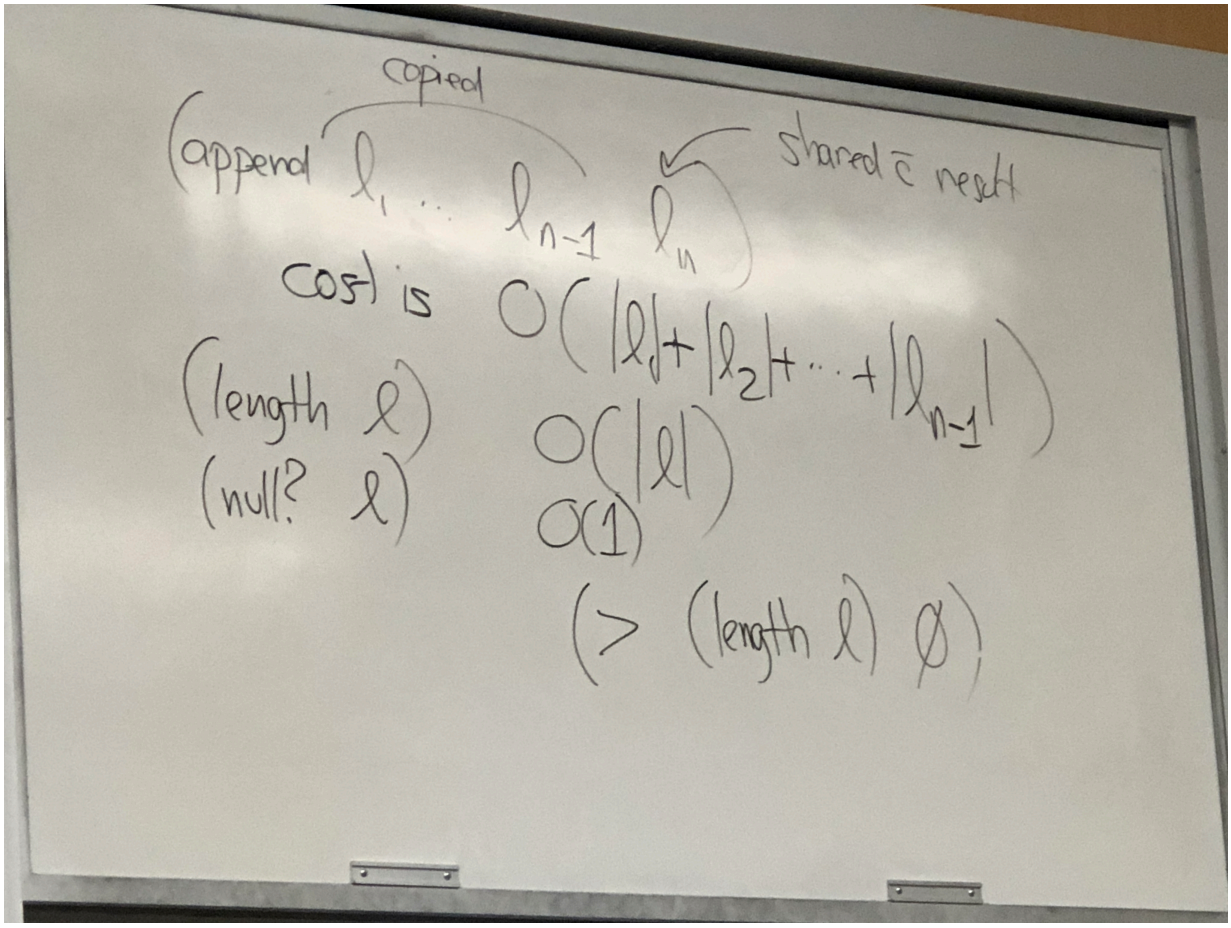
