COLLABORATIVE GARBAGE COLLECTION

Ross Tate

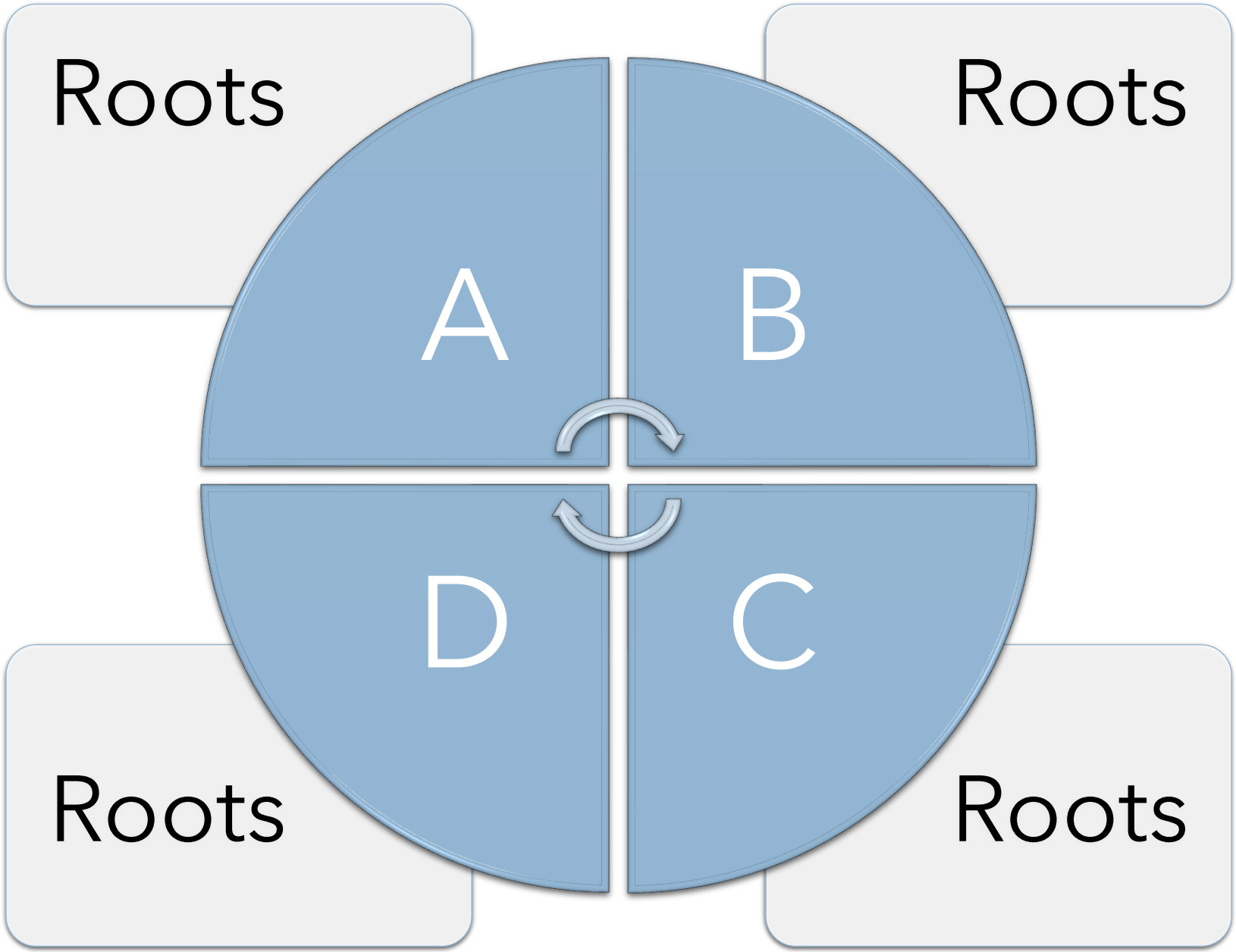
High-Level Need for Garbage Collection

- Wasm programs can each have their own “heaps”, represented in linear memory
- JS programs also have a heap (one per thread), managed by the engine
- Wasm programs can be given references into JS heap via funcrefs and externrefs
  - Sometimes this access is transient, such as when the funcref/externref stays on the wasm stack
  - Other times the access is long term, such as when the funcref/externref is stored in a wasm table
- JS programs can be given references into Wasm “heap” via “handles”
  - i.e. integers that represent something in the wasm “heap”
  - Sometimes the access is transient, such as when the “handle” is invalid after some call finishes
  - Other times the access is long term, such as when the “handle” is valid until explicitly freed
- Problem: Cannot tell when funcrefs/externrefs in tables or long-term handles are no longer needed
  - In particular, cyclic dependencies can form

How to Collect the Garbage?

