On-the-Fly Garbage Collection: An Exercise in Cooperation

Edsger W. Dijkstra, Leslie Lamport, A.J. Martin, C.S. Scholten and E.F.M Steffens

(Communications of the ACM, November 1978)

Rudy Kaplan Depena
CS 395T: Memory Management
March 30th, 2009

Outline

Motivation

Garbage Collection is a useful tool for…

1. Automating Memory Management
2. Making Memory Management less error-prone (Developer doesn't have to think about the memory model or use special keywords.)

Motivation (cont.)

However, using GC is very costly for the mutator because …

1. In the case of McCarthy's Mark-Sweep (and other GC algorithms), collection is performed on the **whole heap**

2. Whole heap collectors incur **long pause times** which hault the mutator from executing and **unpredictable**

Motivation (cont.)

However, using GC is very costly for the mutator because …

1. In the case of McCarthy’s Mark-Sweep (and other GC algorithms), collection is performed on the **whole heap**
2. Whole heap collectors incur **long pause times** which hault the mutator from executing

Motivation (cont.)

Steele* proposes using two (or more) processors …

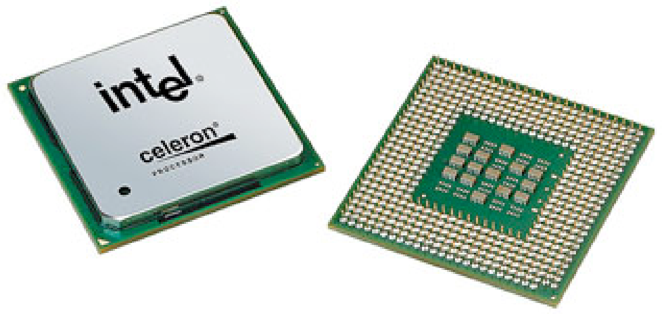
1. One processor is in charge of mutator program
2. One processor is in charge of the garbage collector

Goal: Have little to no pause times interrupting the mutator by housekeeping concurrently. ***However, exclusion and synchronization of processors are an overhead.***

* Multiprocessing Compactifying Garbage Collection (Communications of the ACM, 1975)

Motivation (cont.)

- Dijkstra et. al. proposes the use of multi-processing garbage collection, but **keep exclusion and synchronization constraints weak**
- *Weak constraints mean that we avoid highly frequent mutual exclusion of activities amongst processors. (e.g. - marking objects)*
- The above may help reduce mutator overhead
- *Paper's Goal: Not to make GC better, but to have a **use-case for multi-processor interaction***



Motivation (cont.)

- Although we would like to keep these weak constraints on exclusion *(avoid use of common resource)* and synchronization *(timekeeping)* there are a few unavoidable situations …

- When needing a new object from the free list, the **mutator may be delayed** until collector has appended some objects to the free list

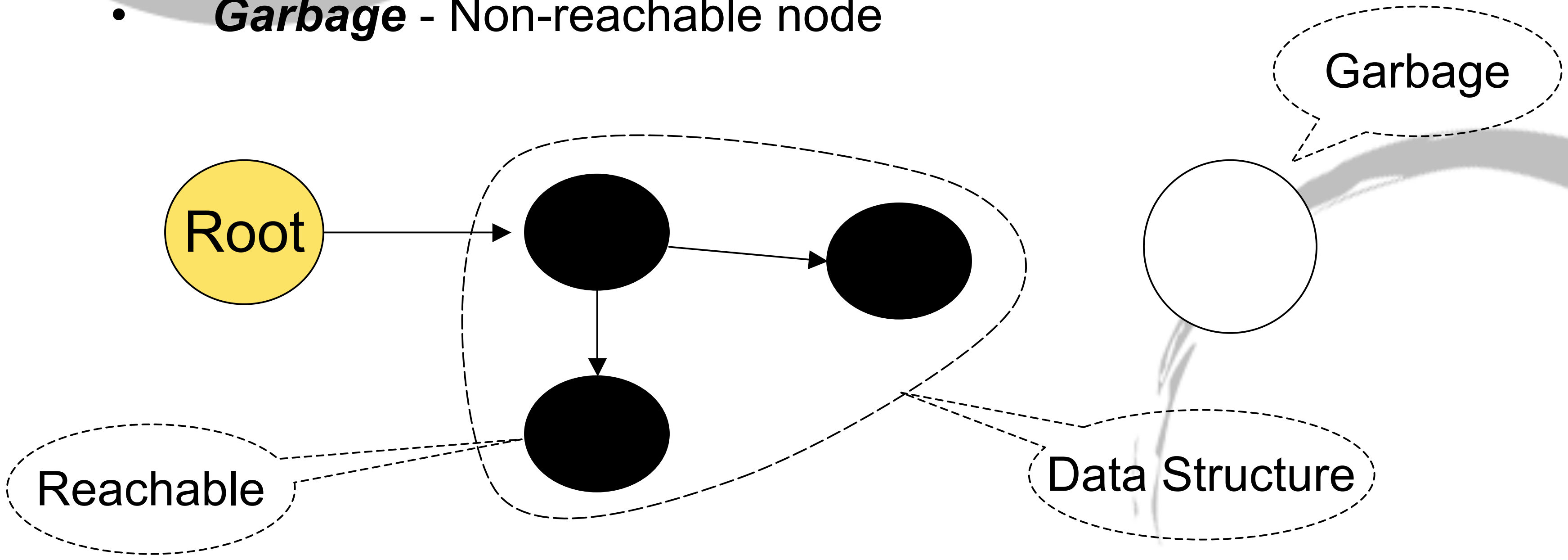
- Between the marking phase and appending/collection phase, it is impossible to guarantee that we append ***all*** garbage: new garbage could have been created between marking and appending phase.

Setting up the Playing Field

- The paper reasons about the problem of managing weak synch and exclusion by using a **Directed Graph** with a few constraints …
  - Fixed number of nodes
  - Each node has at most 2 outgoing edges

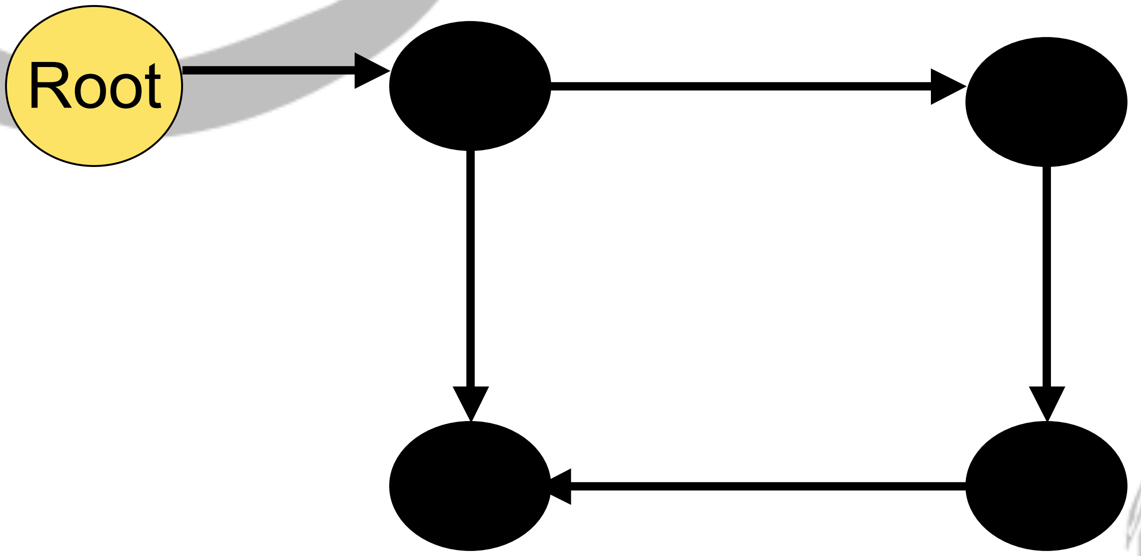
Setting up the Playing Field (cont.)

