10-1

$\langle (M\ N), \mathcal{E}, k \rangle$

$\mapsto \langle M, \mathcal{E}, \underline{fun(N, \mathcal{E}, k)} \rangle$

k' $\qquad$ $k' = new\ fun(N, \mathcal{E}, k)$

$\langle V, \mathcal{E}, arg(\lambda X.M, k) \rangle$

k'' $\qquad$ free (k'')

$\langle M, \underbrace{\mathcal{E}[X \mapsto V]}, k \rangle \longrightarrow new\ Env(\mathcal{E}, X, V)$

"stack discipline" of memory

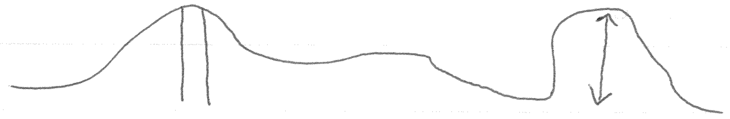
→ predictable memory alloc / free

"heap discipline"

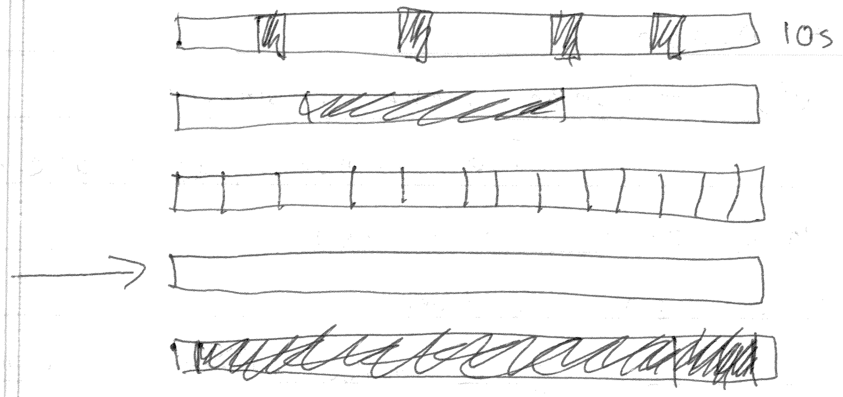
← no prediction

memory management is the study of the heap

- maximize free memory
- all requests for memory are satisfied
  - ↳ infinite memory (not real)
- good simulation under constraints (peak utilization)



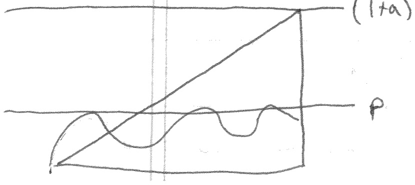
(1+a) P require
(a=overhead)



MMU - max mutator utilization (total not-MM area)

close to 1 good

Pause time - gaps between your program (avg/max size of MM block) (close to 0 good)



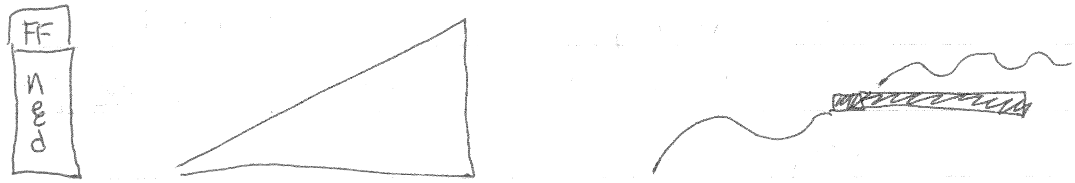
10-2/ MM should be SOUND

memory that is needed must always be available

```
int b[50] =
int a[50] = ...
      x = 53;
  a[x] = 1;
```

memory respects program abstraction

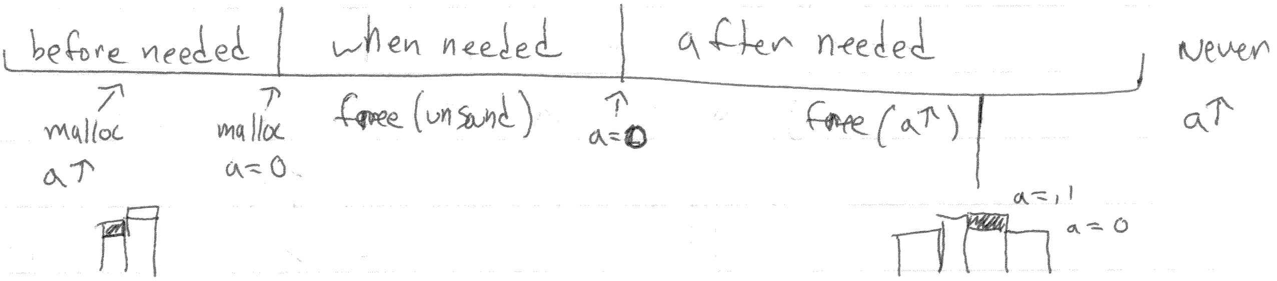
forget to free — a↑ (overhead)



