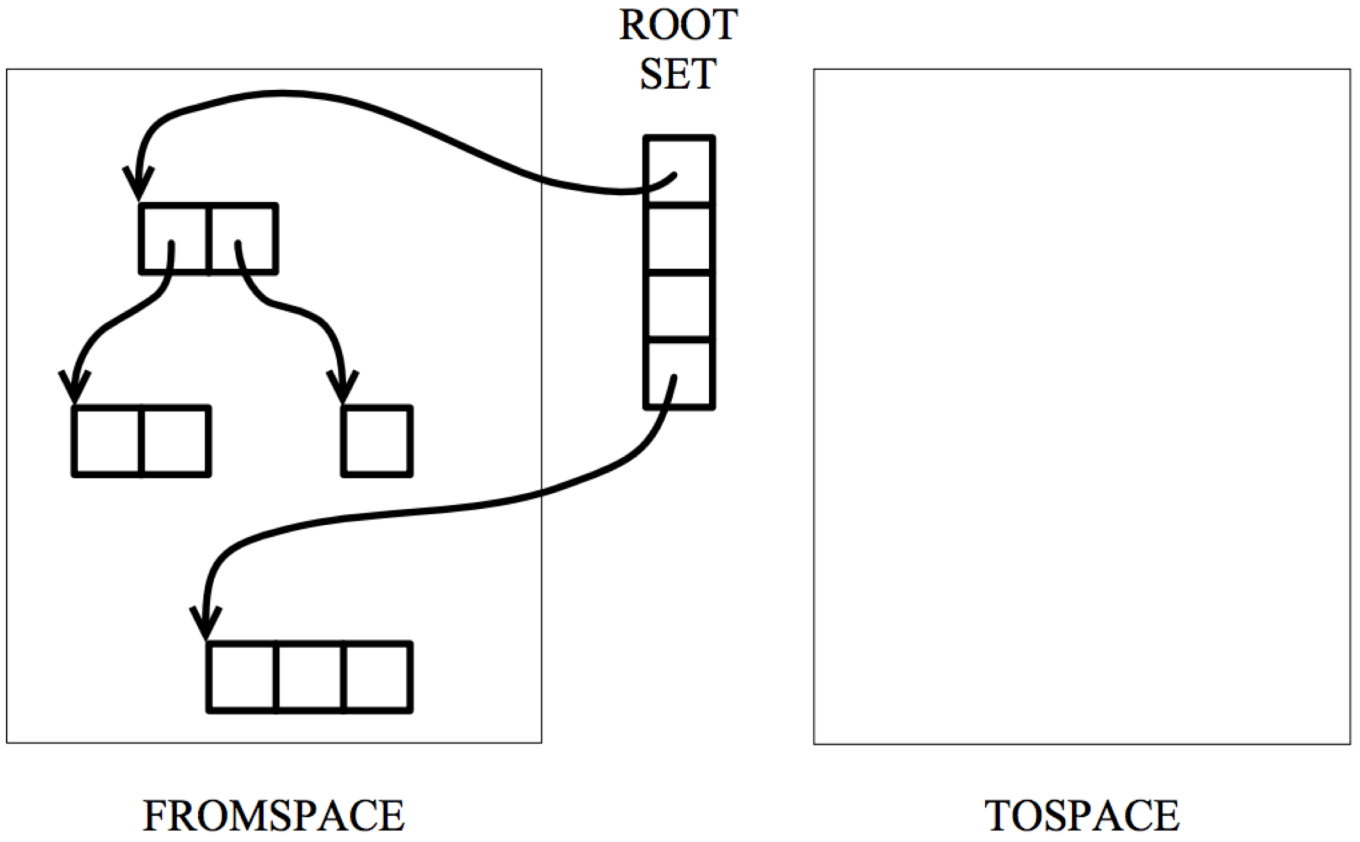
