Lec 22 First Class Functions

More Garbage Collection

▷ Tracing GC "stop the world"

1. pause prog
2. identify objects in heap that are still needed
3. collect useless one somehow

Heap scan : start from regs, global/local vars (= "root pointers") and do DFS

How to interrupt : when calling allocate

Which parts of heap/stack are ptrs :
- maybe assume all ?
- or keep type info in stack

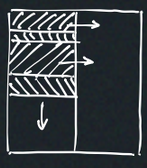
figuring out how large heap arrays are : more info to keep around (and their types)

Mark and sweep :
- reserve bits for each obj
- trace & mark obj
- reclaim unmarked objs

Few langs do this
- stopping is inefficient
- lots of fragmentation
  → possible to run defragmentation

▷ Copying GC

- heap has two semi spaces
- copy reachable into other space
- only allocate in one space



- incremental : put marker in old loc to redirect

▷ Generational Copying GC

Separate older and younger objs into different spaces
GC different spaces at different frequency
Move objs around by age

▷ GC ... and concurrency ...

▷ Type system-informed GC

e.g. linear type sys and each obj can only be used once

First Class Functions

Some languages

▷ C : & operator to get addr of function

```
typedef int optype(int, int);        int * int → int
```

cannot write local func

▷ C1 : func pointers

```
gdef ::= ... | typedef type ft ( type vid, ... )
type ::= ... | ft
unop ::= ... | &
exp  ::= ... | (*exp)(exp, ... )
```

ft considered large type

$$\frac{ft(\tau_1, \ldots) \to \tau \qquad \Gamma(f) = ft}{\Gamma \vdash \&f : ft*} \quad \ldots$$

⋮

HOFs ...