

# Correctness-Preserving Derivation of Concurrent Garbage Collection Algorithms

By Vechev, Yahav, and Bacon

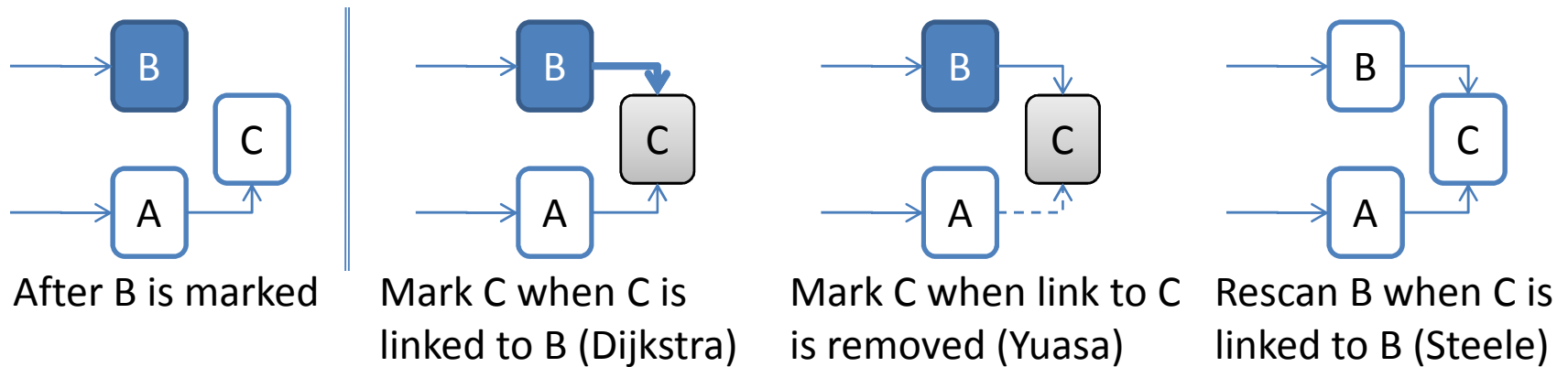
Presented by Sam Harwell

# Outline

- A parametric concurrent collector
- Apex algorithm
- Correctness-preserving transformations
- Further work
- Discussion

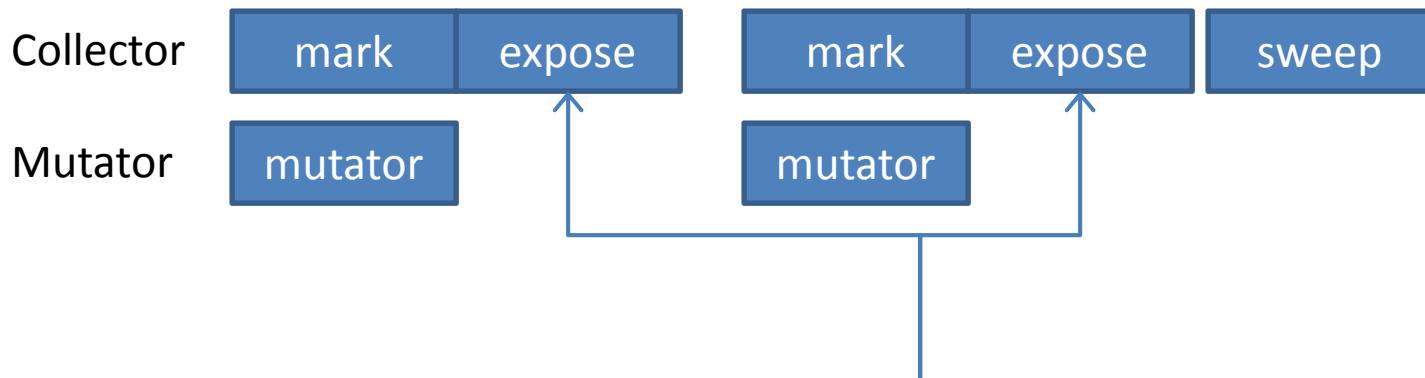
# Concurrency and Hidden Objects

- A concurrently running mutator can “hide” objects from the collector
- Three primary algorithms to address this



# Parametric Concurrent Collector

- A single model for concurrent collectors
- Algorithm differences are encapsulated in the expose function



expose atomically adds items to the mark queue that were altered by the mutator while marking in a way that could have hidden them.