Hand wasm "heap" off to host GC	• The GC Proposal
Enable wasm and host GCs to collaborate	• This talk

THE MODEL



Abstract Model

Many GCs, each with a heap

- With roots
- And cross references into other heaps
  - E.g. externref, handles

GCs are blackboxes

- Can be parallel
- Different implementations

GCs run at any time

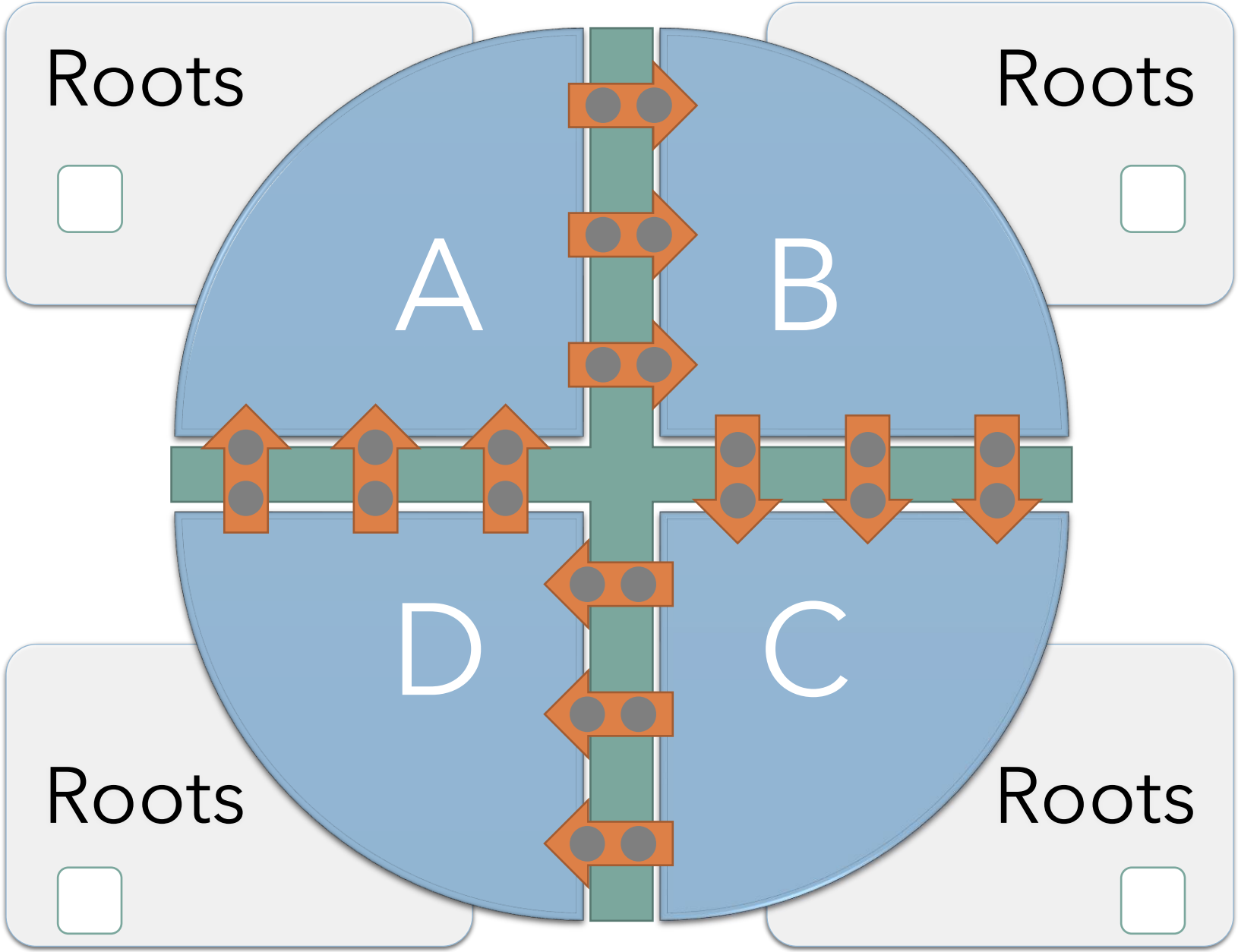
- Including in parallel

Can we clean the cycles?

THE ALGORITHM

High-Level Algorithm

- Add a centralized “cross-reference manager”
  - Responsible only for cross references
  - Cleans cross-reference garbage using “epochs”
- During an epoch:
  - Each GC runs many many times
  - Each run updates cross-reference “color” information
- An epoch ends when
  - cross-reference manager sees that “color” information has finished propagating
- When an epoch ends
  - All cross-references that are still “unknown” to be needed can be cleaned up
    - Would have been colored as “needed” if it were actually needed
  - Color information is cleared and a new epoch begins



