Last time

$$\Gamma \vdash M : ?$$

Polymorphic type inference and mutable state

```
let r = ref None in
    r := Some "boom";
    match !r with
      None -> ()
    | Some f -> f ()
```

The value restriction

Only generalize if M is a *syntactic value*

```
let x = M in N
```

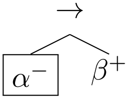
Relaxing the value restriction: variance

Variance describes the input/output behaviour of a type parameter:

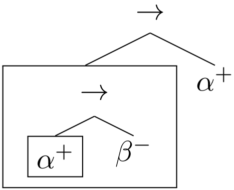
> Parameters for output types are positive / covariant
> Parameters for input types are negative / contravariant

The positions of the parameter in the definition determine variance.

```
'a -> 'b
```



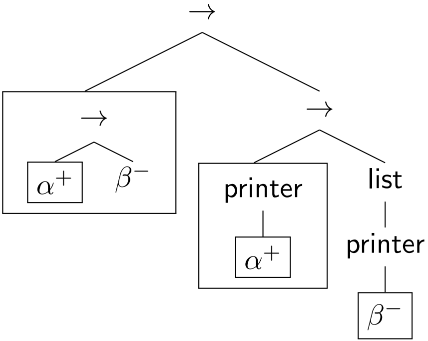

```
('a -> 'b) -> 'a
```



Relaxing the value restriction: variance

```
type 'a printer = 'a -> string

('a -> 'b) -> 'a printer -> 'b printer list
```



Relaxing the value restriction: the rules

It's safe to generalize if we are only reading polymorphic values.

- covariant type variables
- invariant type variables
- contravariant type variables
- bivariant type variables

Relaxing the value restriction: the rules

It's safe to generalize if we are only reading polymorphic values.

- covariant type variables ✓
- invariant type variables
- contravariant type variables
- bivariant type variables

Relaxing the value restriction: the rules

It's safe to generalize if we are only reading polymorphic values.

- covariant type variables ✓
- invariant type variables ✗
- contravariant type variables
- bivariant type variables

Relaxing the value restriction: the rules

It's safe to generalize if we are only reading polymorphic values.

- covariant type variables ✓
- invariant type variables ✗
- contravariant type variables ✗
- bivariant type variables

Relaxing the value restriction: the rules

It's safe to generalize if we are only reading polymorphic values.

- covariant type variables ✓
- invariant type variables ✗
- contravariant type variables ✗
- bivariant type variables ✓

This time

$$\Gamma \vdash A$$

A suggestive notation

$$A \to B$$
$$\forall \alpha . A$$
$$\exists \alpha . A$$

A suggestive notation

$$A \to B$$
$$\forall \alpha . A$$
$$\exists \alpha . A$$
$$A \times B$$
$$A + B$$

A suggestive notation

$$A \supset B$$
$$\forall\alpha.A$$
$$\exists\alpha.A$$
$$A \wedge B$$
$$A \vee B$$

A suggestive notation

$$A \supset B$$
$$\forall\alpha.A$$
$$\exists\alpha.A$$
$$A \wedge B$$
$$A \vee B$$

Types *correspond to* **propositions**

What logic?

$\lambda^{\rightarrow}$

$$\mathcal{B} \qquad A \supset B \qquad A \wedge B \qquad A \vee B$$

What logic?

$\lambda^{\rightarrow}$ corresponds to **propositional logic**

$$\mathcal{B} \qquad A \supset B \qquad A \wedge B \qquad A \vee B$$

What logic?

$\lambda^{\rightarrow}$ corresponds to **propositional logic**

$$\mathcal{B} \qquad A \supset B \qquad A \wedge B \qquad A \vee B$$

System F

$$\forall \alpha . A \qquad \exists \alpha . A$$

What logic?

$\lambda^{\rightarrow}$ corresponds to **propositional logic**

$$\mathcal{B} \qquad A \supset B \qquad A \wedge B \qquad A \vee B$$

System F corresponds to **second-order propositional logic**

$$\forall\alpha.A \qquad \exists\alpha.A$$

What logic?

