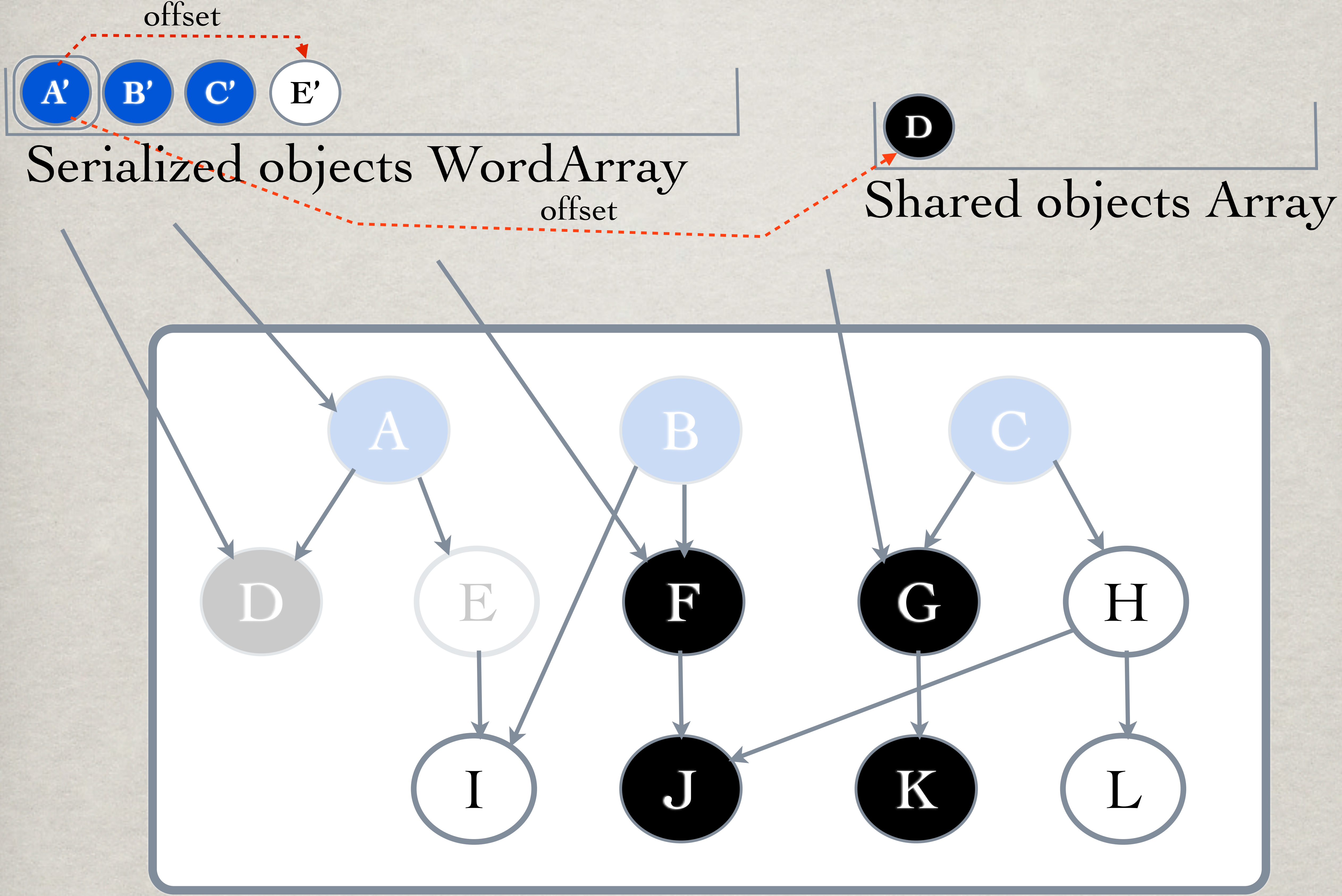


Serialized objects WordArray

offset

Shared objects Array

Subgraph traverse



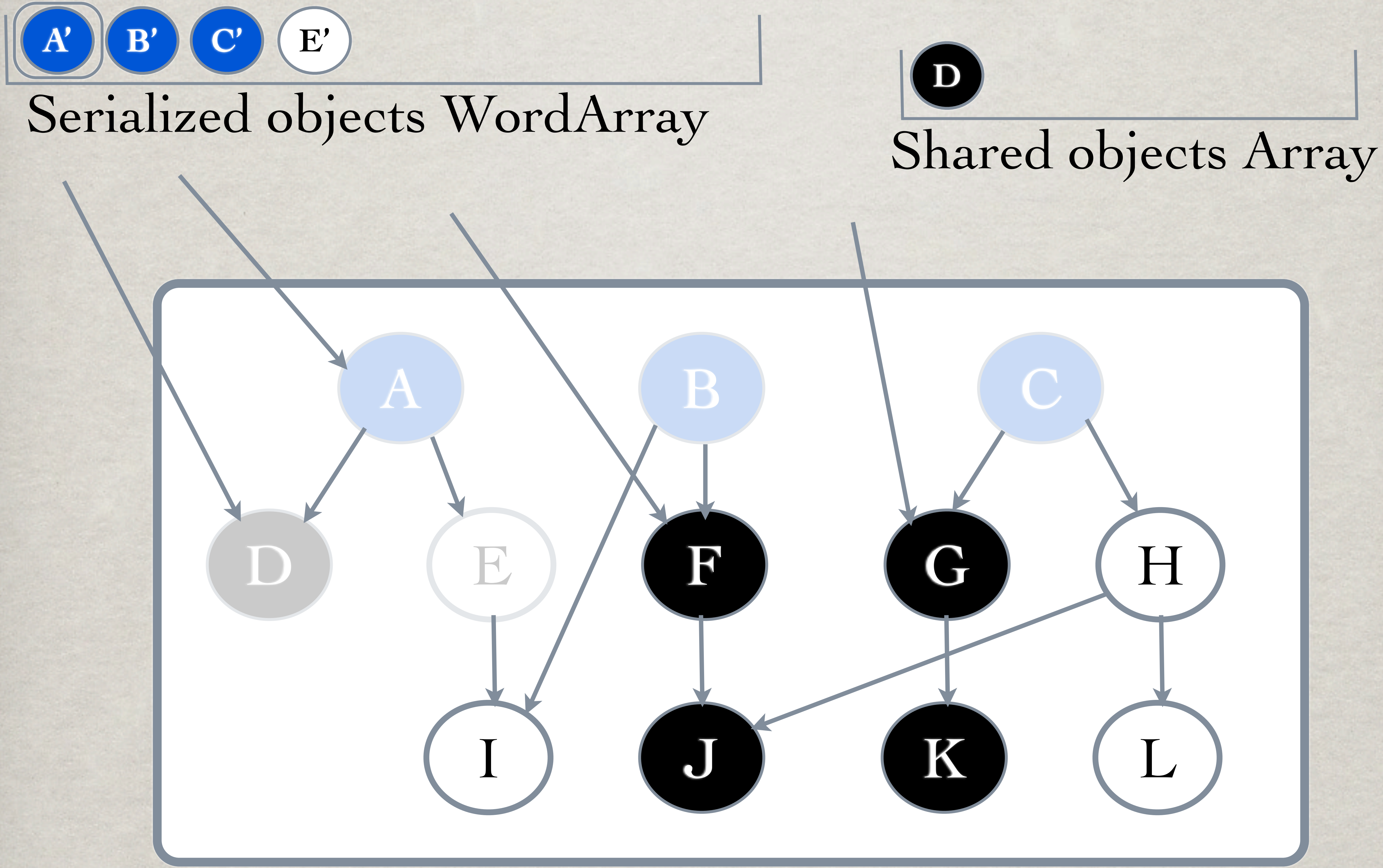