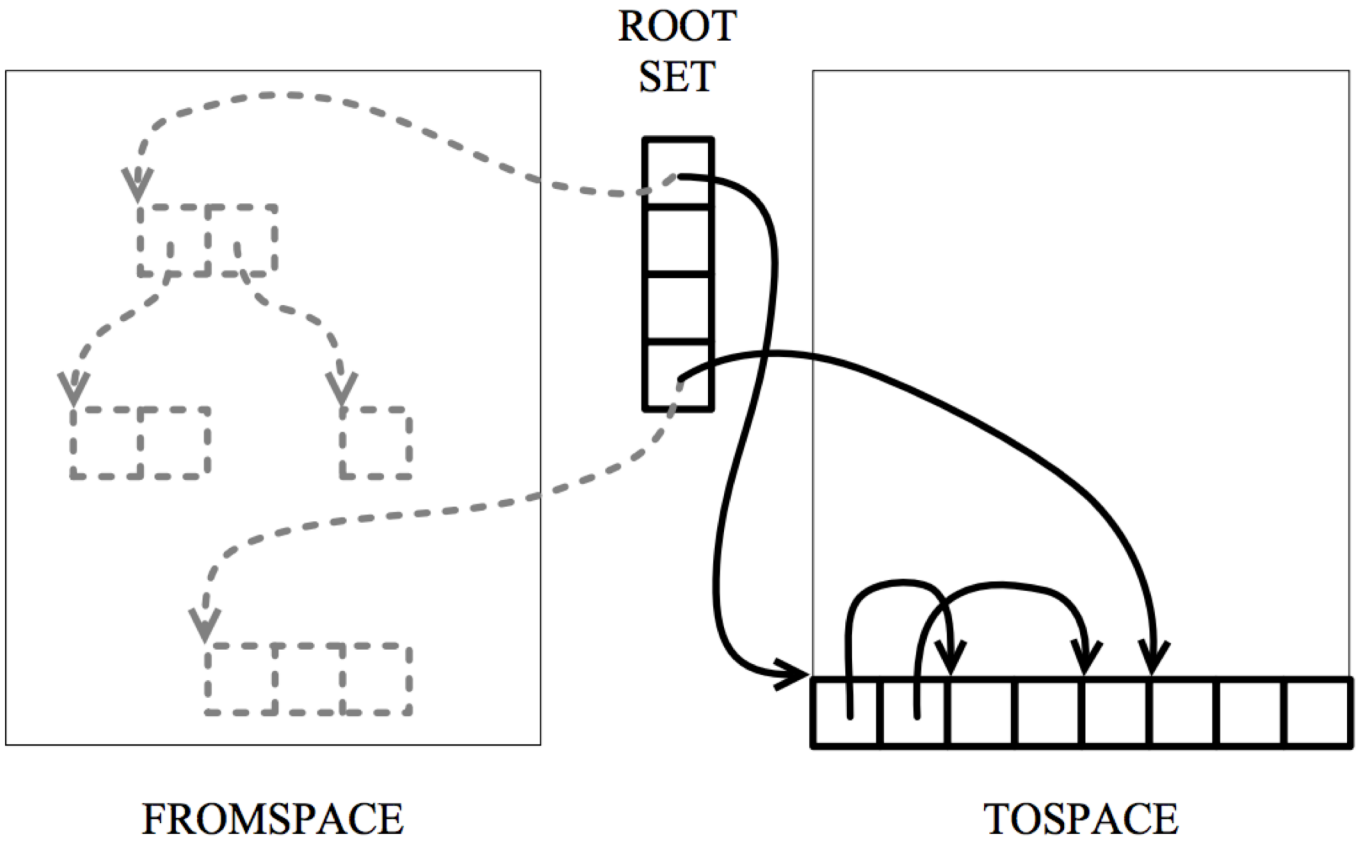
Stop-and-copy; Incremental collectors

