This time

$$\Gamma \vdash A$$

A suggestive notation

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

Curry-Howard

A suggestive notation

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

Curry-Howard

A suggestive notation

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

Curry-Howard

A suggestive notation

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

Types *correspond to* ***propositions***

Curry-Howard

A suggestive notation

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

Types *correspond to* ***propositions***

(Part 1 of the ***Curry-Howard*** correspondence)

$\lambda^{\rightarrow}$

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

What about **first-order logic**?

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

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

What about **first-order logic**?

What logic?

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

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

System F

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

What about **first-order logic**?

What logic?

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

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

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

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

What about **first-order logic**?

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

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

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

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

System Fω

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

What about **first-order logic**?

What logic?

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

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

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**?

What logic?

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

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

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 logic

$$P \to Q$$

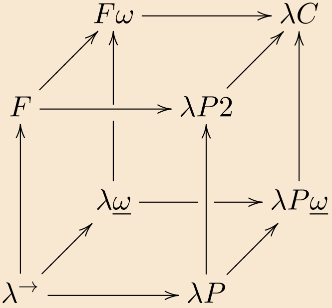
$$(\forall P. P \to P) \to (\exists Q. Q \to Q)$$

Predicate logic (FOPL)

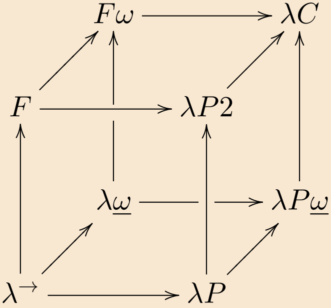
$$P(x)$$

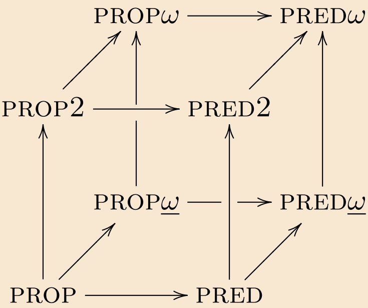
$$\forall x \in A. P(x)$$

Lambda and logic cubes



Lambda and logic cubes





More suggestive notation

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

Terms *correspond to* ***proofs***

Curry-Howard

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***

Curry-Howard

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

Curry-Howard

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***

(Part 2 of the ***Curry-Howard*** correspondence)

Inference rules for $\rightarrow$

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

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda \mathsf{x}{:}A.M : A \rightarrow B}\ \rightarrow\textsf{-intro} \qquad\qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B}$$

$$\frac{\Gamma \vdash M : A \rightarrow B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}\ \rightarrow\textsf{-elim} \qquad\qquad \frac{\Gamma \vdash A \rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

Inference rules for $\times$

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

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

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

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***

Curry-Howard

Continuing the correspondence

Types *correspond to* ***propositions***

Programs *correspond to* ***proofs***

Evaluation *corresponds to* ***proof simplification***

Curry-Howard

Continuing the correspondence

Types *correspond to* ***propositions***

Programs *correspond to* ***proofs***

Evaluation *corresponds to* ***proof simplification***

(The three-part ***Curry-Howard*** correspondence)

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

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

A program from a proof

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

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

Right subtree:

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

A program from a proof

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

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

Right subtree:

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

A program from a proof

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

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

Right subtree:

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

A program from a proof

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

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

Right subtree:

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

A program from a proof

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

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

Right subtree:

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

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

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

Right subtree:

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

Left subtree:

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

A program from a proof

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

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

Right subtree:

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

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

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

$$\dfrac{\qquad\qquad\qquad\qquad\qquad\qquad}{\Gamma \vdash \exists\alpha.P\alpha}\ \to\text{-elim}$$

Right subtree:

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

Left subtree:

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

A program from a proof

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

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

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:

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

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

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

Right subtree:

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

Left subtree:

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

Finally:

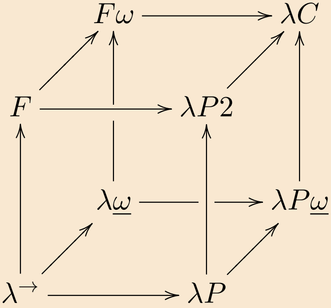
$$\dfrac{\begin{array}{c}\Gamma \vdash H\ [\exists\alpha.V\alpha] : (\forall\alpha.P\alpha \to \exists\alpha.P\alpha) \to \exists\alpha.P\alpha \\ \Gamma \vdash \Lambda\alpha.\lambda v : P\alpha.\text{pack } \alpha, v \text{ as } \exists\alpha.P\alpha : \forall\alpha.P\alpha \to \exists\alpha.P\alpha\end{array}}{\Gamma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

Is it useful?