offset

Serialized objects WordArray

offset

Shared objects Array

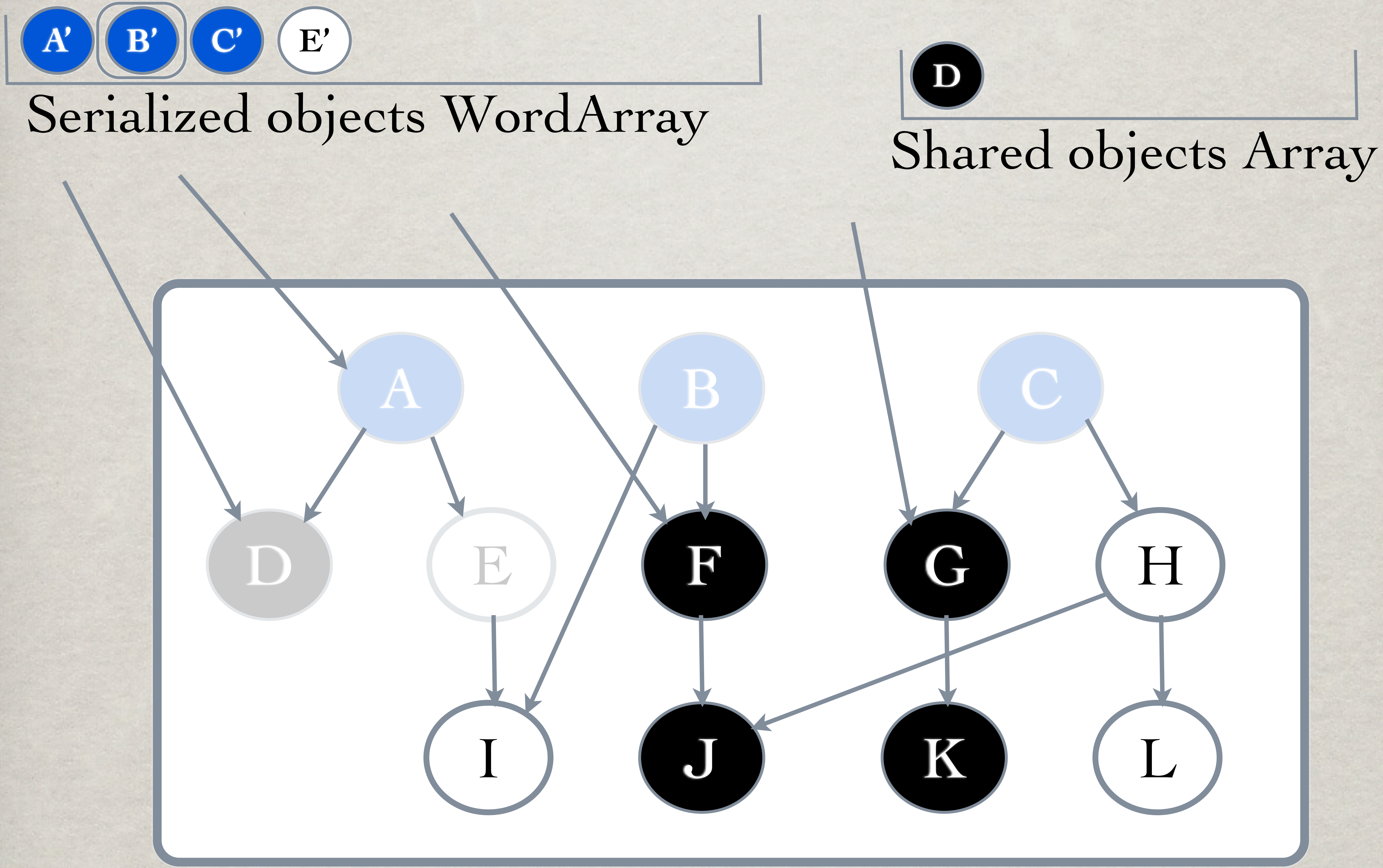
Subgraph traverse



Serialized objects WordArray

