Starvation-Free Heap Size for Replication-Based Incremental Compacting Garbage Collection

Tomoharu Ugawa (The University of Electro-Communications)
Hideya Iwasaki (The University of Electro-Communications)
Taiichi Yuasa (Kyoto University)

Starvation-Free Heap Size for Replication-Based Incremental Compacting Garbage Collection

Tomoharu Ugawa (The University of Electro-Communications)
Hideya Iwasaki (The University of Electro-Communications)
Taiichi Yuasa (Kyoto University)

Motivation

- Real-time GC is indispensable for real-time applications
- Real-time GC requires much memory if the application allocates objects of various sizes
  - Real-time mark-sweep GC
    → suffer from fragmentation
  - Real-time semi-space copying GC
    → twice as much memory as live objects
  - Real-time mark-compact GC
    → large overhead (execution time)

Partial Compaction

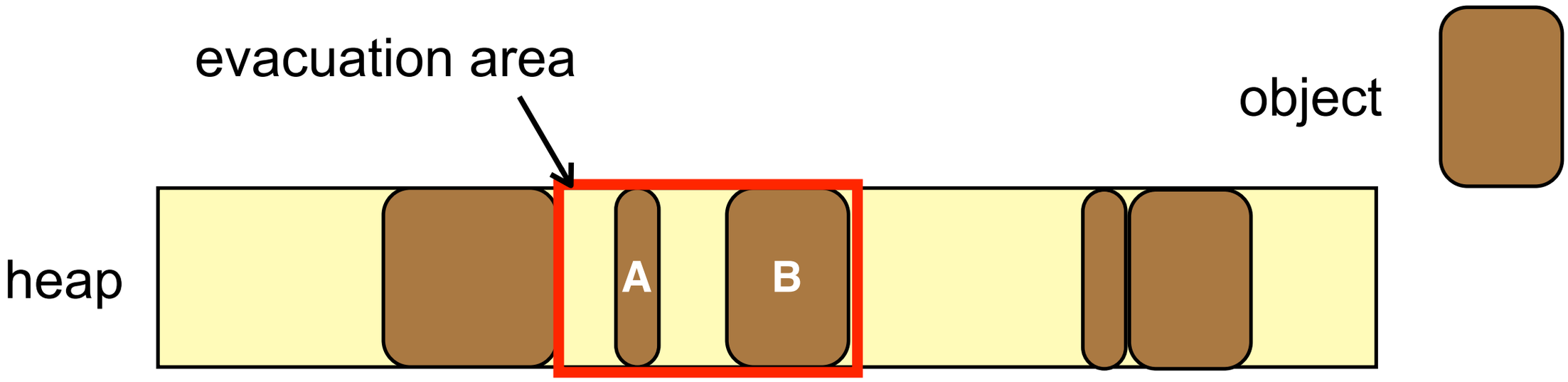
We developed a “partial compaction” by evacuation

- To be combined with mark-sweep GC
- Evacuate only a part of the heap
  - to create a large contiguous free area
  - to eliminate fragmentation

Partial Compaction

after mark-sweep GC,

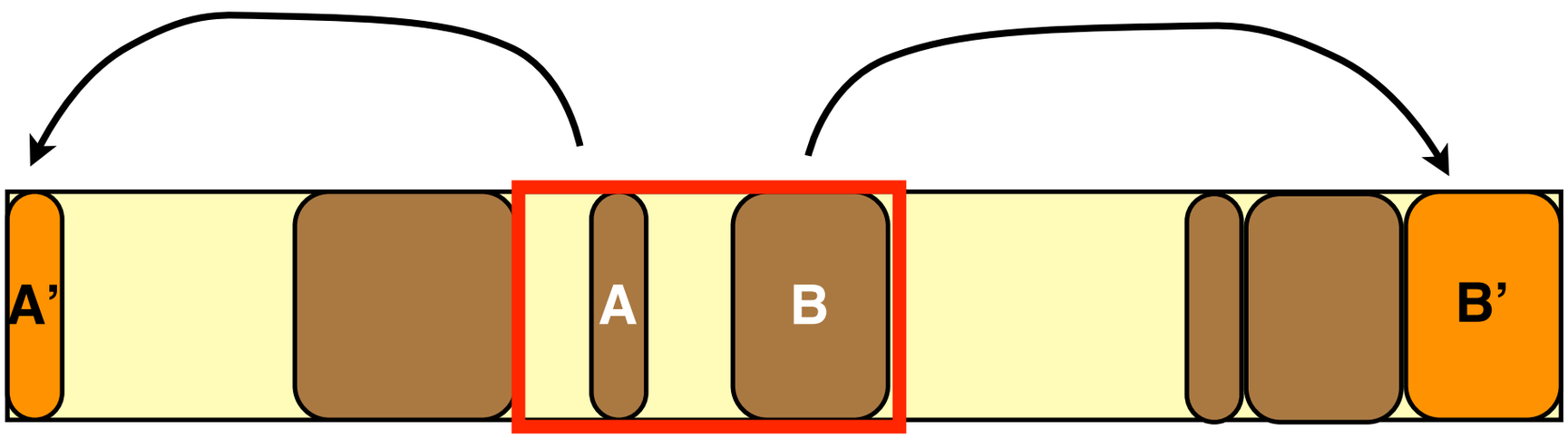
1. Choose an area as the evacuation area (EA)
2. Relocate objects in the EA
3. Reclaim the EA as a large contiguous free area



Partial Compaction

after mark-sweep GC,

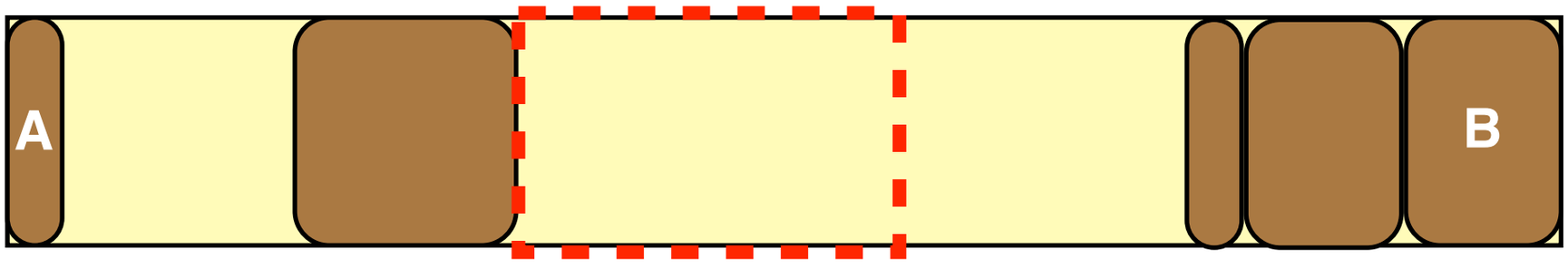
1. Choose an area as the evacuation area (EA)
2. Relocate objects in the EA
3. Reclaim the EA as a large contiguous free area



Partial Compaction

after mark-sweep GC,

1. Choose an area as the evacuation area (EA)
2. Relocate objects in the EA
3. Reclaim the EA as a large contiguous free area

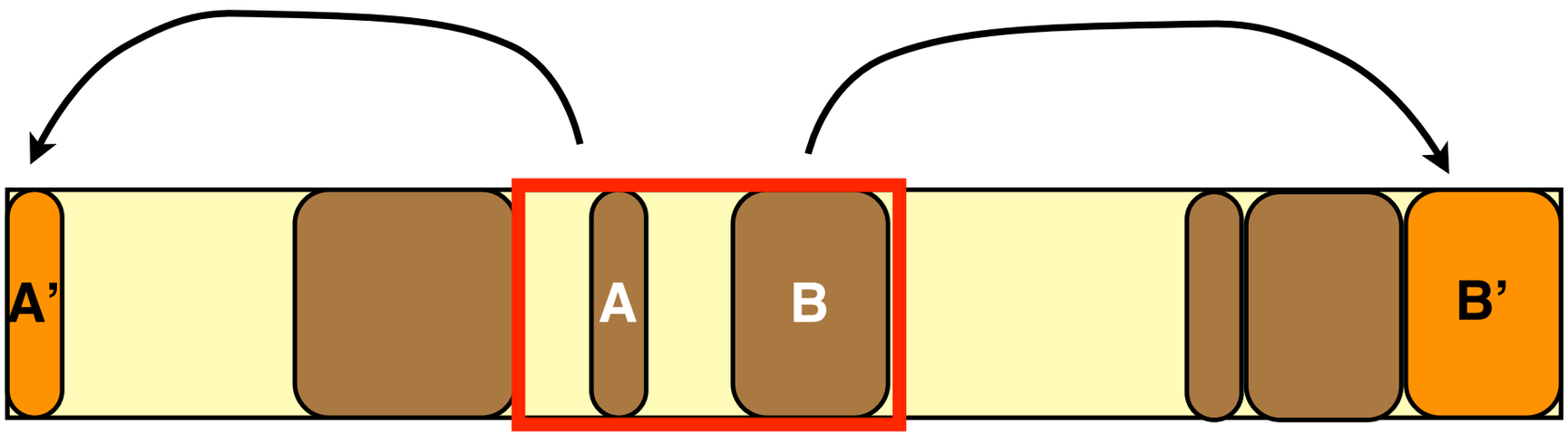


Required Memory

EA is only a part of the heap
→ space for copies can be
smaller than semi-space copying GC

Smaller EA

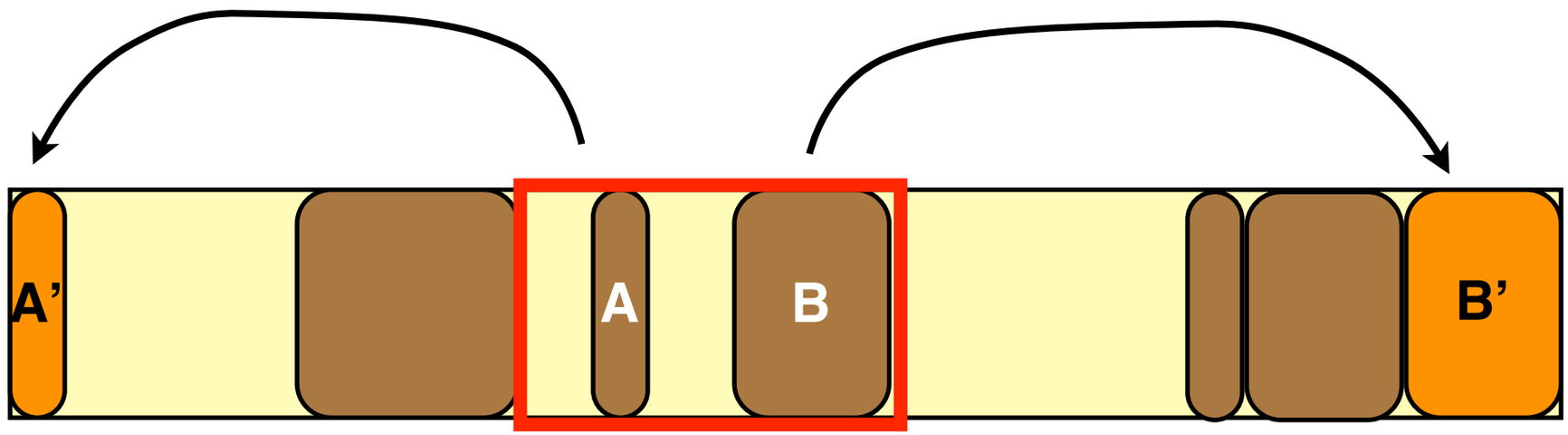
- smaller space for copies
- less effective for eliminating fragmentation



