Syntax

Lambda calculus syntax

$$t, t' ::= x \mid \lambda x.t \mid t\ t'$$

Syntax

variables | functions | function application

`t, t'` $::=$	`x`	`\x -> t`	`t t'`	Haskell
`t, t'` $::=$	`x`	`fun (x) -> t`	`t t'`	OCaml
`t, t'` $::=$	`X`	`fun (X) => t end`	`t(t')`	Erlang
`t, t'` $::=$	`x`	`x -> t`	`t.apply(t)`	Java

Typing rules
(linear)

Typing syntax and relation

Church syntax adds a type "signature"

$$t ::= x \mid \lambda(x : A).t \mid t\ t$$

Type syntax $A, B ::= A \to B$ $\mid \texttt{Int} \mid \texttt{Bool} \mid \texttt{....}$

cf Haskell: **`t -> t'`**

In a full language we'd want more…

Typing lets us relate expressions to types, e.g.

$$\lambda(x : A).x : A \to A$$

Cf.:

```
id :: a -> a
id = \x -> x
```

Quick exercise:

Q: What is the type of this lambda term?

$$\lambda(x : A).\lambda(y : B).x$$

A: $\lambda(x : A).\lambda(y : B).x : A \to (B \to A)$

Cf.:

```
const :: a -> b -> a
const x y = x
```

Q: What is the type of this lambda term?

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

A: *It depends!*

Typing syntax and relation

Typing *judgement* with *assumptions* about variable types

$$y : B \vdash \lambda(x : A).y : A \to B$$

Assumptions **Term** **Type**

Syntax of assumptions

$$\Gamma ::= \Gamma, x : A \mid \emptyset$$

Typing *judgement* form: $\Gamma \vdash t : A$

Typing rules

Defined inductively

Base case:

$$\frac{}{\textbf{conclusions}}$$

Inductive step:

$$\frac{\textbf{premises (inductive hypotheses)}}{\textbf{conclusions}}$$

$$\text{var}\ \frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}$$

A term which is just one variable, takes its type from the context

$$\text{abs}\ \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda(x : A)\,.\,t : A \to B}$$

Binds a variable out of the context

$$\text{app}\ \frac{\Gamma \vdash t_1 : A \to B \qquad \Gamma \vdash t_2 : A}{\Gamma \vdash t_1\ t_2 : B}$$

Shares the context between two sub terms

Example

$$\lambda(x : A).\lambda(y : A \to B).y\ x : \qquad \textbf{???}$$

Example

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{x : A \vdash x : A}\,\text{var} \qquad \dfrac{}{y : A \to B \vdash y : A \to B}\,\text{var}}{x : A, y : A \to B \vdash y\ x : B}\,\text{app}}{x : A \vdash \lambda(y : A \to B).y\ x : (A \to B) \to B}\,\text{abs}}{\emptyset \vdash \lambda(x : A).\lambda(y : A \to B).y\ x : A \to ((A \to B) \to B)}\,\text{abs}
$$