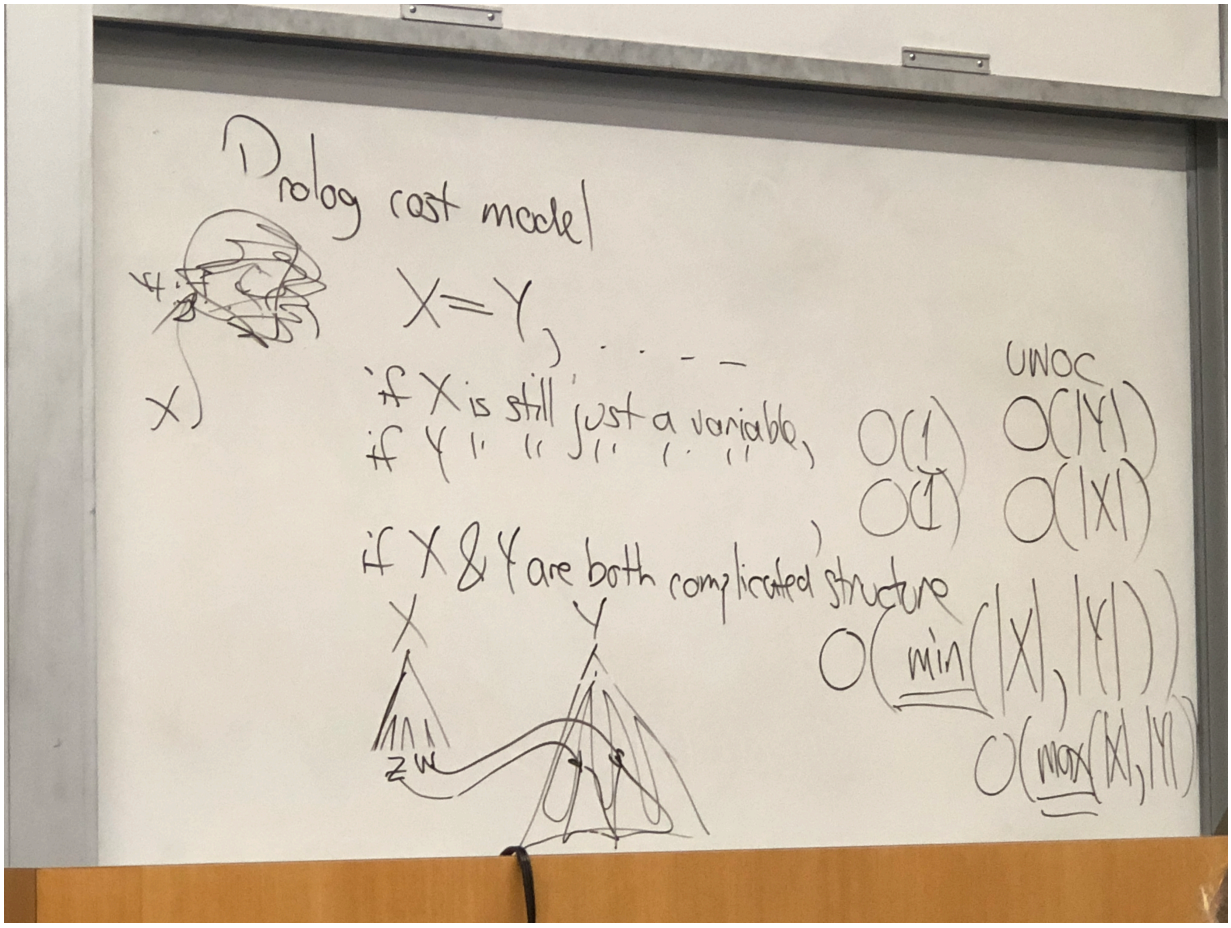