- A few tidbits in terminology …
  - ***Roots*** - A fixed set of nodes
  - ***Reachable*** - Node is reachable if reachable if there exists a directed path from one root to the node.
  - ***Data Structure*** - Subgraph of all reachable nodes
  - ***Garbage*** - Non-reachable node



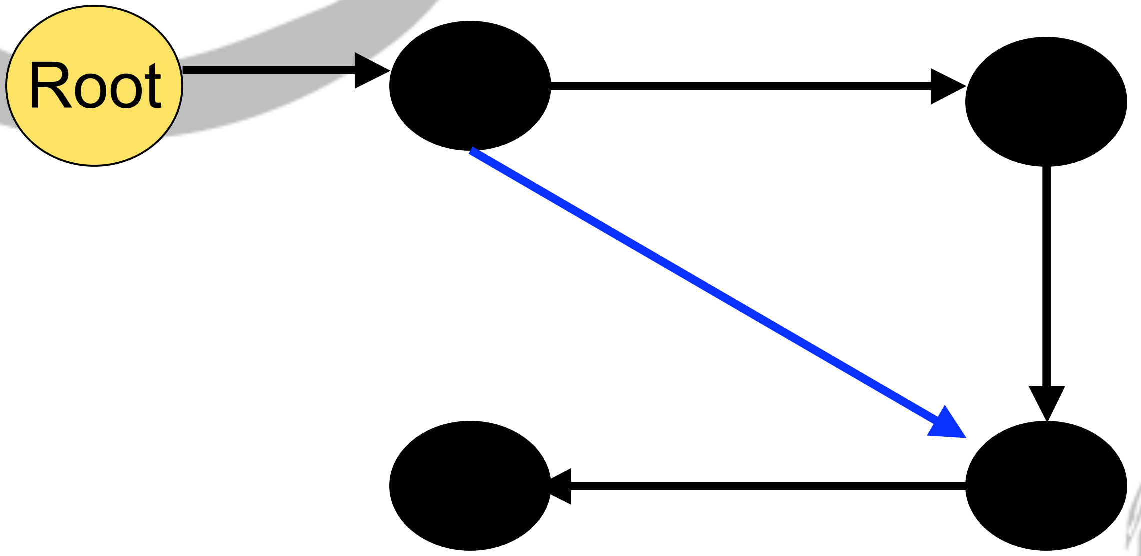
Setting up the Playing Field (cont.)

1. Redirect an edge from one reachable node to another



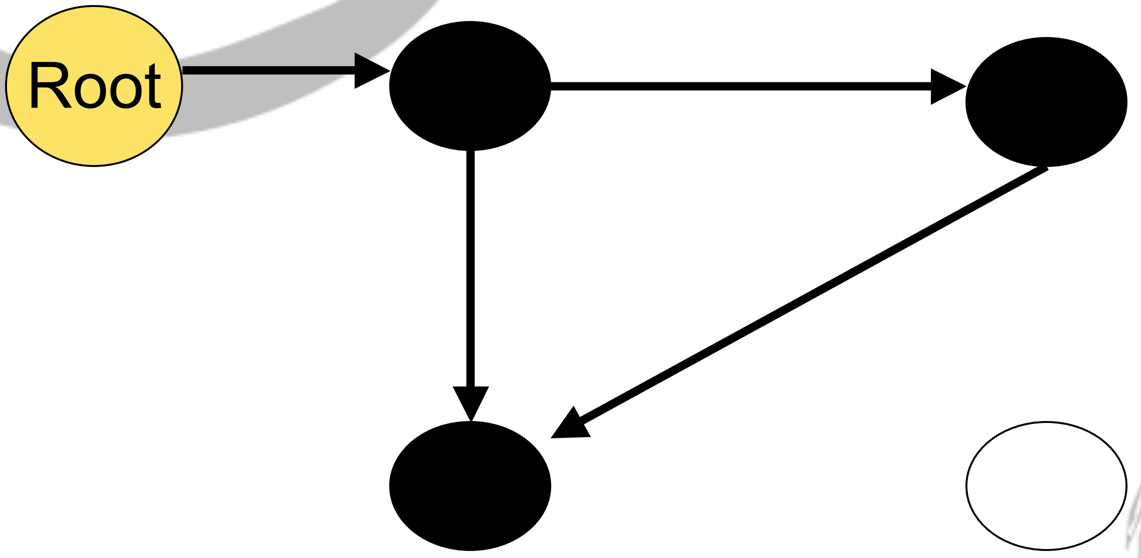
Setting up the Playing Field (cont.)

1. Redirect an edge from one reachable node to another



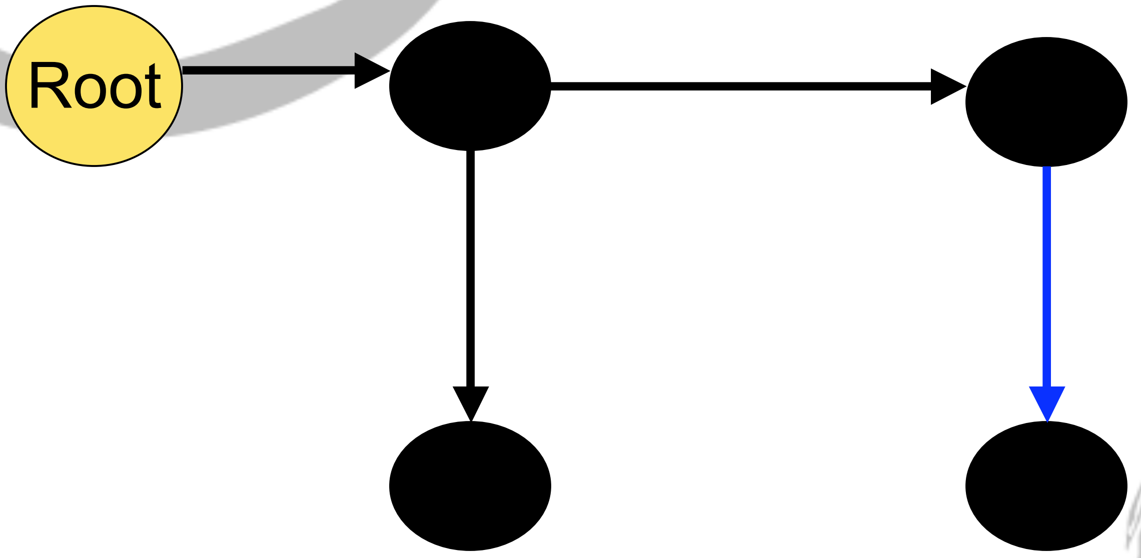
Setting up the Playing Field (cont.)

