Welcome to our Presentation

- Oliver Jessner, Andreas Scheicher
- University of Salzburg, department of Computer Sciences

- Garbage Collection in V8

What is V8 and what is a Garbage Collector?

A Garbage Collector is a program that produces dangling pointers

- V8 is the JavaScript Engine of Node.js, in all variants of Chromium and Opera.

What is V8 and what is a Garbage Collector?

A Garbage Collector is a program that produces dangling pointers

- V8 is the JavaScript Engine of Node.js, in all variants of Chromium and Opera.
- V8 is developed by Google and the Open Source Community

What is V8 and what is a Garbage Collector?

A Garbage Collector is a program that produces dangling pointers

- V8 is the JavaScript Engine of Node.js, in all variants of Chromium and Opera.
- V8 is developed by Google and the Open Source Community
- A Garbage Collector is a program that provides automatic memory management.

Modern Garbage Collectors

- Cost of execution time circa 5%

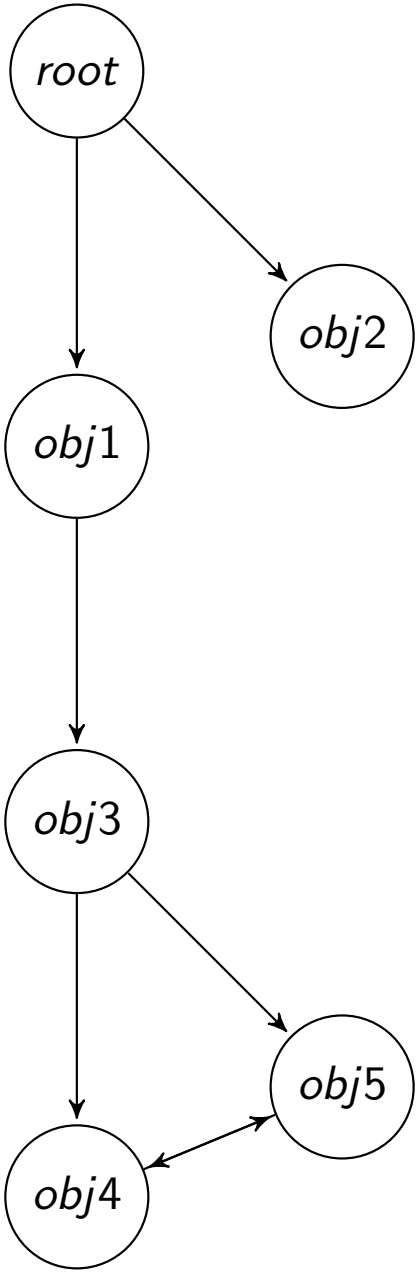
Modern Garbage Collectors

- Cost of execution time circa 5%
- Multithreading

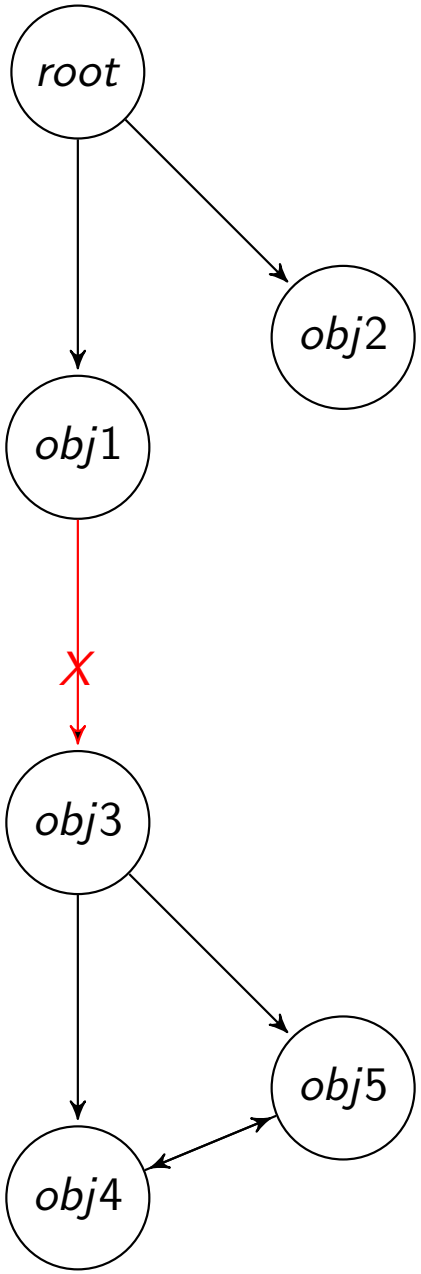
Modern Garbage Collectors

- Cost of execution time circa 5%
- Multithreading
- Mixed Strategies

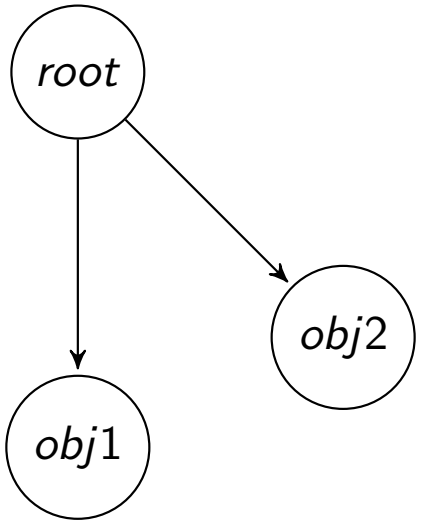
How does a Object Graph look like?

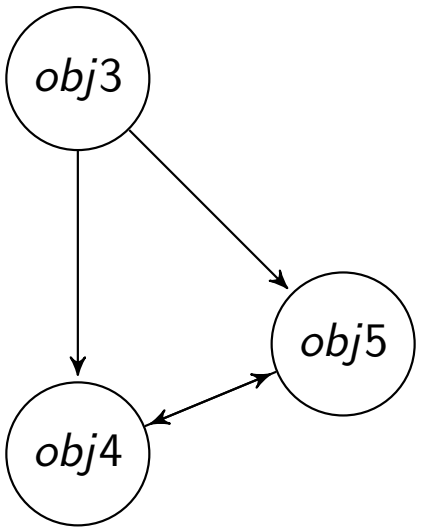


Now a dereference takes place



The Objects can not be reached from the root node





How does a Simple Garbage Collector work?

```
1     var new = function(ref){
2         ref = allocate();
3
4         if(ref === null){
5             mark();
6             sweep();
7             ref = allocate();
8         }
9
10        if(ref === null)
11            throw new outOfMemoryException();
12    }
```

Mark

```
1     var mark = function(){
2         var ref;
3         for(var obj : heap){
4             ref = obj.address;
5
6             if(ref && ref.Unmarked){
7                 ref.mark();
8                 recursiveMark(ref);
9             }
10        }
11    }
```

Sweep

```
1     var sweep = function(){
2         foreach(var obj : heap){
3             if(obj.isMarked)
4                 obj.UnMark();
5             else
6                 free(obj);
7         }
8     }
```

What kind of Garbage Collector are existing?

- Tracing Garbage Collection

What kind of Garbage Collector are existing?

- Tracing Garbage Collection
  1. Mark and Sweep
  2. Generational Garbage Collection

What kind of Garbage Collector are existing?

- Tracing Garbage Collection
  1. Mark and Sweep
  2. Generational Garbage Collection
- Reference counting

Which techniques are used in modern languages?

- Tracing Garbage Collection (JavaScript, Python, Ruby, Rust, C#)
  1. Mark and Sweep
  2. Generational Garbage Collection
- Reference counting (C++, PHP, Perl, Vala)

State of the art Generational Garbage Collection

Generational Garbage Collection

What is the difference?

- Divides the heap into more heaps (V8 uses two generations)

Generational Garbage Collection

What is the difference?

- Divides the heap into more heaps (V8 uses two generations)
- More generations lead to less interruption times

Generational Garbage Collection

What is the difference?