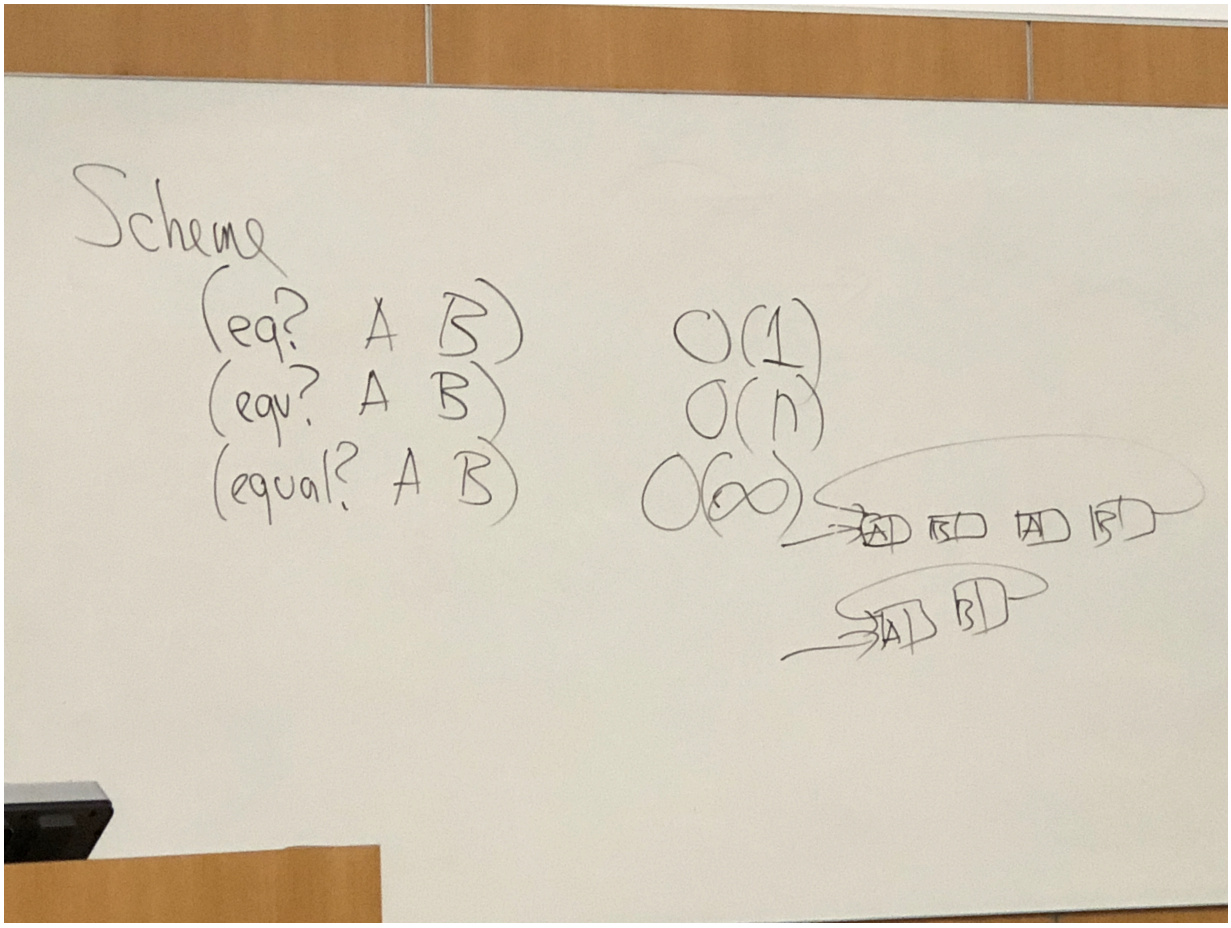

