

CS 252:

Advanced Programming Language Principles



The Simply Typed Lambda Calculus

Prof. Tom Austin

San José State University

REVIEW: Arith Language

```
e ::= true
   | false
   | i
   | succ e
   | pred e
   | iszero e
   | if e then e else e
```

```
v ::= true
   | false
   | i
```

Types for Arith

$$T ::= \text{Bool}$$
$$| \text{Int}$$

Typing rules for Arith (in-class)

Adding lambda functions

$$e ::= \dots$$
$$| \lambda x . e$$
$$| e e$$
$$| x$$
$$v ::= \dots$$
$$| \lambda x . e$$

Small-step evaluation rules for
the Lambda Calculus
(in-class)

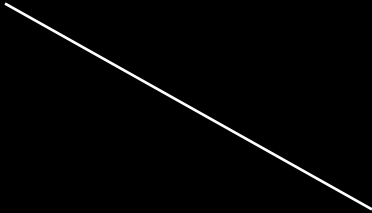
So what is the type of a function?

$$\begin{aligned} T &::= \text{Bool} \\ &| \text{Int} \\ &| T \rightarrow T \end{aligned}$$

Options to determine function type

1. type inference
2. require type annotations

Typing Annotations

$$\lambda x : \mathbf{T} . e$$


Type of the
argument

Arith w/ Lambdas

```
e ::= true | false | i
    | succ e | pred e
    | iszero e
    | if e then e else e
    |  $\lambda x:T.e$ 
    | e e
    | x
```

What is the type of x ?

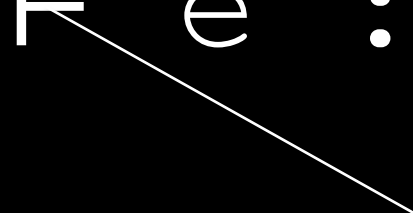
- How do we determine a variable's type in a function?
- We use a *typing environment*:
 - maps variables to types
 - we use the Greek letter gamma: Γ

New typing relation

We need to write our typing rules *given our new typing environment*.

Our revised typing relation is:

$\Gamma \vdash e : T$



This is known as the "turnstile"

Typing rules for the
Simply Typed Lambda Calculus
(in-class)

How powerful is this language?

Implement a typechecker for the Simply Typed Lambda Calculus

Download `stlc.hs`.

Complete the `typecheck` method
following the rules outlined in class.