~~free later than needed~~ free before needed

```
free(p)
p=>x = 22;
```

NOT SOUND



malloc / free + human

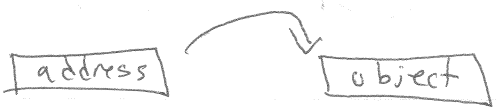
- sound : NO — human makes mistakes
- overhead : "To err is human, to forgive divine"
  - a is unbounded because of human
  - a is one word per allocation
  - O(lg n)
- time: O(lg n) [MMU] → malloc() + free() complexity
- human controls pause time & block length
  - naive — long pauses + big blocks
  - sophisticate — avoid that

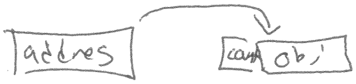
10-3

Smart Pointer (reference counting)

"o.f = ptr"   ptr is used

"o.f = NULL" }
or "o.f = ptr2" } ptr is not used

normal ptr = 

smart ptr = 

when count==0, free()

```
retain (p) {                release (p) {
  p.count++                   p.count--
}                             if (p.count == 0)
                                free
                            }
```

GOAL: SOUNDNESS

NO: human still puts retain/release

- AUTOMATIC is better, but still not sound

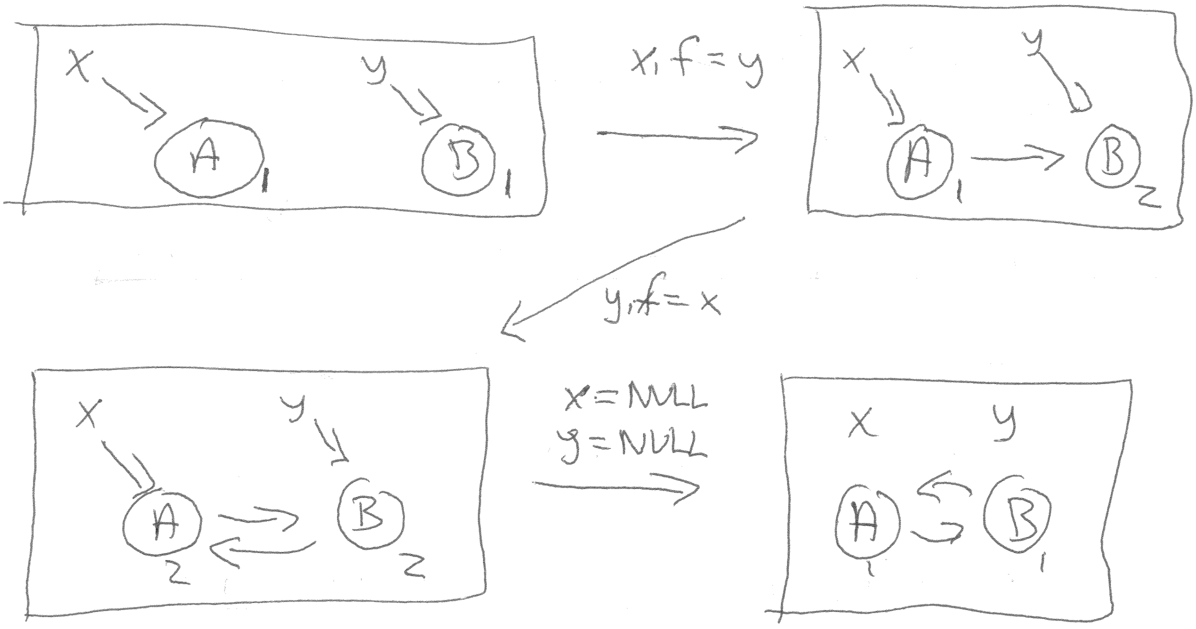
```
= if (p.count ≠ MAX)    |    if (p.count ≠ MAX)
    p.count++           |       p.count--
```

- mem overhead ↑ because of counts

- increase interaction with MM ∝ work
  ↳ very bad cache behavior

+ easier to queue frees

10-4



Cycles are never freed