2. Redirect an edge from one reachable node to a non-reachable node



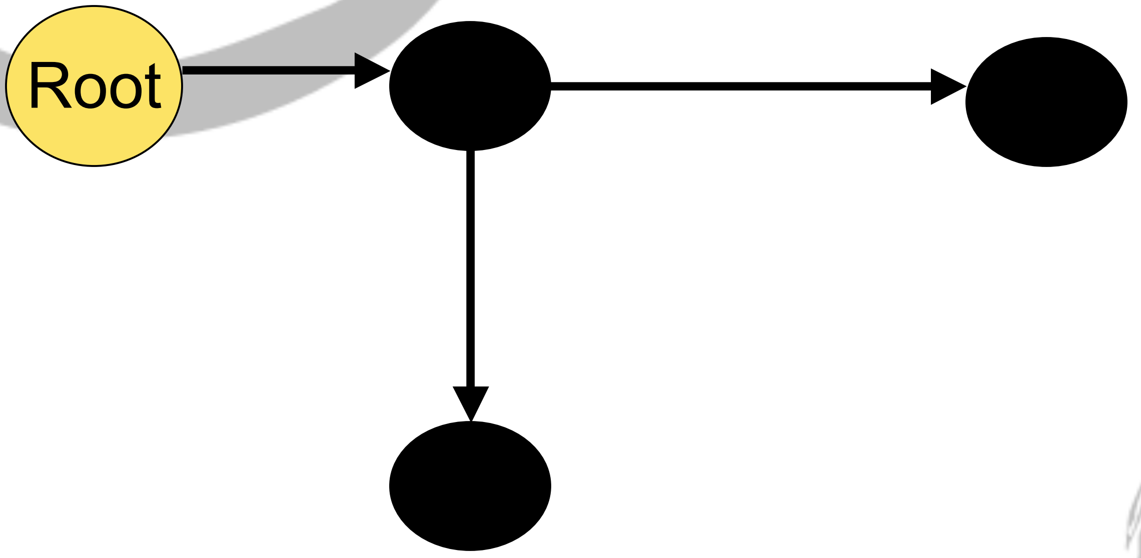
Setting up the Playing Field (cont.)

2. Redirect an edge from one reachable node to a non-reachable node



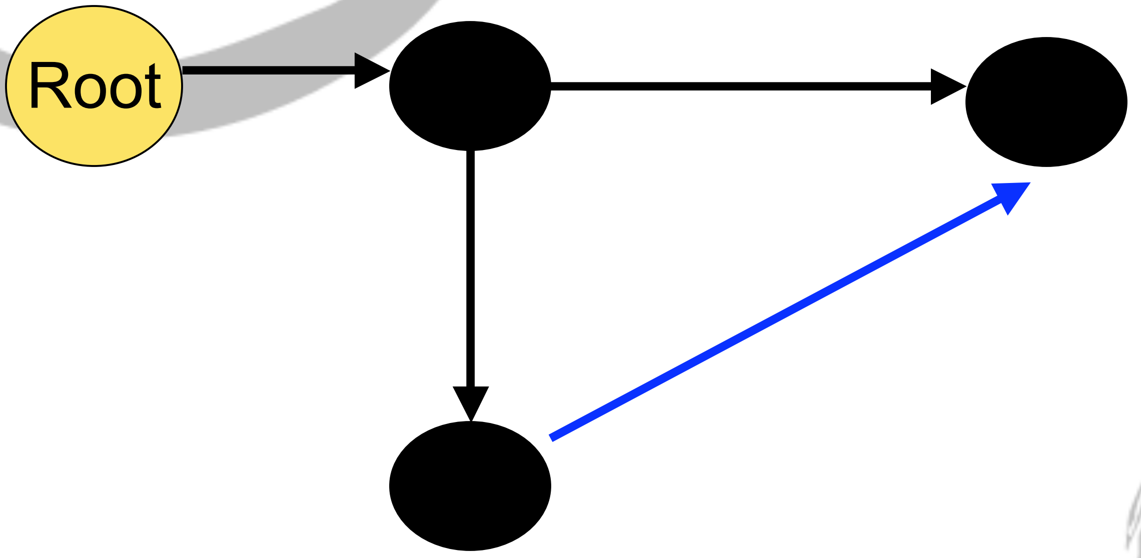
Setting up the Playing Field (cont.)

3. Add missing outgoing edge from one reachable node to another



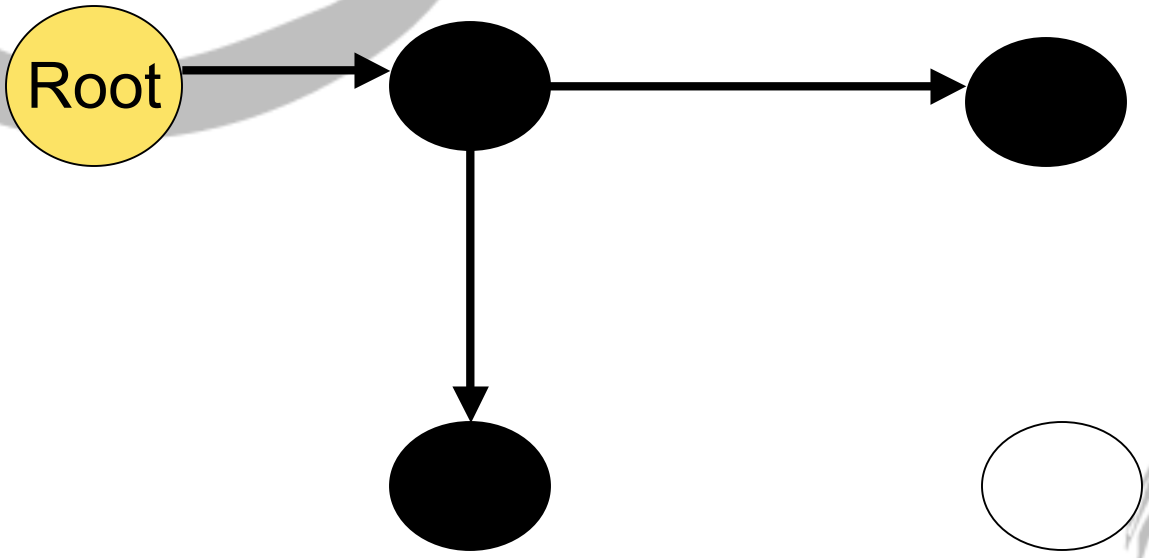
Setting up the Playing Field (cont.)

