CSE114A lecture 15!

- ✓ - announcements
  - Types, type inference, polymorphism
  - if time - example of dynamic scope "in the wild"

Announcements.

- Code walks, round 2 are next week! Sign up! (HW3, HW4)
  - Tuesday - Friday
- midterm - look at Gradescope!
- check Zulip for my announcements about grading methodology
- "Getting To Know You" survey. 20+ minutes or so, but please do it.
- photographer in class

we spent the last couple weeks implementing a language.

- interpreting ASTs into values.
  - parsing strings into ASTs.
    strings: representing programs

Now what? The type system for our language. in particular, type inference.

we want to statically (i.e., without running the program) check that our Nano programs "make sense".

type system - to formalize our intuition about which programs make sense

type inference - compute the type of an expression.

---

language:
- syntax (Expr type)
- semantics (interpreter) - assigns values to expressions.

type system:
✓ - syntax of types
- (static) semantics of our language that assigns types to expressions

consider a "mini-Nano"

$$e ::= n \mid x \mid e_1 + e_2 \mid \backslash x \to e \mid e_1\ e_2 \mid \text{let } x = e_1 \text{ in } e_2$$

(numbers, variables, arithmetic, lambdas, application, let-expressions)

application: aka function call

$$T ::= \text{Int} \mid T_1 \to T_2$$

The syntax of types in mini-Nano.

Now the interesting part: static semantics.

We want to define a typing relation

$$\boxed{\Gamma \vdash e :: T} \qquad T ::= \text{Int} \mid T_1 \to T_2$$

pronounced "In type environment Gamma, expression e has type T"

We define the typing relation using inference rules.

$$\frac{\text{premise1} \quad \dots \quad \text{premise N}}{\text{conclusion}} \quad \leftarrow \text{Things we get to assume}$$

example:

$$\frac{\text{"it always rains on Thursdays"} \qquad \text{"Today is Thursday"}}{\text{"it's raining"}}$$

$$\frac{\Gamma \vdash e_1 :: \text{Int} \qquad \Gamma \vdash e_2 :: \text{Int}}{\Gamma \vdash e_1 + e_2 :: \text{Int}}\ [\text{T-Add}] \qquad (\backslash x \to x) + 1$$

$$\frac{n \in \mathbb{Z}}{\Gamma \vdash n :: \text{Int}}\ [\text{T-Int}]$$

we can give types to: 3, 4, 3 + 4, 3 + (4 + 5)

$$\dfrac{\dfrac{3 \in \mathbb{Z}}{\Gamma \vdash 3 :: \text{Int}}\ [\text{T-Int}] \qquad \dfrac{\dfrac{4 \in \mathbb{Z}}{\Gamma \vdash 4 :: \text{Int}}\ [\text{T-Int}] \quad \dfrac{5 \in \mathbb{Z}}{\Gamma \vdash 5 :: \text{Int}}\ [\text{T-Int}]}{\Gamma \vdash 4 + 5 :: \text{Int}}\ [\text{T-Add}]}{\Gamma \vdash 3 + (4 + 5) :: \boxed{\text{Int}}}\ [\text{T-Add}]$$

↑ This is a typing derivation that proves ~~that~~ the expression $3 + (4+5)$ has type Int.

$$\frac{(x, T) \text{ is in } \Gamma}{\Gamma \vdash x :: \boxed{T}}\ [\text{T-Var}]$$

$(\backslash x \to x)$

let $x = 3$ in $x + 4$

we dealt with variables in our interpreter by looking them up in an environment, binding variables to values.

In our type system, we'll need a type environment, binding variables to types.

[("x", Int), ("f", Int → Int)] ← an example of a type environment

$$\frac{\Gamma, (x, T_1) \vdash e :: T_2}{\Gamma \vdash \backslash x \to e :: \boxed{T_1 \to T_2}}\ [\text{T-Lam}]$$

$$\frac{\Gamma \vdash e_1 :: T_1 \to T_2 \qquad \Gamma \vdash e_2 :: T_1}{\Gamma \vdash e_1\ e_2 :: \boxed{T_2}}\ [\text{T-App}]$$

↑ (This should remind you a little of my "it's raining" example.)

A typing derivation for the expression $(\backslash x \to x)\ 3$

$$\dfrac{\dfrac{\dfrac{(x, \text{Int}) \text{ is in } \Gamma, (x, \text{Int})}{\Gamma, (x, \text{Int}) \vdash x :: \boxed{\text{Int}}}\ [\text{T-Var}]}{\Gamma \vdash (\backslash x \to x) :: \boxed{\text{Int} \to \text{Int}}}\ [\text{T-Lam}] \qquad \dfrac{3 \in \mathbb{Z}}{\Gamma \vdash 3 :: \text{Int}}\ [\text{T-Int}]}{\Gamma \vdash (\backslash x \to x)\ 3 :: \boxed{\text{Int}}}\ [\text{T-App}]$$

[T-Let] next time.