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

Typing syntax and relation

Church syntax adds a type "signature"

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

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

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: $\dfrac{}{\textbf{conclusions}}$

Inductive step: $\dfrac{\textbf{premises (inductive hypotheses)}}{\textbf{conclusions}}$

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

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

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

Binds a variable out of the context

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

Shares the context between two sub terms

Example

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

Example

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

PCF
Programming Computable Functions

Dana Scott / Gordon Plotkin 1977

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

$$\mid \mathsf{zero} \mid \mathsf{succ}\ t \mid \mathsf{fix}\ t$$

$$\mid \mathsf{case}\ t\ \mathsf{of}\ \mathsf{zero} \to t_1 \mid \mathsf{succ}\ x \to t_2$$

Syntax

$$A, B ::= A \to B \mid \mathsf{nat}$$

Types

$$\frac{}{\Gamma \vdash \mathsf{zero} : \mathsf{nat}} \qquad \frac{\Gamma \vdash t : \mathsf{nat}}{\Gamma \vdash \mathsf{succ}\ t : \mathsf{nat}} \qquad \frac{\Gamma \vdash t : A \to A}{\Gamma \vdash \mathsf{fix}\ t : A}$$

$$\frac{\Gamma \vdash t : \mathsf{nat} \qquad \Gamma \vdash t_1 : A \qquad \Gamma, y : \mathsf{nat} \vdash t_2 : A}{\Gamma \vdash \mathsf{case}\ t\ \mathsf{of}\ \mathsf{zero} \to t_1 \mid \mathsf{succ}\ y \to t_2 : A}$$

$$\frac{t \rightsquigarrow t'}{(\mathsf{case}\ t\ \mathsf{of}\ \mathsf{zero} \to t_1 \,|\, \mathsf{succ}\ y \to t_2) \rightsquigarrow (\mathsf{case}\ t'\ \mathsf{of}\ \mathsf{zero} \to t_1 \,|\, \mathsf{succ}\ y \to t_2)}$$

$$\frac{}{(\mathsf{case}\ \mathsf{zero}\ \mathsf{of}\ \mathsf{zero} \to t_1 \,|\, \mathsf{succ}\ y \to t_2) \rightsquigarrow t_1}$$

$$\frac{}{(\mathsf{case}\ \mathsf{succ}\ t \mathsf{of}\ \mathsf{zero} \to t_1 \,|\, \mathsf{succ}\ y \to t_2) \rightsquigarrow t_2[t/y]}$$

$$\frac{t \rightsquigarrow t'}{\mathsf{succ}\ t \rightsquigarrow \mathsf{succ}\ t'}$$

$$\frac{}{\mathsf{fix}\ t \rightsquigarrow t\,(\mathsf{fix}\ t)}$$