Abstract Data Structure

Crossref Manager

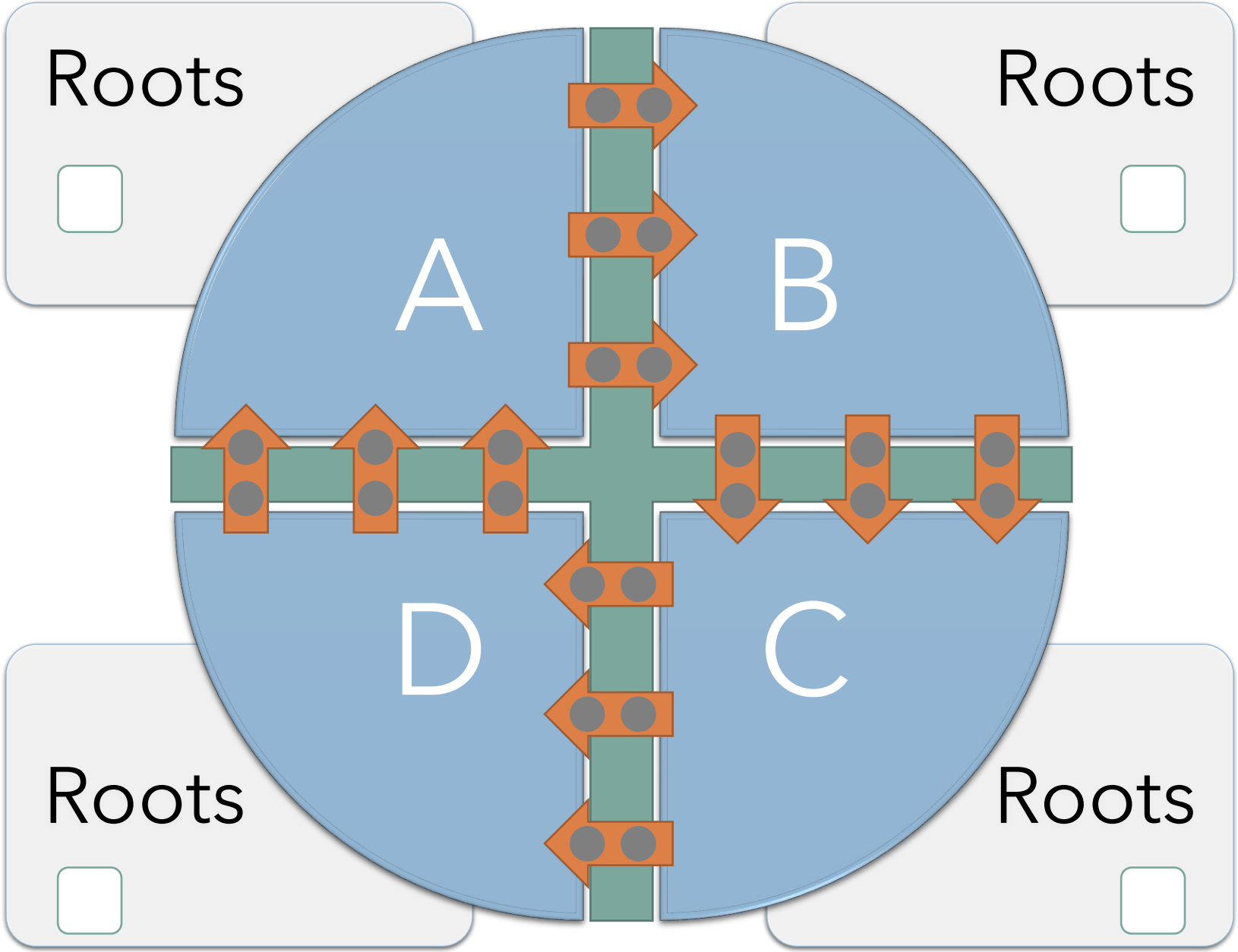
- Centralized
- Register crossrefs

For each crossref:

- Have “out” and “in” marks
- Each taking on 3 colors
  - Black = needed
  - White = unneeded
  - Grey = “it depends”
  - Grey at beginning of epoch

For each heap:

- Have a “roots” checkbox
  - Unchecked at beginning of epoch



Abstract GC Algorithm

Color outgoing “out” marks

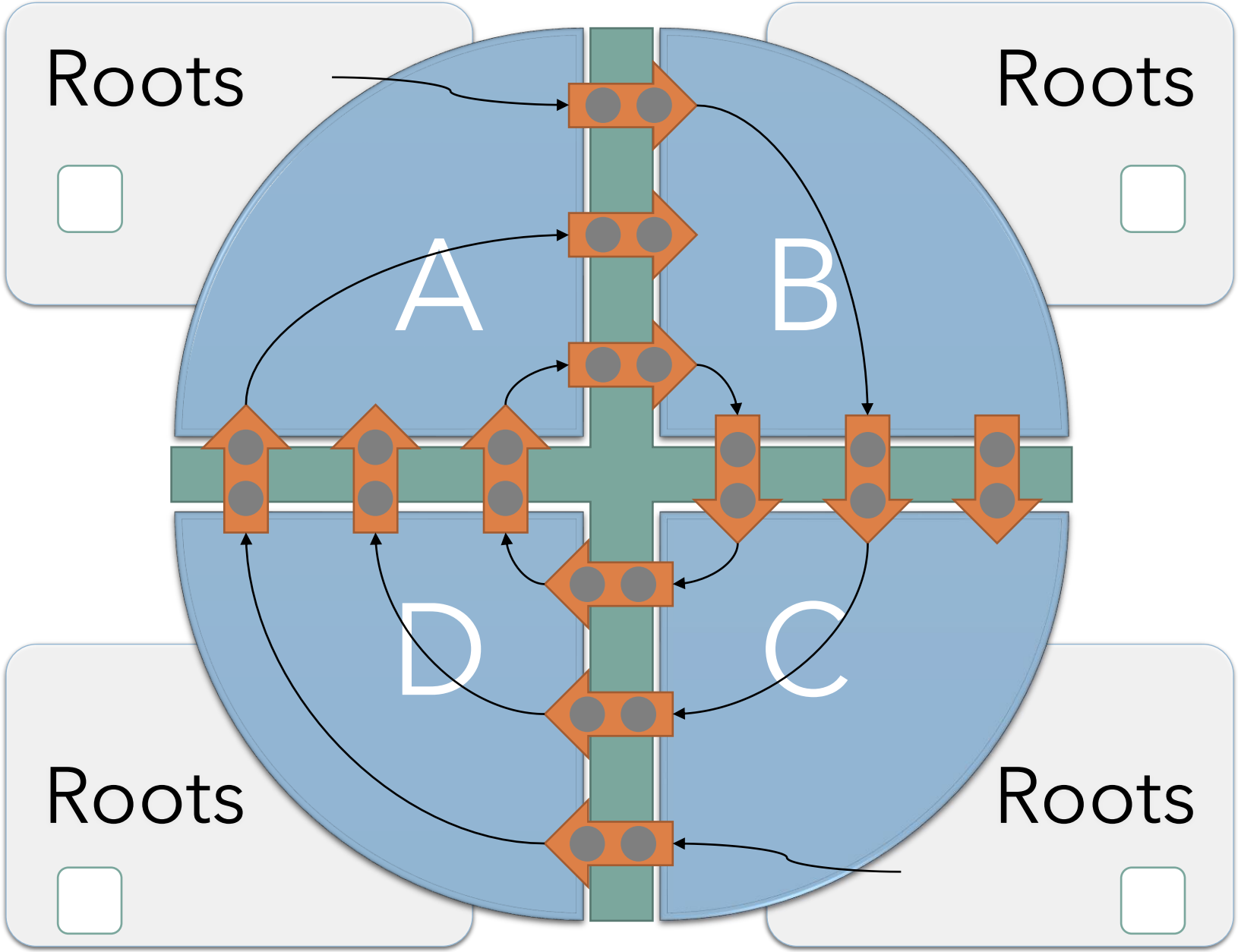
- Black if reachable from root or from incoming crossref with black “out” mark
- White only if unreachable from root and incoming crossrefs with non-white “out” mark

Check “roots” box

- Only if “out” marks reachable from roots are black

Color incoming “in” marks

- Black only if “out” marks reachable from it are black
- White only if that crossref’s “out” mark is white