3. Add missing outgoing edge from one reachable node to another



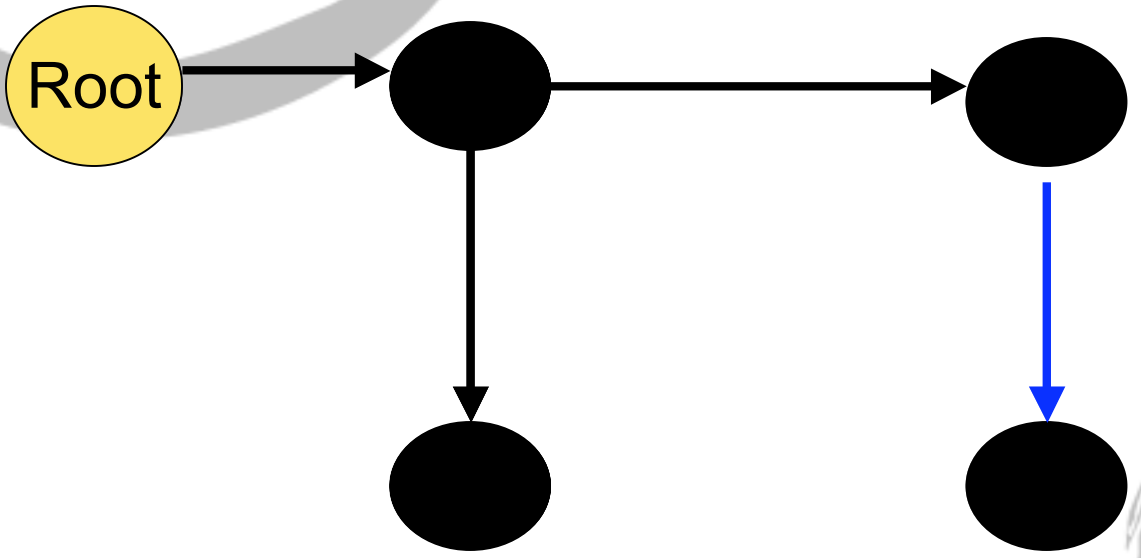
Setting up the Playing Field (cont.)

4. Add missing edge from reachable node to non-reachable node



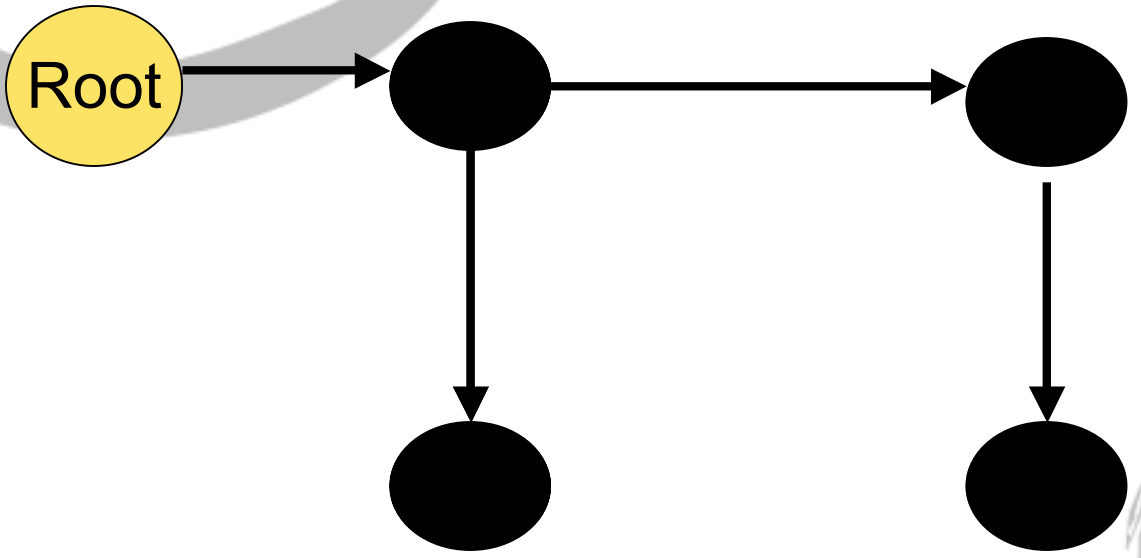
Setting up the Playing Field (cont.)

4. Add missing edge from reachable node to non-reachable node



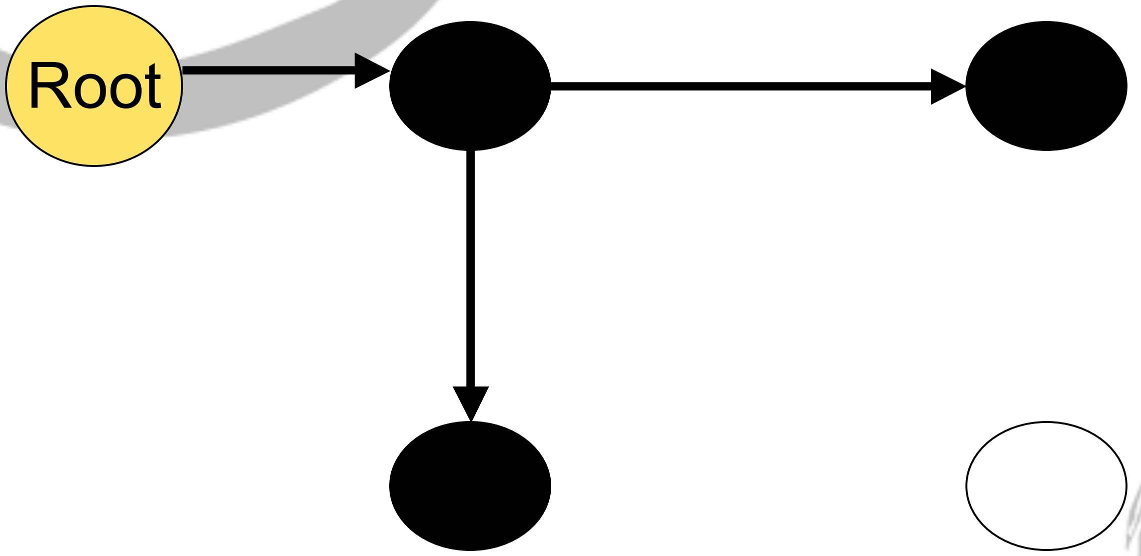
Setting up the Playing Field (cont.)

5. Remove the edge of a reachable node



Setting up the Playing Field (cont.)

5. Remove the edge of a reachable node



Discussion

- The authors set a special root node called *NIL* whose outgoing edges point to itself
- The authors also represent a formerly missing edge now as an edge with *NIL* as the logic
- They say that *NIL* allows them to view data structure modifications of types (3) and (5) as special cases of (1).
- Why does *NIL* help with this ???

Remember:

(1) Redirect an outgoing edge of reachable node to already reachable node

(3) Add an edge pointing from reachable node to a reachable node

(5) Remove edge of reachable node

Setting up the Playing Field (cont.)

Rules that **make** garbage:

(1) Redirect edge from one reachable node toward another

(2) Redirect edge from reachable node toward a non-reachable node

(5) Remove edge of reachable node

Rules that **recycle** garbage:

(2) Redirect edge from reachable node toward a non-reachable node

(4) Add missing edge from reachable node to non-reachable node

Setting up the Playing Field (cont.)

- Suppose we have two programs A and B.
- In this paper, **atomic operations** are denoted with angle brackets:
  - A: <z:=0> and B: <z:=3>
- We say that A is **coarser-grained** than B if B is the result of replacing an atomic operation of A by a piece of program containing at least two atomic operations and having the same net effect as the original operation:
  - A: <z:=0>
  - B: <z_0:= 0>
    <z_1:= 0>
- If correctness is proved on finer-grained solution, this implies that coarse-grained solution is correct also.

Discussion

- A simplification that the authors made is treating the **free list not as a list of garbage**, but as part of the data structure
- Link *NIL* and free nodes such that they are reachable from special root nodes.
- A modification of (2) is now replaced by a sequence of modifications of type (1).
- Why does this simplification allow them to replace (2) with a sequence of (1) modifications ???