Shared objects Array

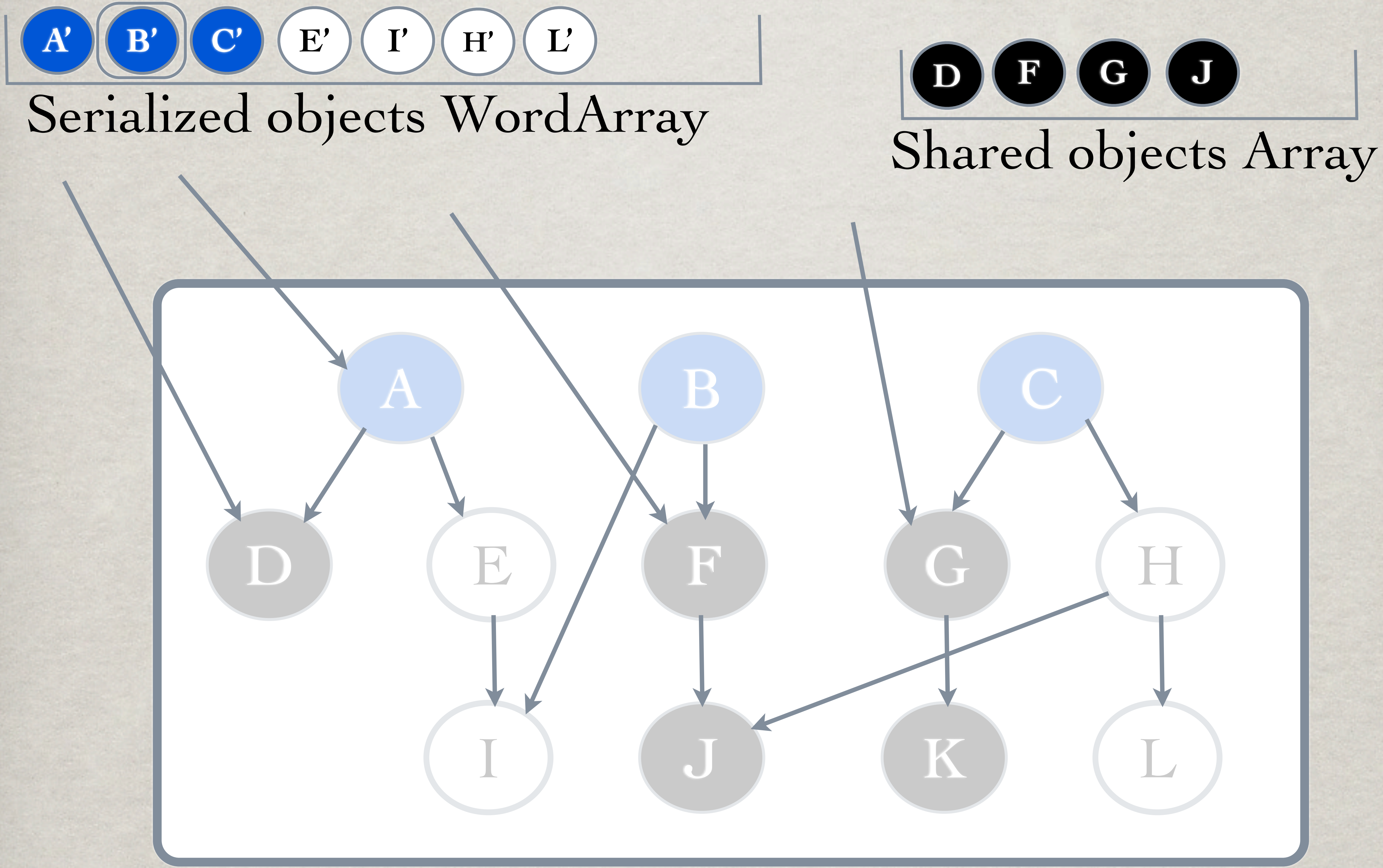
Subgraph traverse



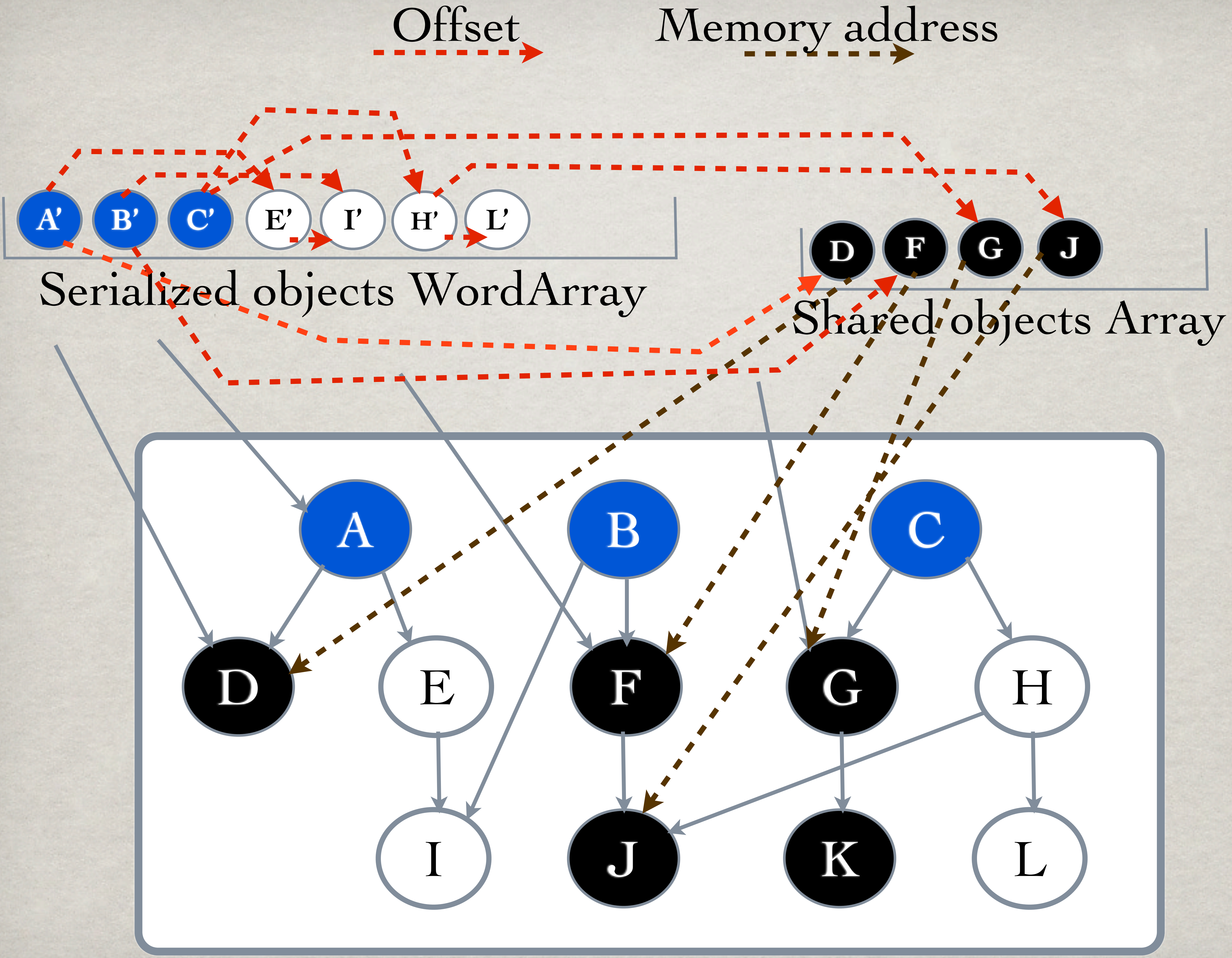
Serialized objects WordArray