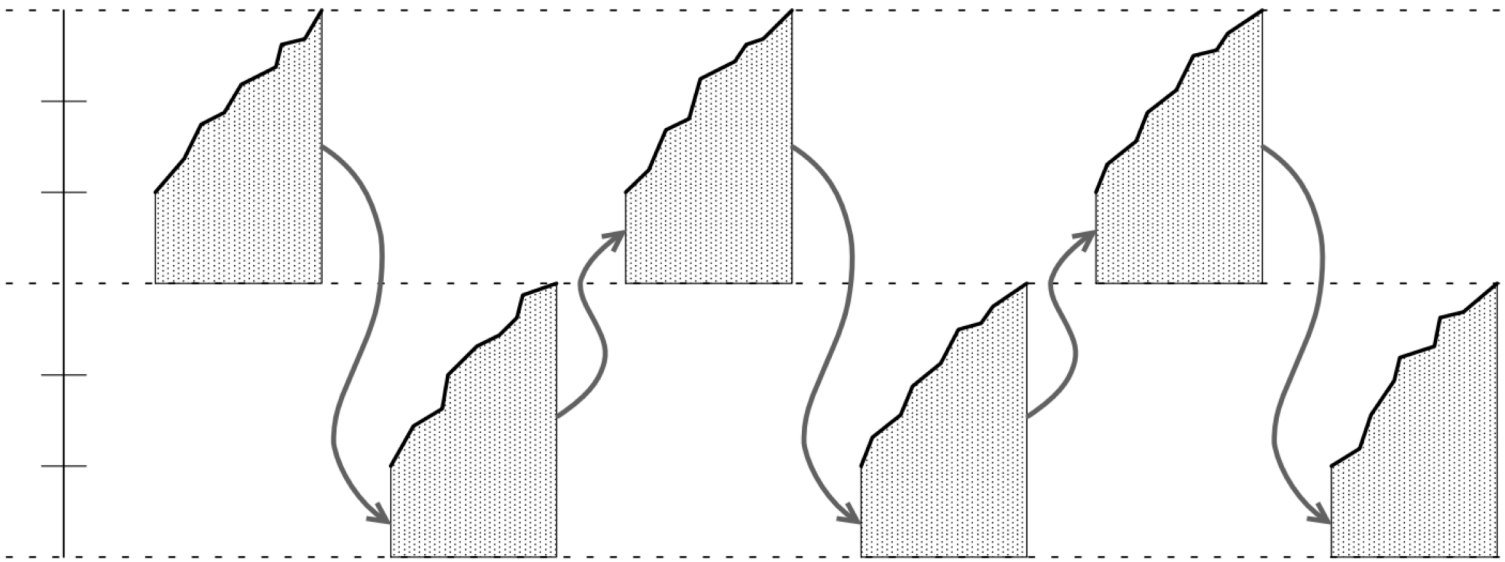
Stop-and-copy



Stop-and-copy



Stop-and-copy memory usage



Multi-space variants are possible, and common

Main advantage: time is only spent on live objects.
Objects are immediately copied during traversal. ← WARNING!