GC Example

A colors AB1-out black

A checks roots

D checks roots

C colors CD-1-out black

C checks roots

B colors BC2-out black

B colors BC1-out white

B checks roots

B colors AB1-in black

C colors BC1-in white

C colors CD2-out black

D colors DA1-out black

D colors CD1-in black

C colors BC2-in black

A colors AB2-out black

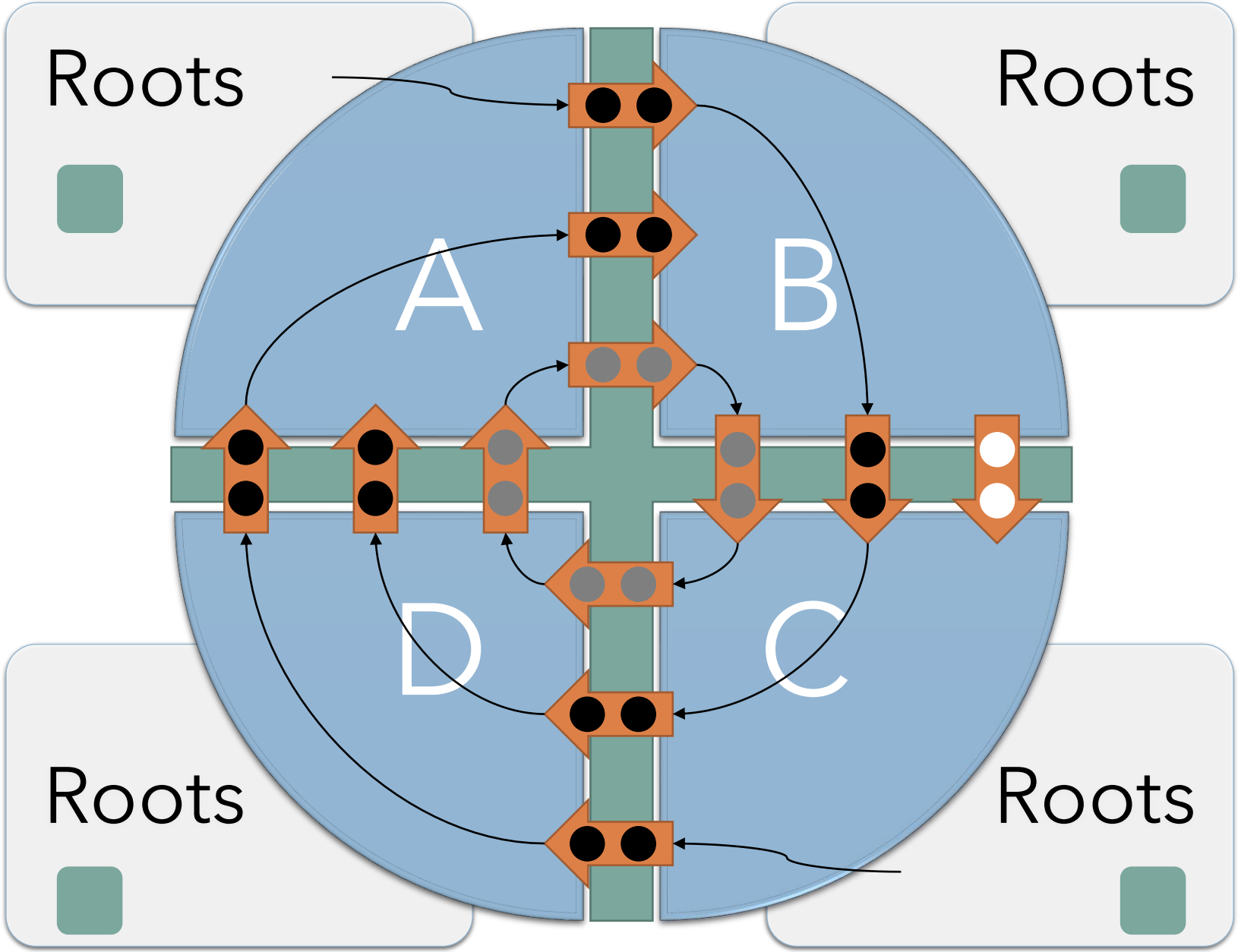
A colors DA1-in black

D colors DA2-out black

D colors CD2-in black

A colors DA2-in black

B colors AB2-in black



GC Example

A colors AB1-out black

A checks roots

D checks roots

C colors CD-1-out black

C checks roots

B colors BC2-out black