Problems

- How large the EA should be?
- Where in the heap should the GC choose the EA?
- How large the heap should be to avoid starvation?

starvation:
failure of allocation due to lack of memory or lack of contiguous memory



Outline

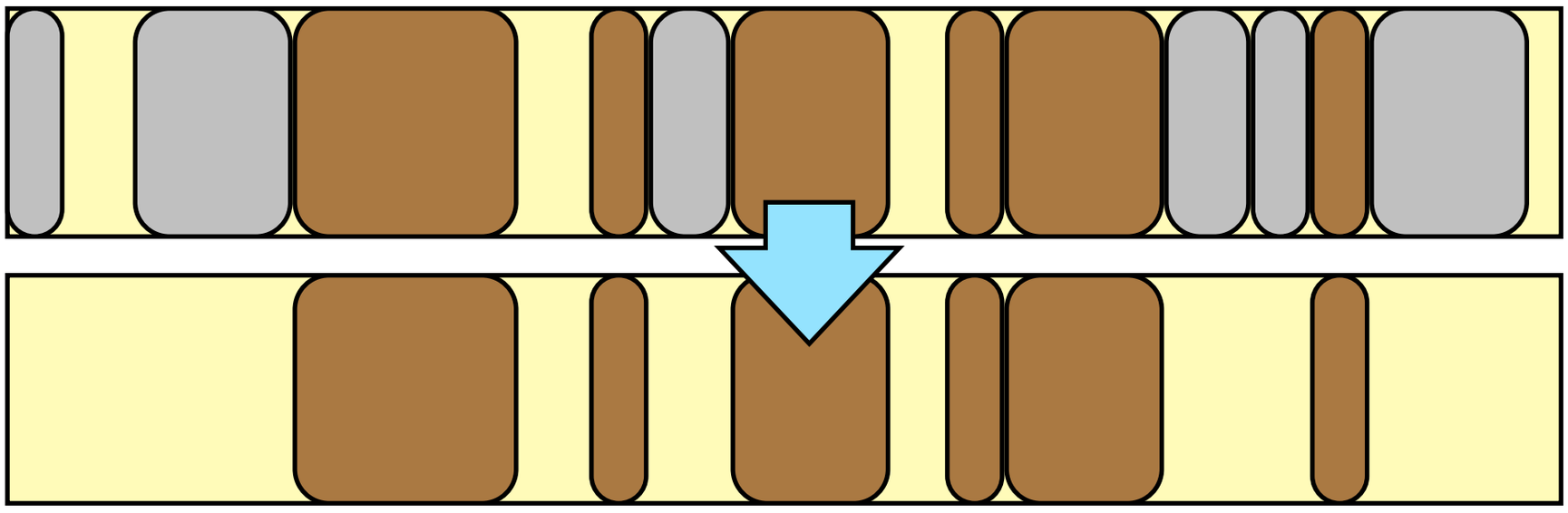
- Related work
- Model of application, allocator, and GC
- Basic Idea
  - Strategy
  - Analysis
- Practical setting
- Future work and summary

Related Work

Real-time mark-sweep GC

- incremental update [Steele ’75]
- on-the-fly [Dijkstra et al. ’78]
- snapshot-at-the-beginning [Yuasa ’90]

cannot bound heap size in cases
where objects of various sizes are allocated

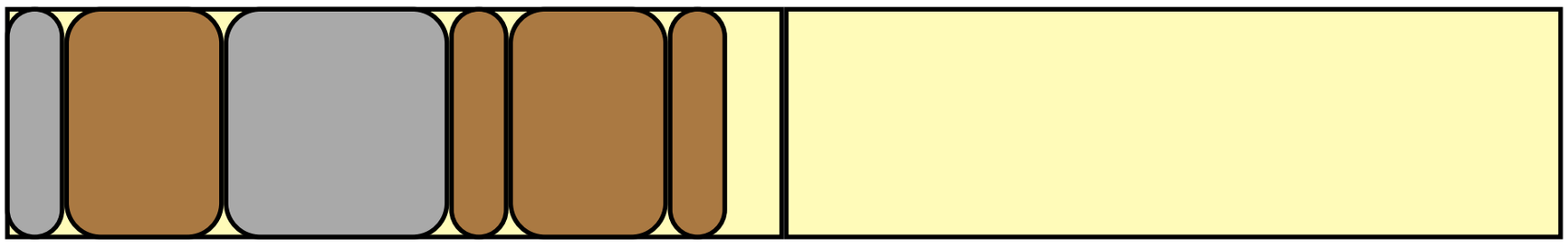


Related Work

Real-time semi-space copying GC

- with read barrier [Baker ’78]
- replication-based [Nettles and O’Toole ’93]
- in multi-processor settings [Cheng et al. ’01]

require more than twice as much memory as the amount of live objects



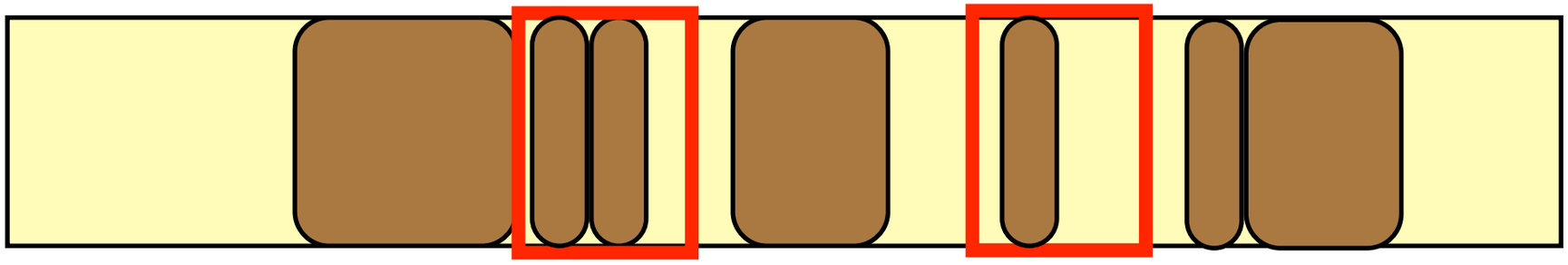
Related Work

Partial compaction by evacuation (1/3)

- Metronome [Bacon ’03]
- Immix [Blackburn and McKinley ’08]

choose several discontiguous areas as the EA

→ cannot allocate large memory chunk

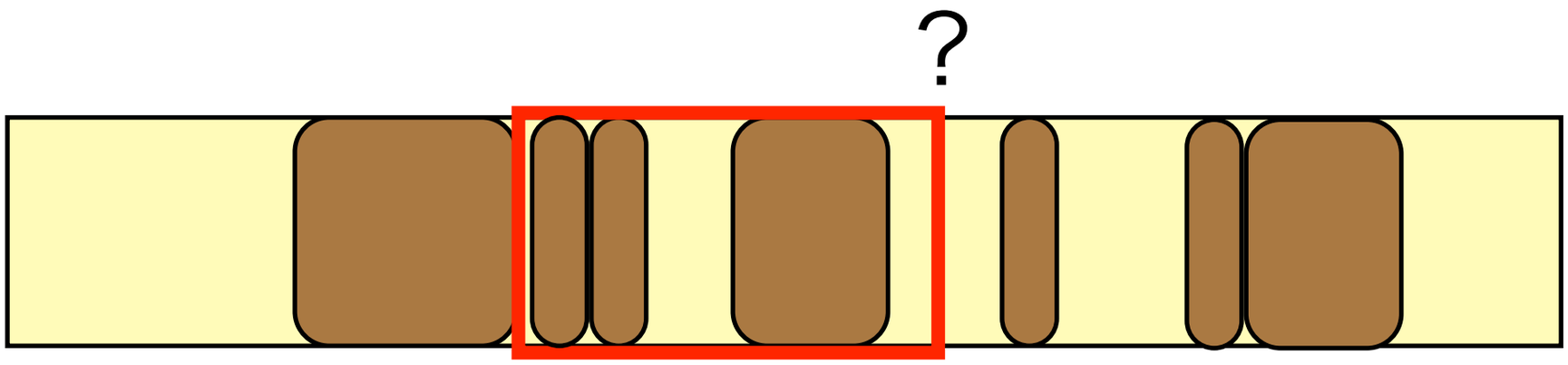


Related Work

Partial compaction by evacuation (2/3)

- Sapphire [Hudson and Moss ’01]
- Open VM [Kalibera ’09]

do not have any idea about where to choose the EA



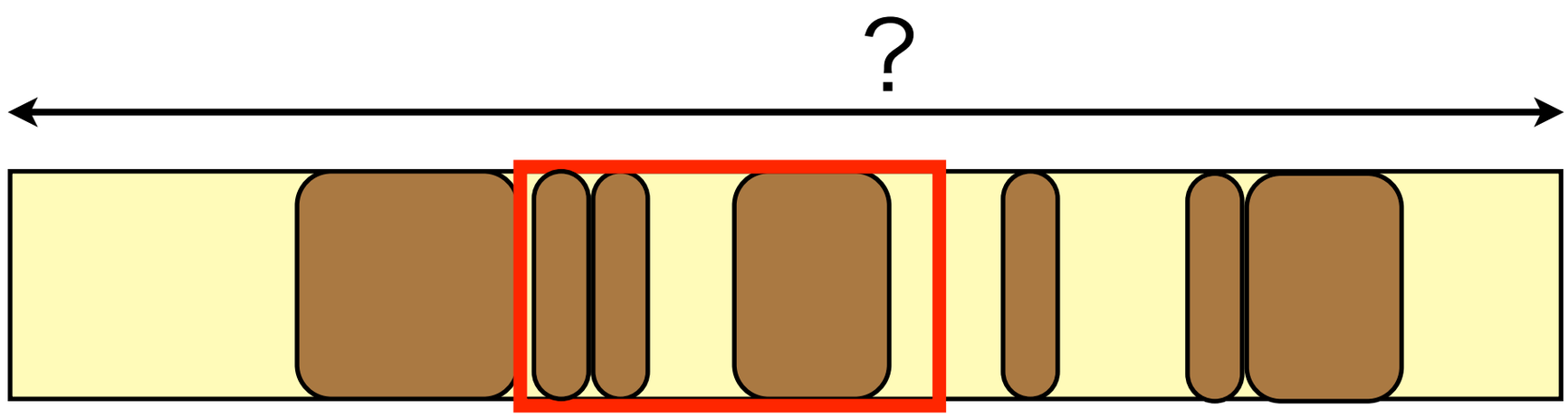
Related Work

Partial compaction by evacuation (3/3)

