

CSE 114A Lecture 3

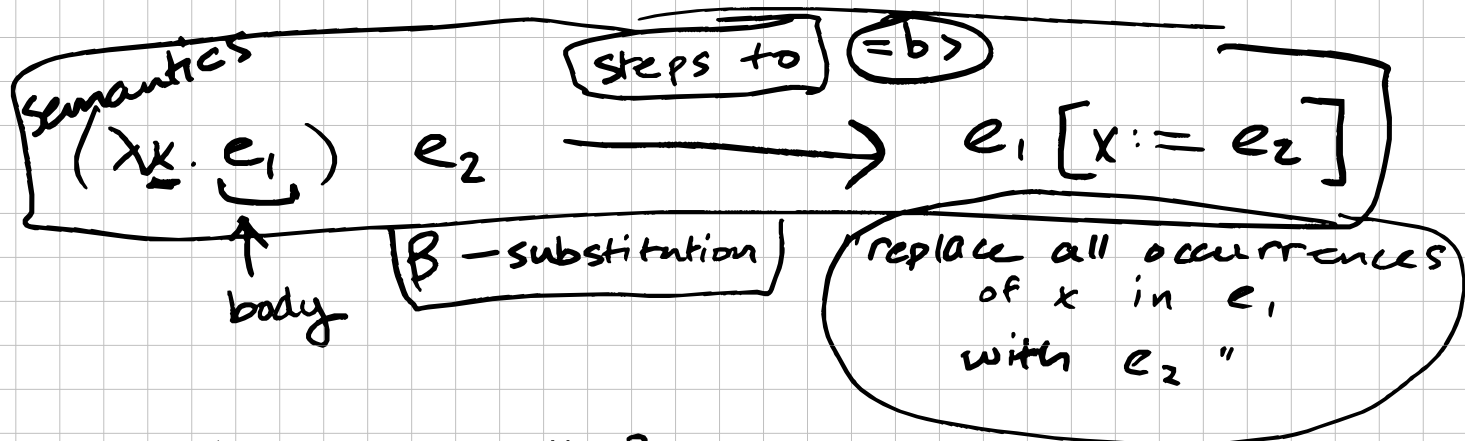
agenda:

- ✓ Announcements
- ✓ Recap last time
 - lambda calc syntax & substitution rule
 - scope of variables; free & bound variables
 - capture-avoiding substitution
 - renaming
- Live coding in ELSA!
 - back to booleans / conditionals
 - numbers
 - pairs
 - recursion

lambda calculus

Syntax $e ::= x \mid \lambda x. e \mid e_1 e_2$

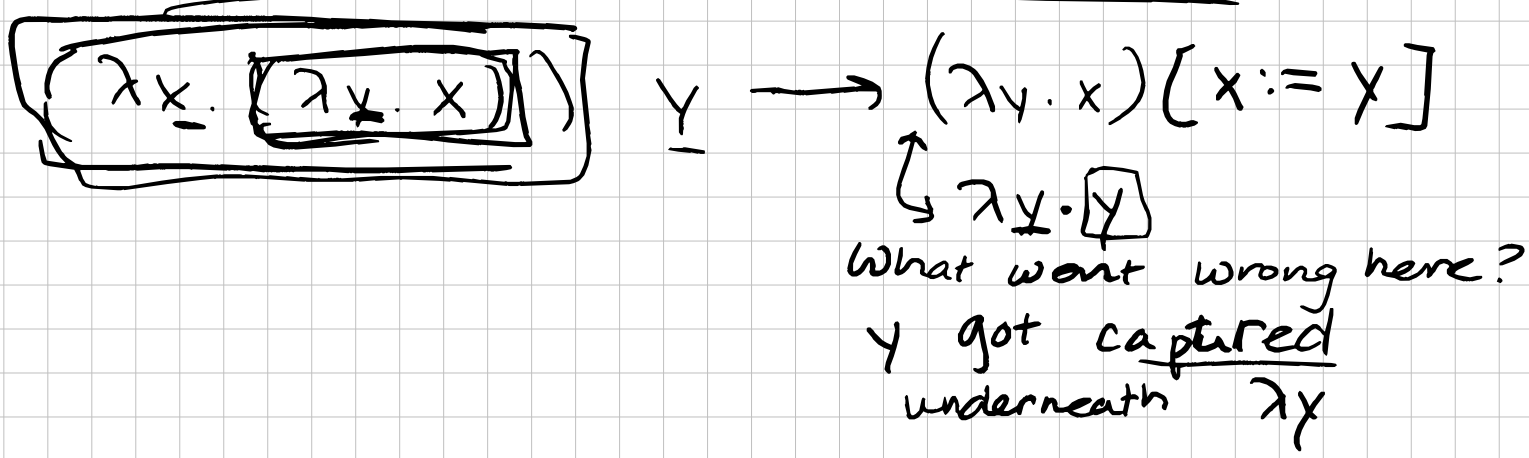
↑ variables function definition aka lambda abstraction function application aka function call



What's a "value"?

