TECHNISCHE UNIVERSITÄT MÜNCHEN

FAKULTÄT FÜR INFORMATIK

Garbage Collection & Reference Counting

Clemens Ruck & Alex Egger

Summer Term 2017

Overview

Methods of Memory Management

1. Shortcomings of Manual Memory Management
2. Garbage Collection
3. Rust’s approach

Memory Management in Rust

1. Stack Allocation
2. Heap Allocation with Box<T>
3. Reference counting in Rust
4. The ’unsafe’ keyword
5. The ’unsafe’ keyword

Common shortcomings of manual memory management

Garbage Collection

Example - Mark & Sweep

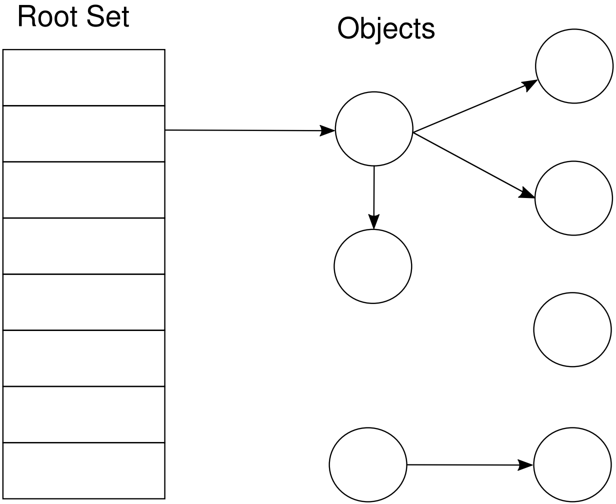


Figure: A graph-represenation of alive objects.

Example - Mark & Sweep

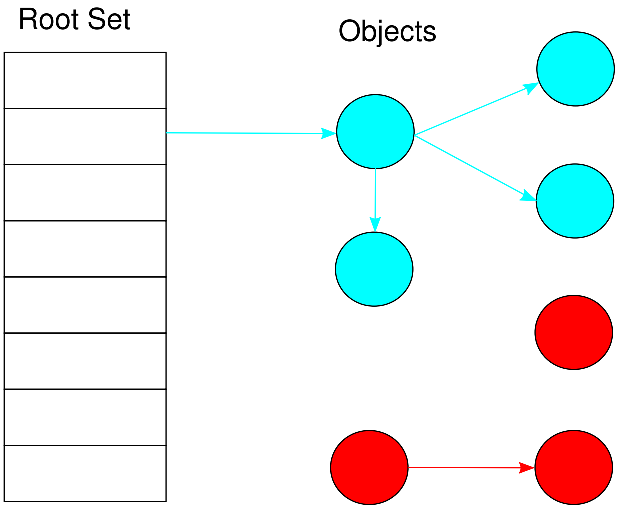


Figure: The 'Mark' stage of the algorithm.

Example - Mark & Sweep

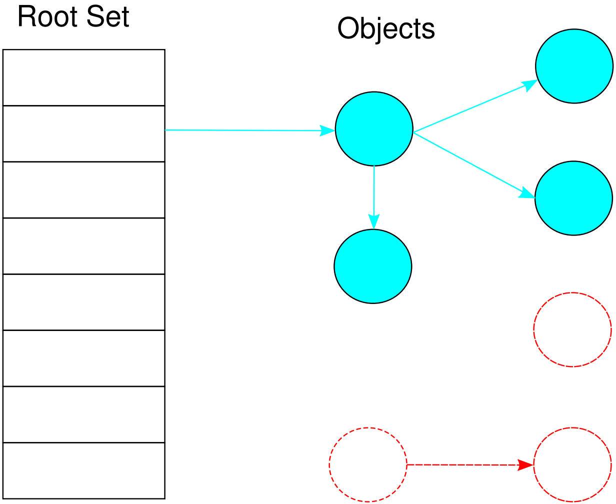


Figure: The 'Sweep' stage of the algorithm.

Memory management in Rust

Stack Allocation

Address	Name	Value
0	x	42

```rust
fn main() {
    let x = 42;
    other();
}

fn other() {
        let y = 27;
        let z = 99;
}
```

Stack Allocation

After line 8:

Address	Name	Value
2	z	99
1	y	27
0	x	42

```
1 fn main() {
2         let x = 42;
3         other();
4 }
5
6 fn other() {
7         let y = 27;
8         let z = 99;
9 }
```

Stack Allocation

After line 3:

Address	Name	Value
0	x	42

```
1  fn main() {
2          let x = 42;
3          other();
4  }
5
6  fn other() {
7          let y = 27;
8          let z = 99;
9  }
```