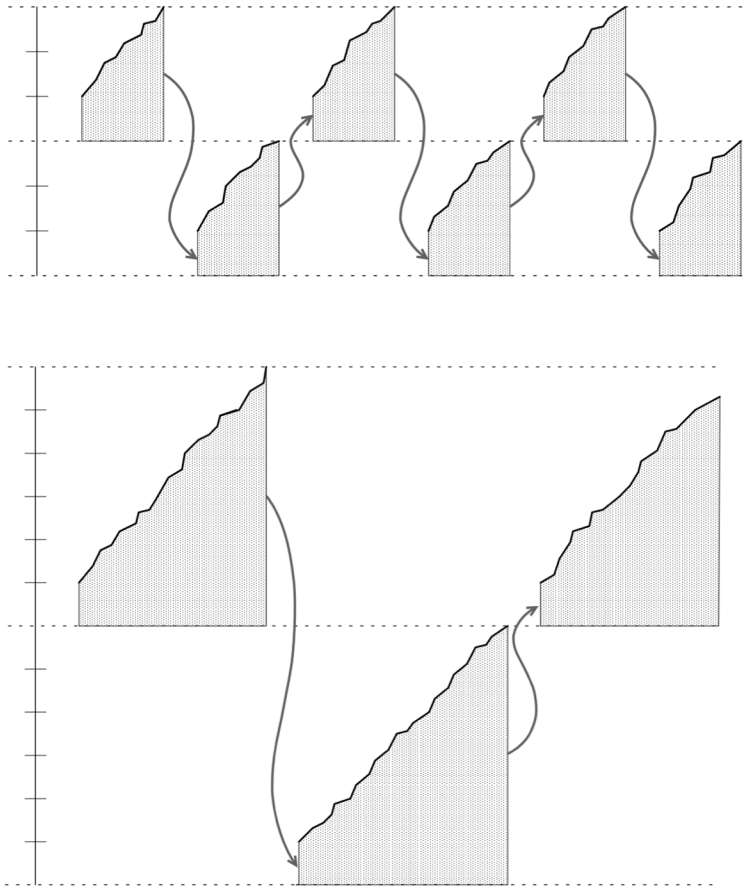
Stop-and-copy frequency

Copy frequency is proportional to the size of the semispace

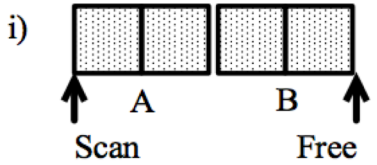
By decreasing the frequency of collection, we are increasing the average age of objects in the semispace.

Objects that become garbage before collection *never* have to be copied!

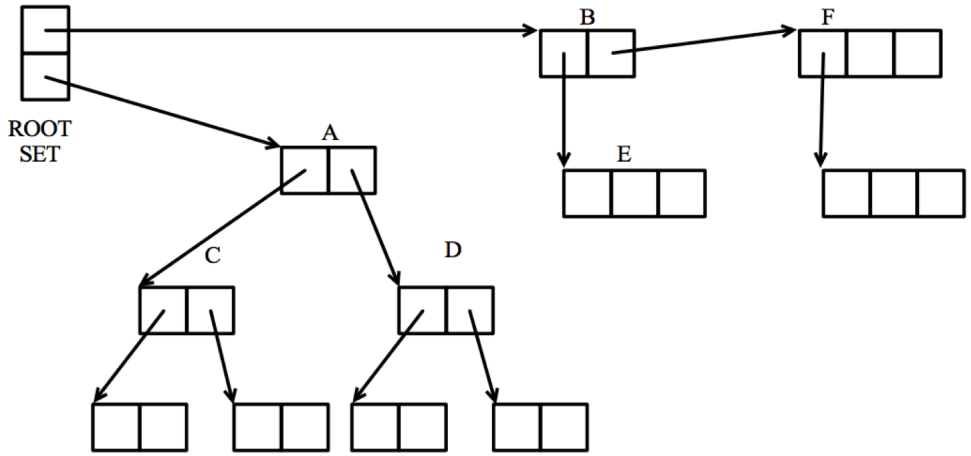
We’re increasing the chances of that happening.



The Cheney algorithm



Each time a pointer into fromspace is encountered, that object is copied to the end of the queue, and the pointer is updated. The free pointer is then advanced, and the scan continues.



When an object is transported into tospace, a *forwarding pointer* is installed in the old version of the object

Non-copying implicit collection

Observation:

The fromspace and tospace are conceptually *sets.* Any other implementation of sets would work just fine

Idea:

As long as you can tell which set something is supposed to be in, you don't have to copy things around!

