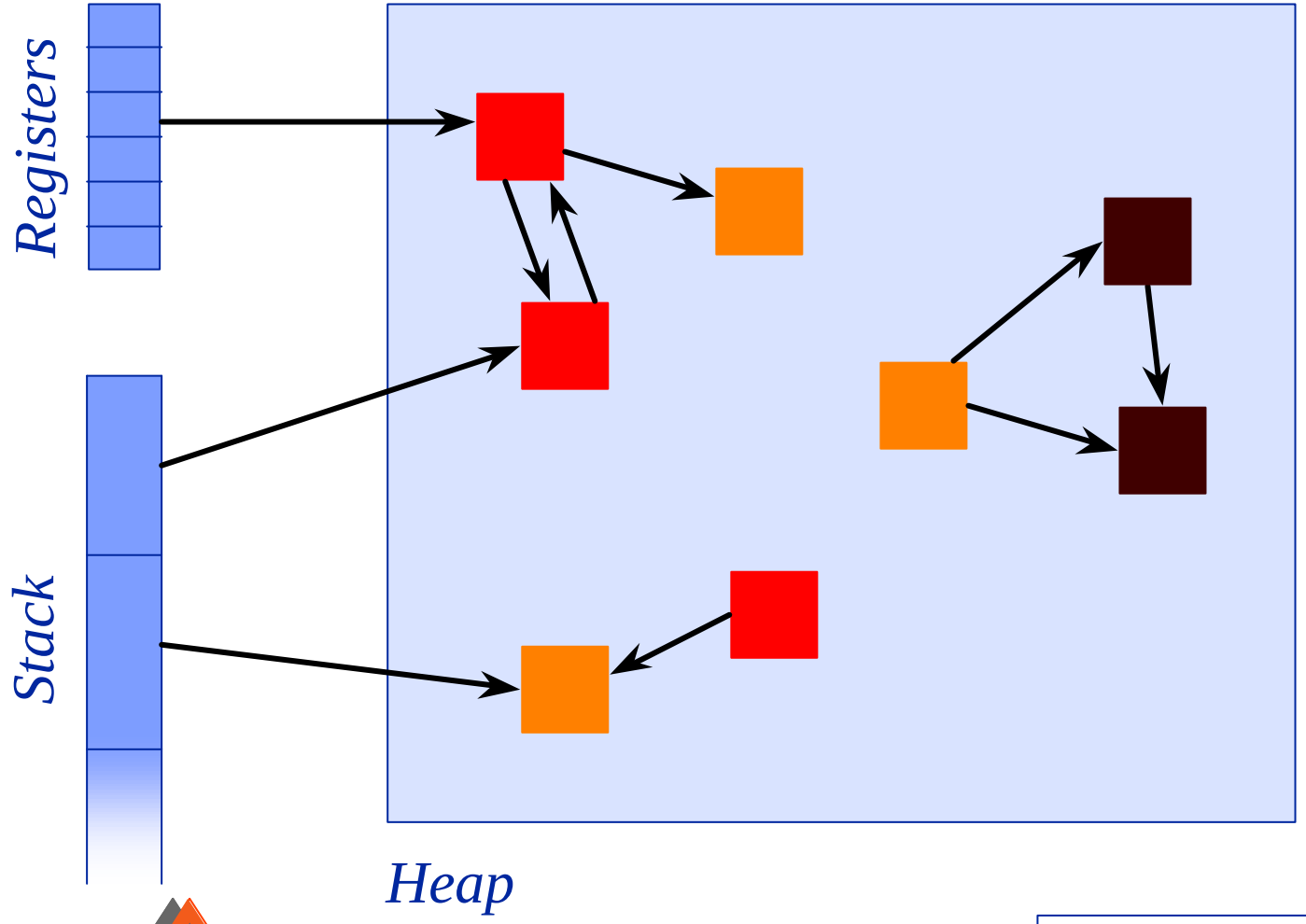
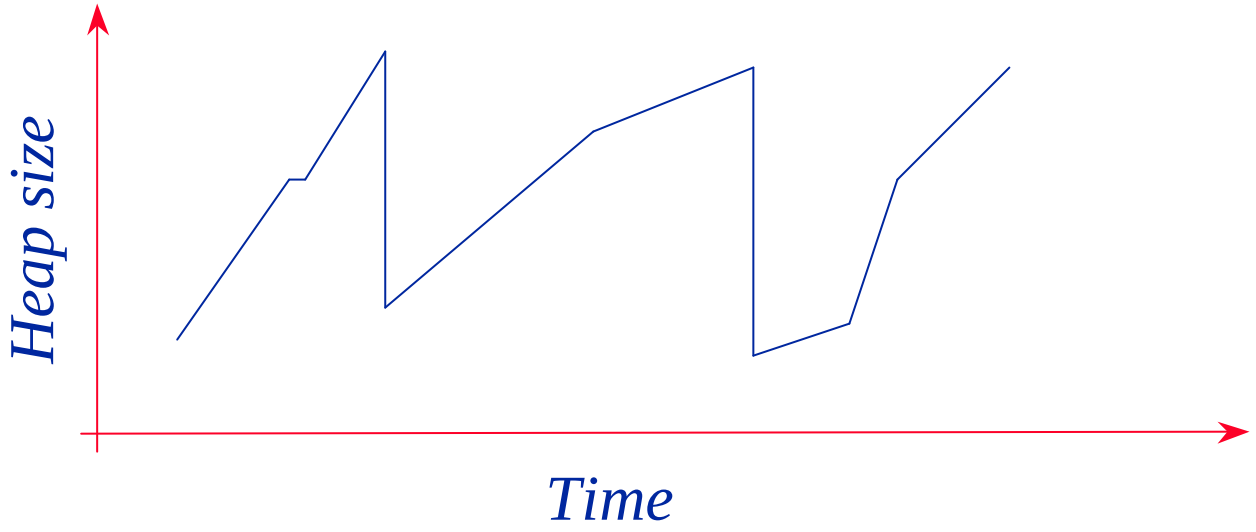
Using type information in automatic storage management