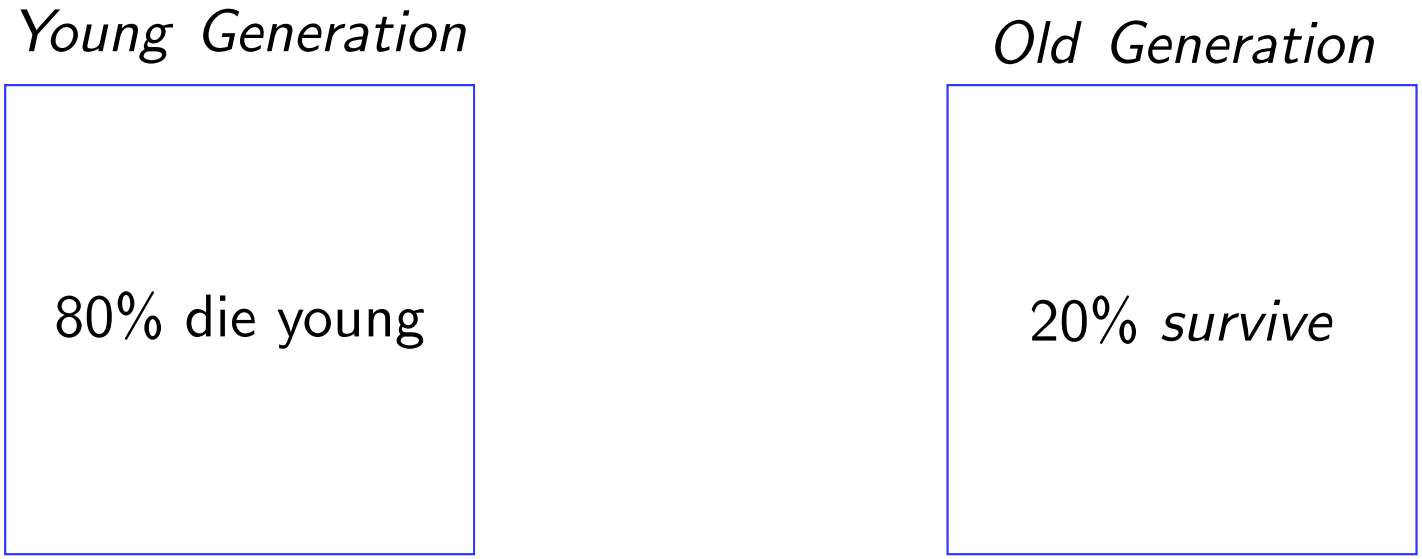
- Divides the heap into more heaps (V8 uses two generations)
- More generations lead to less interruption times
- Mark and Copy/Semi Space

Generational Garbage Collection

What is the difference?

- Divides the heap into more heaps (V8 uses two generations)
- More generations lead to less interruption times
- Mark and Copy/Semi Space
- Mark and Compact

More generations less Stop-the-World interrupts



What are the differences between the Old- and Young-generation?

Young Generation

- Fast collection

What are the differences between the Old- and Young-generation?

Young Generation

- Fast collection
- Frequent collection

What are the differences between the Old- and Young-generation?

Young Generation

- Fast collection
- Frequent collection
- Needs more space

Why is the collection of the Young Generation faster?