# Apex Algorithm

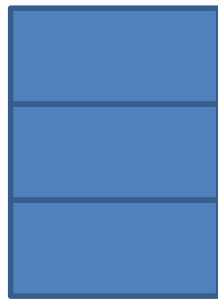
- An instance of the parallel concurrent collector
- Goals
  - As easy as possible to show correctness
  - Well-defined as a base for transformations
- Characteristics
  - Sacrifices performance for precision and clarity
  - Implements expose as rescanning, similar to Steele

# Dimensions: Algorithm Parameters

- **Wavefront:** how far has the collector progressed?
- **Policy:** how are modified objects behind the wavefront handled?
- **Threshold:** what is the maximum value for a cross-wavefront count?
- **Protection:** which objects must be traced to ensure all live objects are found?
- **Allocation:** how are newly allocated objects handled?

# Wavefront Dimension

- If the fields of object  $o$  are tracked precisely in correct algorithm, it is correct but less precise to track  $o$  whole object instead.



Per-Field Wavefront  
-Exact information  
-More expensive  
-More garbage collected



Per-Object Wavefront  
-Approximate information  
-Less expensive  
-Less garbage collected

# Policy Dimension

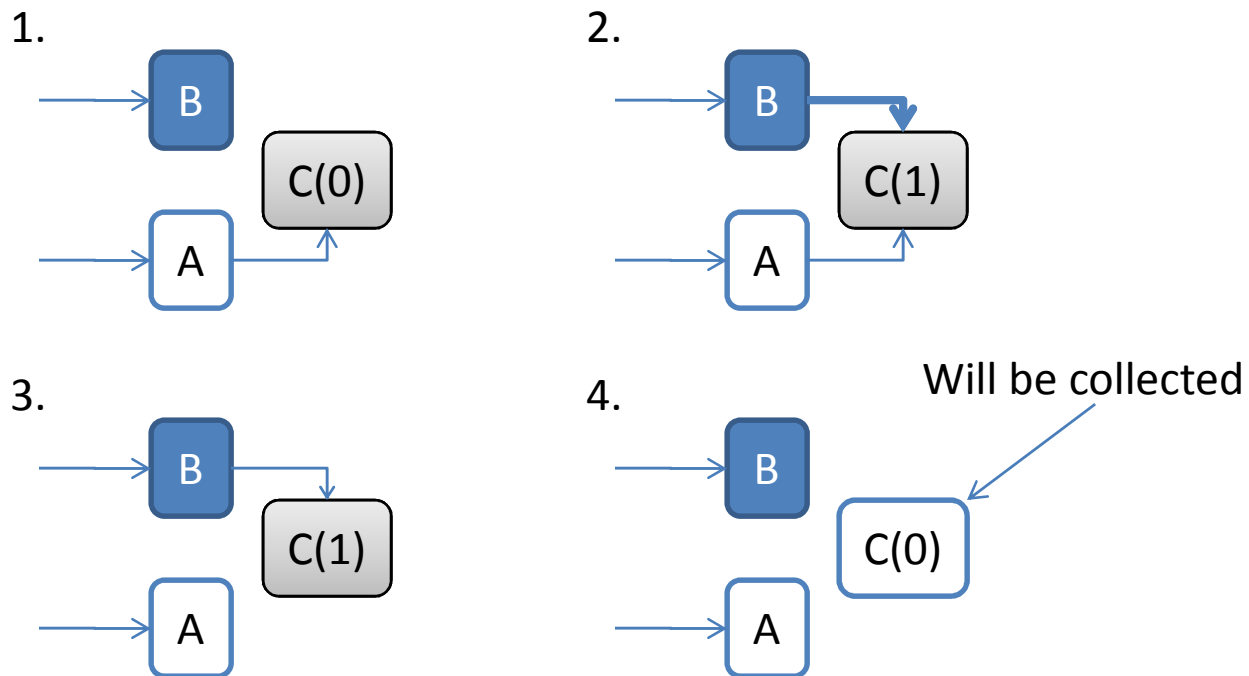
- Modifications to a field can be found by:
  - Atomically scanning the heap (SR)
  - Scanning the log (LR)
- Moving items from SR to LR is a correctness-preserving and precision-reducing (CPPR) transformation