B colors BC1-out white

B checks roots

B colors AB1-in black

C colors BC1-in white

C colors CD2-out black

D colors DA1-out black

D colors CD1-in black

C colors BC2-in black

A colors AB2-out black

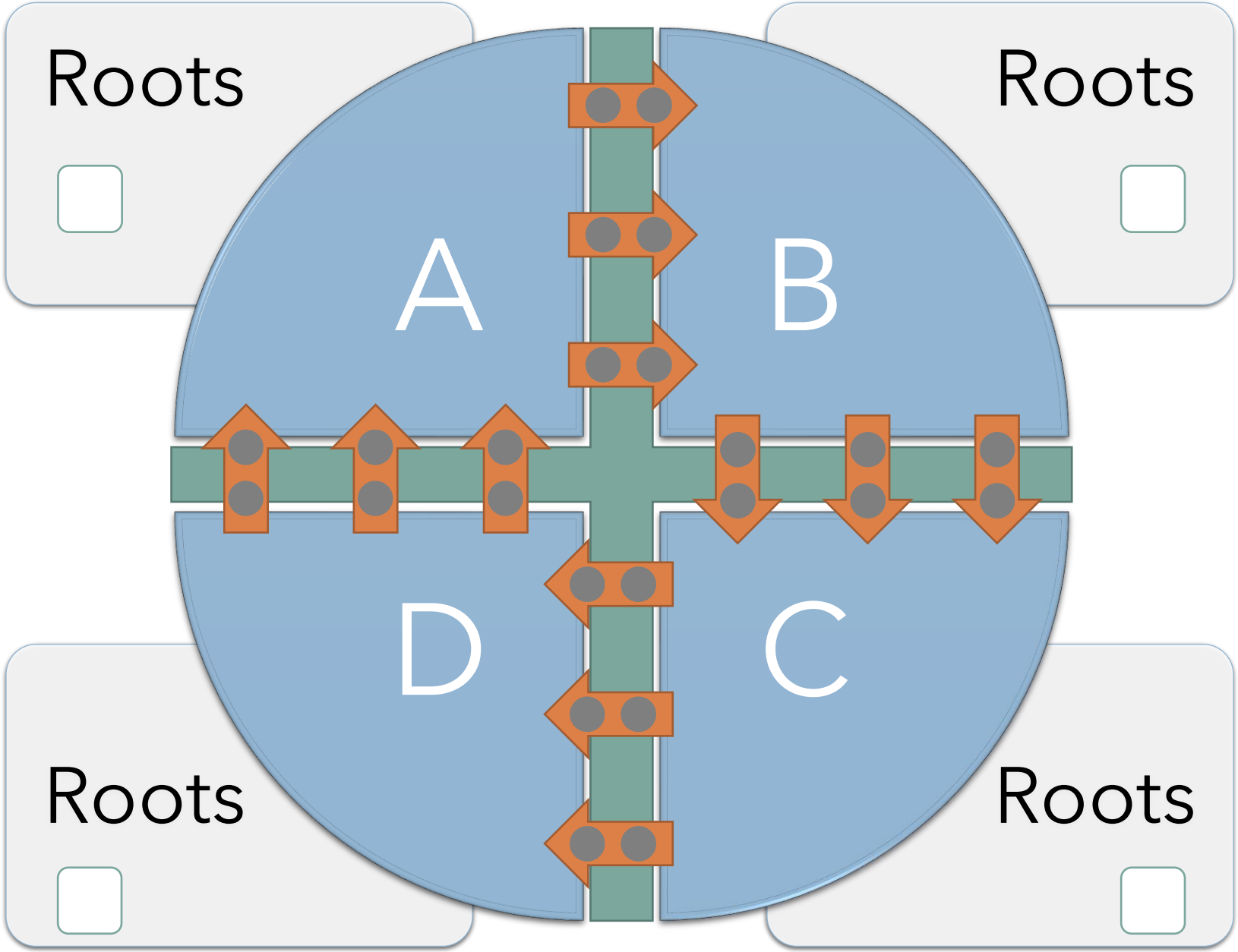
A colors DA1-in black

D colors DA2-out black

D colors CD2-in black

A colors DA2-in black

B colors AB2-in black



Abstract CRM Algorithm

At start of epoch

- All “roots” checkboxes are cleared
- All crossref marks are grey or white

During epoch

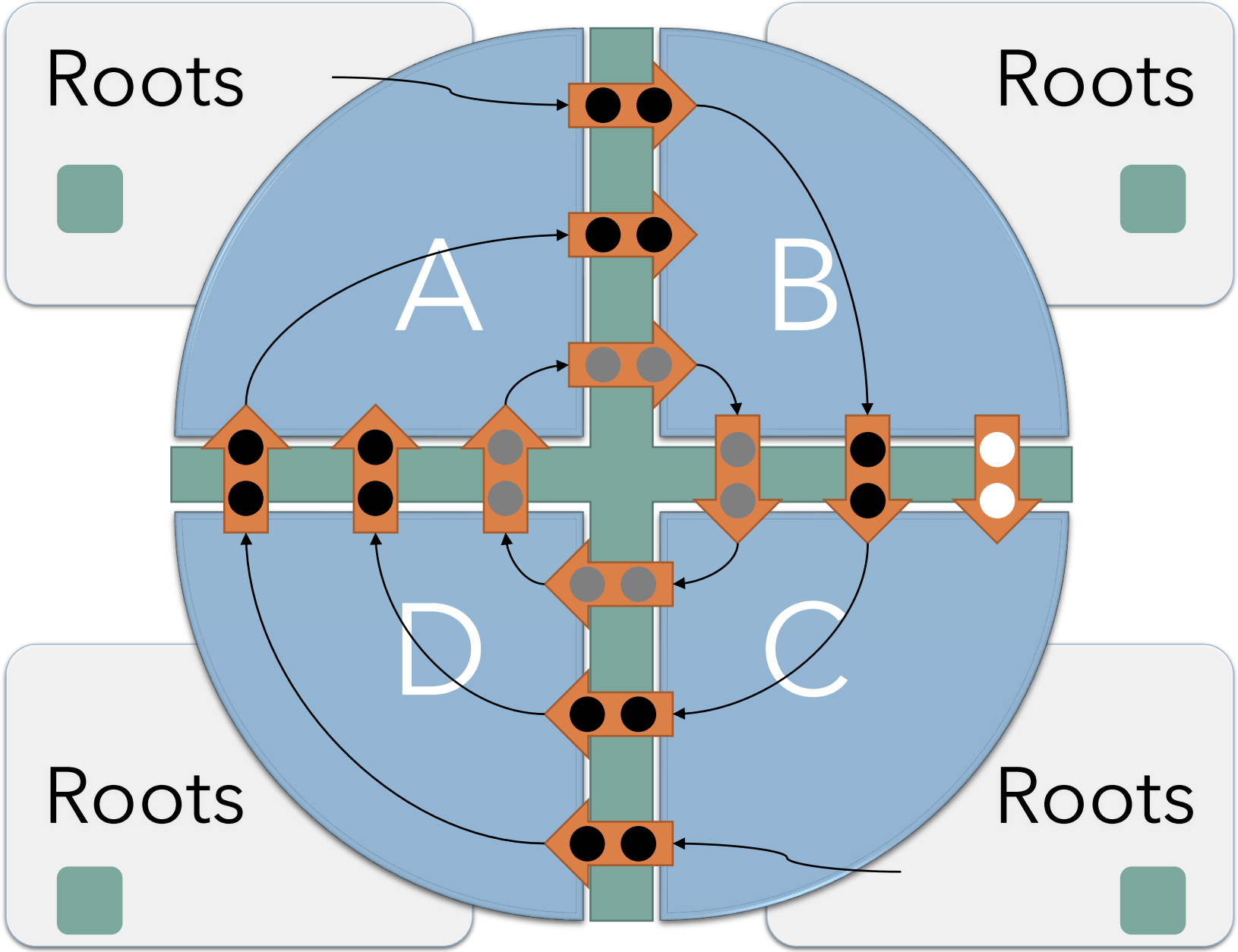
- Color newly created crossref marks black
- Free crossrefs with white “out” and “in” marks