Remember:

(1) Redirect an outgoing edge of reachable node to already reachable node

(2) Redirect an outgoing edge of reachable nodes towards a not yet reachable one with outgoing edges

Setting up the Playing Field (cont.)

- Now because of *NIL* and free list being apart of the data structure, mutator and collector activities are now repeated executions of (1)

- Mutator: “redirect an edge of reachable node to already reachable one”
- Collector:
  - Mark Phase: “mark all reachable nodes”
  - Append Phase: “append all unmarked nodes ot the free list and remove markings from all marked nodes”

- Two **correctness criteria** must be satisfied when mutator and collector cooperate together
- *CC1: Every garbage node is eventually appended to free list.*
- *CC2: Appending a garbage node to the free list is the collector’s only change to the shape of the data structure*

Remember:

(1) Redirect an outgoing edge of reachable node to already reachable node

Setting up the Playing Field (cont.)

- The final goal was a fine-grained solution that would accept the following atomic operations …

  1. Redirect an edge
  2. Find the successor of a node
  3. Test/Set attribute of a node

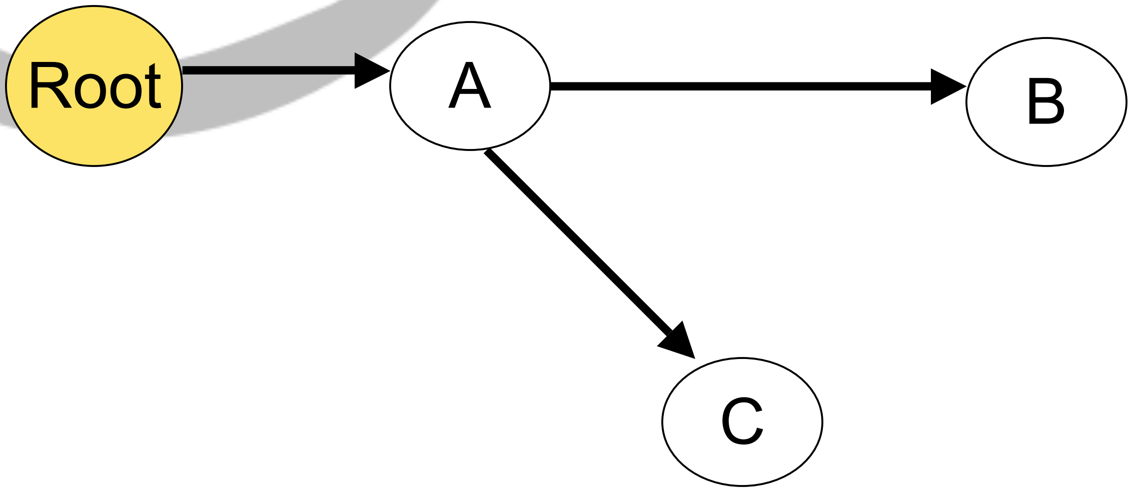
  *4. Append node nr.i to the free list

- The authors want to find a coarse grained solution and then transform it to a finer grained solution.

* Operation available to the collector

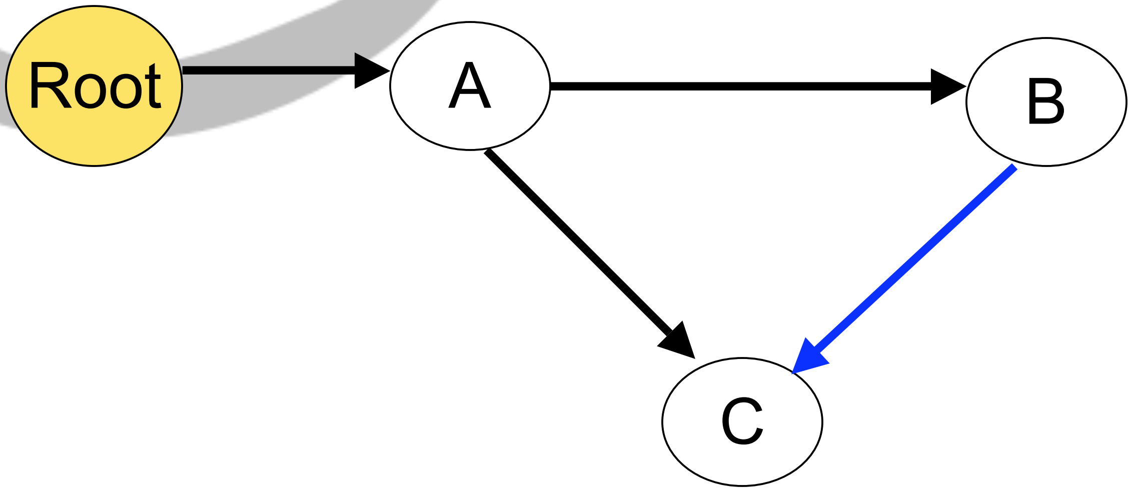
First Coarse-Grained Solution

- Counter-Example shows that goal of “no extra mutator overhead” is unattainable.



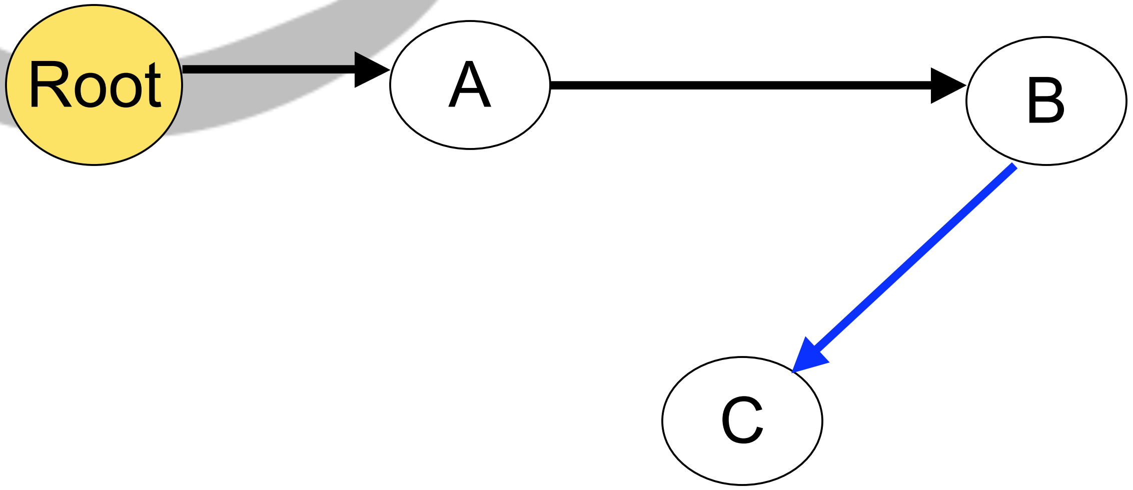
First Coarse-Grained Solution

- Counter-Example shows that goal of “no extra mutator overhead” is unattainable.