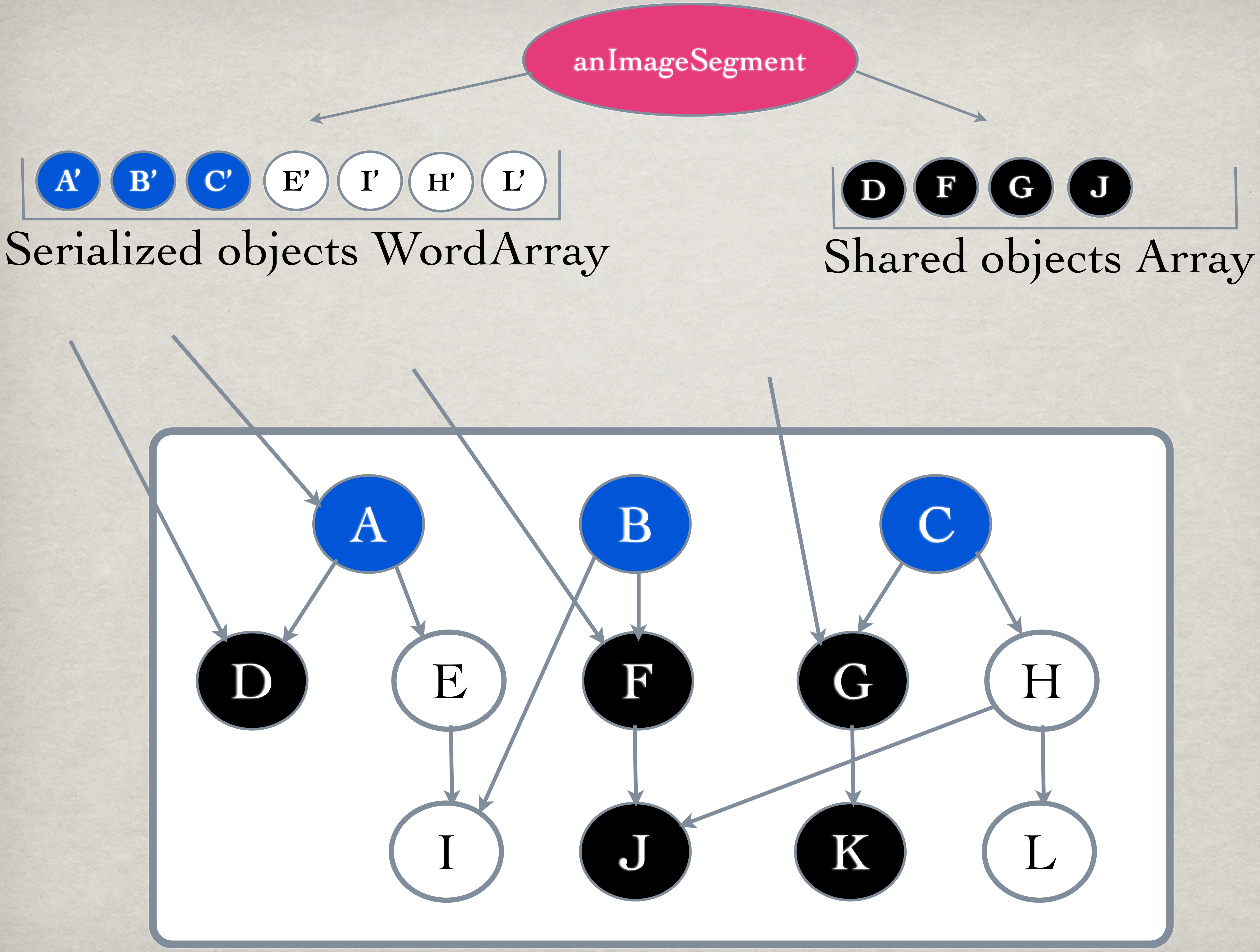
Shared objects Array

Subgraph traverse


Serialized objects WordArray

Shared objects Array