copied
shared & reset
(append $l_1 \cdots l_{n-1}$ l_n)
cost is $O(|l_1| + |l_2| + \cdots + |l_{n-1}|)$
(length l) $O(|l|)$
(null? l) $O(1)$
(> (length l) $\emptyset$)

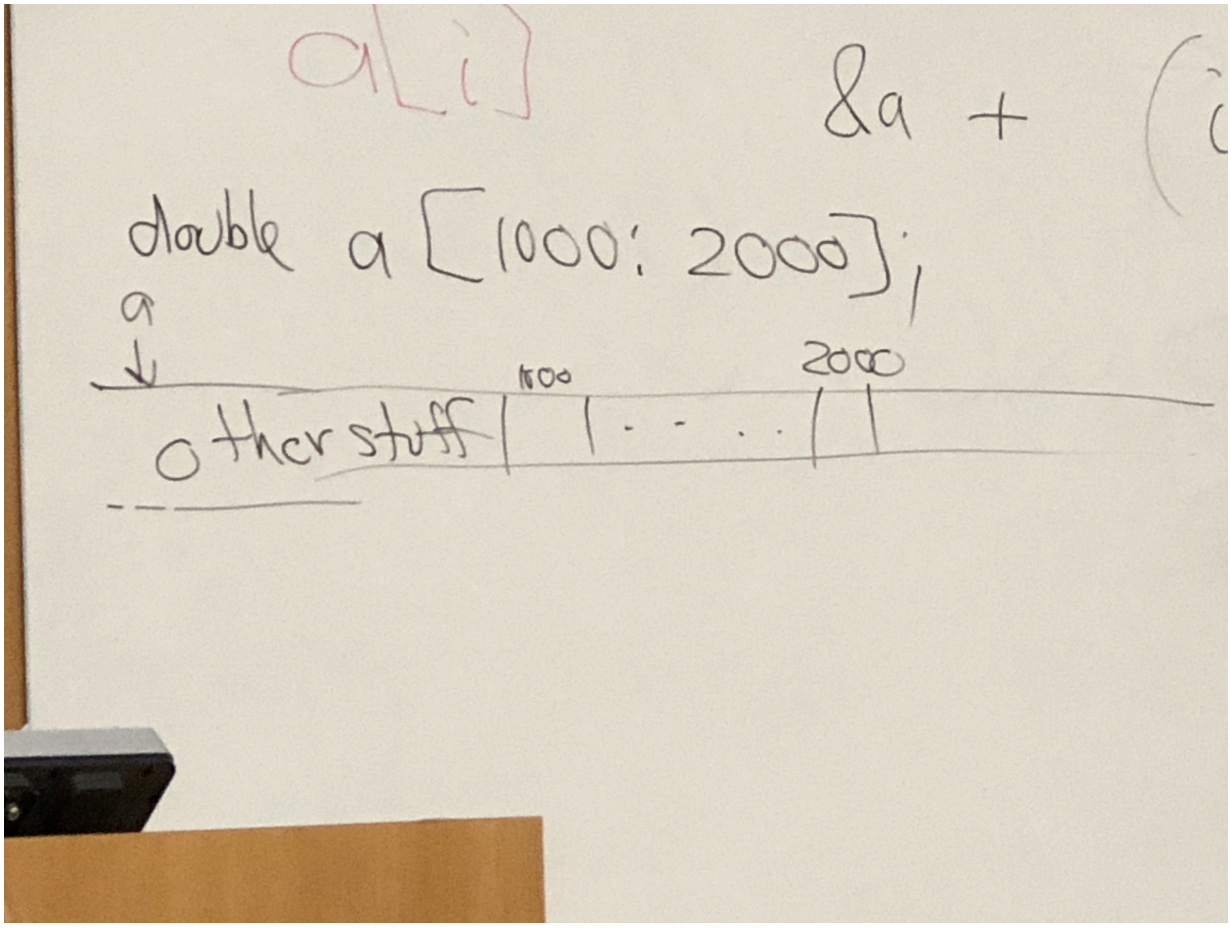

Prolog cost model
X=Y, ...
if X is still just a variable,
if Y " " " " ,
if X & Y are both complicated structure
X
Y
Z
W
UNOC
O(1) O(|Y|)
O(1) O(|X|)
O(min(|X|, |Y|))
O(max(|X|, |Y|))