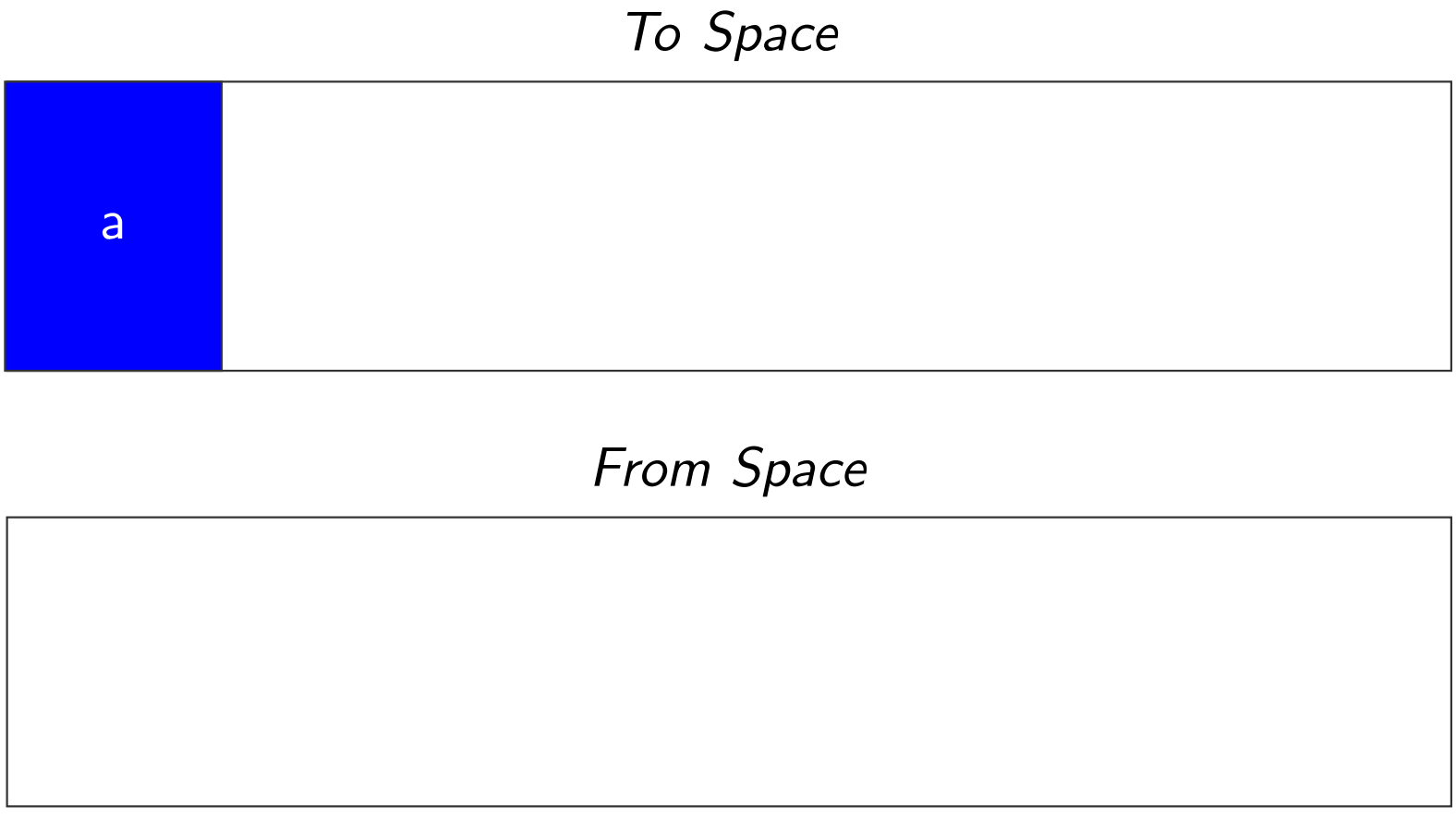
Keyword, Mark and Copy

To Space

From Space

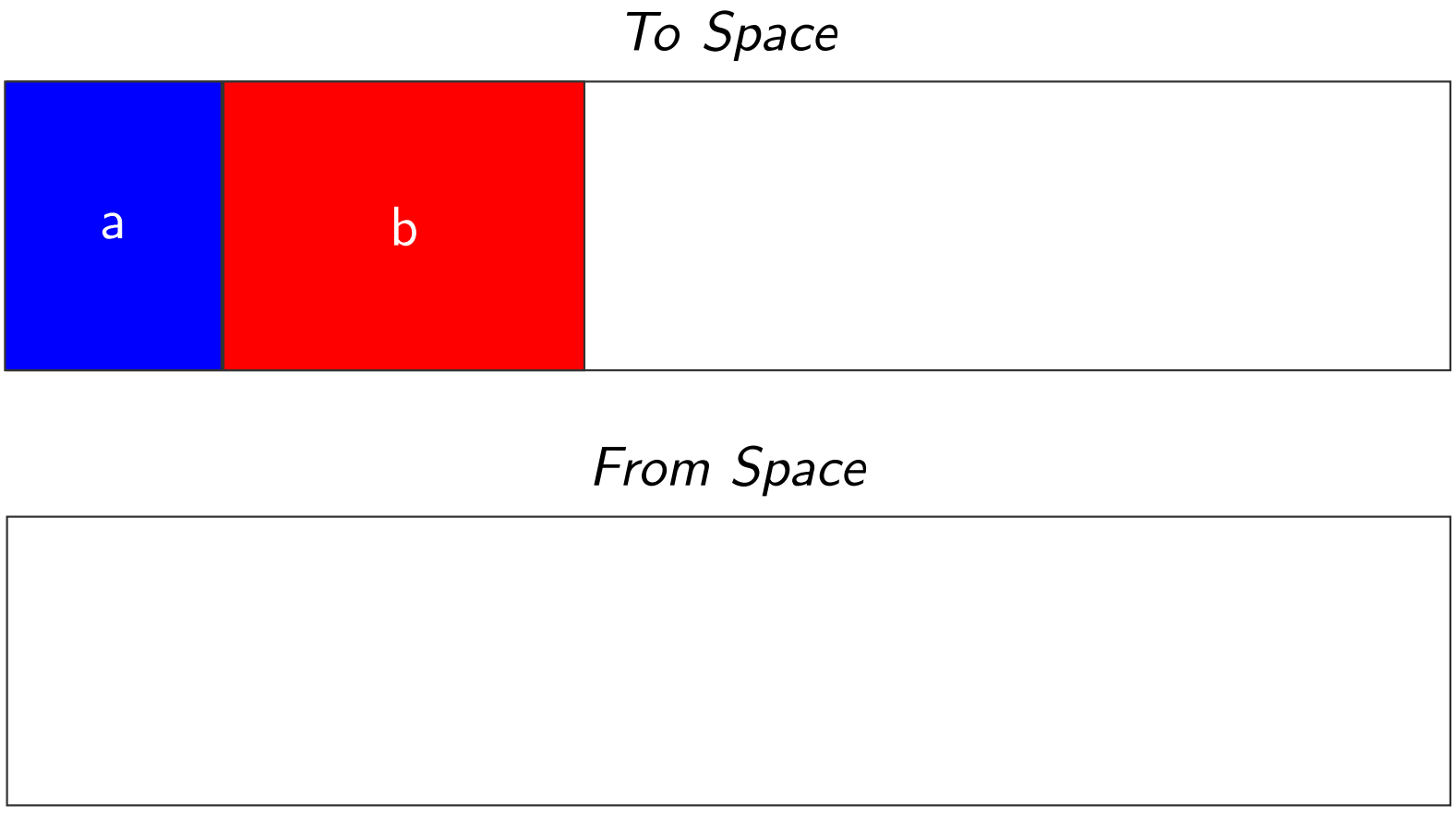
Why is the collection of the Young Generation faster?

Keyword, Mark and Copy



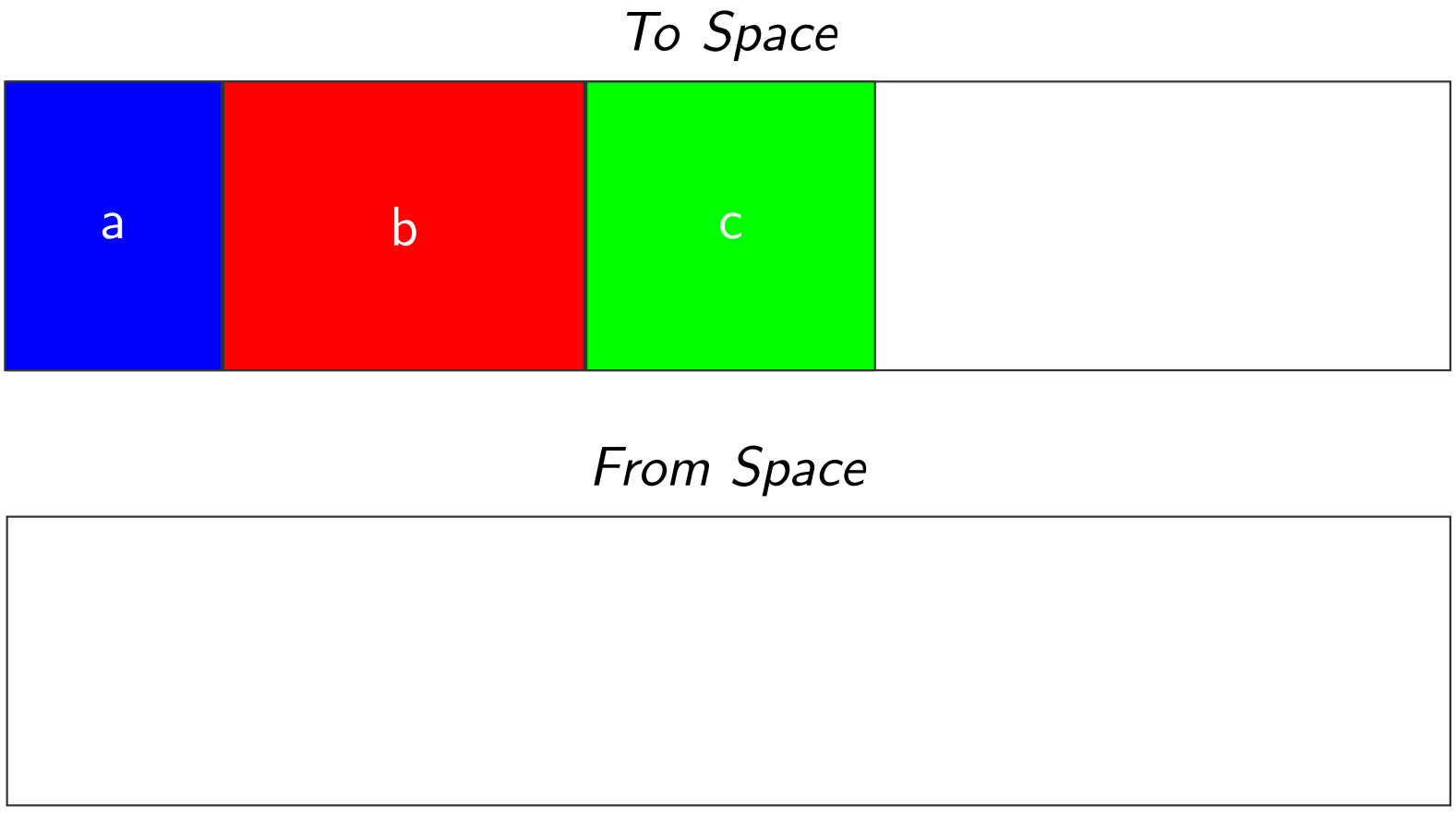
Why is the collection of the Young Generation faster?