$\lambda^{\rightarrow}$ corresponds to **propositional logic**

$$\mathcal{B} \qquad A \supset B \qquad A \wedge B \qquad A \vee B$$

System F corresponds to **second-order propositional logic**

$$\forall \alpha . A \qquad \exists \alpha . A$$

System Fω

$$\lambda \alpha . A \qquad A\ B$$

What logic?

$\lambda^{\rightarrow}$ corresponds to **propositional logic**

$$\mathcal{B} \qquad A \supset B \qquad A \wedge B \qquad A \vee B$$

System F corresponds to **second-order propositional logic**

$$\forall \alpha . A \qquad \exists \alpha . A$$

System Fω corresponds to **higher-order propositional logic**

$$\lambda \alpha . A \qquad A\ B$$

What logic?

$\lambda^{\rightarrow}$ corresponds to **propositional logic**

$$\mathcal{B} \qquad A \supset B \qquad A \wedge B \qquad A \vee B$$

System F corresponds to **second-order propositional logic**

$$\forall\alpha.A \qquad \exists\alpha.A$$

System Fω corresponds to **higher-order propositional logic**

$$\lambda\alpha.A \qquad A\ B$$

What about first-order logic?

Propositional vs predicate

Propositional logic

$$P \to Q$$

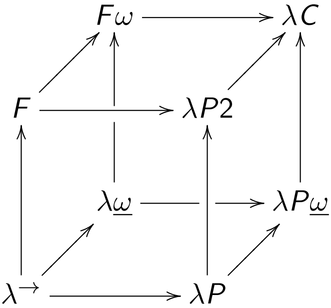
$$(\forall P.P \to P) \to (\exists Q.Q \to Q)$$

Predicate logic (FOPL)

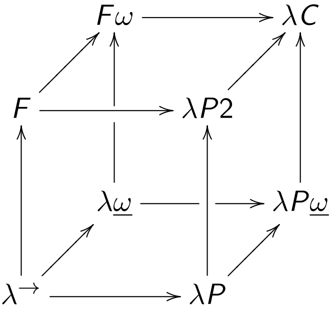
$$P(x)$$

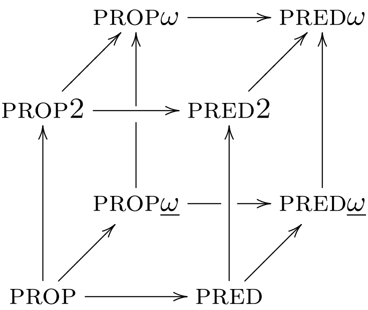
$$\forall x \in A.P(x)$$

Lambda and logic cubes



Lambda and logic cubes





More suggestive notation

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}$$

More suggestive notation

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B} \qquad\qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

More suggestive notation

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B} \qquad\qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

Terms *correspond to* **proofs**

Inference rules for $\to$

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A}\ \text{tvar} \qquad\qquad \frac{A \in \Gamma}{\Gamma \vdash A}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : A \to B}\ \to\text{-intro} \qquad\qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$$

$$\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}\ \to\text{-elim} \qquad\qquad \frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

Inference rules for $\times$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash \langle M, N \rangle : A \times B}\ \times\text{-intro} \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-intro}$$

$$\frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \text{fst } M : A}\ \times\text{-elim-1} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-elim-1}$$

$$\frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \text{snd } M : B}\ \times\text{-elim-2} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-elim-2}$$

Classical vs intuitionistic logic

Classical logic

Emphasis on **truth**

Truth values: $\top$, $\bot$

$A \vee \neg A$ always holds

Intuitionistic logic

Emphasis on **proof**

Proofs inhabit propositions

$A \vee \neg A$ doesn't hold in general

Brouwer-Heyting-Kolmogorov (BHK) interpretation

A proof of $A \to B$:

a function that builds a proof of B from a proof of A.

A proof of $A \wedge B$:

a pair of a proof of A and a proof of B.