One big problem

For many real-time systems, it's unacceptable to simply stop and spend a lot of time collecting garbage

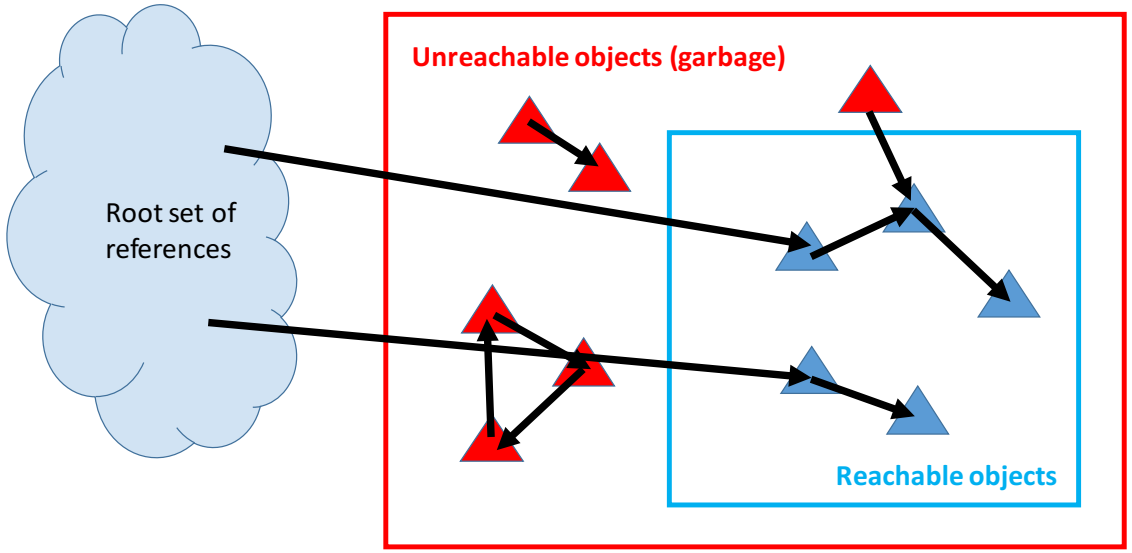
Solution: incremental GC

What happens if the program (*mutator*) changes the liveness graph in the middle of a traversal?

M&S: multiple readers, single writer

Collector

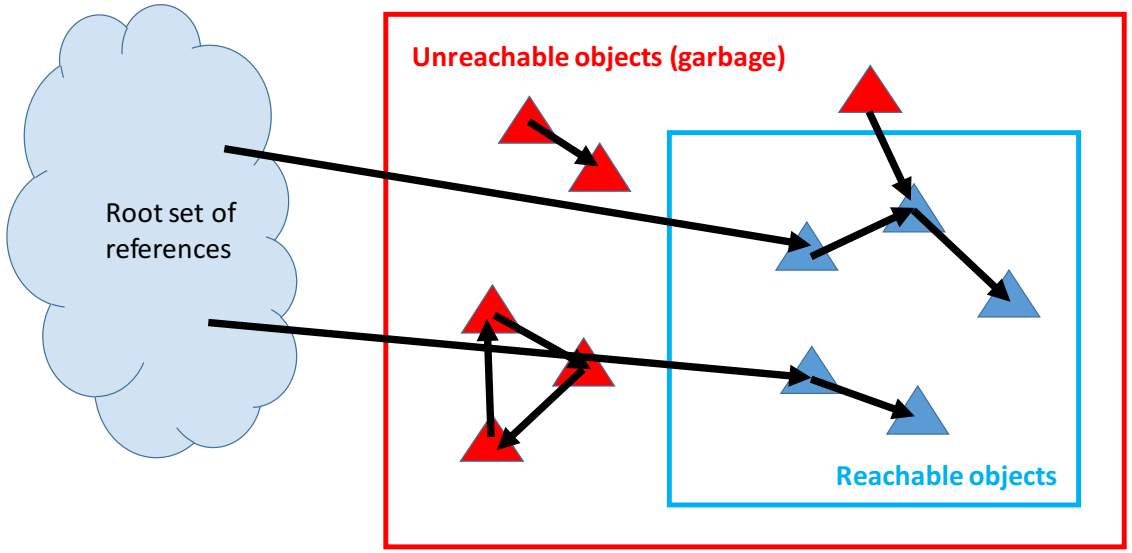
Mutator
(the program)