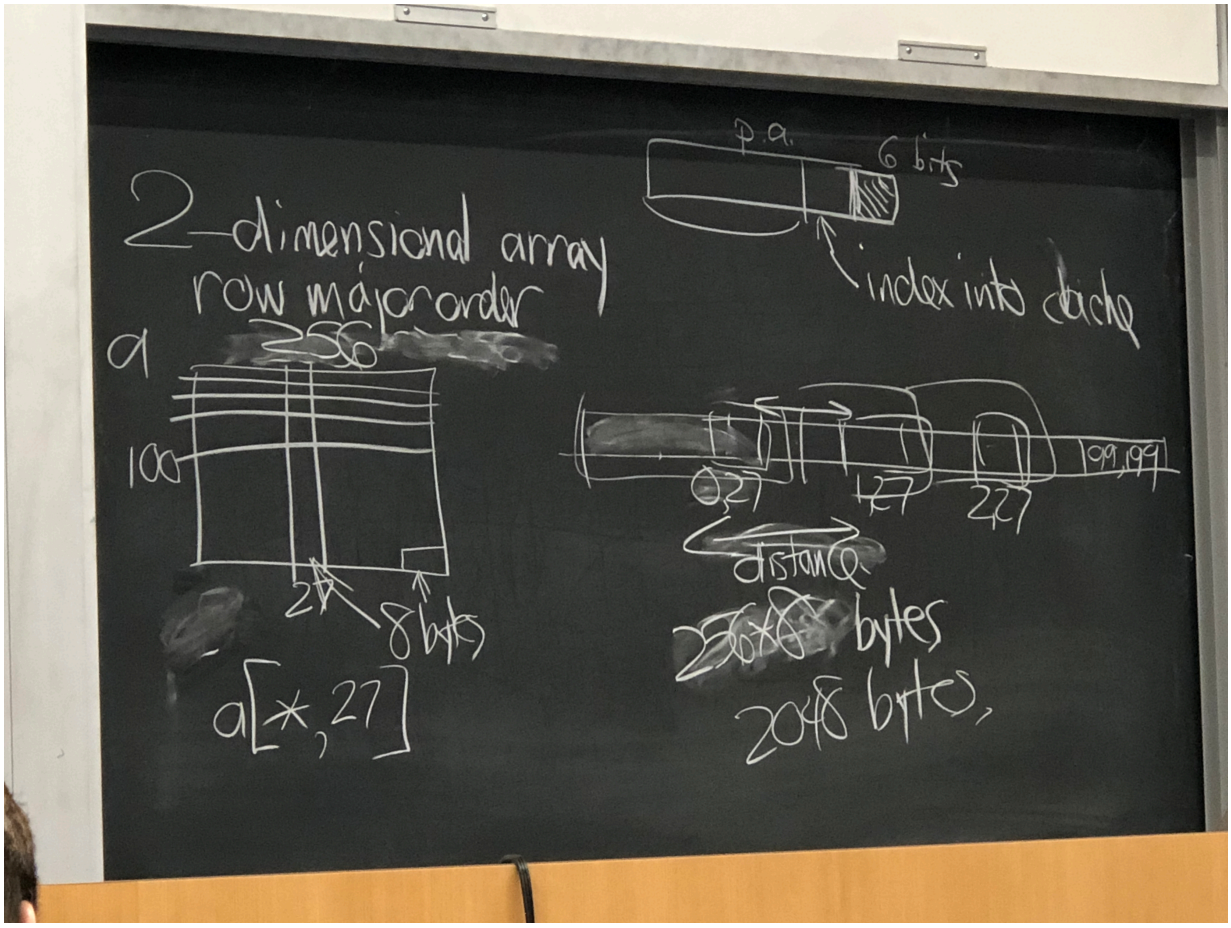
Array access cost
Powers of 2
a[i]
&a + (i - lb(a)) * sizeof a[0]

double a[1000: 2000];

- if we use a power of 2, then we can get rid of multiply and replace with shift (better performance)
-



Non-Powers of 2
Can actually be beneficial if we have 2D array
Want to access a column
But entries are 200*8 bytes away from each other, not power of 2 so access should not be power of 2



What if we increase shit to power of 2
200 -> 256
256*8 bytes away from each other: fucks up our placement of memory into cache
Kills spatial/temporal locality

Semantics
What does a program mean? (as opposed to syntax, what does a program look like)