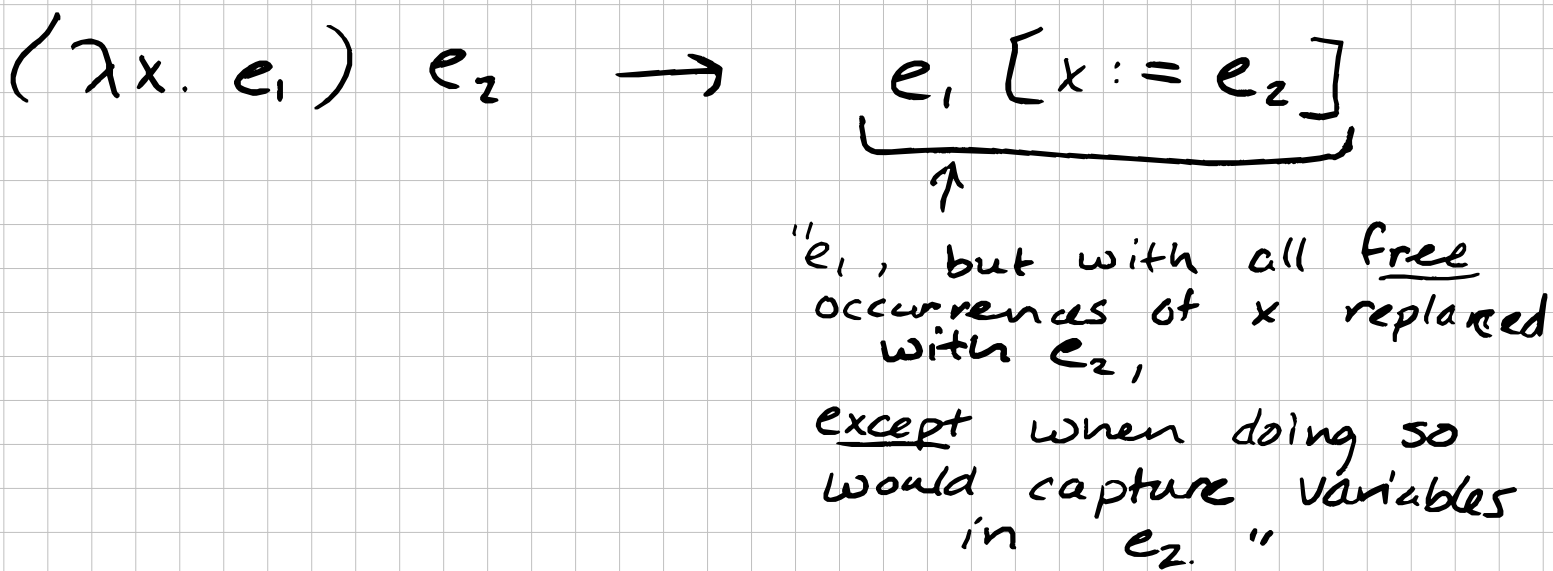
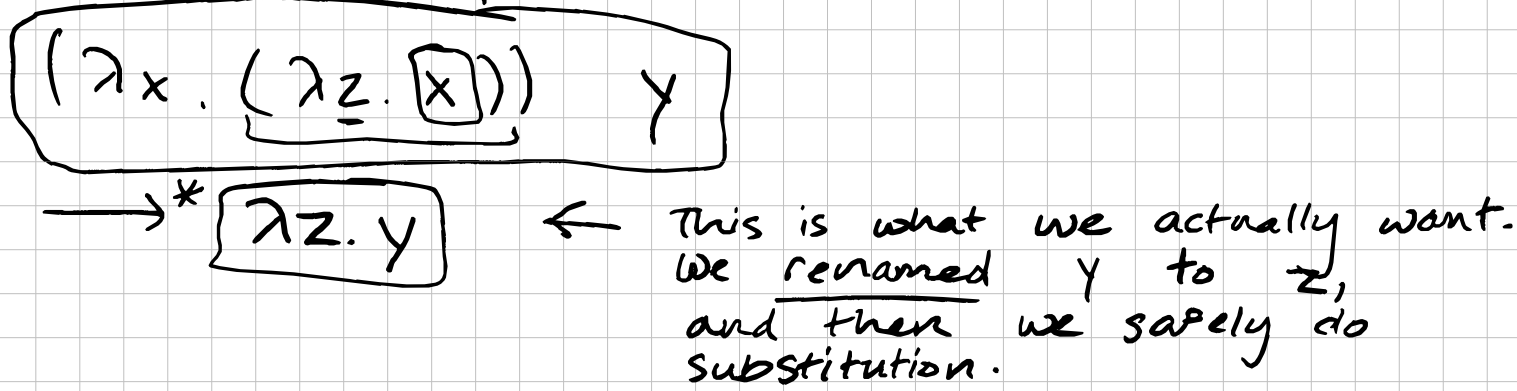
$\lambda x. e$ ← this is a value!

Even if this e here is some large expression, if the expression can't be evaluated further, we have a value.



$(\lambda x. (\lambda y. x)) y$

↑ we should rename y to something that won't accidentally capture a variable.



This is called capture-avoiding substitution.

Free and bound variables

$\lambda x. x$ x is bound by λx .
No free variables.

$\lambda x. (\lambda y. x)$ x is bound by λx .
No free variables.

$\lambda x. \lambda y. z y$ y is bound by λy .
 z is free.

$\lambda z. \lambda x. \lambda y. z y$ Now z is no longer free.