$\neg A$

means $A \to \bot$

$\bot$

has no proof

Continuing the correspondence

Types *correspond to* **propositions**

Programs *correspond to* **proofs**

Continuing the correspondence

Types *correspond to* **propositions**

Programs *correspond to* **proofs**

Evaluation *corresponds to* **proof simplification**

Who should care?

Language designers

e.g. *linear logic*: restrictions on structural rules
corresponds to a language with resource management guarantees

Logicians

since results about programming languages transfer "for free"
e.g. strong normalization implies consistency

Authors (and users) of proof assistants

e.g. Coq and other tools based on type theory

Programmers?

Logical equivalences

$$\forall\beta.(\forall\alpha.(P\alpha \to \beta)) \to \beta \qquad \leftrightarrow \qquad \exists\alpha.P\alpha$$

$$\forall\beta.(P \to \beta) \wedge (Q \to \beta) \to \beta \qquad \leftrightarrow \qquad P \vee Q$$

Proof: we must show

$$\forall\beta.(\forall\alpha.(P\alpha \to \beta)) \to \beta \vdash \exists\alpha.P\alpha$$

$$\exists\alpha.P\alpha \vdash \forall\beta.(\forall\alpha.(P\alpha \to \beta)) \to \beta$$

etc.

A proof

Let $\Gamma = \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta$. Then

$$
\frac{
\dfrac{\dfrac{}{\Gamma \vdash \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta}}{\Gamma \vdash (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\;\forall\text{-elim}
\qquad
\dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, P\alpha \vdash \exists\alpha.P\alpha}\;\exists\text{-intro}}{\Gamma, \alpha \vdash P\alpha \to \exists\alpha.P\alpha}\;\to\text{-intro}}{\Gamma \vdash \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\;\forall\text{-intro}
}{\Gamma \vdash \exists\alpha.P\alpha}\;\to\text{-elim}
$$

A program from a proof

Let $\Gamma = \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta$. Then

$$
\dfrac{
\dfrac{\overline{\Gamma \vdash \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta}}{\Gamma \vdash (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\ \forall\text{-elim}
\qquad
\dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, P\alpha \vdash \exists\alpha.P\alpha}\ \exists\text{-intro}}{\Gamma, \alpha \vdash P\alpha \to \exists\alpha.P\alpha}\ \to\text{-intro}}{\Gamma \vdash \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\ \forall\text{-intro}
}{\Gamma \vdash \exists\alpha.P\alpha}\ \to\text{-elim}
$$

Right subtree:

$$
\dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, P\alpha \vdash \exists\alpha.P\alpha}\ \exists\text{-intro}}{\Gamma, \alpha \vdash P\alpha \to \exists\alpha.P\alpha}\ \to\text{-intro}}{\Gamma \vdash \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\ \forall\text{-intro}
$$

A program from a proof

Let $\Gamma = \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta$. Then

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta}}{\Gamma \vdash (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\ \forall\text{-elim} \qquad \dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, P\alpha \vdash \exists\alpha.P\alpha}\ \exists\text{-intro}}{\Gamma, \alpha \vdash P\alpha \to \exists\alpha.P\alpha}\ \to\text{-intro}}{\Gamma \vdash \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\ \forall\text{-intro}}{\Gamma \vdash \exists\alpha.P\alpha}\ \to\text{-elim}$$

Right subtree:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, v : P\alpha \vdash \text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \exists\alpha.P\alpha}\ \exists\text{-intro}}{\Gamma, \alpha \vdash P\alpha \to \exists\alpha.P\alpha}\ \to\text{-intro}}{\Gamma \vdash \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\ \forall\text{-intro}$$

A program from a proof

Let $\Gamma = \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta$. Then

$$\dfrac{\dfrac{}{\Gamma \vdash \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta}}{\Gamma \vdash (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\ \forall\text{-elim} \qquad \dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, P\alpha \vdash \exists\alpha.P\alpha}\ \exists\text{-intro}}{\Gamma, \alpha \vdash P\alpha \to \exists\alpha.P\alpha}\ \to\text{-intro}}{\Gamma \vdash \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\ \forall\text{-intro}$$
$$\overline{\qquad\Gamma \vdash \exists\alpha.P\alpha\qquad}\ \to\text{-elim}$$