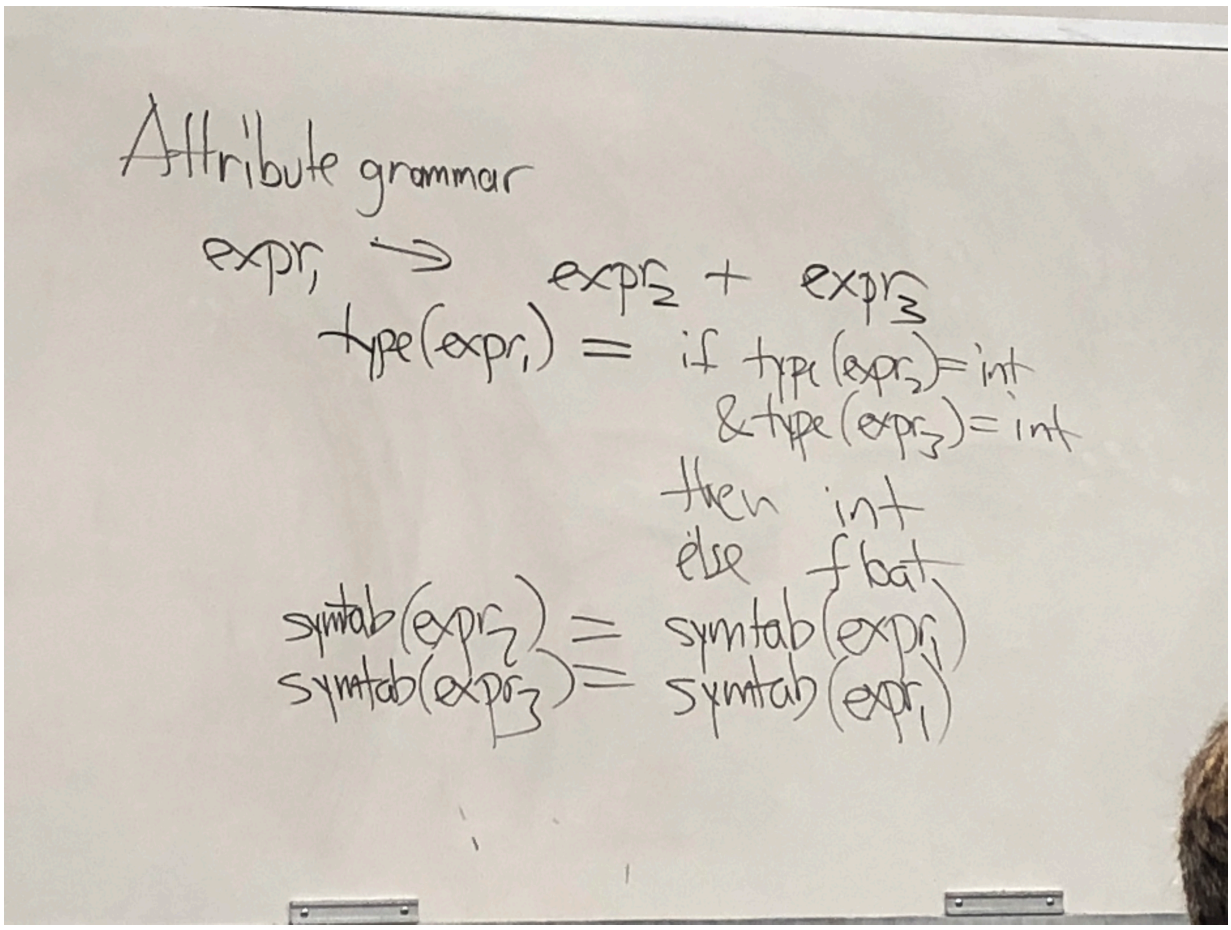
static semantics
- how much you can figure out about a program while it's not running
- harder than syntax to find out
- **attribute grammars**

dynamic semantics
- meaning when the program is running
- hardest to find out
- **operational semantics**
- **axiomatic semantics**

- **denotational semantics**

Static Semantics:
Attribute Grammar
expr -> expr + expr

- we know is ambiguous: but don't care right now because that's a syntax problem



Attribute grammar is harder to find out, but once you have it you can write a compiler and