Keyword, Mark and Copy



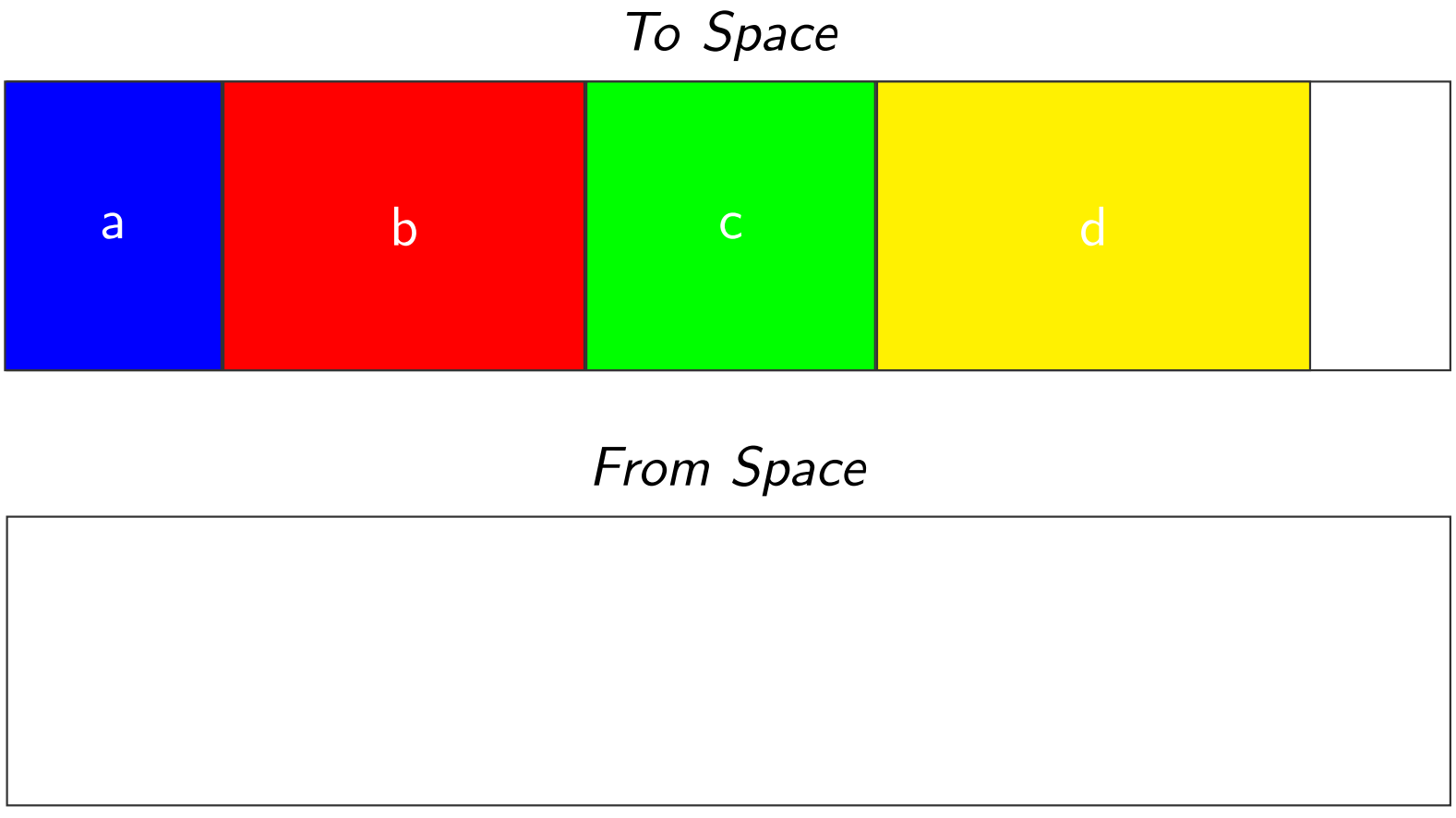
Why is the collection of the Young Generation faster?

Keyword, Mark and Copy



Why is the collection of the Young Generation faster?

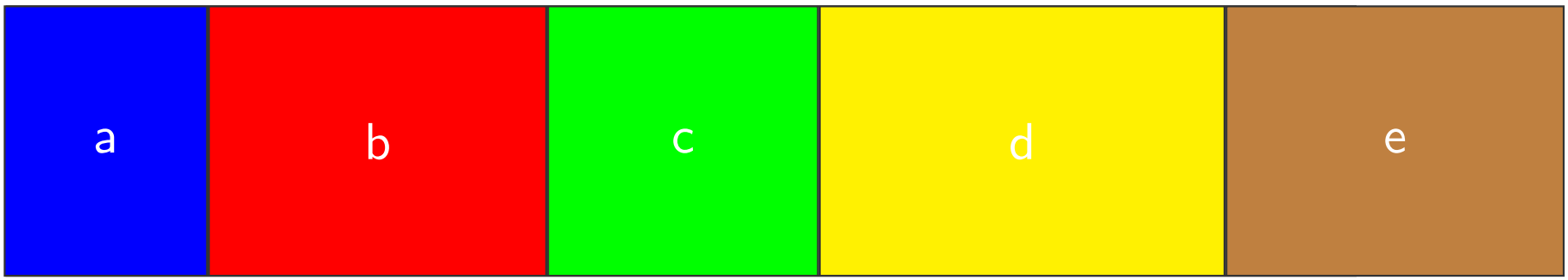
Keyword, Mark and Copy



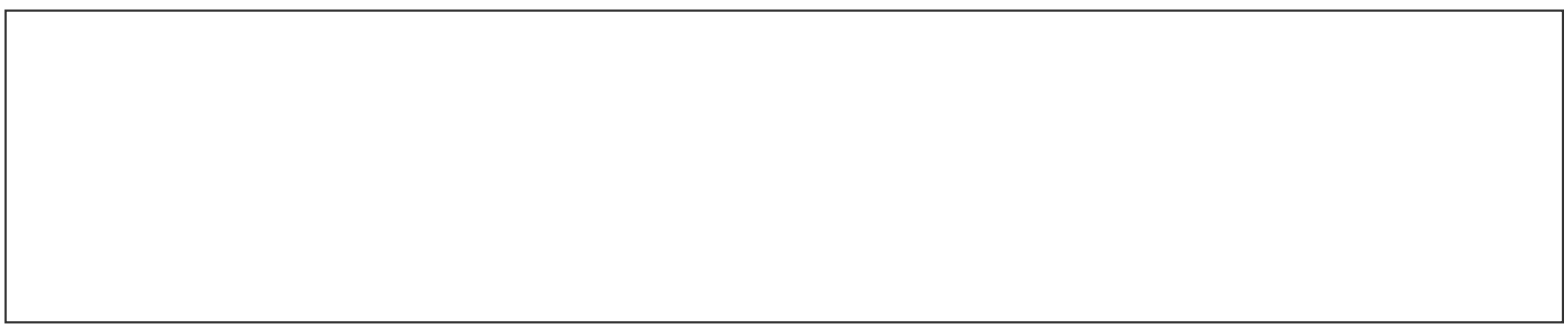
Why is the collection of the Young Generation faster?