- Replication-based compaction [Ugawa et al. ’10]

chooses the EA that is effective for defragmentation

Cannot bound the heap size



Outline

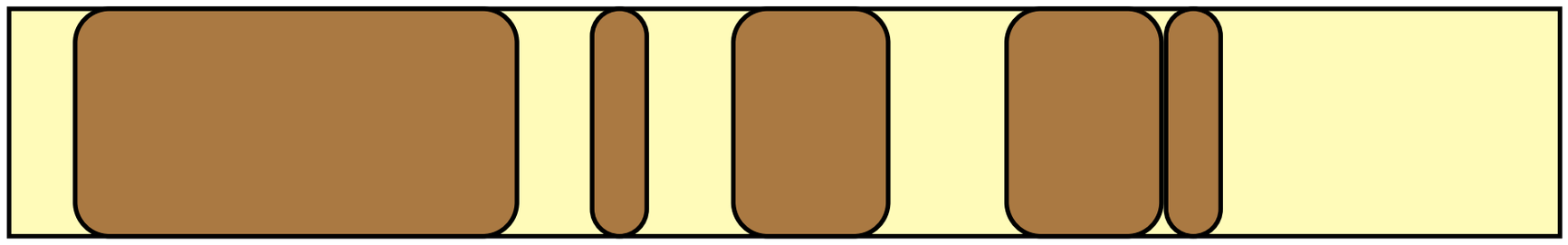
- Related work
- Model of application, allocator, and GC
- Basic Idea
  - Strategy
  - Analysis
- Practical setting
- Future work and summary

Application Model

- Application creates objects of different sizes
- All objects can be relocated
  - No ambiguous pointers
  - No pinned objects
- Max(amount of reachable (i.e. live) objects) ≤ R

Allocator Model

- A single “unstructured” heap
  - Not divided into pages or segments
    (page boundaries do not have significant meaning)
    - Objects can be allocated anywhere in the heap
- Every object occupies a contiguous memory area regardless of its size

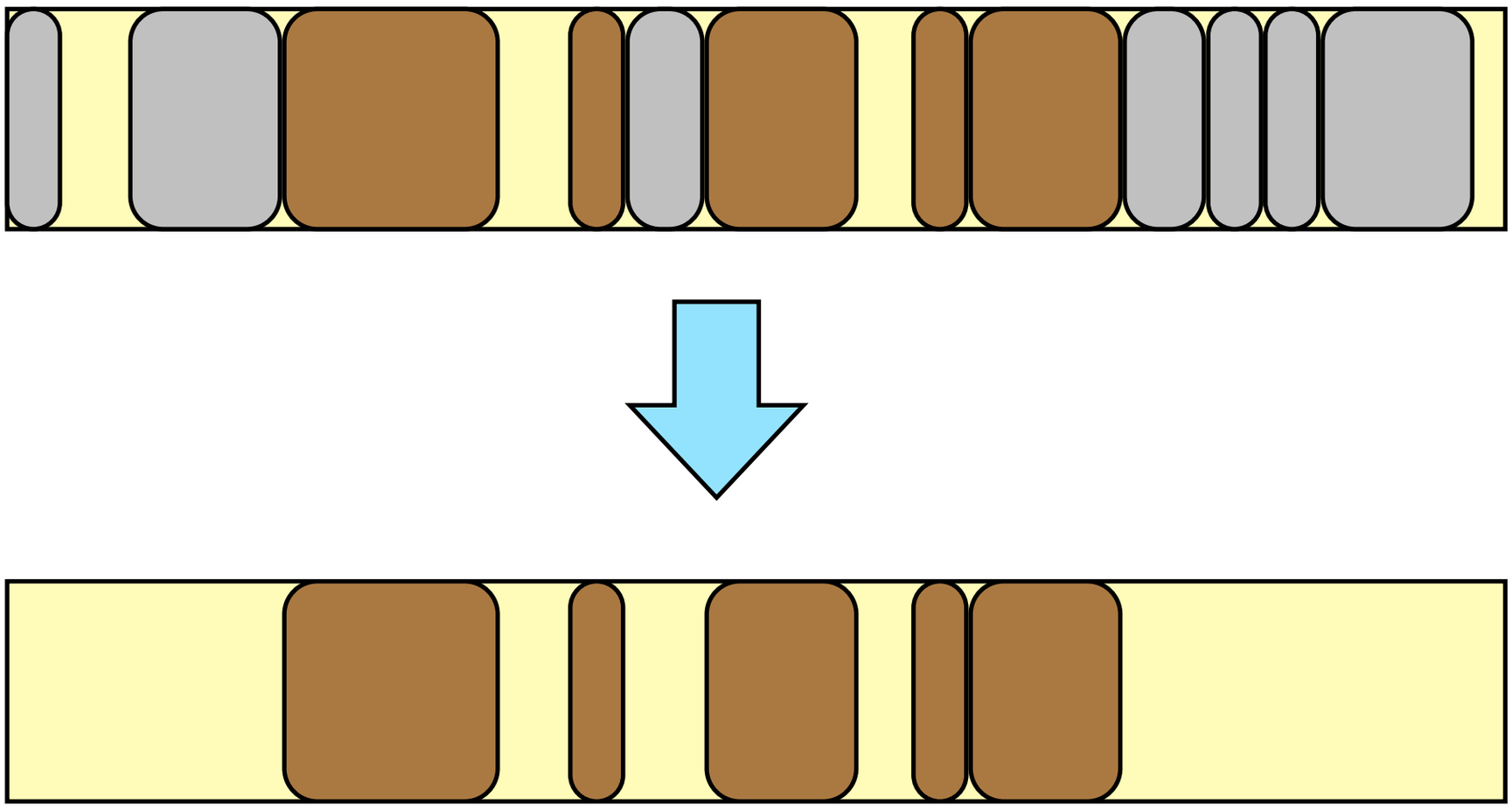


Collector Model

- Compaction after mark-sweep GC
  1. Mark phase
  2. Sweep phase
  3. Compaction phase
- Incremental GC
  - Perform GC little by little each time an object is allocated
  - Amount of allocation during a single GC cycle ≤ a

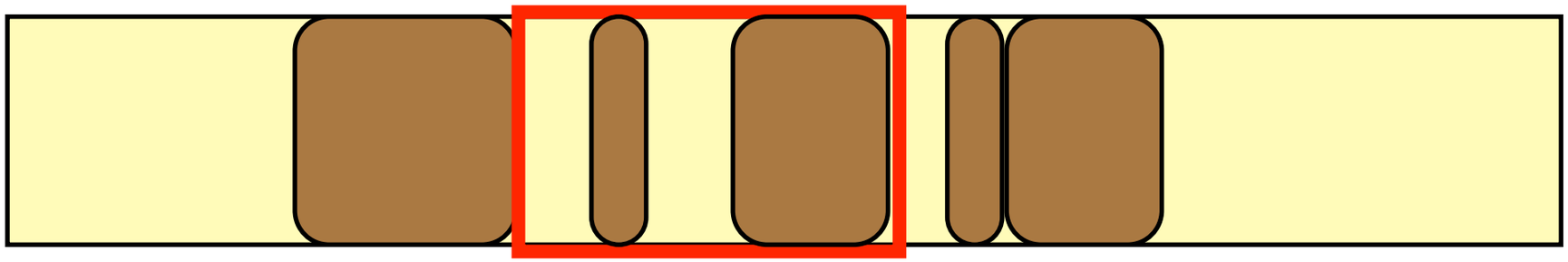
Before Compaction

- Mark and sweep phases collect garbage
  - Independent of barriers for incremental marking



Compaction Steps

Step 1: Choose the EA (location and size)

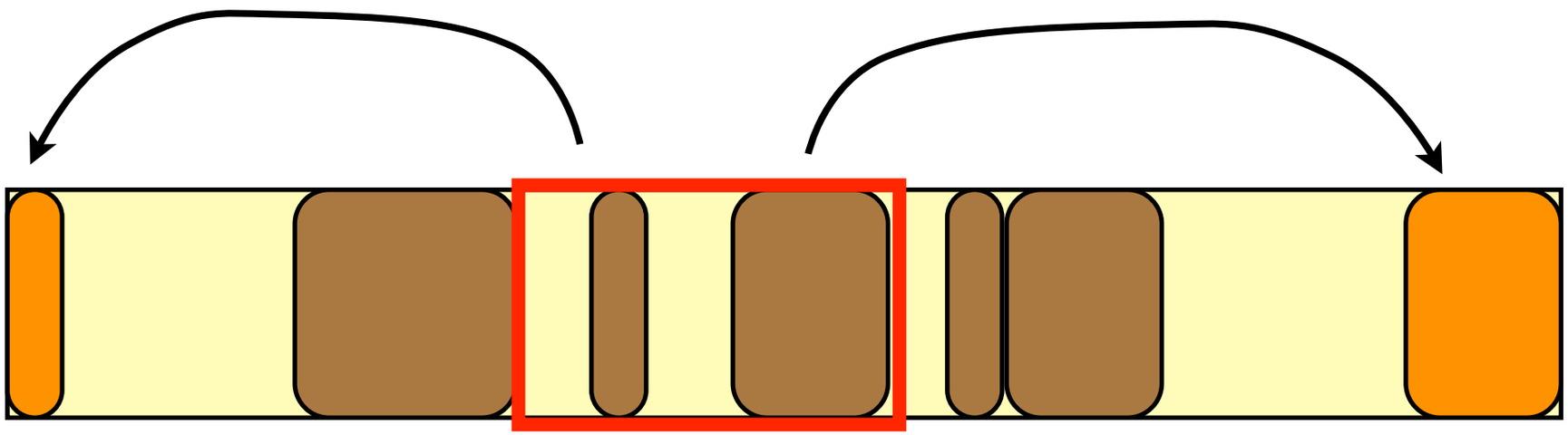


Compaction Steps

Step 1: Choose the EA (location and size)

Step 2: Relocate objects in the EA

- Independent of barrier for relocating incrementally



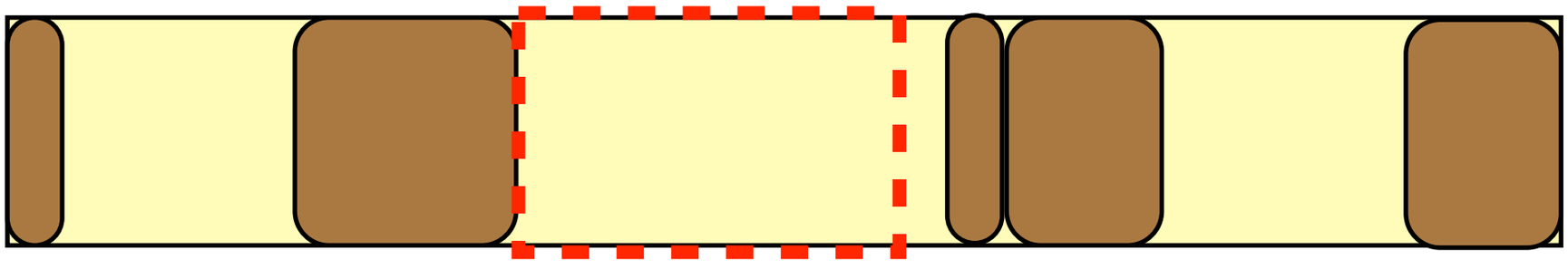
Compaction Steps

Step 1: Choose the EA (location and size)