Copying: multiple readers, multiple writers

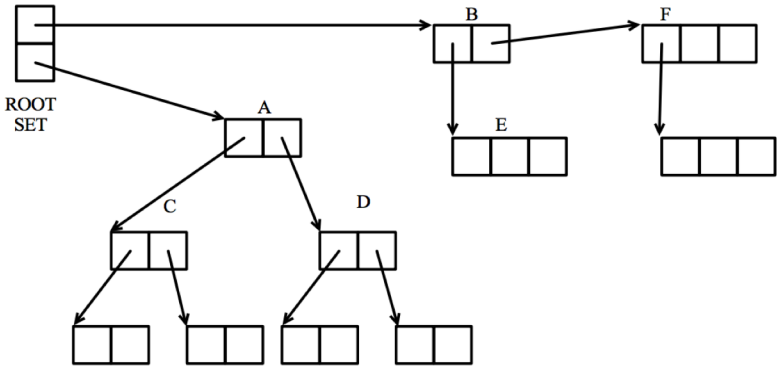
Collector

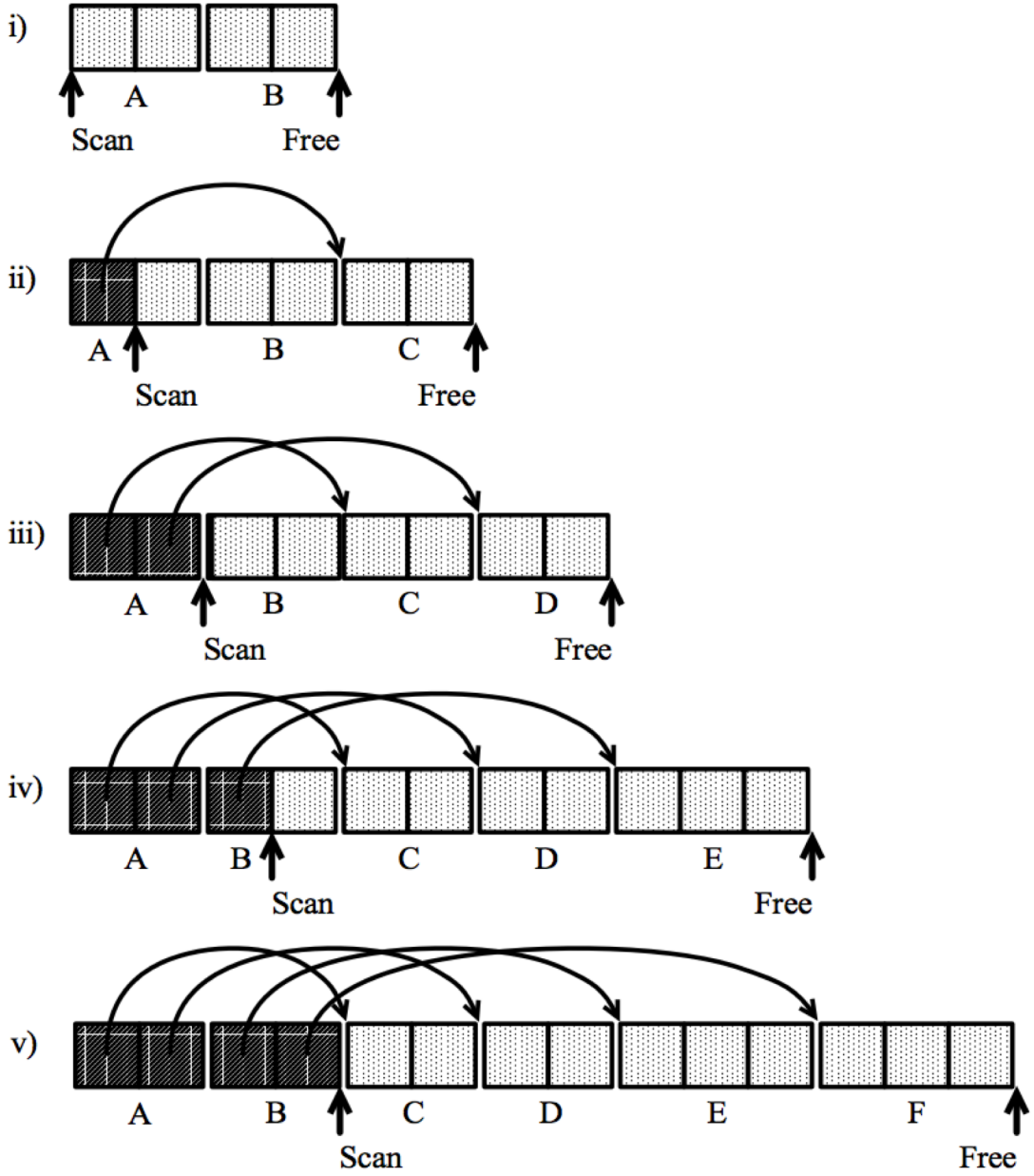
Mutator
(the program)



