Lecture 05.29.26. FRIDAY

- Announcements
  - Posted the slides link on Ed Discussion
  - I will post my notes later today
  - Next week? Post a poll of what you want for last lectures
  - Assignment due saturday
- Agenda
  - Finish up unification
  - Type inference algorithm.

* UNIFICATION
  - Type variables, a, b, c
  - Type substitution
    [a/Int, b/c → c] substitution
  - Applying a substitution
    [a/Int, b/c → c]
    a → a Int → Int
  - If there are no type variables in the type that the substitution is being applied to, then it stays the same.
    a → d Int → d

* UNIFIERS
  a → Int
  Bool → b
  } [a/Bool, b/Int]
  Bool → Int

  ? Question ?
  ~~[Bool/a, Int/b]~~ ?
  - This is not a substitution
  - A substitution only maps type variables (a, b, c, ... etc.) to types
  - Bool + Int are not type variables so they can't go on LHS
  - Caveats
    - If the types I have are the same to begin with, then an empty substitution will unify them. a → b, a → b []
    - a, b (type variables) and a to different then [a/b,] + [b/a] are unifiers.
    - If I have type variable + another type. a, Int → Bool then [a/ Int → Bool]

? Question?
- Suppose I have a
  Int → a
  Can I unify these? Why or why not?
  - yes?
  - No?
  [a / Int → a] ? No
  - If we apply this a to Int → a
  - If we apply this to Int → a
    Int → Int → a
  * You can't unify a variable with any type that contains free occurences of that variable.
    - "occurs check"

- Examples

1	a + Int	✓	[a/Int]
2	a → a + Int → Int	✓	[a/Int]
3	a + Int → Int	✓	[a/Int → Int]
4	a + b → c	✓	[a/b → c]
5	a → Int + Int → b	✓	[a/Int, b/Int]
6	Int + Int	✓	[]
7	a + a	✓	[]
8	a + b	✓	[a/b]
9	Int + Int → Int	✗	
10	Int + a → a	✗	

- Question
  can we use a-step for this
  Add my slides.