Tim Harris

Traditional heap organization



Problem : reclamation is very bursty

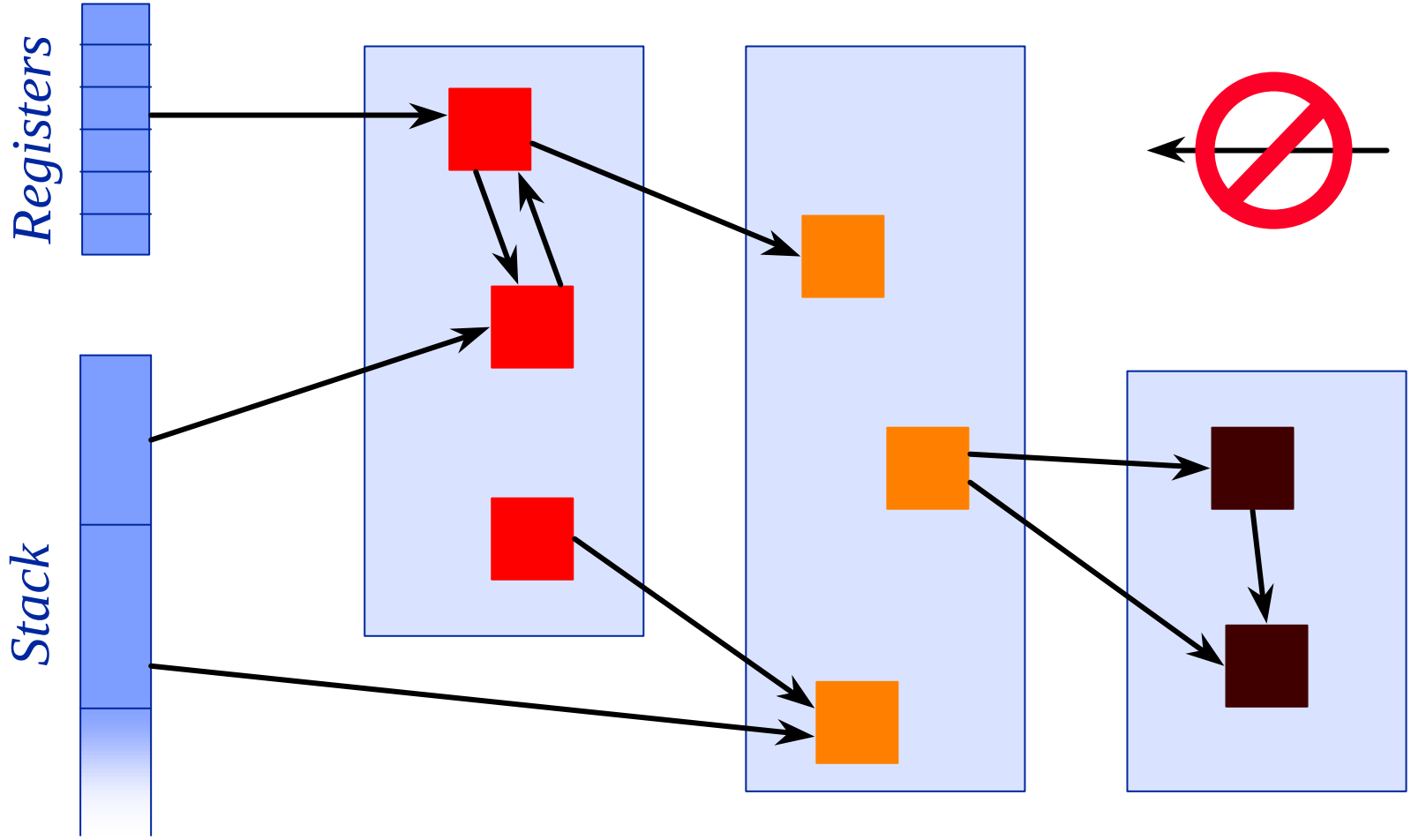


- The whole heap has to be checked before any storage can be reclaimed.

Generational collection

- Idea: separate newly created objects
  - Evidence that they are likely to soon become garbage
- Scan and reclaim space used by new objects without examining the entire heap
  - How are references from old to new objects handled?
  - When does an object become ‘old’?
- How else could the heap be divided?

Alternative : use type information



Potential benefits

- Some storage can be reclaimed after each section of the heap has been scanned
- Different sections can be scanned with different algorithms and maybe concurrently by different thread
- Some sections can be scanned more frequently than others

Remaining problems

- Can the classes for non-trivial applications be effectively divided into these sections?
- What happens when new classes are introduced at run-time?
- How are ‘generic’ references handled?
  - `java.lang.Object[]` in Java
  - `(void *)` in C++