End epoch only if

- All “roots” boxes are checked
- All black “out” marks have black “in” marks

End epoch by

- Coloring all grey “out” marks white
- Coloring all black marks grey
- Clearing all “roots” checkboxes
- (bikeshedding details for parallelism)



CRM Example

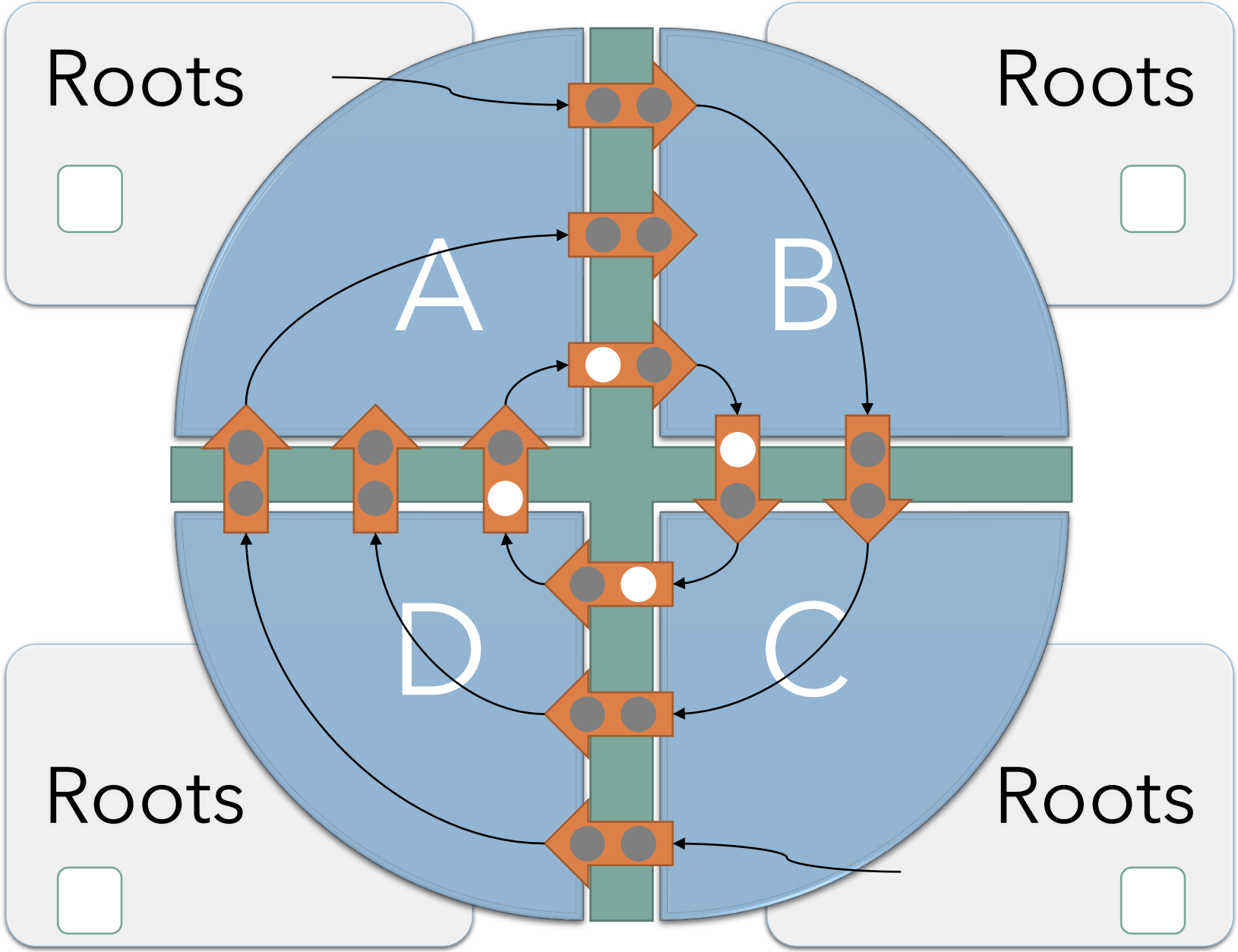
All-white crossref can be freed

Epoch can end because all crossrefs with black “out” mark also have black “in” mark

