Memory Management

- **Perfect**

	live	dead
not deleted	✓	– – –
deleted	– – –	✓

- **Manual management**

	live	dead
not deleted	✓	✗ memory leak
deleted	✗ ✗ dangling pointers	✓

- **Automatic garbage collection**

	live	dead
not deleted		
deleted		

Manual vs. Automatic

	Manual	Automatic
Languages	C, C++	Lisp, Java, ML
Advantages	performance	no programmers' effort no dangling pointers less memory leaks
Choice?		

Performance Metric for Garbage Collectors

II. Reference Counting

- **Free objects as they transition from "reachable" to "unreachable"**
- **Keep a count of pointers to each object**
- **Zero reference -> not reachable**
  - When the reference count of an object = 0
    - delete object
    - subtract reference counts of objects it points to
    - recurse if necessary
- **Not reachable -> zero reference?**
- **Cost**
  - overhead for each statement that changes ref. counts

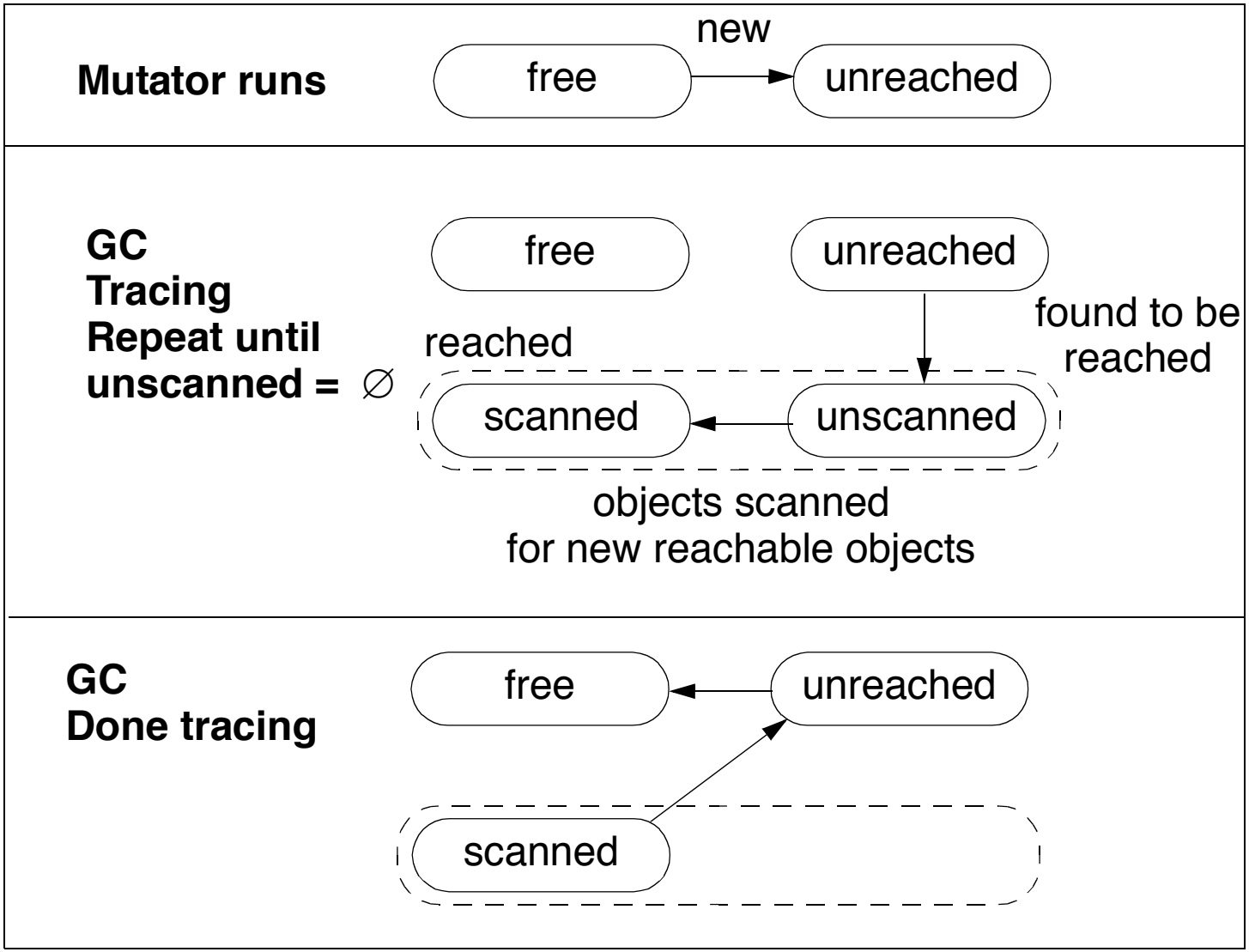
III. Why is Trace-Based GC Hard?

- **Reasons**
  - Requires complementing the reachability set - that's a large set
  - Interacts with resource management: memory

Trace-based GC

- **Reachable objects**
  - Root set: (directly accessible by prog. without deref'ing pointers)
    - objects on the stack, globals, static field members
  - + objects reached transitively from ptrs in the root set.
- **Complication due to compiler optimizations**
  - Registers may hold pointers
  - Optimizations (e.g. strength reduction, common subexpressions) may generate pointers to the middle of an object
  - Solutions
    - ensure that a "base pointer" is available in the root set
    - compiler writes out information to decipher registers and compiler-generated variables (may restrict the program points where GC is allowed)

Trace-Based GC: Memory Life-Cycle



A basic GC: Mark and Sweep

- **Data structures**
  - Free: a list of free space
  - Unscanned: a work list
  - A reach bit per object, set to 0 at the beginning
- **Algorithm**
  - Put objects in root set in Unscanned, set their reach bit
  - While Unscanned $\neq \varnothing$
    - remove object o from Unscanned
    - scan o for newly reached objects
      put in Unscanned, set their reach bit
  - Free = $\varnothing$
  - Sweep through heap space
    - unset reach bit: put in Free
    - otherwise: unset reach bit
- **Cost**
  - Mark: visit all reachable objects; sweep: visit all objects