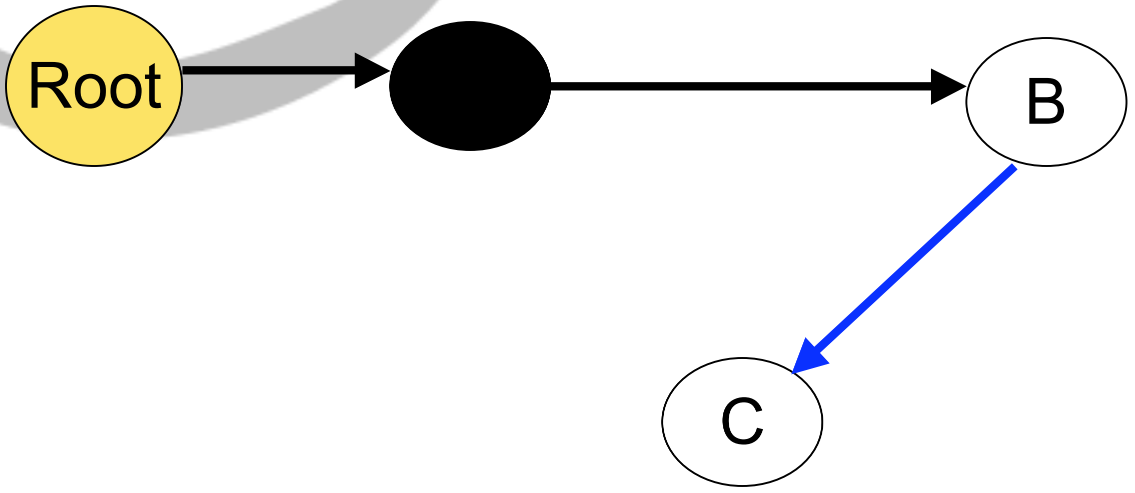
First Coarse-Grained Solution

- Counter-Example shows that goal of “no extra mutator overhead” is unattainable.



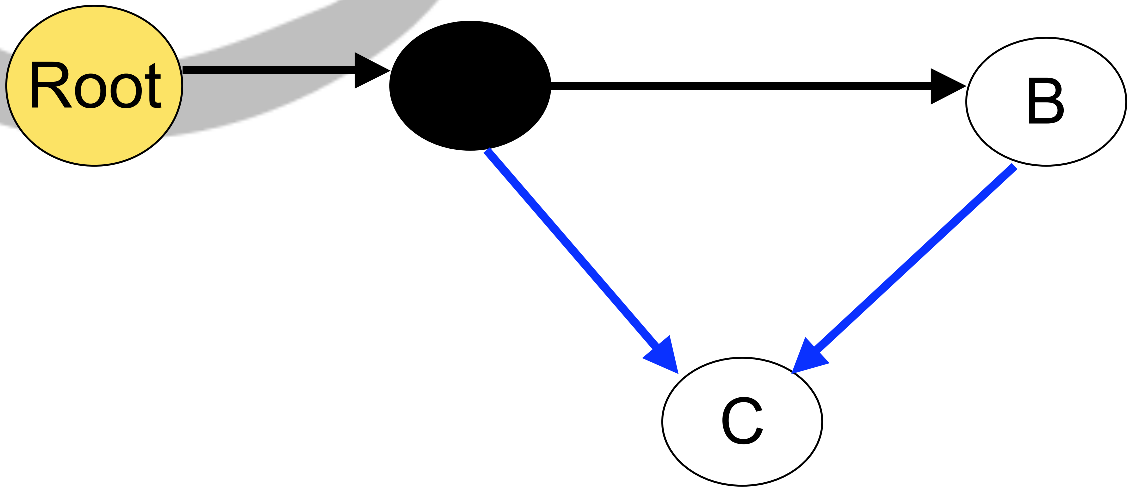
First Coarse-Grained Solution

- Counter-Example shows that goal of “no extra mutator overhead” is unattainable.



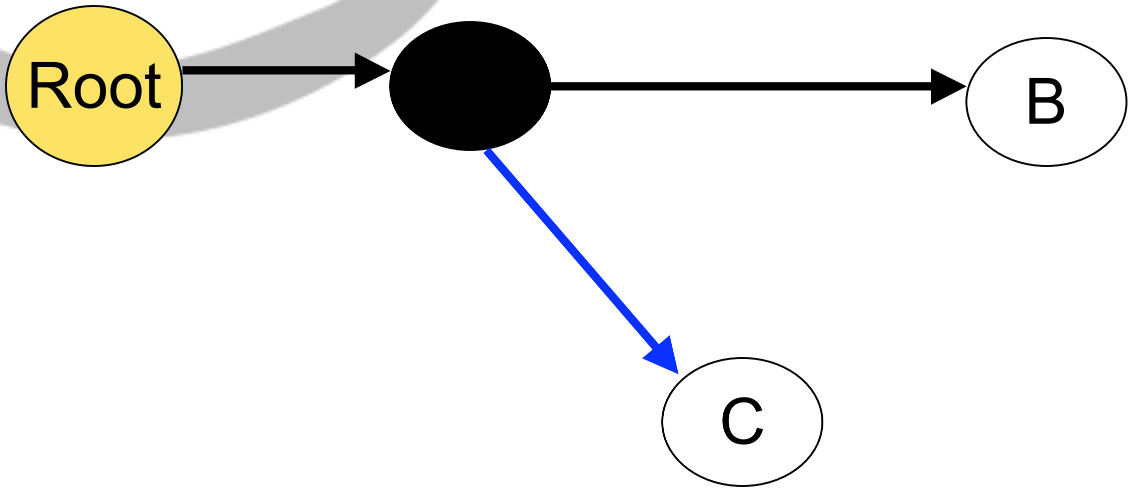
First Coarse-Grained Solution

- Counter-Example shows that goal of “no extra mutator overhead” is unattainable.



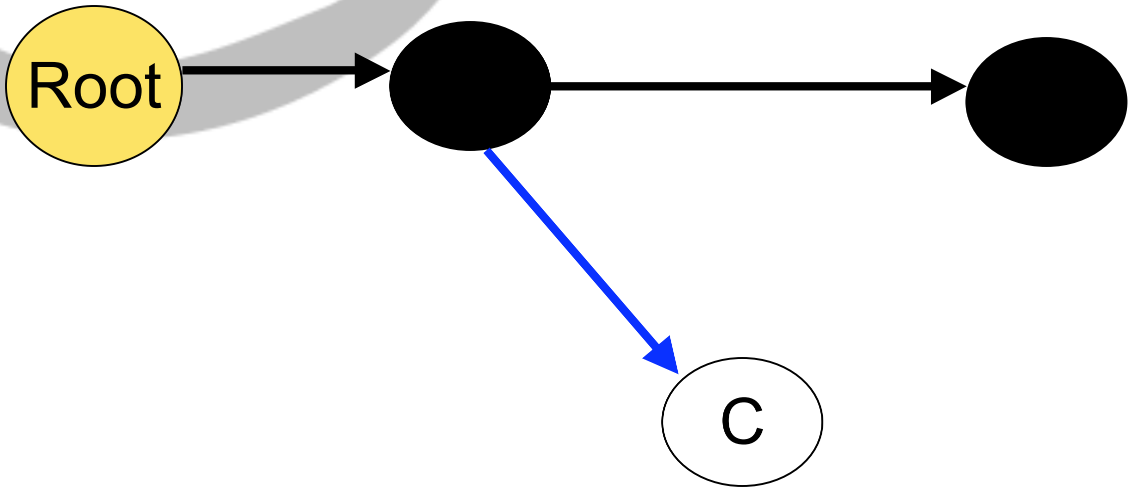
First Coarse-Grained Solution

- Counter-Example shows that goal of “no extra mutator overhead” is unattainable.