Keyword, Mark and Copy

To Space



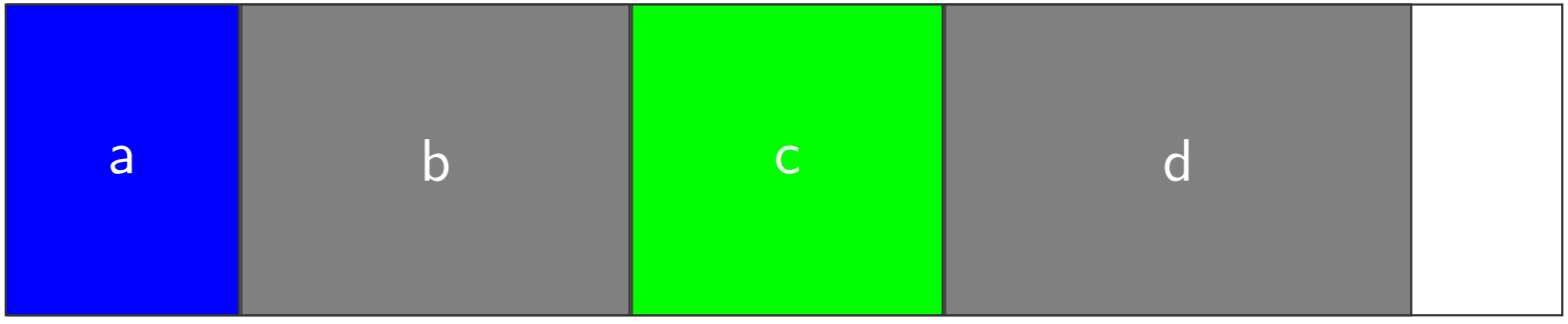
From Space



Now the mark phase starts

Keyword, Mark and Copy

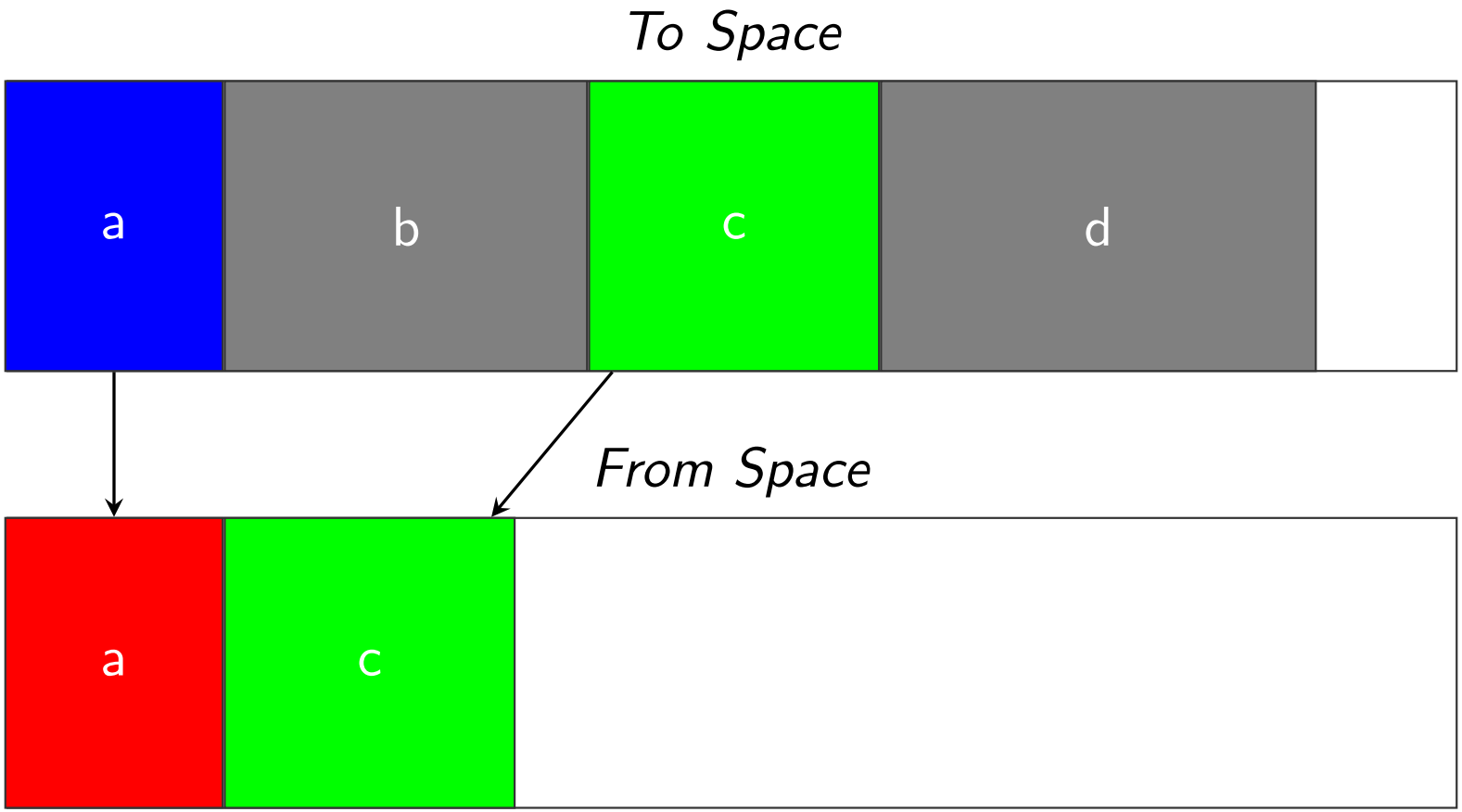
To Space



From Space

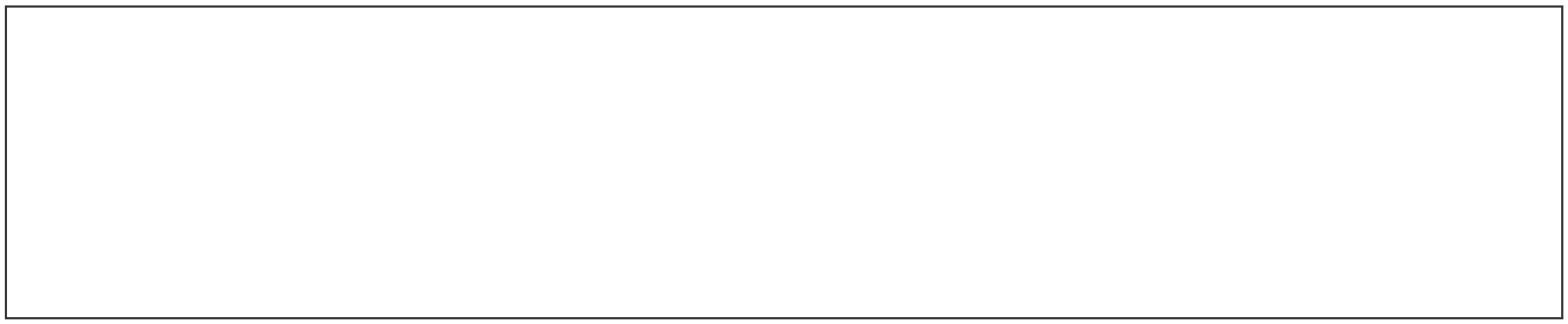
Now the mark phase starts

Keyword, Mark and Copy

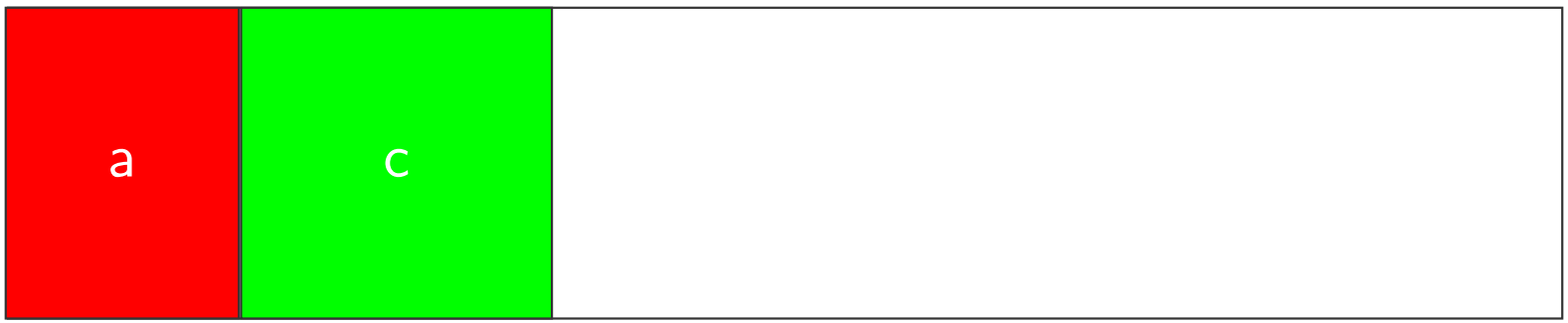


Delete toSpace
Keyword, Mark and Copy

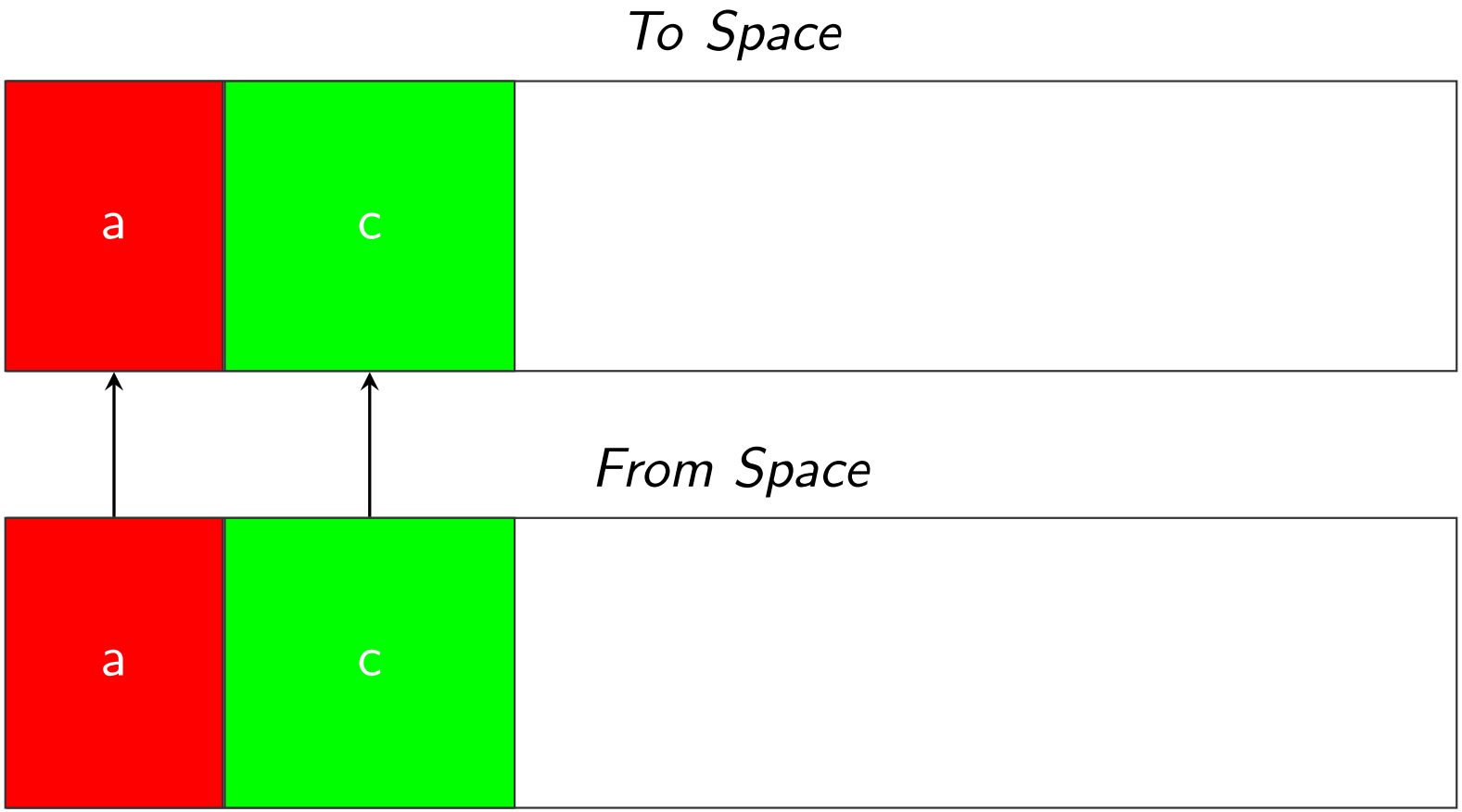
To Space



From Space



Delete toSpace
Keyword, Mark and Copy