- specify type checking
- scope checking

Dynamic Semantics :
Operational Semantics

- uses abstract imperative interpreter

Axiomatic Semantics

- uses axioms and rules of inference to define stuff

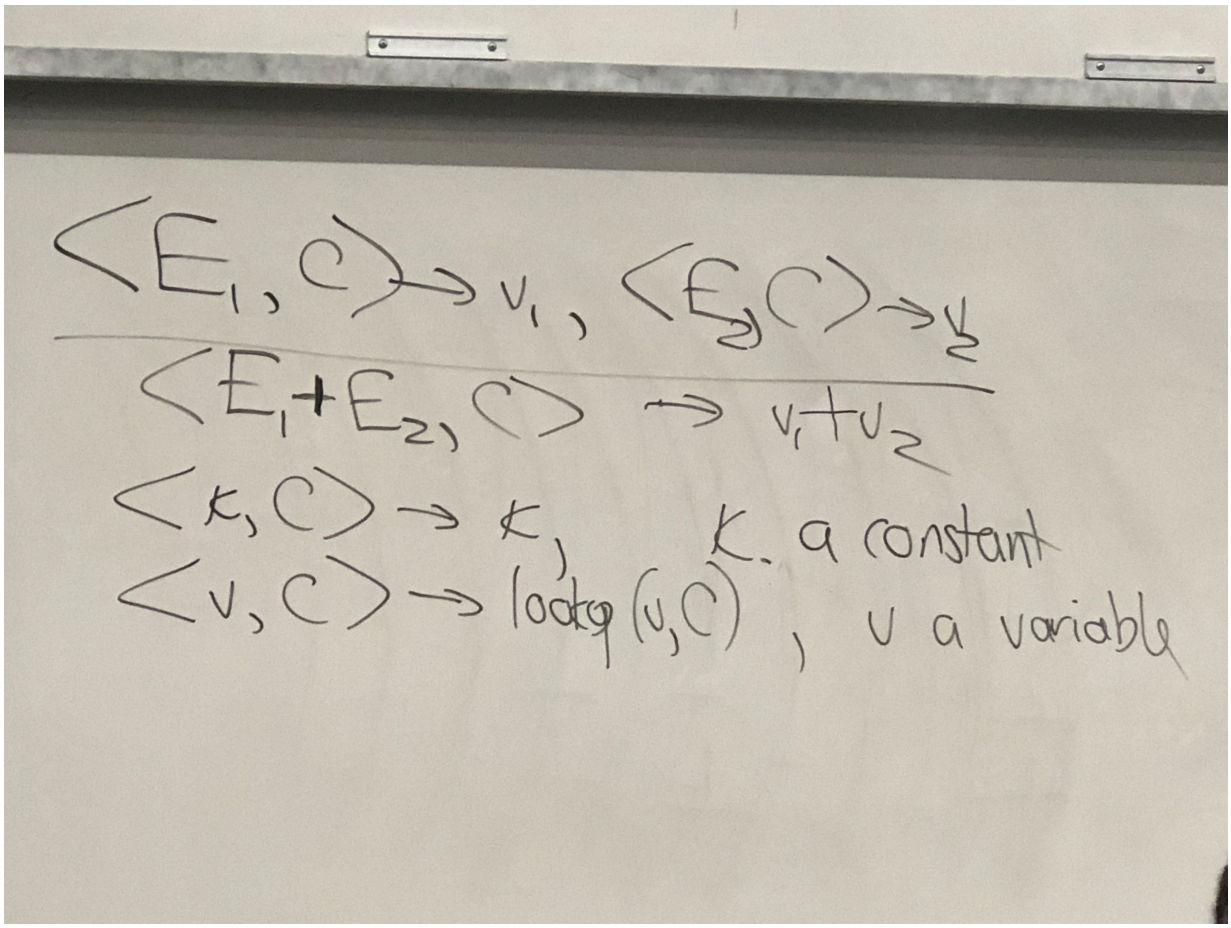
Denotational semantics

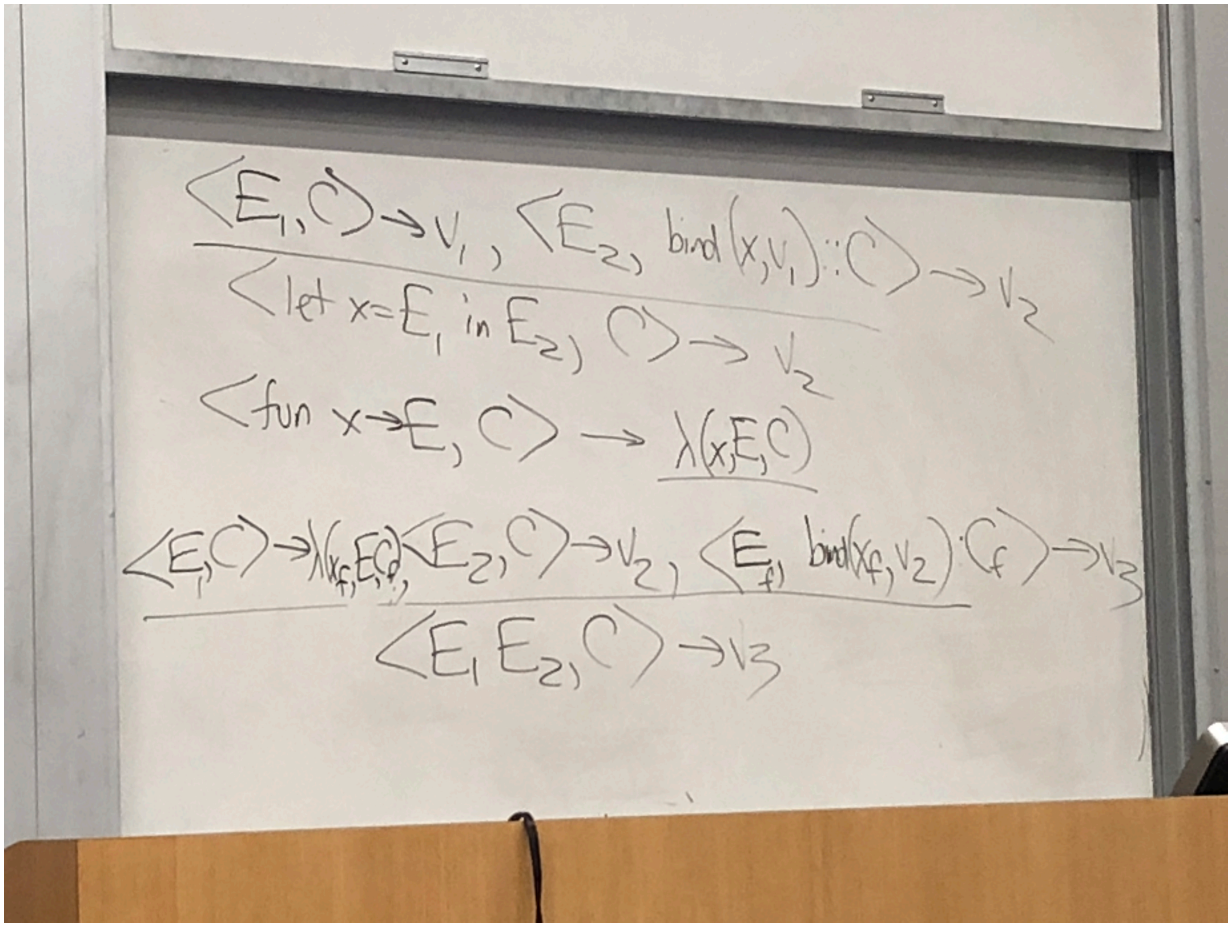
- maps programs to functions (mathematical)
- hardest one

Types of languages we've studies (language paradigms)

- logic
- functional
- imperative

People use specific semantics to talk about each paradigm of language: i.e. functional uses denotational semantics usually





the fact that the bottom line is c_f at the end rather than c, we know that we have static scoping (why??)
Semantics can show how to figure out if there's static scoping or dynamic scoping, more information, etc etc.