Right subtree:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, v : P\alpha \vdash \text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \exists\alpha.P\alpha}\ \exists\text{-intro}}{\Gamma, \alpha \vdash \lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : P\alpha \to \exists\alpha.P\alpha}\ \to\text{-intro}}{\Gamma \vdash \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\ \forall\text{-intro}$$

A program from a proof

Let $\Gamma = \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta$. Then

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta}}{\Gamma \vdash (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\ \forall\text{-elim} \qquad \dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, P\alpha \vdash \exists\alpha.P\alpha}\ \exists\text{-intro}}{\Gamma, \alpha \vdash P\alpha \to \exists\alpha.P\alpha}\ \to\text{-intro}}{\Gamma \vdash \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\ \forall\text{-intro}}{\Gamma \vdash \exists\alpha.P\alpha}\ \to\text{-elim}$$

Right subtree:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, v : P\alpha \vdash \text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \exists\alpha.P\alpha}\ \exists\text{-intro}}{\Gamma, \alpha \vdash \lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : P\alpha \to \exists\alpha.P\alpha}\ \to\text{-intro}}{\Gamma \vdash \Lambda\alpha.\lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\ \forall\text{-intro}$$

A program from a proof

Let $\Gamma = \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta$. Then

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta}}{\Gamma \vdash (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\ \forall\text{-elim} \qquad \dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, P\alpha \vdash \exists\alpha.P\alpha}\ \exists\text{-intro}}{\Gamma, \alpha \vdash P\alpha \to \exists\alpha.P\alpha}\ \to\text{-intro}}{\Gamma \vdash \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\ \forall\text{-intro}}{\Gamma \vdash \exists\alpha.P\alpha}\ \to\text{-elim}$$

Right subtree:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, v : P\alpha \vdash \text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \exists\alpha.P\alpha}\ \exists\text{-intro}}{\Gamma, \alpha \vdash \lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : P\alpha \to \exists\alpha.P\alpha}\ \to\text{-intro}}{\Gamma \vdash \Lambda\alpha.\lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\ \forall\text{-intro}$$

Left subtree:

$$\dfrac{\dfrac{}{\Gamma \vdash \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta}}{\Gamma \vdash (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\ \forall\text{-elim}$$

A program from a proof

Let $\Gamma = H : \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta$. Then

$$
\dfrac{
\dfrac{\dfrac{}{\Gamma \vdash \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta}}{\Gamma \vdash (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\;\forall\text{-elim}
\qquad
\dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, P\alpha \vdash \exists\alpha.P\alpha}\;\exists\text{-intro}}{\Gamma, \alpha \vdash P\alpha \to \exists\alpha.P\alpha}\;\to\text{-intro}}{\Gamma \vdash \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\;\forall\text{-intro}
}{\Gamma \vdash \exists\alpha.P\alpha}\;\to\text{-elim}
$$

Right subtree:

$$
\dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, v : P\alpha \vdash \text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \exists\alpha.P\alpha}\;\exists\text{-intro}}{\Gamma, \alpha \vdash \lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : P\alpha \to \exists\alpha.P\alpha}\;\to\text{-intro}}{\Gamma \vdash \Lambda\alpha.\lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\;\forall\text{-intro}
$$

Left subtree:

$$
\dfrac{\dfrac{}{\Gamma \vdash H : \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta}}{\Gamma \vdash (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\;\forall\text{-elim}
$$

A program from a proof

Let $\Gamma = H : \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta$. Then

$$\dfrac{\dfrac{}{\Gamma \vdash \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta}}{\Gamma \vdash (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\ \forall\text{-elim} \quad \dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, P\alpha \vdash \exists\alpha.P\alpha}\ \exists\text{-intro}}{\Gamma, \alpha \vdash P\alpha \to \exists\alpha.P\alpha}\ \to\text{-intro}}{\Gamma \vdash \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\ \forall\text{-intro}$$
$$\overline{\Gamma \vdash \exists\alpha.P\alpha}\ \to\text{-elim}$$