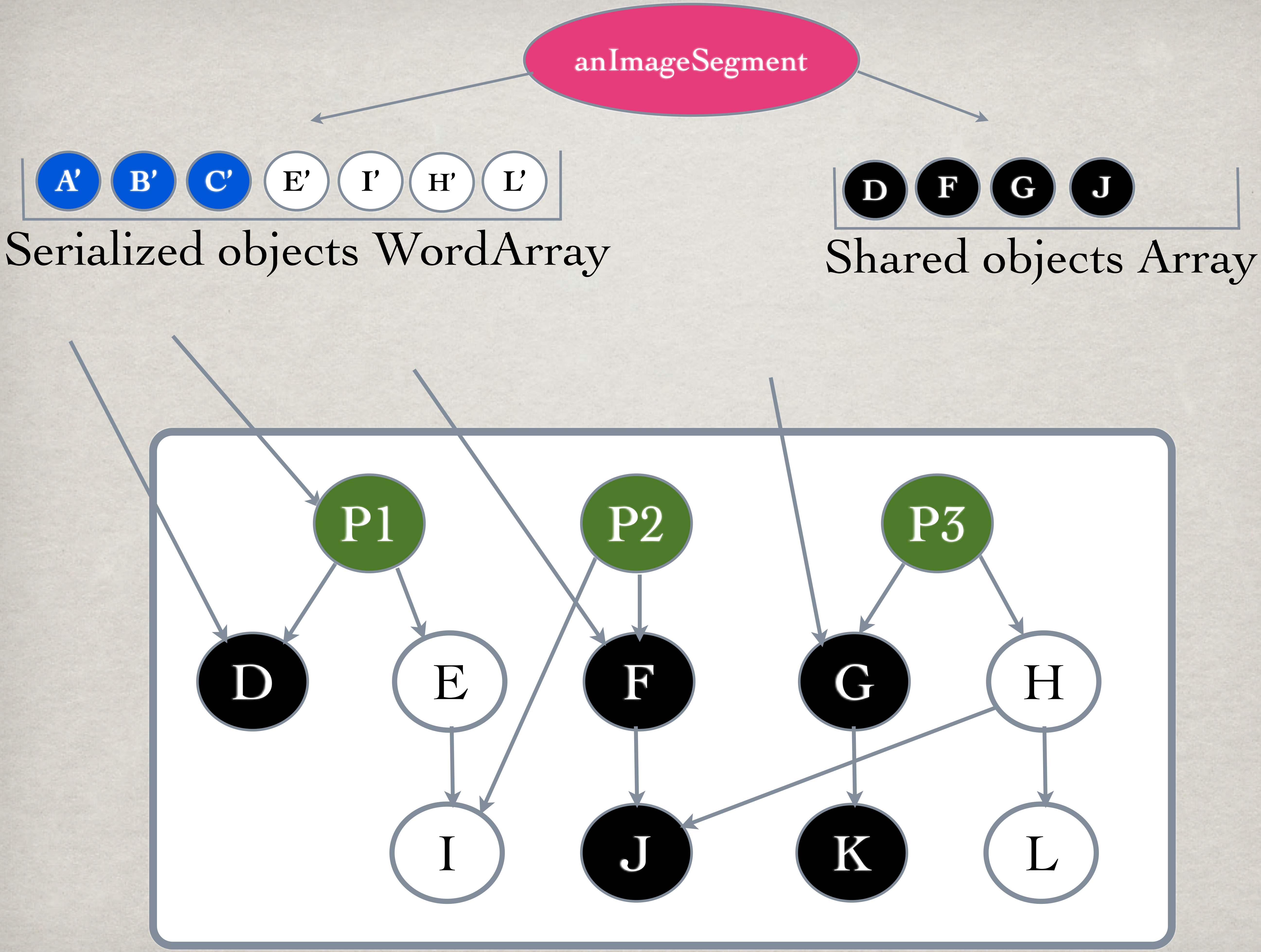

Offset
Memory address
A'
B'
C'
E'
I'
H'
L'
Serialized objects WordArray
D
F
G
J
Shared objects Array
A
B
C
D
E
F
G
H
I
J
K
L