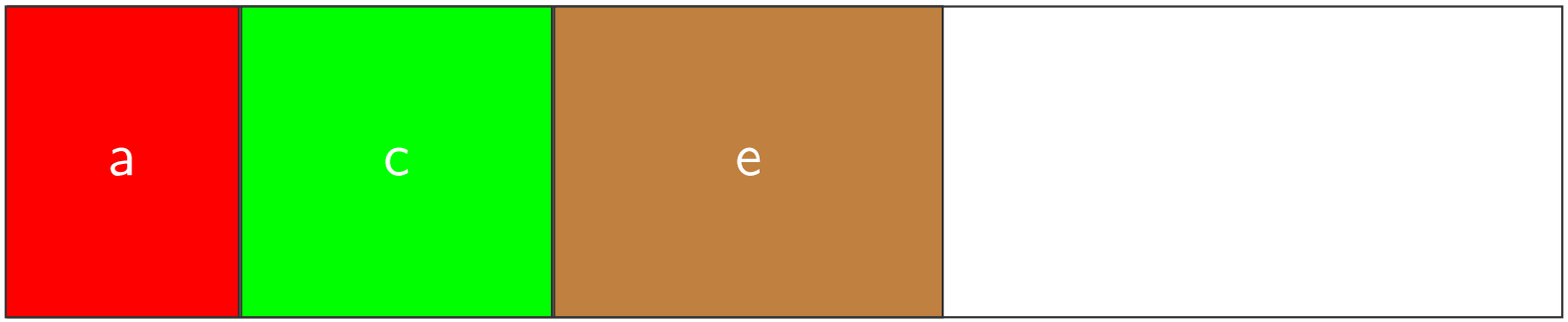
Delete fromSpace
Keyword, Mark and Copy

To Space

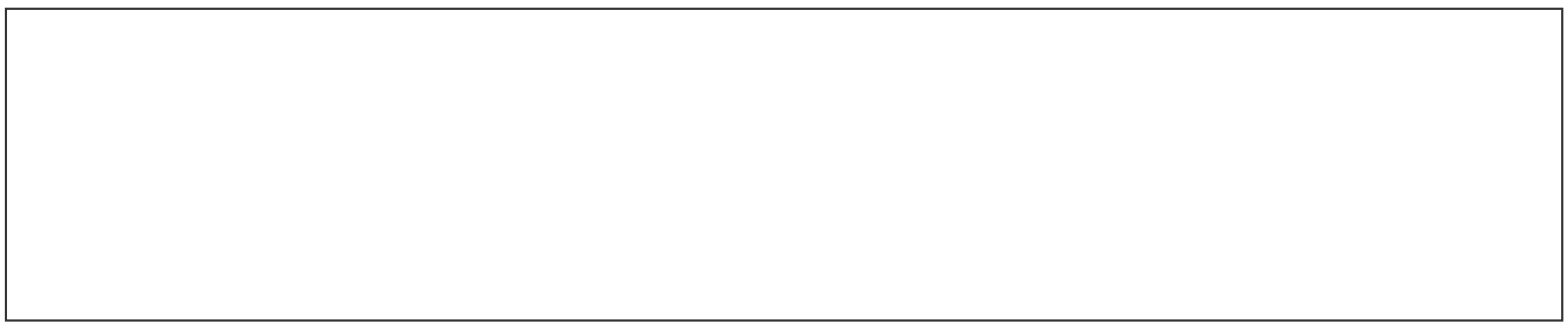
From Space

Delete fromSpace
Keyword, Mark and Copy

To Space



From Space



What are the differences between the Old- and Young-generation?

Old Generation

- Slower collection

What are the differences between the Old- and Young-generation?

Old Generation

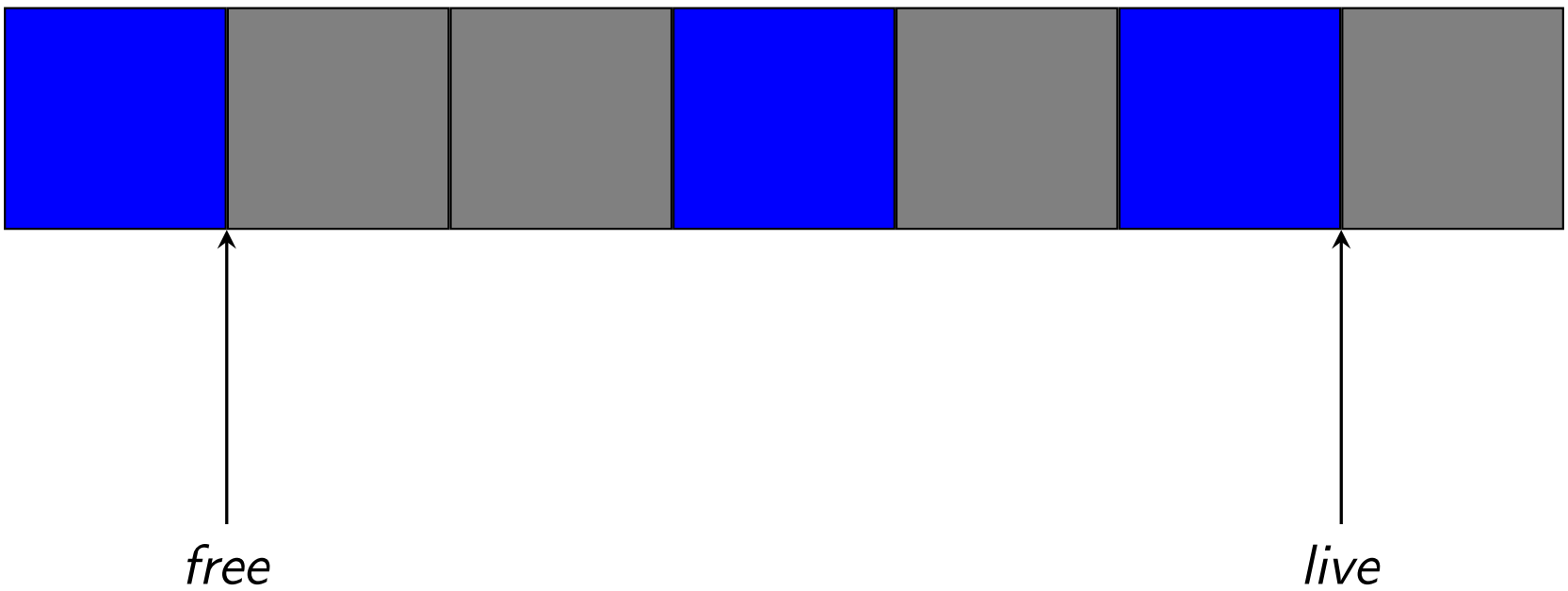
- Slower collection
- Infrequent collection