Step 2: Relocate objects in the EA

- Independent of barrier for relocating incrementally

Step 3: Reclaim the EA as a large contiguous free area



Goal

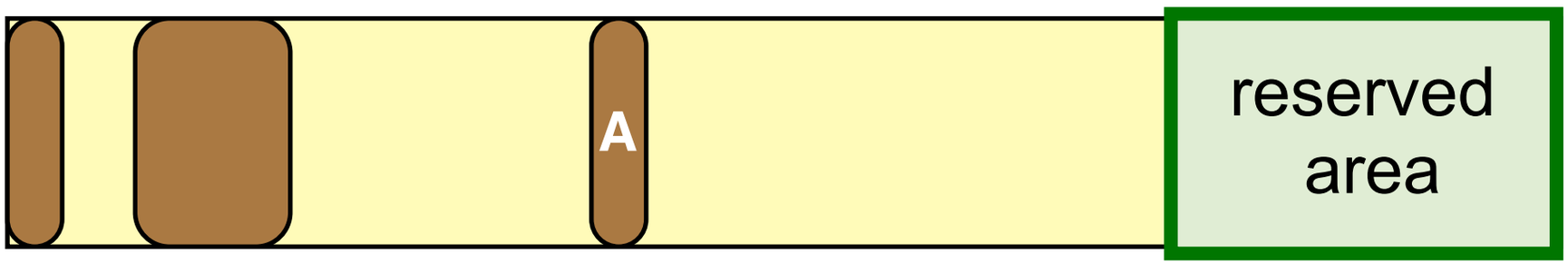
- Give a strategy for choosing the EA
- Estimate the required heap size to avoid starvation

Outline

- Related work
- Model of our GC
- Basic Idea
  - Strategy
  - Analysis
- Practical setting
- Future work and summary

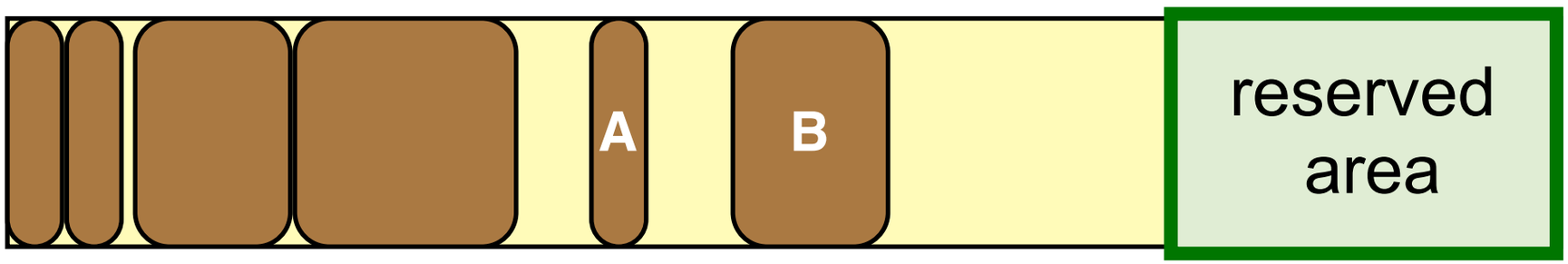
Reserved Free Area

- Reserve a contiguous free area for the next GC cycle
  - Allocation in the area is prohibited until the next GC cycle is triggered



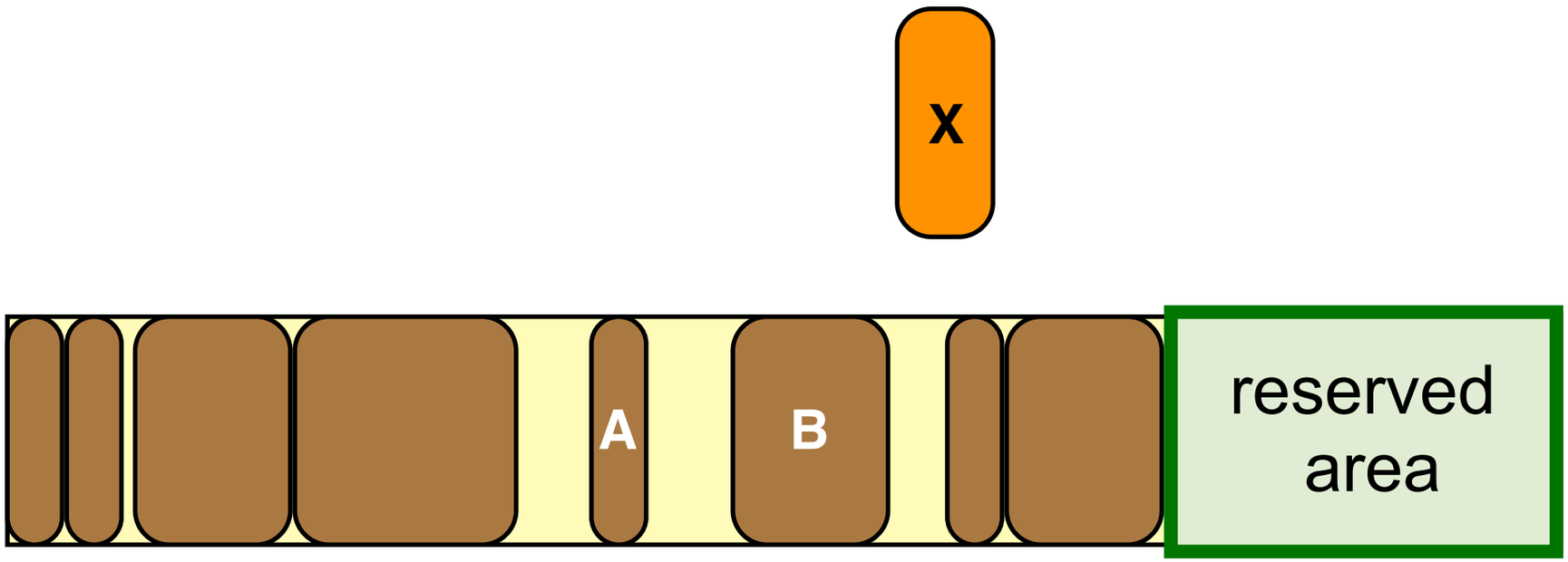
Reserved Free Area

- Reserve a contiguous free area for the next GC cycle
  - Allocation in the area is prohibited until the next GC cycle is triggered



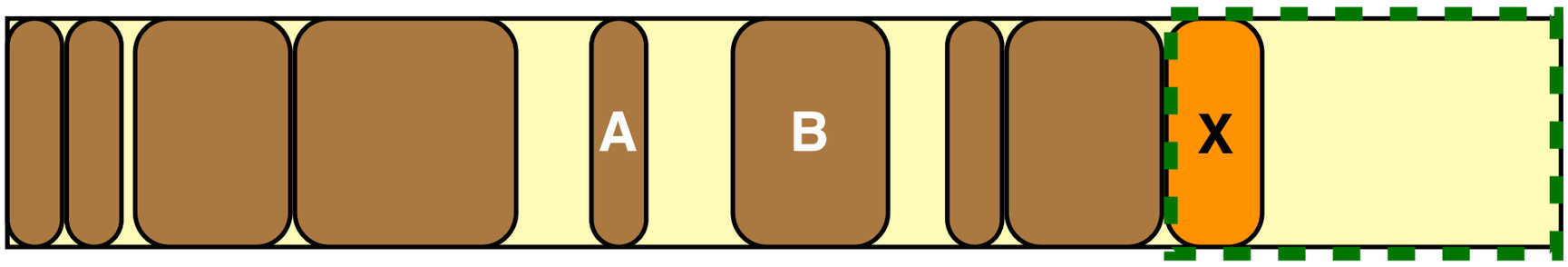
Trigger GC cycle

- When an allocation of an object, **x**, is failed
  - Allocate the object **x** in the reserved free area
  - Trigger the GC cycle



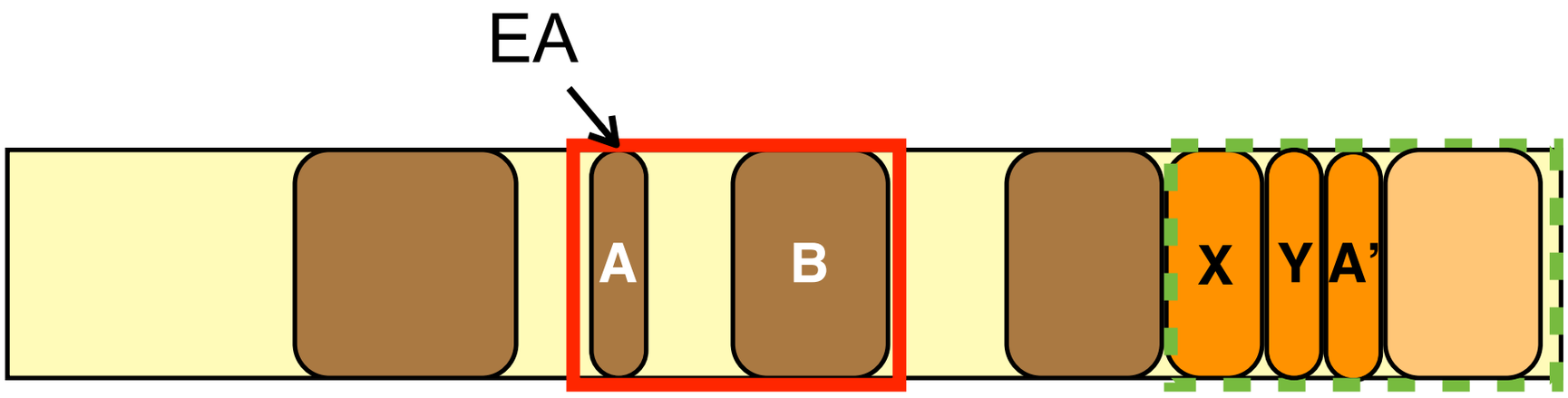
Trigger GC cycle

- When an allocation of an object, **x**, is failed
  - Allocate the object **x** in the reserved free area
  - Trigger the GC cycle