anImageSegment

Serialized objects WordArray

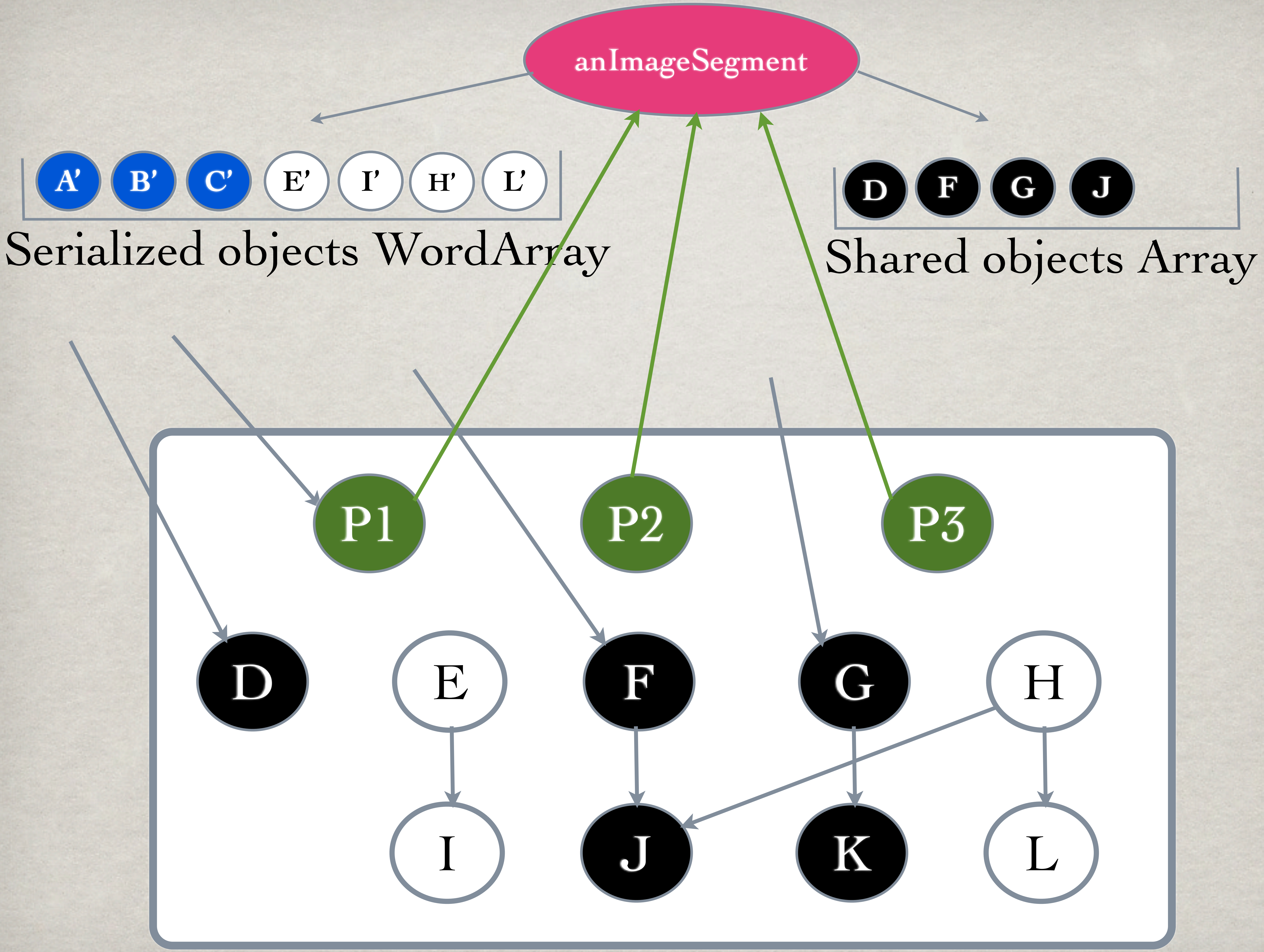
Shared objects Array

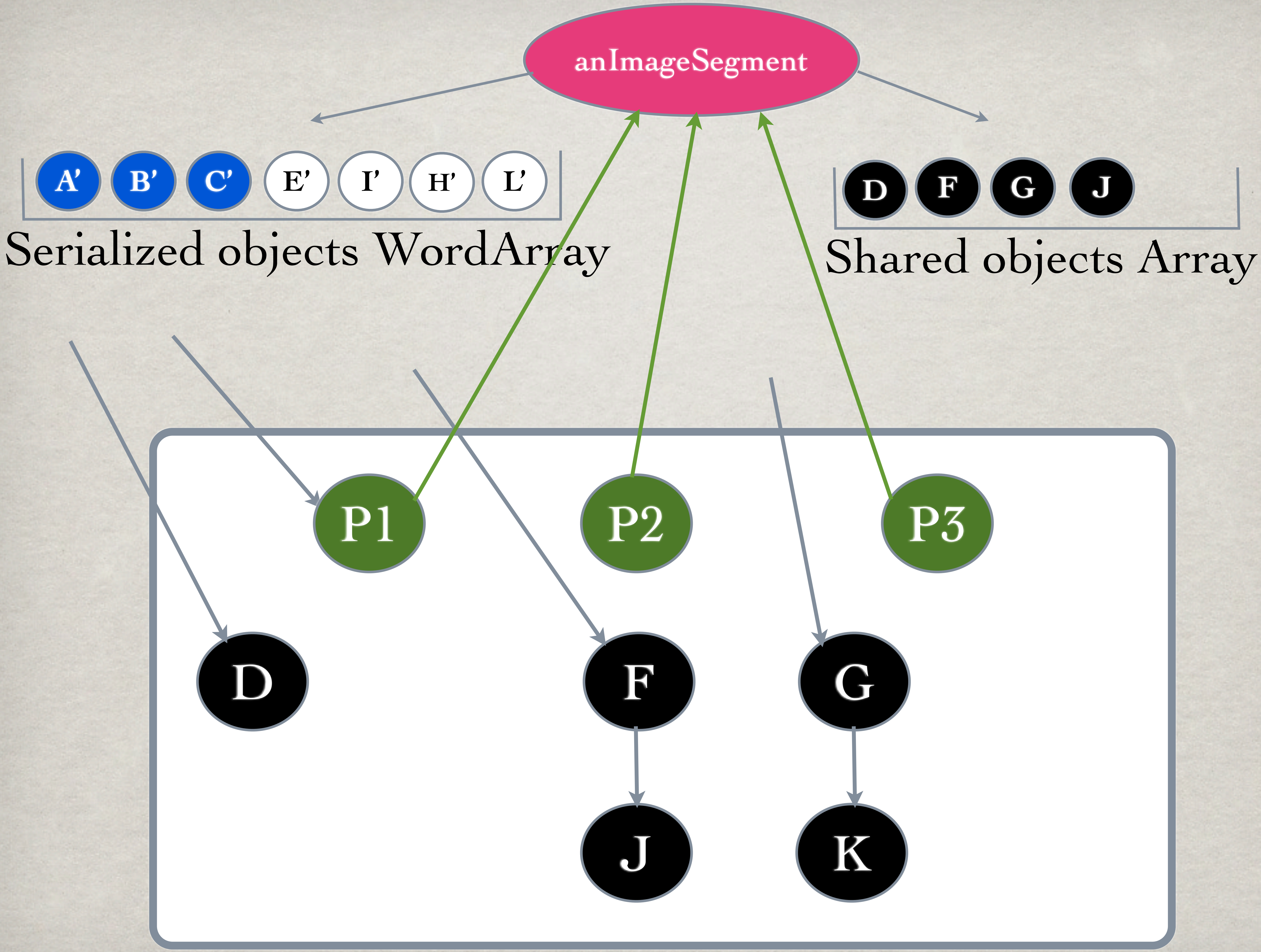


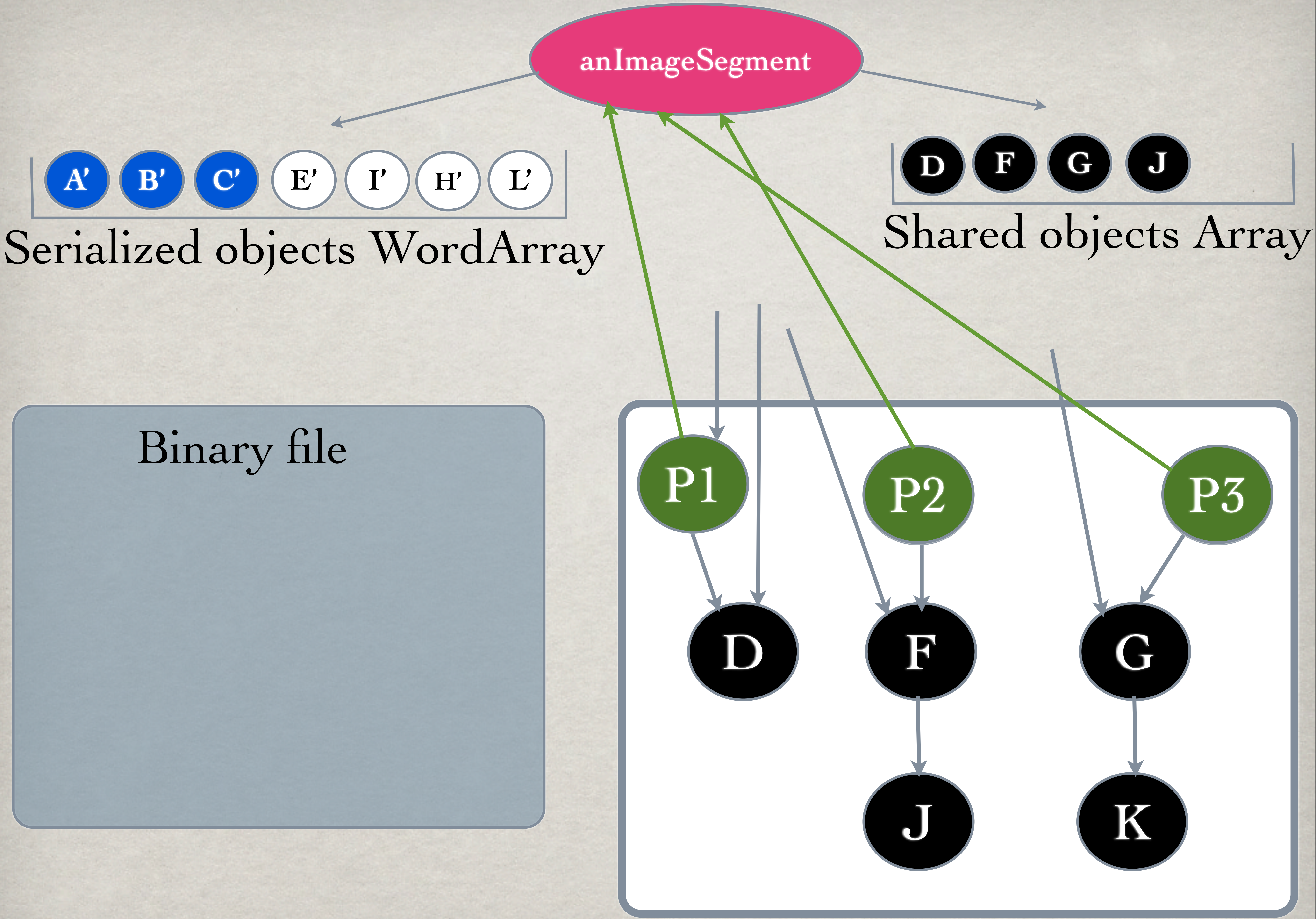
anImageSegment

Serialized objects WordArray

Shared objects Array


anImageSegment
A'
B'
C'
E'
I'
H'
L'
Serialized objects WordArray
D
F
G
J
Shared objects Array
P1
P2
P3
D
E
F
G
H
I
J
K
L


anImageSegment
A'
B'
C'
E'
I'
H'
L'
Serialized objects WordArray
D
F
G
J
Shared objects Array
P1
P2
P3
D
F
G
J
K