What are the differences between the Old- and Young-generation?

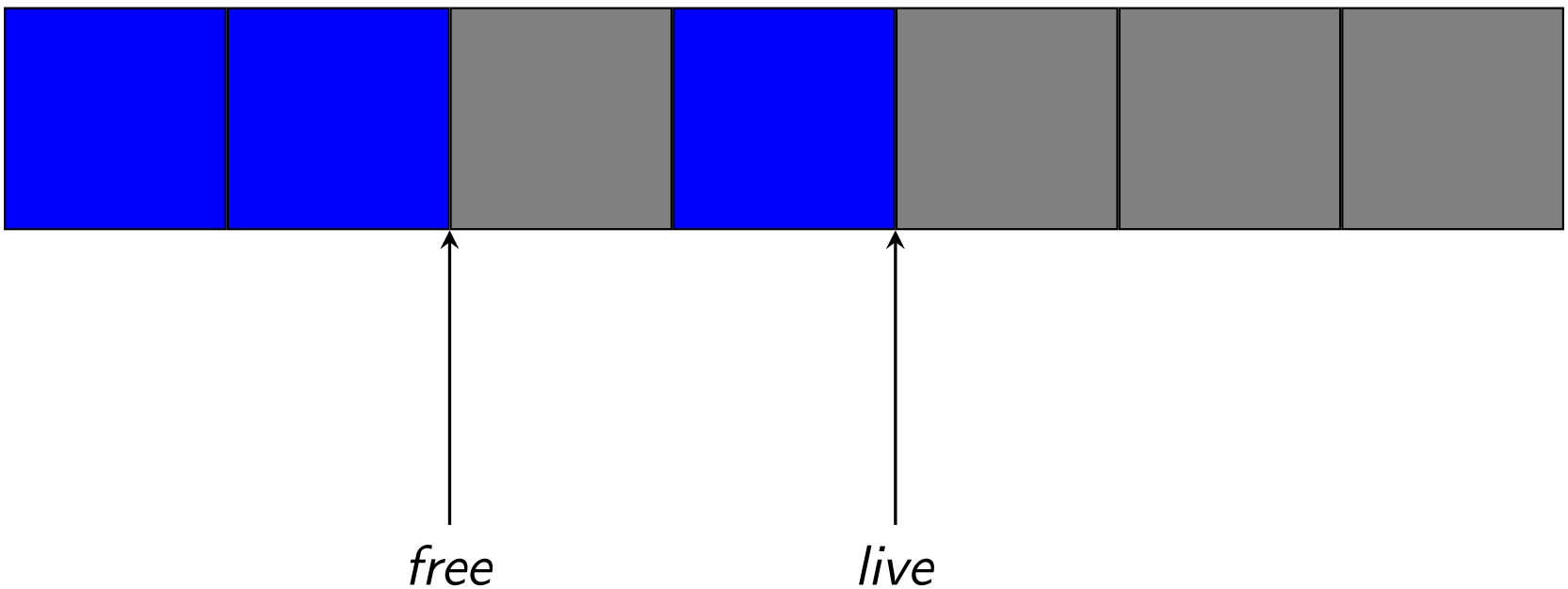
Old Generation

- Slower collection
- Infrequent collection
- Needs less space

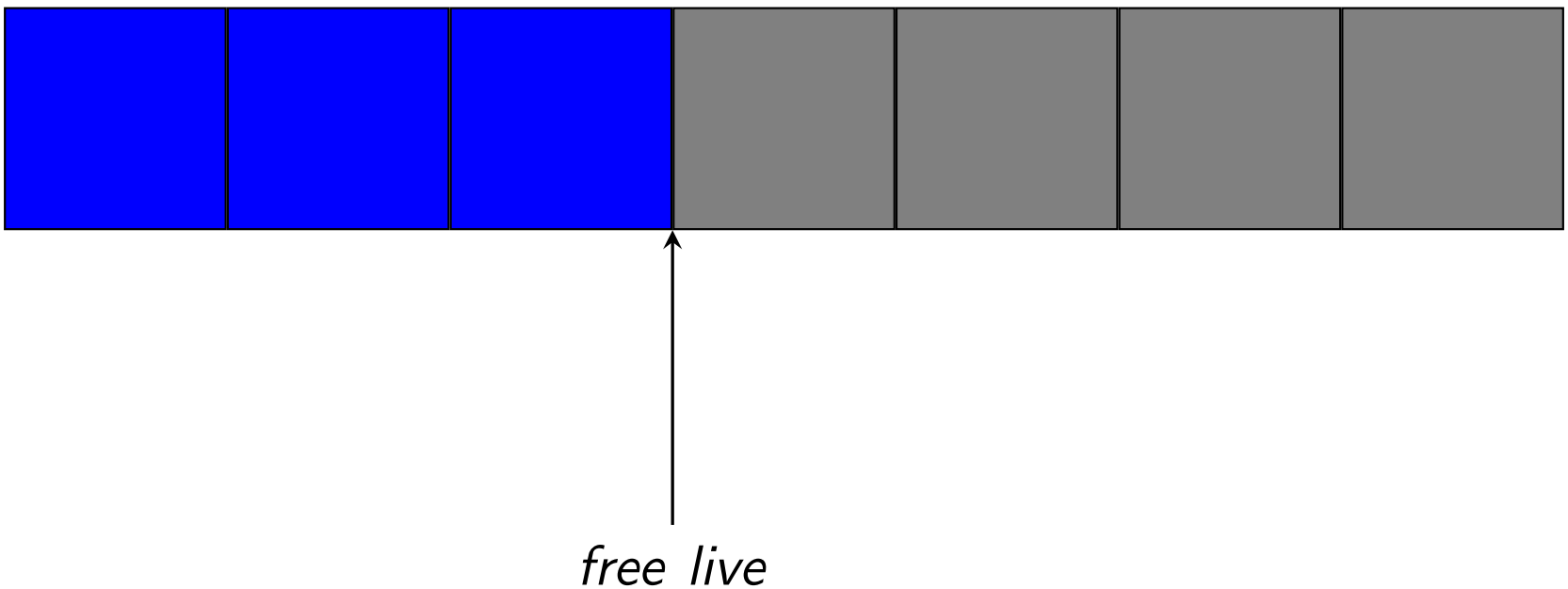
Keyword Mark and Compact



Keyword Mark and Compact

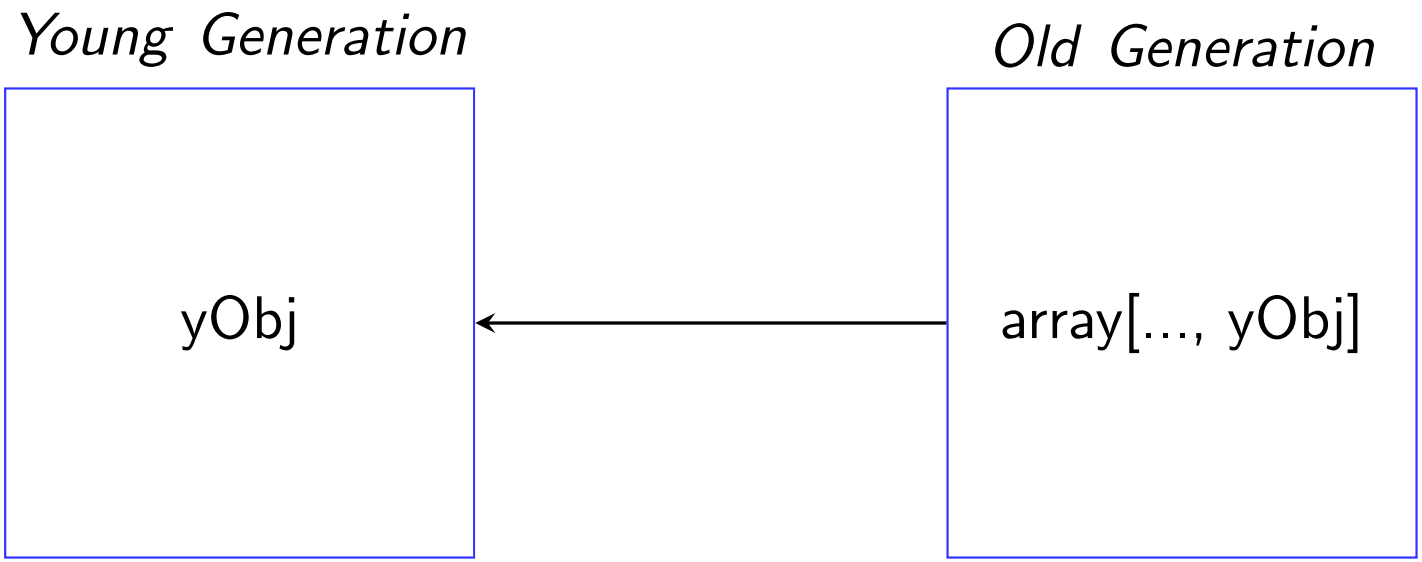


Keyword Mark and Compact