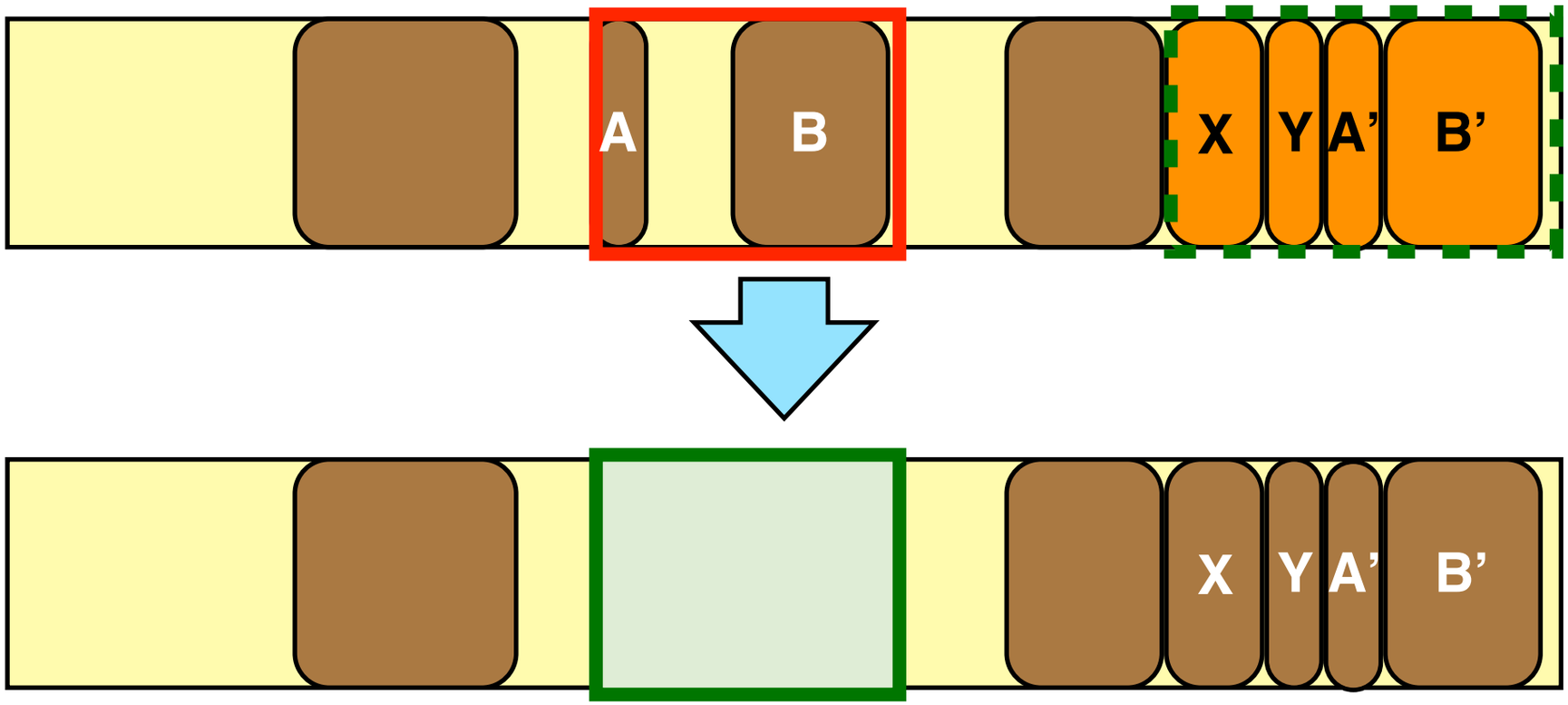
GC cycle

- The reserved free area behaves as the to-space
  - copying objects in the evacuation area
    - **A’**, **B’**
  - allocation by the application
    - **X**, **Y**



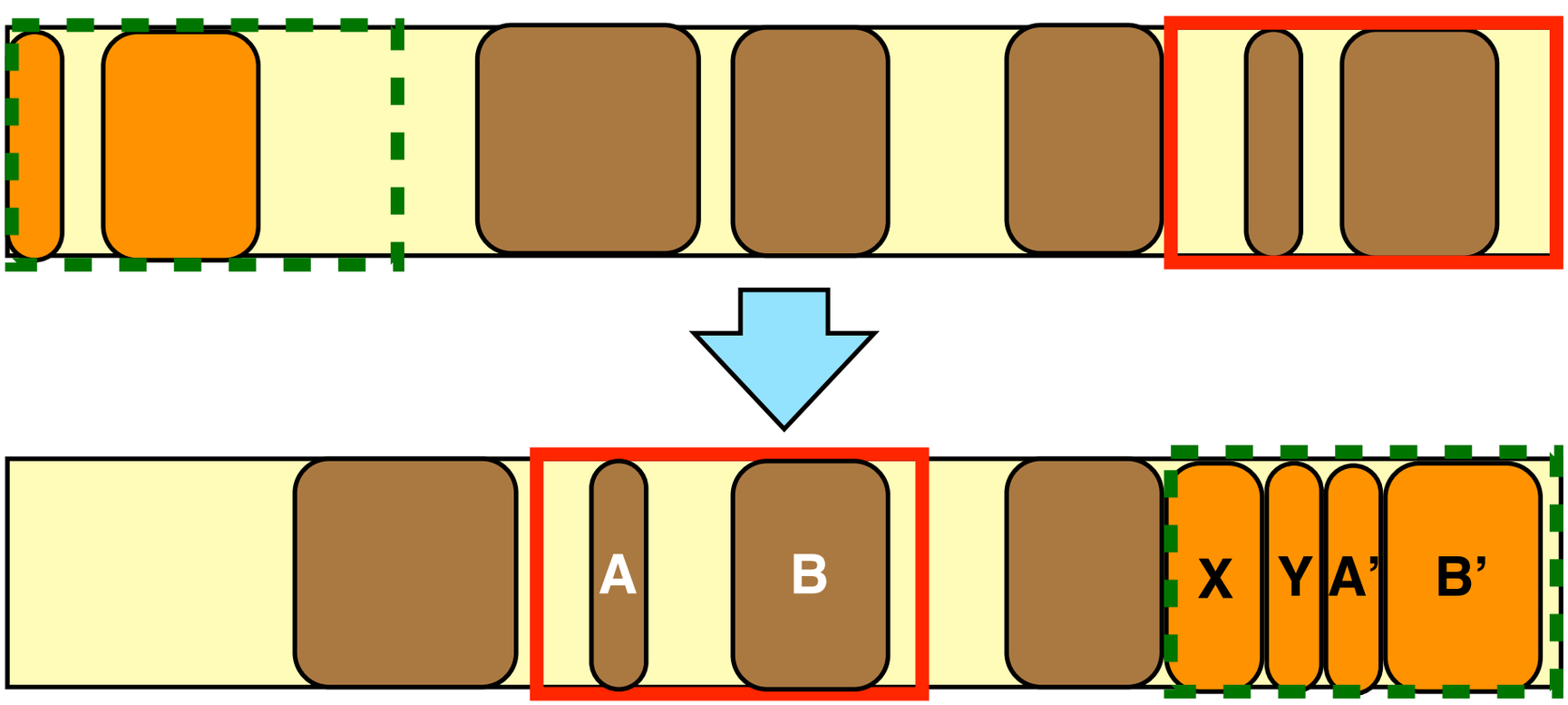
Next Reserved Area

- Use the EA as the next reserved free area
- Reserved free area has the same size as the previous EA



Size of Evacuation Area

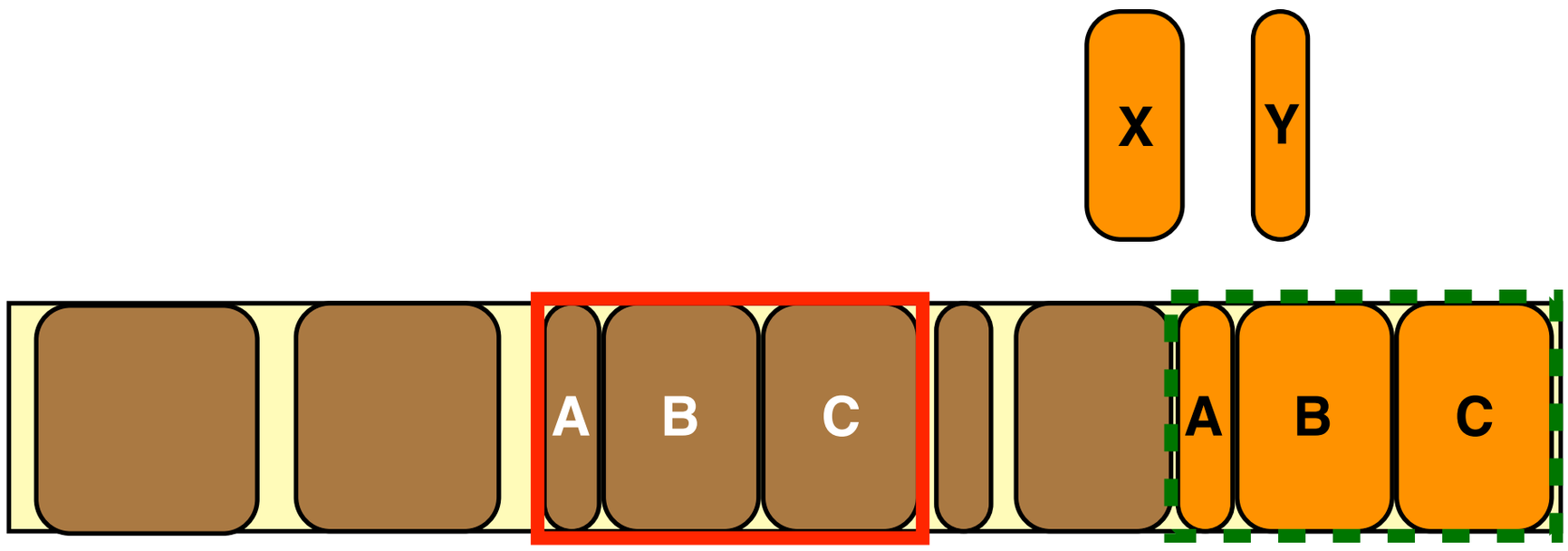
- Always choose the EA of a fixed size
- sizeof(from space) == sizeof(to space)



Overflow

If we chose the EA full of live objects,
the reserved free area would be exhausted