Right subtree:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, v : P\alpha \vdash \text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \exists\alpha.P\alpha}\ \exists\text{-intro}}{\Gamma, \alpha \vdash \lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : P\alpha \to \exists\alpha.P\alpha}\ \to\text{-intro}}{\Gamma \vdash \Lambda\alpha.\lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\ \forall\text{-intro}$$

Left subtree:

$$\dfrac{\dfrac{}{\Gamma \vdash H : \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta}}{\Gamma \vdash H\ [\exists\alpha.V\alpha] : (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\ \forall\text{-elim}$$

A program from a proof

Let $\Gamma = H : \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta$. Then

$$\dfrac{\dfrac{}{\Gamma \vdash \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta}}{\Gamma \vdash (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\ \forall\text{-elim} \quad \dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, P\alpha \vdash \exists\alpha.P\alpha}\ \exists\text{-intro}}{\Gamma, \alpha \vdash P\alpha \to \exists\alpha.P\alpha}\ \to\text{-intro}}{\Gamma \vdash \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\ \forall\text{-intro}$$
$$\overline{\qquad\qquad\Gamma \vdash \exists\alpha.P\alpha\qquad\qquad}\ \to\text{-elim}$$

Right subtree:

$$\dfrac{\ldots}{\Gamma \vdash \Lambda\alpha.\lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\ \forall\text{-intro}$$

Left subtree:

$$\dfrac{\ldots}{\Gamma \vdash H\ [\exists\alpha.V\alpha] : (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\ \forall\text{-elim}$$

Finally:

$$\dfrac{\begin{array}{c}\Gamma \vdash H\ [\exists\alpha.V\alpha] : (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha \\ \Gamma \vdash \Lambda\alpha.\lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \forall\alpha.P\alpha \to \exists\alpha.P\alpha\end{array}}{\Gamma \vdash \exists\alpha.P\alpha}\ \to\text{-elim}$$

A program from a proof

Let $\Gamma = H : \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta$. Then

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \forall\beta.(\forall\alpha.P\alpha \to \beta) \to \beta}}{\Gamma \vdash (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\;\forall\text{-elim} \qquad \dfrac{\dfrac{\dfrac{}{\Gamma, \alpha, P\alpha \vdash \exists\alpha.P\alpha}\;\exists\text{-intro}}{\Gamma, \alpha \vdash P\alpha \to \exists\alpha.P\alpha}\;\to\text{-intro}}{\Gamma \vdash \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\;\forall\text{-intro}}{\Gamma \vdash \exists\alpha.P\alpha}\;\to\text{-elim}$$

Right subtree:

$$\dfrac{\dots}{\Gamma \vdash \Lambda\alpha.\lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \forall\alpha.P\alpha \to \exists\alpha.P\alpha}\;\forall\text{-intro}$$

Left subtree:

$$\dfrac{\dots}{\Gamma \vdash H\,[\exists\alpha.V\alpha] : (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha}\;\forall\text{-elim}$$

Finally:

$$\dfrac{\begin{array}{c}\Gamma \vdash H\,[\exists\alpha.V\alpha] : (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha \\ \Gamma \vdash \Lambda\alpha.\lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \forall\alpha.P\alpha \to \exists\alpha.P\alpha\end{array}}{\Gamma \vdash H\,[\exists\alpha.V\alpha]\,(\Lambda\alpha.\lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha) : \exists\alpha.P\alpha}\;\to\text{-elim}$$

Is it useful?

$$\forall \beta.(P \to \beta) \wedge (Q \to \beta) \to \beta \qquad \leftrightarrow \qquad P \vee Q$$

These type equivalences can be useful in constructing programs.

The data type encodings we saw last week can be derived this way.

In the exercises: two ways of building an HTML renderer.

We'll revisit when we look at domain-specific languages.

Closing thoughts

The correspondence suggests a way of thinking about programming
— and a way of systematically constructing (some) programs

However, propositional logic is quite weak
(and our types are often uninformative)

We'll have richer types available later (GADTs, monads),
at which point we'll revisit the question of usefulness

Next time

Abstraction