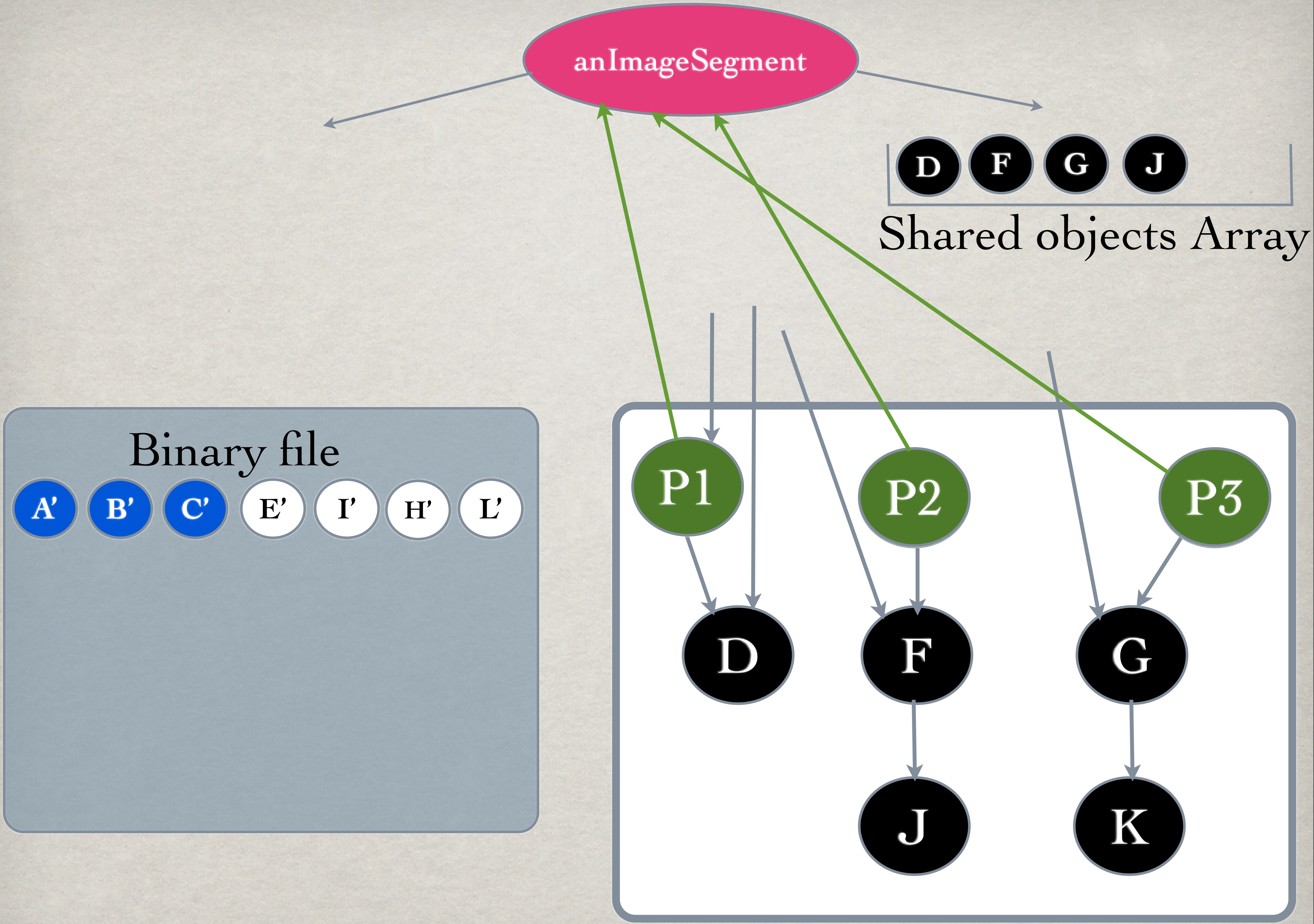


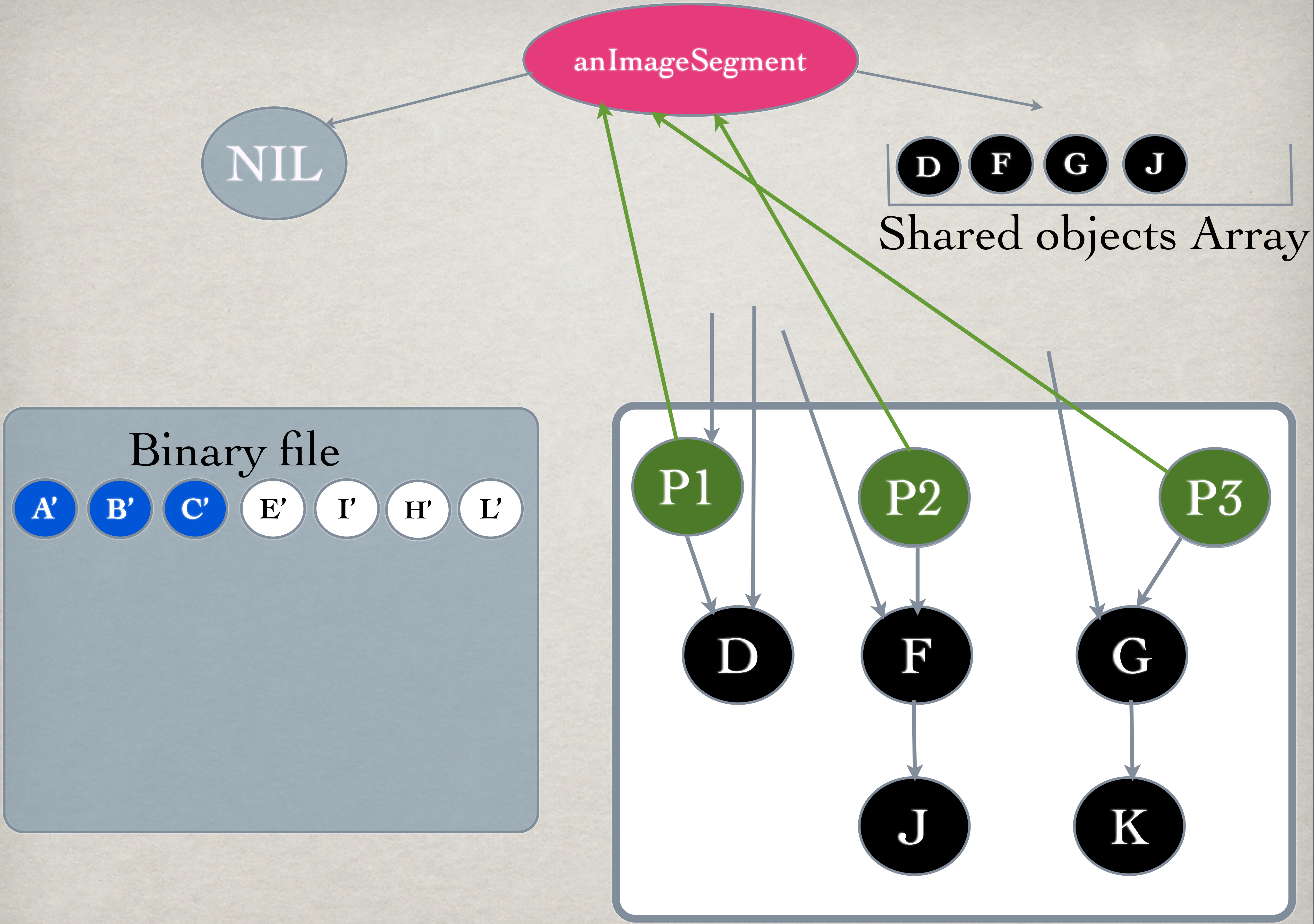
anImageSegment

Serialized objects WordArray

Shared objects Array

Binary file


anImageSegment
D
F
G
J
Shared objects Array
Binary file
A'
B'
C'
E'
I'
H'
L'
P1
P2
P3
D
F
G
J
K


anImageSegment
NIL
D
F
G
J
Shared objects Array
Binary file
A'
B'
C'
E'
I'
H'
L'
P1
P2
P3
D
F
G
J
K

ImageSegment conclusions

- ✓ Good speed.
- ✓ Graph traverse is done in VM side.
- ✓ Good use of GC facilities.
- ✗ You have to be aware of shared objects.
- ✗ Bad granularity level.
- ✗ Implicit needed information in object graphs.

Thanks!

Mariano Martinez Peck
marianopeck@gmail.com

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE INRIA

Ecole d'Ingénieurs Centre de Recherche Mines de Douai LILLE EURORÉGION