# Threshold Dimension

- Counts cross-wavefront (behind->ahead) pointers
- Only objects with positive counts are added to the pending mark queue
- Threshold: once an object's count reaches a fixed  $C$ , it is treated as having a count of  $\infty$  and always added to the queue
- Increasing the threshold is a CPPR transformation

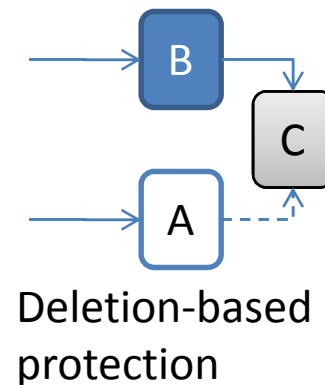
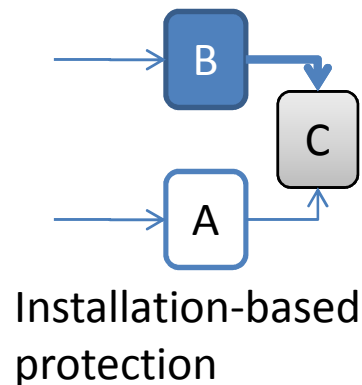
# Threshold Dimension

- If the count can be greater than one, then C might be collected even after it is store in the mark queue



# Protection Dimension

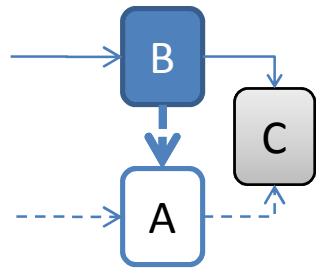
- Moving an item from installation-based protection (IS) to deletion-based protection (DS) is a weakly precision-reducing and correctness-preserving transformation



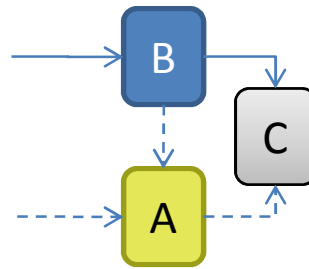
# Allocation Dimension

- Apex allocates objects ahead of the wavefront (white)
- Define a new, unmarked *yellow* state that is treated as behind the wavefront for any IS pointers stored into the object
- Allocating an object yellow instead of white is a CPPR transformation
- Allocating an object black instead of yellow is a further CPPR transformation, and is used in Metronome

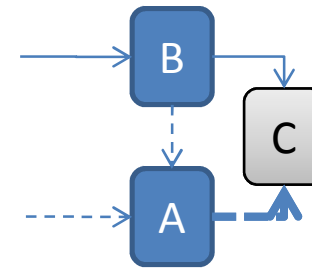
# Allocation Dimension



Allocate white



Allocate yellow



Allocate black

- > Previous pointer
- - -> New pointer, cross-wavefront count incremented
- - - -> New pointer, no need to increment

# Future Work

- Automatically apply combinations of the various transforms and examine the result
- Relax atomicity constraints, such as assuming the expose update is atomic
- Shown at a conference in 2007

# Discussion

- Completeness?
- Coverage of the problem domain?
- Termination?