- Because application also allocates objects in the area

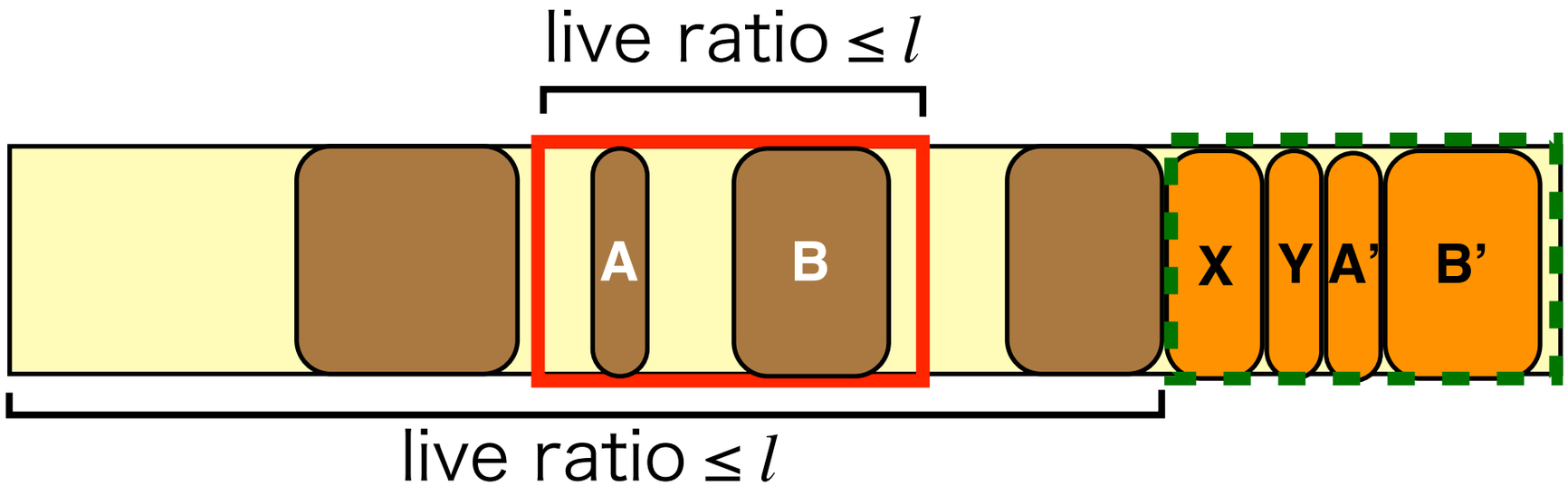


Live Ratio

Define *live ratio* :
fraction of the area occupied by live objects

Limit the live ratio of the EA by l

- We can guarantee that there is an area whose live ratio is less than l by adjusting the heap size (because we know the amount of live objects $\leq R$)



Estimation

We will estimate

- H — heap size
- E — size of the EA
- l — upper bound of the live ratio of the heap

We are given

- R — maximum amount of live objects
- a — upper bound of the amount of objects allocated during a single GC cycle

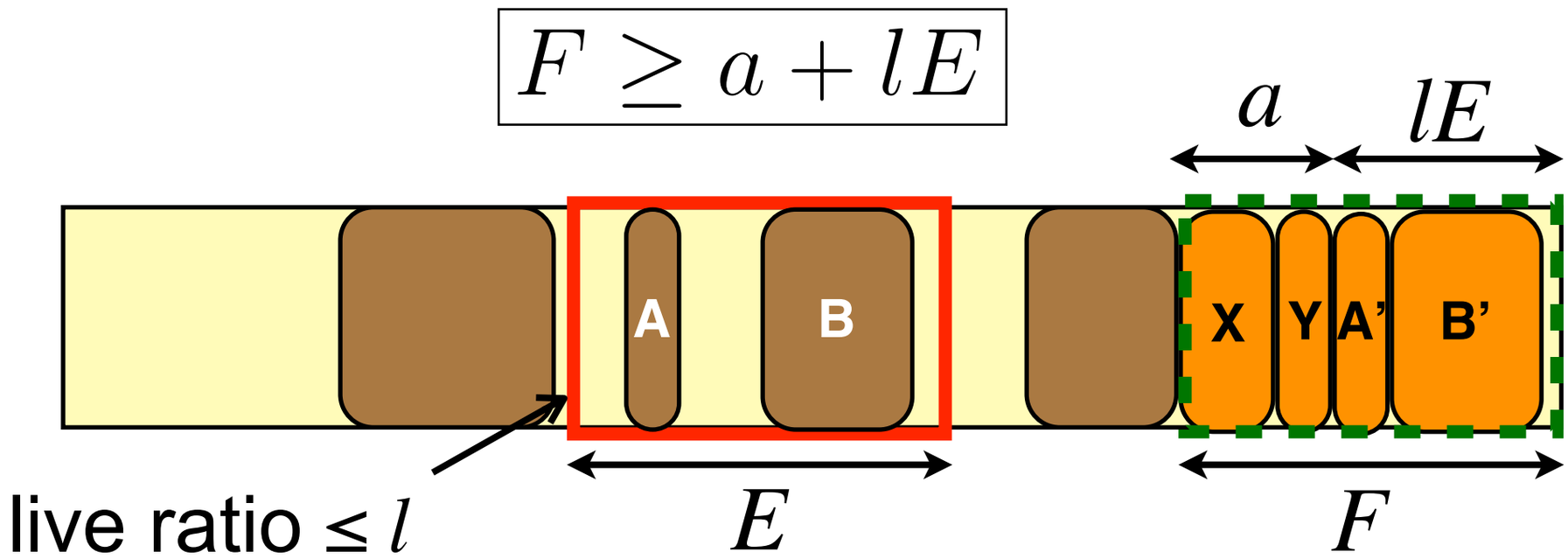
Size of Reserved Free Area

Memory consumption during GC ≤ $a + lE$

- a: Allocation by the application
- lE: Objects in the EA
  (to be copied to the reserved area)