Grey “out” marks colored white

Black marks colored grey

“Roots” checkboxes cleared



Combined Example

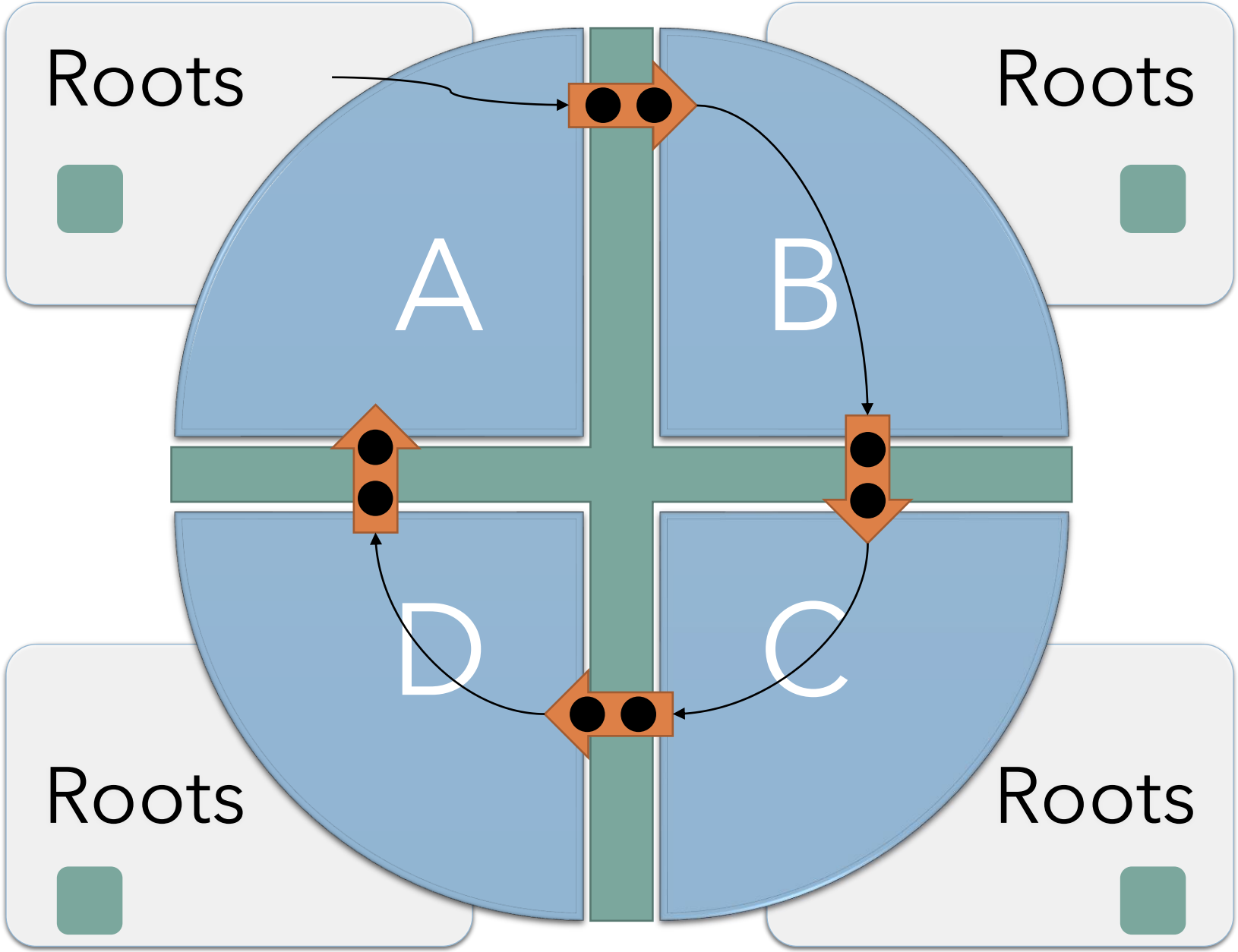
Inside “in”s eventually colored white by each GC and eventually freed by CRM

Fast forward…

Suppose program C drops the root to CD1

Then “out” of CD1 can be colored white

- which can propagate through D, A, & B, collecting garbage before epoch ends



Combined Example

Inside "in"s eventually colored white by each GC and eventually freed by CRM

Fast forward…

Suppose program C drops the root to CD1

Then "out" of CD1 can be colored white

- which can propagate through D, A, & B, collecting garbage before epoch ends

THE GUARANTEES

Sound and Complete Garbage Collection

- Provided all garbage collectors are complete
  - (meaning they only color cross references black if they are appropriately reachable)
  - And given sufficient time to execute
  - Then the epoch will eventually end
  - At which point all cross references that were unreachable from any roots at the beginning of the epoch
  - Will be freed or have their “out” mark be colored white at the end of the epoch
- Provided all garbage collectors are sound
  - (meaning they update “in” marks and “roots” only if reachable cross references have been colored black)
  - Then only cross references that are unreachable from any roots at the end of the epoch
  - Will be freed or have their “out” mark be colored white at the end of the epoch