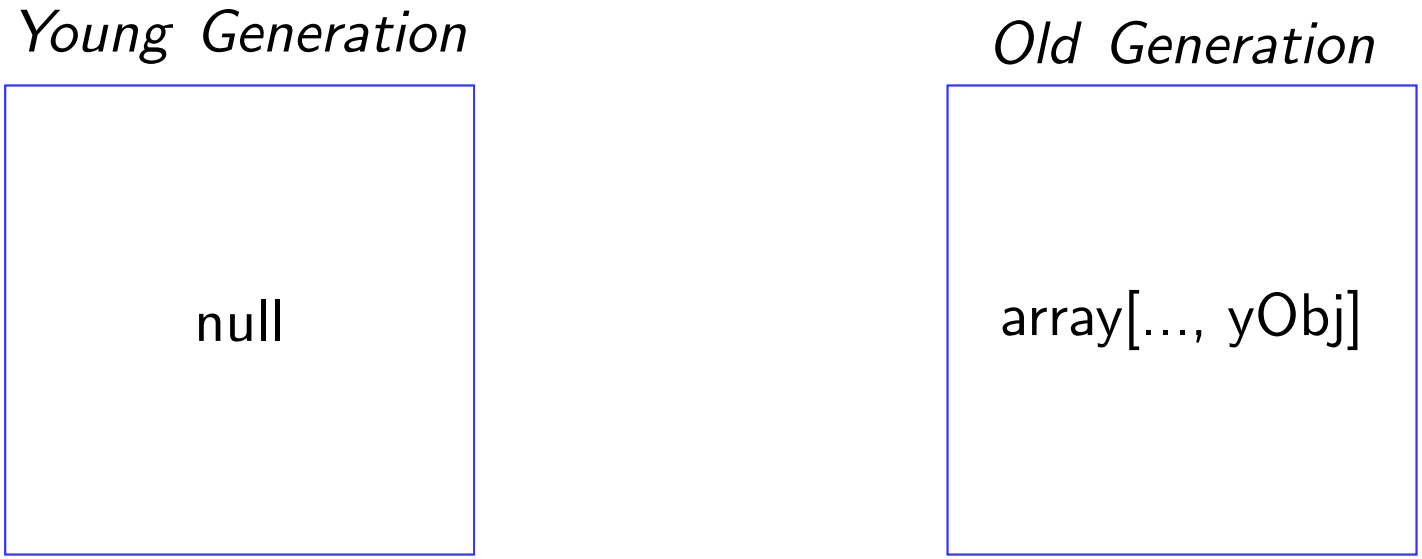
There is one big flaw with Generations

Intergenerational References



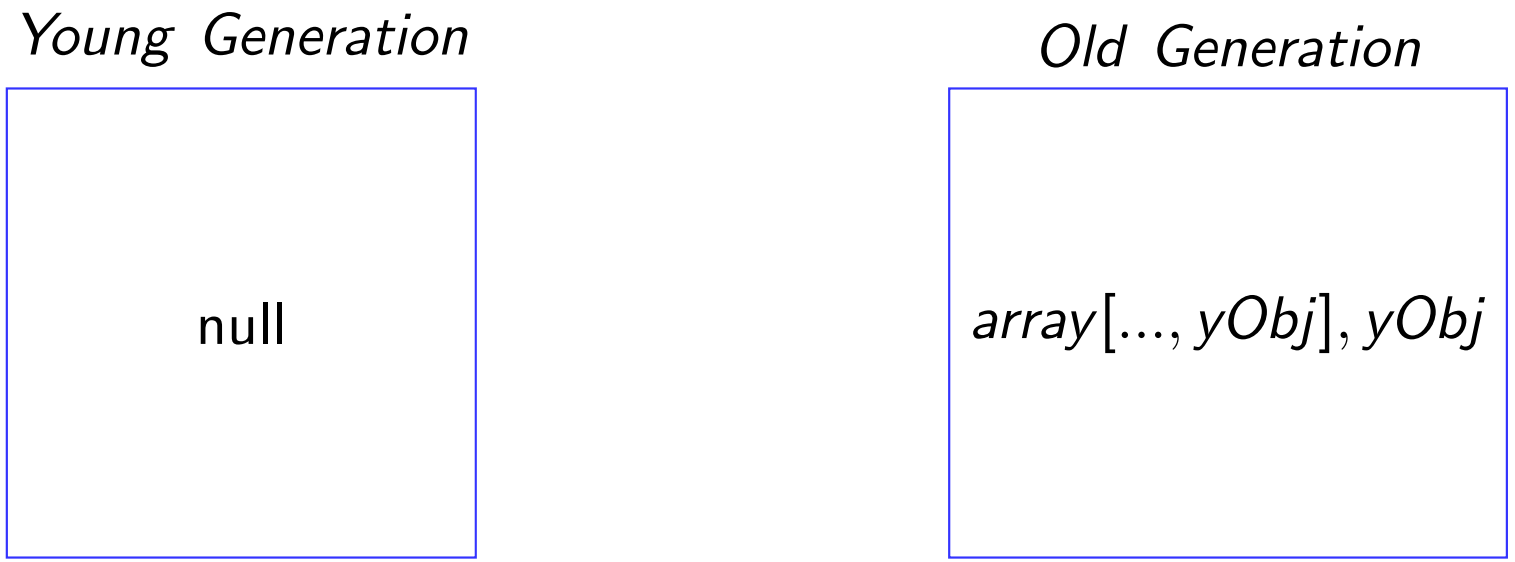
Now the Young Generation gets collected

We have a null reference in the Old Generation



There is one big flaw with Generations

Intergenerational References



Thank you for your attention

Any further questions?