Size of the reserved free area: F

$$F \geq a + lE$$



Size of EA

Size of the EA, E, should be as large as the size of the reserved free area, F

- Because the EA is used as the next reserved free area

$$\boxed{E = F}$$

By $\boxed{F \geq a + lE}$,

$$\boxed{E \geq \frac{a}{1-l}}$$

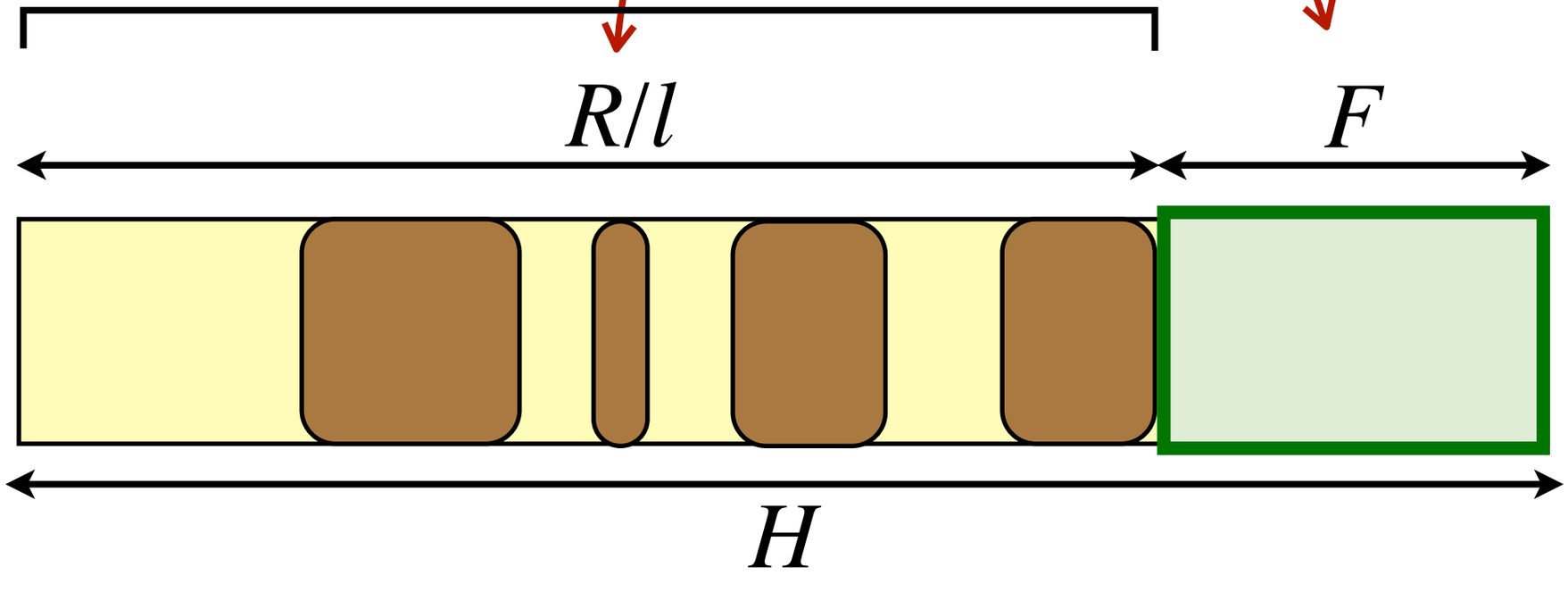
Heap Size

Required heap size H is

$$H = \frac{R}{l} + \frac{a}{1-l}$$

amount of live objects $\leq R$

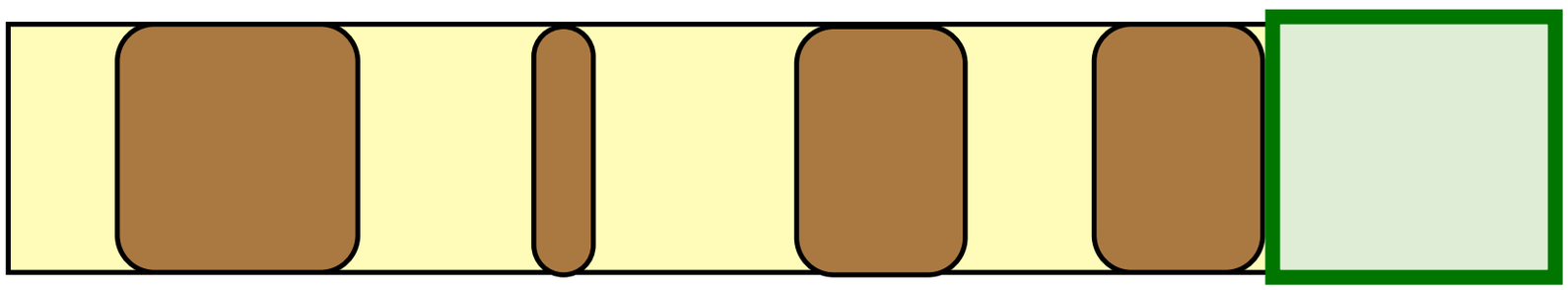
live ratio $\leq l$



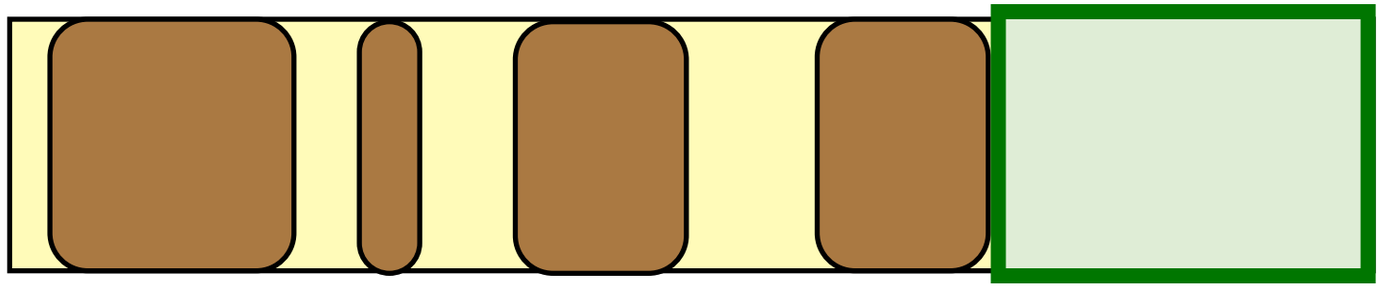
Adjusting Live Ratio

$$H = \frac{R}{l} + \frac{a}{1-l}$$

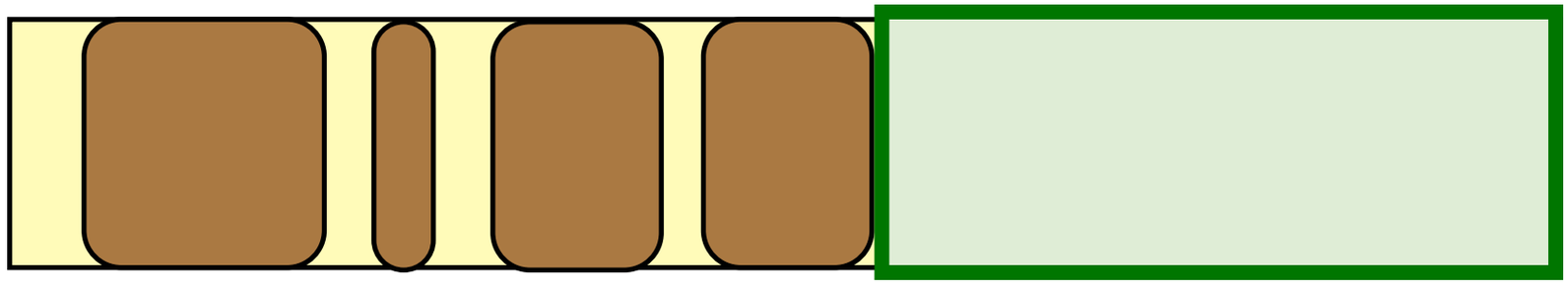
too low:



appropriate:



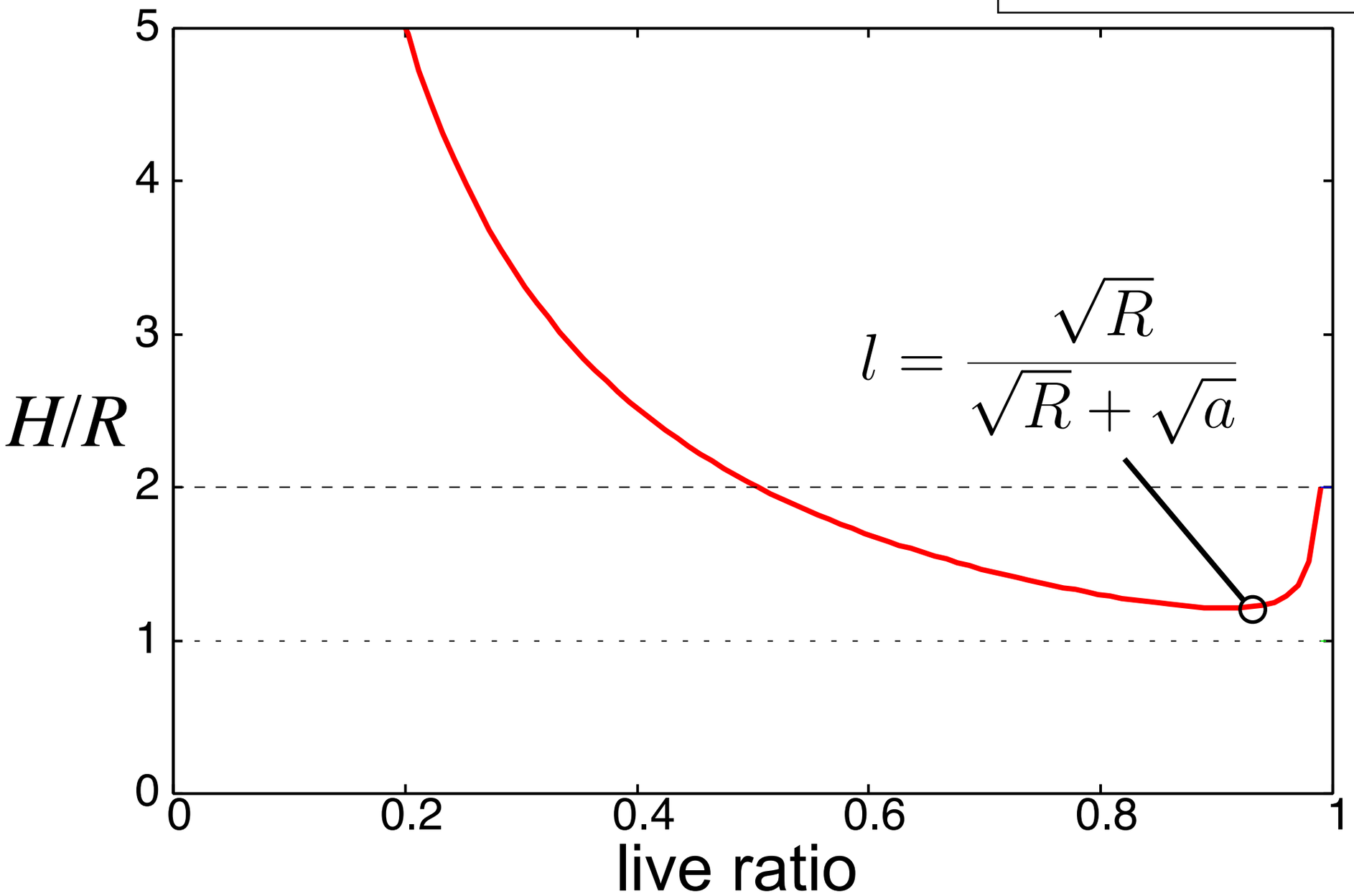
too high:



Appropriate Live Ratio

Normalized heap size
when $a/R = 1\%$

$$H = \frac{R}{l} + \frac{a}{1-l}$$



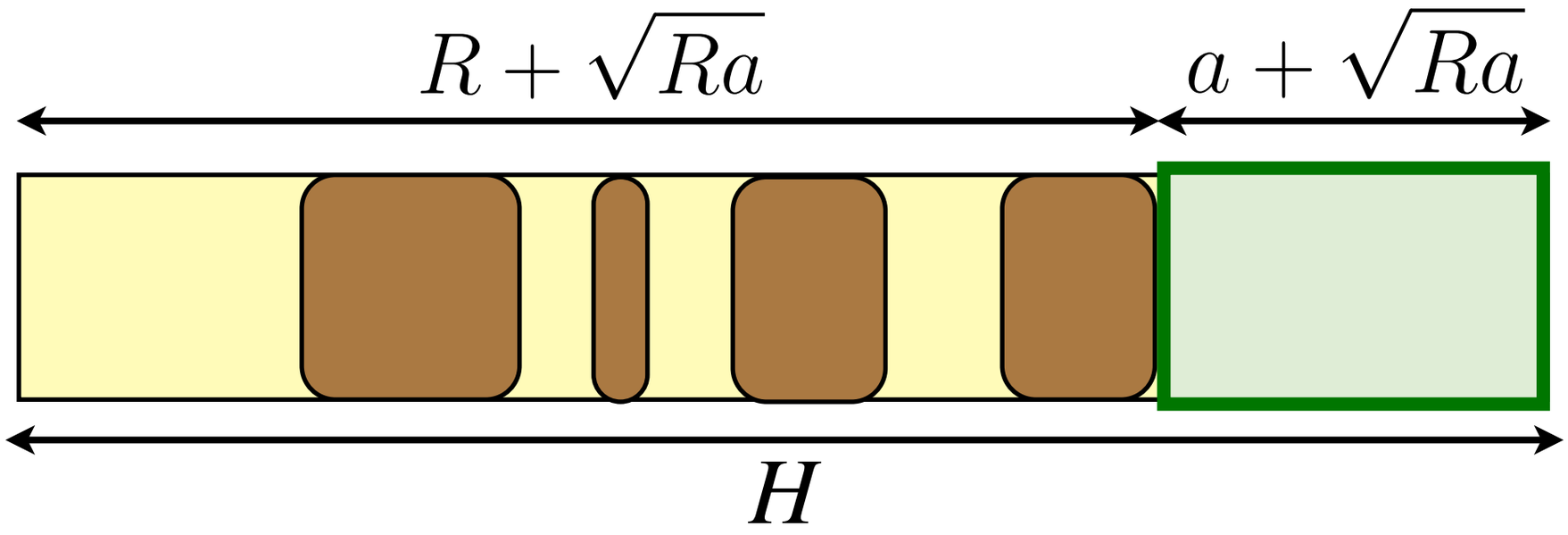
Result

When the live ratio is appropriate,

$$l = \frac{\sqrt{R}}{\sqrt{R} + \sqrt{a}}$$

$$H = R + 2\sqrt{Ra} + a$$

$$E = F = a + \sqrt{Ra}$$



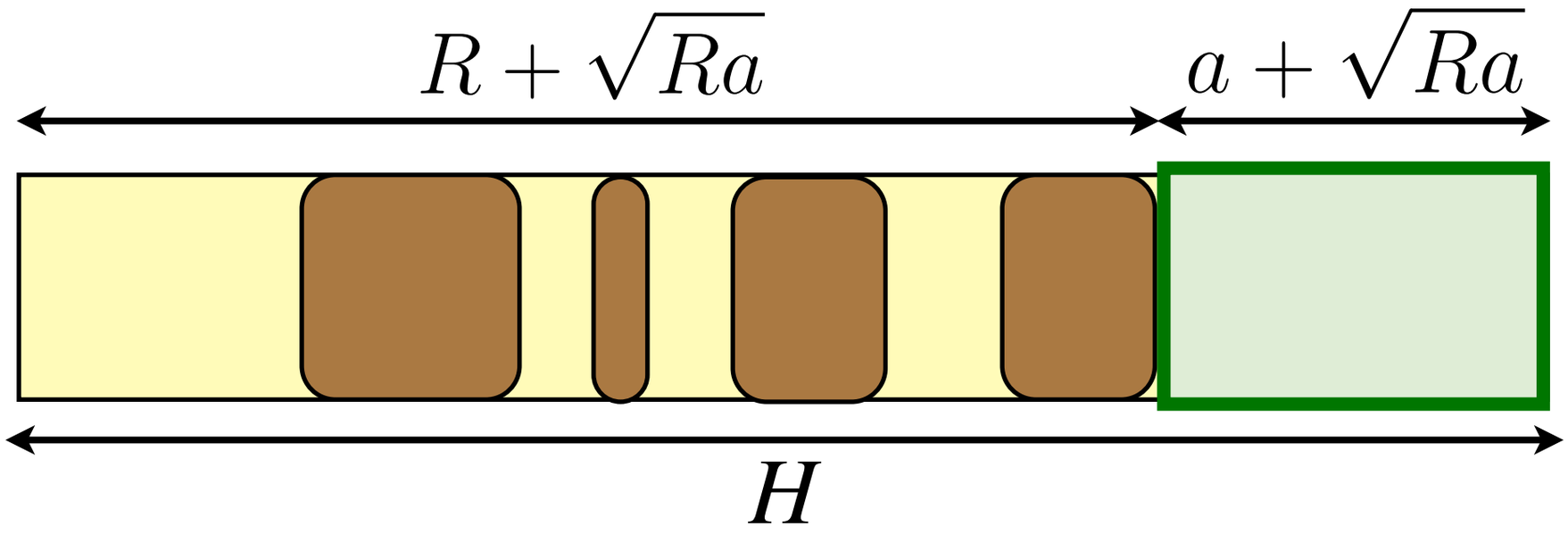
Result

When the live ratio is appropriate,

$$l = \frac{\sqrt{R}}{\sqrt{R}+\sqrt{a}} \qquad lE = \frac{\sqrt{R}(a+\sqrt{Ra})}{\sqrt{R}+\sqrt{a}}$$