Tri-coloring

- **White:** objects subject to collection
- **Black:** objects that will be retained
- **Gray:** an object has been reached by the traversal, *but its descendants may not have been.*

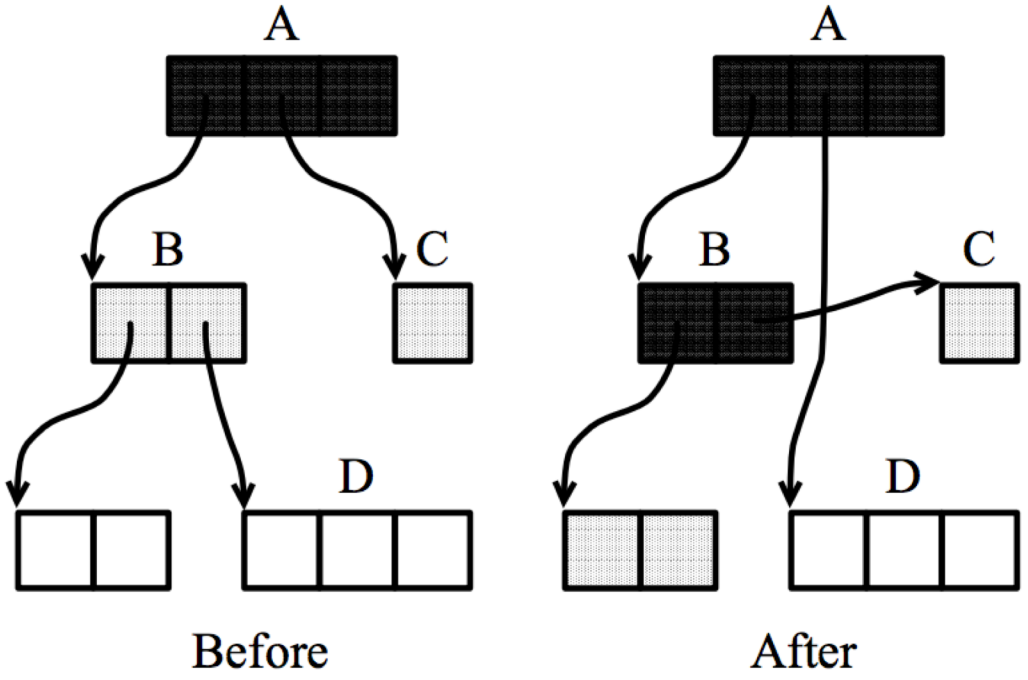




Tri-coloring invariant

- The mutator must preserve the invariant that no black object hold a pointer directly to a white object

- Two basic approaches to coordination
  - Read barrier
    - detects when the mutator attempts to access a pointer to a white object, and immediately colors the object gray (copies it over)
  - Write barrier
    - when the program attempts to write a pointer into an object, the write is trapped or recorded



A violation of the tricolor invariant