First Coarse-Grained Solution

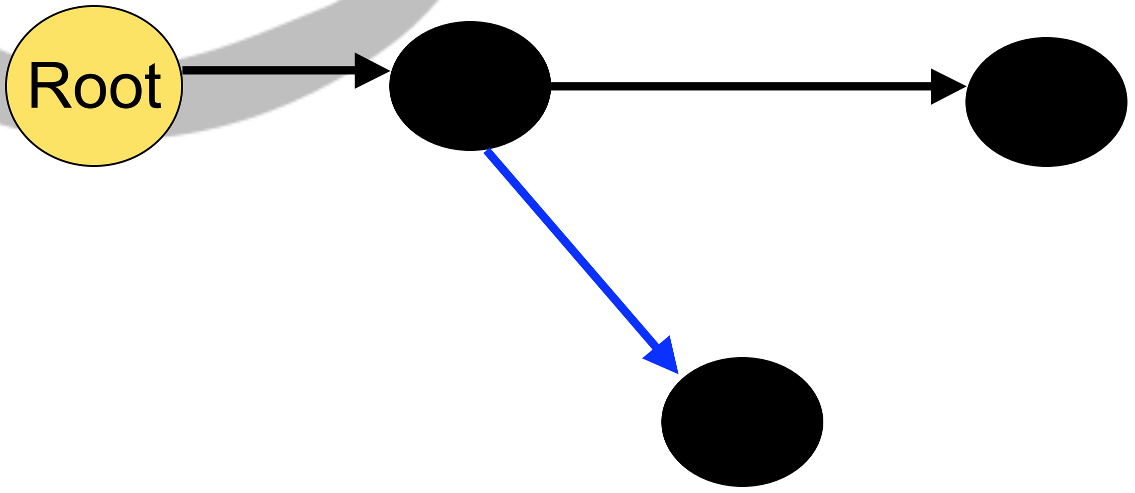
- Counter-Example shows that goal of “no extra mutator overhead” is unattainable.



<u>Now C has not been marked by collector and it is garbage !!!!</u>

First Coarse-Grained Solution

- Counter-Example shows that goal of “no extra mutator overhead” is unattainable.



Solution: ***Mutator*** *may have to mark target of edge it redirects. So how does it mark a node?*

First Coarse-Grained Solution

- Marking described in colors
- WHITE : *Garbage or not yet marked*
- **Black :** Marked and reachable
- *Gray :* Intermediate color given to current node that is being marked. Signifies reachability.

- Burdens of Proof:
  - **Progress**: Based on a monotonicity argument
    - No Node will become a lighter color
  - **Safety**: Based on invariant that is true and will not be destroyed during repetition
    - *P1: No edge points from a black node to a white one*

First Coarse-Grained Solution

- *Gray :* Marking a node gray is a way of protecting the mutator and/or collector from not violating invariant, P1
- A **coarse grained mutator** will repeatedly "shade a node" by making it gray if it is white and leaving it unchanged if it is already gray or black.
- A **coarse grained collector** will also make all reachable nodes black while keeping P1 an invariant during its marking phase.
- The **marking phase terminates** when all gray nodes are black
- Thus, we may change *(1)* to atomic operation *M1*
- *M1: <redirect an outgoing edge of a reachable node towards an already reachable one, and shade the new target>*

Remember:
P1 No edge points from a black node to a white node
(1) Redirect an outgoing edge of a reachable node towards another already reachable node.

First Coarse-Grained Solution

The Garbage collector’s **Append Phase** terminates when all black **nodes have been changed to white** and all previously **white nodes have been appended to free list.**

```
appending phase:
do i < M -> {all nodes with a number < i are nonblack; all nodes with
     a number >= i are nongray, and are garbage, if white}
     <c:= color of node nr.i>:
     if c = white -> <append node nr.i to the free list>
     |  c = black -> <make node nr.i white>
     fi;
     i := i + 1
od {there are no black nodes}
```

Remember that all these guarantees are to make sure that

1. GC is not leaking memory
2. Processors are playing nice and not giving the mutator unecessary overhead

First Coarse-Grained Solution

- To satisfy these goals we have to demonstrate that CC1 and CC2 are being met
- To prove CC2 we must show that ***a white node with a number >= i is garbage***

1. is an invariant

Proof: The only way a violation happens is by making a nongarbage node white, since i increases, this never happens

2. Holds when the collector enters its append cycle

Remember:
CC1 Every garbage node is eventually appended to the free list
CC2 Appending a garbage node to the free list is the collector’s only modification of the shape of the data structure
P1 No edge points from a black node to a white node

First Coarse-Grained Solution

2. ***a white node with a number >= i is garbage*** holds when the collector enters its append cycle

   Proof: We must show all white nodes are garbage.

   - *All nodes are white before marking. This implies P1 holds.*
   - *Repeated use of M1 is safe so this also preserves P1*
   - *Since the mark phase leaves no gray nodes available and leaves all reachable nodes black this means all white nodes are garbage.*
   - *So the white node with a number >= i is garbage*
   - *Thus, CC2 holds*

Remember:

CC1 Every garbage node is eventually appended to the free list

CC2 Appending a garbage node to the free list is the collector's only modification of the shape of the data structure

P1 No edge points from a black node to a white node

M1 <redirect an outgoing edge of a reachable node towards an already reachable one and shate the new target>

First Coarse-Grained Solution

We must now prove CC1 by showing that both phases (mark & append) terminate properly

Proof of Append's Termination:

- At the end of the marking phase there are no gray nodes and every white node is garbage.
- Since the mutator can't shade a garbage/white node, and shading a black node has no effect, then every node is black or white during the append phase.
- So append is sure to end when there are no black nodes left because there are no gray nodes to leave us in ambiguity.

Remember:

CC1 Every garbage node is eventually appended to the free list

CC2 Appending a garbage node to the free list is the collector's only modification of the shape of the data structure

P1 No edge points from a black node to a white node

M1 <redirect an outgoing edge of a reachable node towards an already reachable one and shade the new target>

First Coarse-Grained Solution

We must now prove CC1 by showing that both phases (mark & append) terminate properly

Proof of Marking Termination:

- Before the marking cycle begins, all roots are shaded.
- Once the marking cycle begins
  - If current node == root, then color black and shade all successors because root has already been shaded gray
  - If current node != root, then color black and shade all successors because current node has already been shaded from previous cycle iteration
- After last cycle iteration no nodes are gray, so either they have been touched and are black or they are white and hence garbage.
- Since there are no gray nodes, then marking cycle ends
- Note: Since white nodes are garbage, then there is no edge from black node to white node and P1 has kept its invariance

CC1 is shown since marking phase ends by guaranteeing all white nodes are garbage and append phase shows that all garbage is on free list and black nodes that do remain are turned into new white nodes for the next GC.

Remember:
CC1 Every garbage node is eventually appended to the free list
P1 No edge points from a black node to a white node

First Coarse-Grained Solution

- **Since both CC1 and CC2 are met**,
- **Discussion: Why does proving CC1 and CC2 ensure correctness?**
- P1 was kept invariant during the marking cycle making the solution course-grained.
- In the **Fine-Grained solution, the authors also try and keep P1 as an invariant** and split M1 into two atomic operations
  - One for edge redirection
  - One for shading a new target
- **However, the finer grain solution contains a bug.**
  - There is an edge from node A to node B and the mutator shades B
  - A collector's full mark and append phase *eventually changes B to white!*
  - Another collector's mark/append phase takes B and **erroneously appends B** to the free list even though it was supposed to be reachable by the mutator
- We must come up with another coarse-grained mutator that will fix this bug on our fine-grained solution.
- **Discussion: Why does this bug exist?**

Remember:
CC1 Every garbage node is eventually appended to the free list
CC2 Appending a garbage node to the free list is the collector's only modification of the shape of the data structure
P1 No edge points from a black node to a white node
M1 <redirect an outgoing edge of a reachable node towards an already reachable one and shade the new target>

Second Coarse-Grained Solution

P1 must be replaced by weaker invariants P2 and P3.