Advantages & Limitations

Heap Allocation

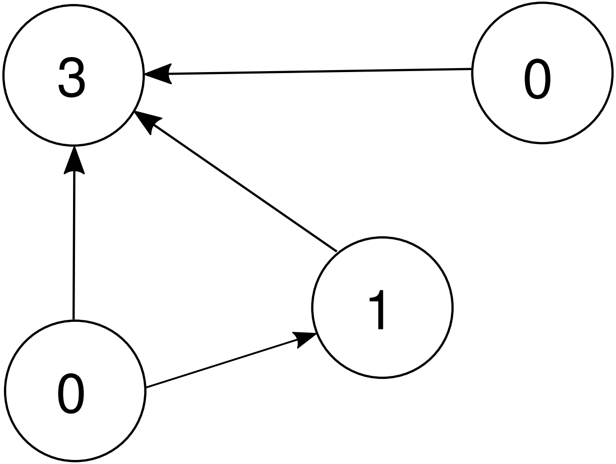
Workings of the Heap

Box<T> in Rust

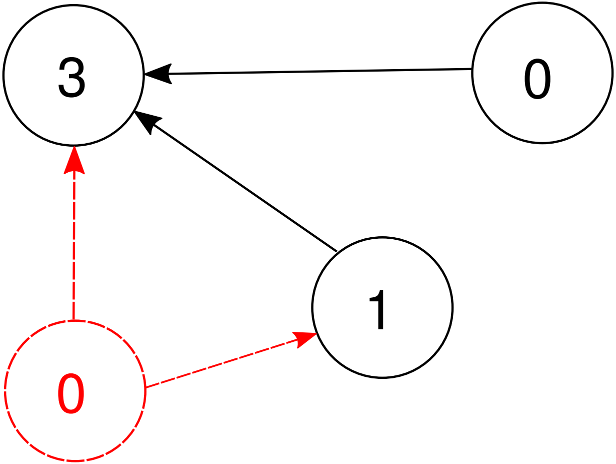
Use cases

Comparison: Heap vs. Stack

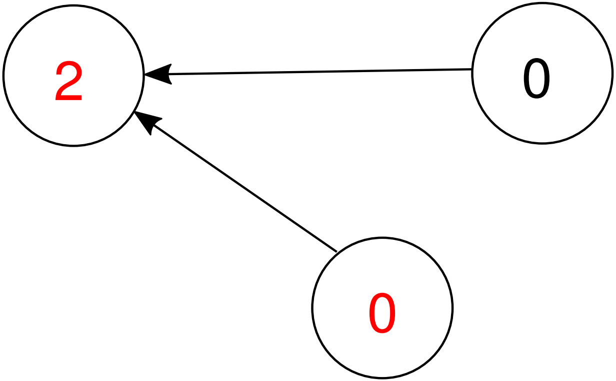
Reference Counting - Example



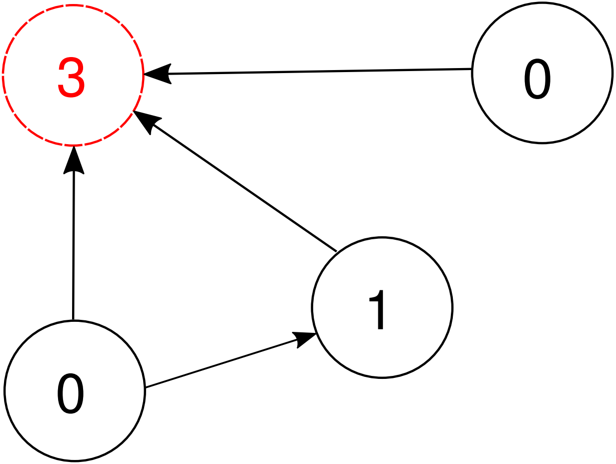
Reference Counting - Example



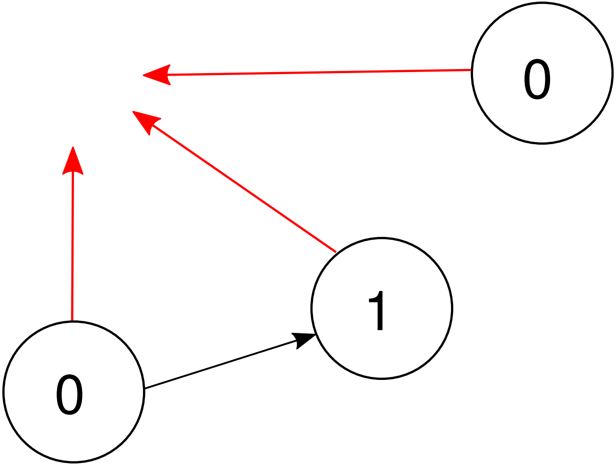
Reference Counting - Example



Reference Counting - Bad example



Reference Counting - Bad example



Limitations

⚠ Reference cycles can never be reclaimed!

Rc<T> (and Weak<T>) in Rust

Example

Limitations of Rc<T>

Multi-threaded Reference Counting with Arc<T>

Example

The ’unsafe’ keyword

Use cases