$$H = R + 2\sqrt{Ra} + a \qquad = \sqrt{Ra}$$

$$E = F = a + \sqrt{Ra}$$



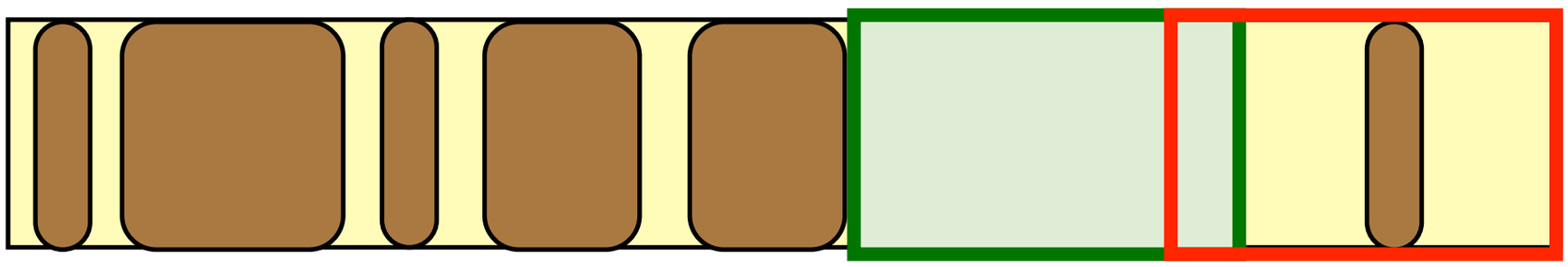
Outline

- Related work
- Model of our GC
- Basic Idea
  - Strategy
  - Analysis
- Practical setting
- Future work and summary

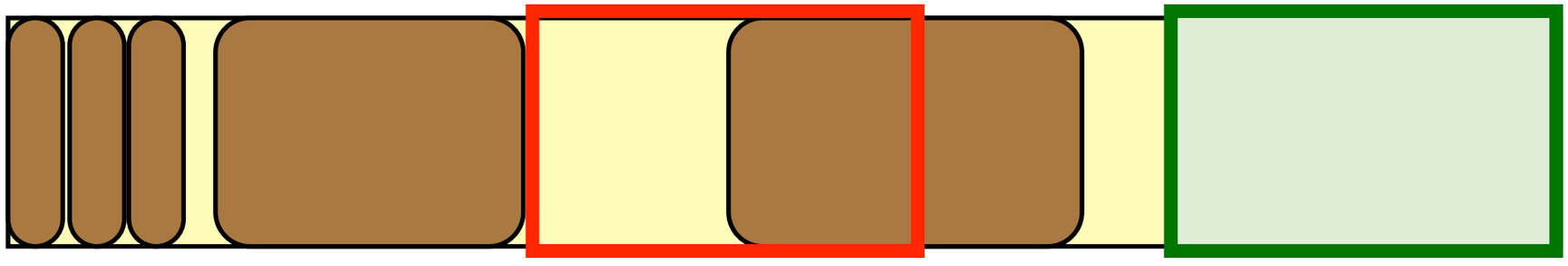
Practical Settings

We looked over two cases where we cannot choose the candidate of the EA

case 1: candidate overlaps with the reserved area



case 2: candidate contains a part of an object

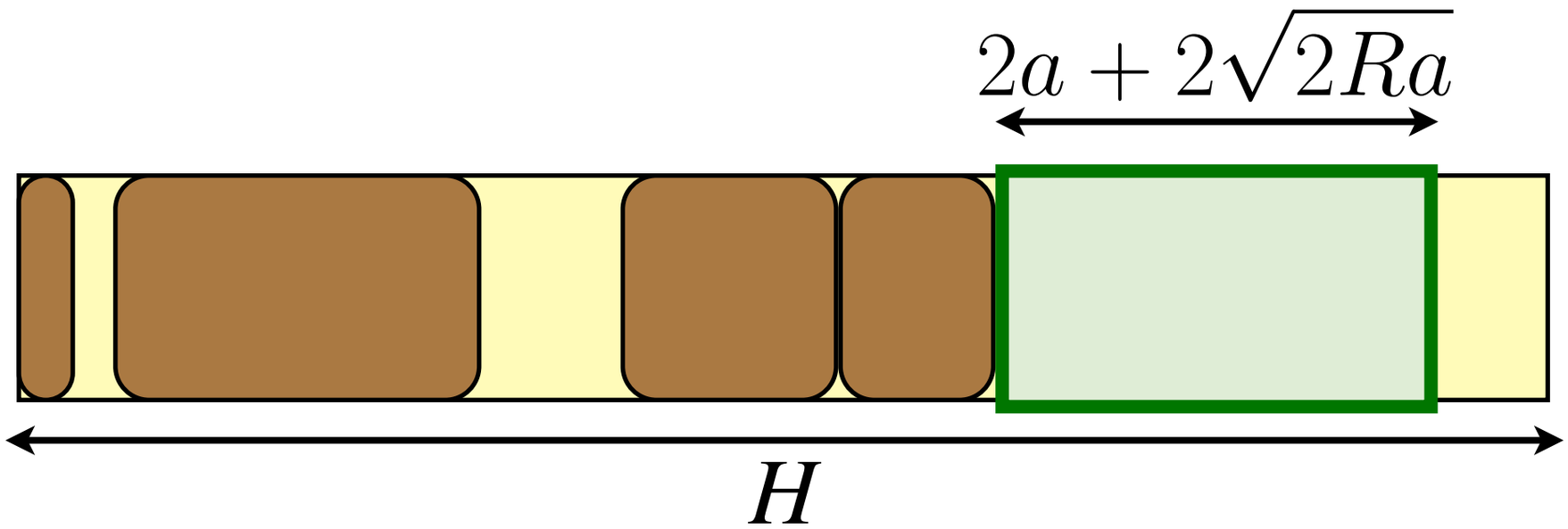


Revised Estimation

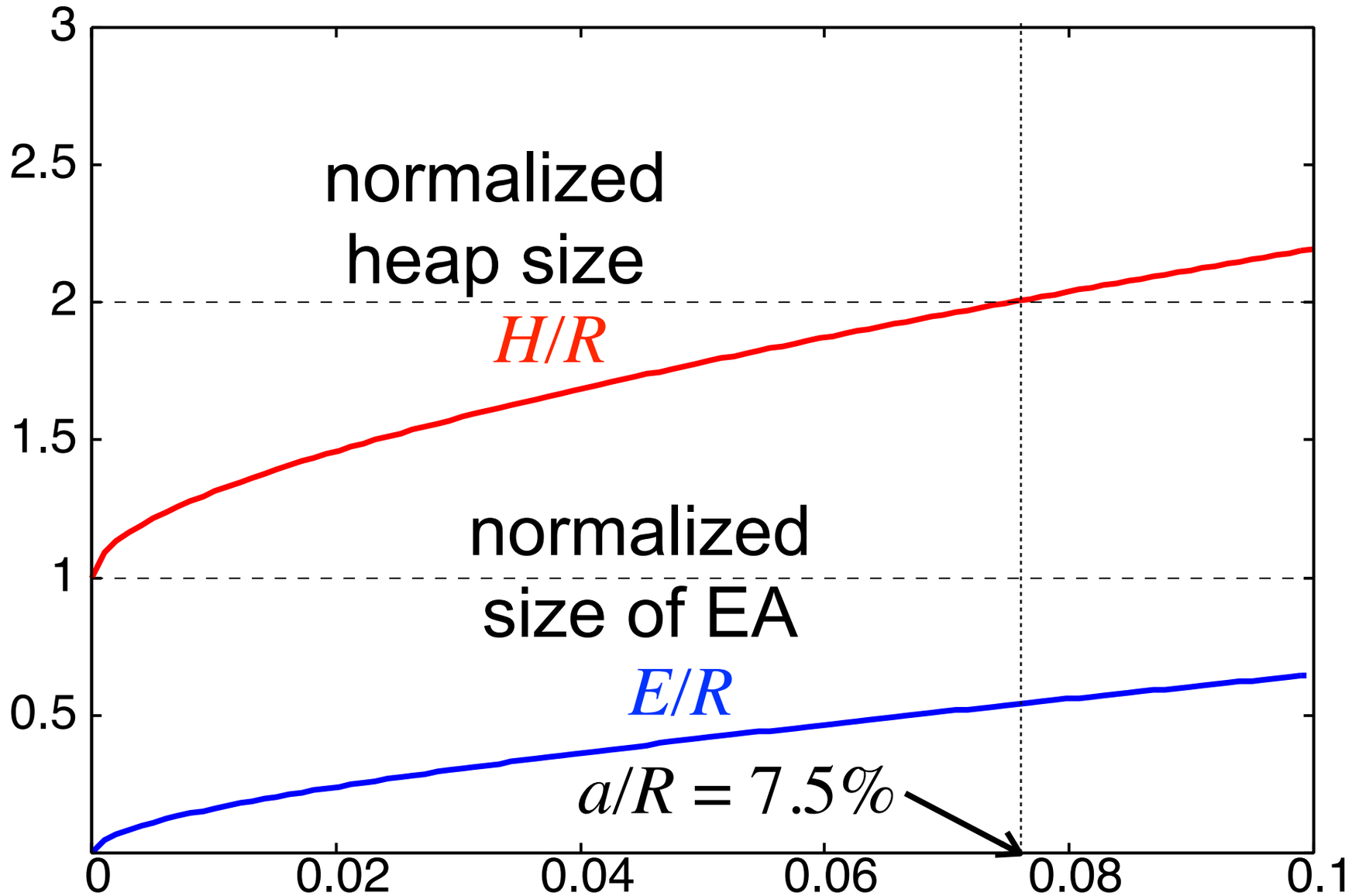
$$l = \frac{\sqrt{R}}{\sqrt{R} + \sqrt{2a}}$$

$$H = R + 2\sqrt{2Ra} + 3a$$

$$E = F = 2a + 2\sqrt{2Ra}$$



Required Heap Size



a/R; upper bound of allocation during a GC cycle

Future Work

- Implementation

Summary

- Gave a strategy to choose the EA
- Estimated the required heap size to avoid starvation
- Applicable to any partial compaction by evacuation
  - Independent of barrier for incremental marking
  - Independent of barrier for incremental relocation

Thank you

E-mail to: ugawa@cs.uec.ac.jp