Definition: Propagation Path = A path consisting of only white targets, starting with gray source nodes

- *P2 "for each white reachable node, there is a propagation path leading to it"*
  - *Corollary 1: If each root is gray or black, the absence of edges from black to white implies that P2 is satisfied*

Discussion: Why is P2 satisfied from corallary 1?

- *P3 "only the last edge placed by the mutator may lead from a black node to a white one"*
  - *Corollary 2: If there are no black nodes, P3 is trivially satisfied. Hence it holds at the beginning of the marking cycle.*
- Both collector phases will leave P2 and P3 as invariants in tact.
- Now that we have P2 and P3 defined, we must find a mutator operation that leaves them as invariants

Remember:
P1 No edge points from a black node to a white node
M1 <redirect an outgoing edge of a reachable node towards an already reachable one and shade the new target>

Second Coarse-Grained Solution

M2: <shade the target of the previously redirected edge and redirect an outgoing edge of a reachable node towards a reachable node>

- M2 prevents previous edge to have white target by shading old target, **thus sustaining P3**
- Every white reachable node which had a propagation path before M2, has a propagation path after M2.
- If the node whose successor is redefined as black then the outgoing edge was not part of propagation path, so no path needs to be maintained after M2.
- **So P2 holds.**

Discussion: Why does establishing a new M2 help lead us to a fine grained solution?

Remember:
P1 No edge points from a black node to a white node
P2 "for each white reachable node, there is a propagation path leading to it"
P3 only the last edge placed by the mutator may lead from a black node to a white one
M1 <redirect an outgoing edge of a reachable node towards an already reachable one and shade the new target>

Fine Grained Solution

- We will use the second grained solution as a stepping stone.
- For fine grained mutator **we must split up M2** into ..
  - *M2.1 <shade the target of the previous redirected edge>*
  - *M2.2 <redirect an outgoing edge of a reachable node towards a reachable node>*
- In the previous mark phase, it repetitively used an atomic operator …
  - *C1: <shade the successors of node nr.i and make node nr.i black>*
- In the fine grained solution **we must split C1**
  - C1.1: <m1:= number of the left-hand successor of node nr.i>
  - C1.2: <shade node nr.m1>
  - C1.3: <m2:= number of the right-hand successor of node nr.i>
  - C1.4: <shade node nr.m2>
  - C1.5: <make node nr.i black>
- We can simplify C1.1 - C1.4 with two rules
  - C1.1a: <shade the left-hand successor of node nr.i>
  - C1.3a: <shard the right-hand successor of node nr.i>

Remember:
M2: <shade the target of the previously redirected edge and redirect an outgoing edge of a reachable node towards a reachable node>

Fine Grained Solution

- Thus, we can replace C1 with the final fine grained set of atomic actions
- In the fine grained solution **we must split C1**
  - C1.1a: <shade the left-hand successor of node nr.i>
  - C1.3a: <shard the right-hand successor of node nr.i>
  - C1.5: <make node nr.i black>
- Also **to prove correctness of the fine-grained solution** we must **prove that P2 and P3** are still invariants
- We prove P2 and P3, by proving the invariance of a stronger relation.

Remember:
C1: <shade the successors of node nr.i and make node nr.i black>
P2 "for each white reachable node, there is a propagation path leading to it"
P3 only the last edge placed by the mutator may lead from a black node to a white one

Fine Grained Solution

- To prove P2 and P3, we must first define a notion called C-edges whose sources have been detected gray by the marking cycle.
- *P2a: "every root is gray or black and for each white reachable node there is a propagation path leading to it, containing no C-edges"*
- *P3a: "there exists at most one edge 'E' satisfying*
  - *pr: "(E is a black-to-white edge)* ***or***

    (E is a C-edge with a white target);"
- Proof:
  - P2a holds from corollary 1 and the set of C-edges is empty at the start of the mark cycle.
  - **Discussion: Why is this valid?**
  - P3a trivially holds at start of mark cycle because no E edges exist because no C-edges and no black nodes.

Remember:

Corollary 1: If each root is gray or black, the absence of edges from black to white implies that P2 is satisfied

C1: <shade the successors of node nr.i and make node nr.i black>

P2 "for each white reachable node, there is a propagation path leading to it"

P3 only the last edge placed by the mutator may lead from a black node to a white one

Fine Grained Solution

- To prove P2 and P3, we must first define a notion called **C-edges whose sources have been detected gray by the marking cycle.**
- *P2a: "every root is gray or black and for each white reachable node there is a propagation path leading to it, containing no C-edges"*
- *P3a: "there exists at most one edge 'E' satisfying*
  - *pr: "(E is a black-to-white edge)* ***or***
    (E is a C-edge with a white target);"
- Proof:
  - **P2a holds** from corollary 1 and the set of C-edges is empty at the start of the mark cycle. **Discussion: Why is this valid?**
  - **P3a trivially holds** at start of mark cycle because no E edges exist because no C-edges and no black nodes.
  - If M2.1, C1.1a, C1.3a, C1.5 are executed, any propagation path that exists before the operation will exist after. **So P2a holds.**
  - Additionally, the operations in question do not add a black to white edge or a C-edge with white target, **so P3a holds.** (*note: C1.5 may introduce a black to white edge, but that edge was already a C-edge with a white target)*

Remember:
Corollary 1: If each root is gray or black, the absence of edges from black to white implies that P2 is satisfied
M2.1 <shade the target of the previous redirected edge>
C1.1a: <shade the left-hand successor of node nr.i>
C1.3a: <shard the right-hand successor of node nr.i>
C1.5: <make node nr.i black>
P2 "for each white reachable node, there is a propagation path leading to it"
P3 only the last edge placed by the mutator may lead from a black node to a white one

Fine Grained Solution

- Proof of P2a and P3a (cont.):
- **P3a holds** because if it held before M2.2 it will hold after
- Also **P2a holds** because
  - M2.2 does not create C-edges
  - if C-edge or black source existed before M2.2, it does not belong to propagation path without C-edges.

- **Since P2a and P3a hold for both mutator and collector, we have proven our fine grained solution**.
- **Since we prove our fined grain solution, this means that CC1 and CC2 hold and that we have successfully forced our two processors to cooperate with each other !!!!**

Remember:
Corollary 1: If each root is gray or black, the absence of edges from black to white implies that P2 is satisfied
M2.1 <shade the target of the previous redirected edge>
M2.2 <redirect an outgoing edge of a reachable node towards a reachable node>
P3a: “there exists at most one edge ‘E’ satisfying pr: “(E is a black-to-white edge) or (E is a C-edge with a white target);”
P2a: “every root is gray or black and for each white reachable node there is a propagation path leading to it, containing no C-edges”
C-edges whose sources have been detected gray by the marking cycle