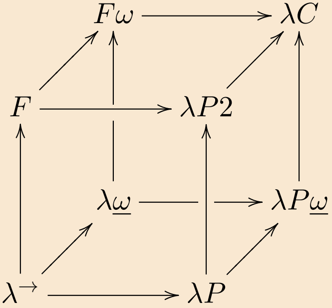
$$\forall \beta.(P \to \beta) \land (Q \to \beta) \to \beta \qquad \leftrightarrow \qquad P \lor Q$$

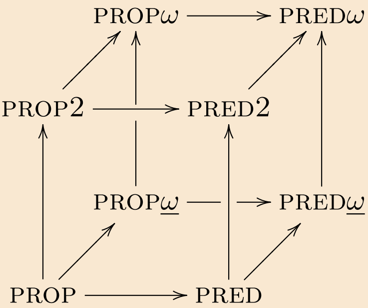
These type equivalences can be useful in constructing programs.

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


$F\omega$
λC
F
$\lambda P2$
$\lambda\underline{\omega}$
$\lambda P\underline{\omega}$
$\lambda^{\rightarrow}$
λP

Lambda and logic cubes





Typing in λC

$$\frac{\Gamma, x : M, \Delta \vdash *}{\Gamma, x : M, \Delta \vdash x : M} \text{ tvar}$$

$$\frac{\Gamma, x : M \vdash N : P}{\Gamma \vdash \lambda x : M.N : \Pi x : M.P} \; \Pi\text{-intro}$$

$$\frac{\begin{array}{c}\Gamma \vdash M : \Pi x : P.Q \\ \Gamma \vdash N : P\end{array}}{\Gamma \vdash M\ N : Q[x := N]} \; \Pi\text{-elim}$$

Typing in λC

$$\frac{\Gamma, x : M, \Delta \vdash *}{\Gamma, x : M, \Delta \vdash x : M}\ \text{tvar}$$

$$\frac{\Gamma, x : M \vdash N : P}{\Gamma \vdash \lambda x : M.N : \Pi x : M.P}\ \Pi\text{-intro}$$

bound variables appear in types

$$\frac{\Gamma \vdash M : \Pi x : P.Q \qquad \Gamma \vdash N : P}{\Gamma \vdash M\ N : Q[x := N]}\ \Pi\text{-elim}$$

arguments substituted into types

Inference rules in λC

$$\frac{\Gamma, x : M, \Delta \vdash *}{\Gamma, x : M, \Delta \vdash x : M}\ \text{tvar}$$

$$\frac{x : M \in \Gamma}{\Gamma \vdash x : M}$$

$$\frac{\Gamma, x : M \vdash N : P}{\Gamma \vdash \lambda x : M.N : \Pi x : M.P}\ \Pi\text{-intro}$$

$$\frac{\Gamma, x : M \vdash P}{\Gamma \vdash \forall x \in M.P(x)}$$

$$\frac{\Gamma \vdash M : \Pi x : P.Q \qquad \Gamma \vdash N : P}{\Gamma \vdash M\ N : Q[x := N]}\ \Pi\text{-elim}$$

$$\frac{\Gamma \vdash \forall x \in P.Q(x) \qquad \Gamma \vdash N : P}{\Gamma \vdash Q(N)}$$

Proposition as Type / Specification

$$\forall x \in N\ .\exists y \in N\ .(x =_N y * 2) \vee (x =_N y * 2 + 1)$$

Proof (on Paper)

$$\forall x \in N\ .\exists y \in N\ .(x =_N y * 2) \vee (x =_N y * 2 + 1)$$

By structural induction on x.

- Case 0: By def. $0 =_N 0 * 2$. By $\vee$-intro and $\exists$-intro.
- Case $x + 1$: We prove

  $$\exists y \in N\ .(x + 1 =_N y * 2) \vee (x + 1 =_N y * 2 + 1)$$

  from the assumption

  $$\exists y \in N\ .(x =_N y * 2) \vee (x =_N y * 2 + 1)$$

  By $\exists$-elim., then case analysis then substitution or elementary analysis, and as before $\vee$-intro and $\exists$-intro.

  - If $x =_N y * 2$, then $x + 1 =_N y * 2 + 1$.
  - If $x =_N y * 2 + 1$, then $x + 1 =_N (y + 1) * 2$.

Proof as Program (...)

$$
\begin{array}{l}
\lambda x\,.\,\mathsf{natrec}(x\,,\\
\quad \langle 0\,,\ \ \mathbf{inl}(\mathsf{id}(0))\rangle\,,\\
\quad (x\,,\ z_1)\,\mathsf{split}(z_1\,,\\
\qquad (y\,,\ z_2)\,\mathsf{when}(z_2\,,\\
\qquad\quad (z_3)\langle y\,,\ \ \mathbf{inr}(\mathsf{subst}(z_3\,,\ \ \mathsf{id}(x+1)))\rangle\,,\\
\qquad\quad (z_4)\langle y+1\,,\ \ \mathbf{inl}(c(x,y,z_4))\rangle)))
\end{array}
$$

Proof as Program (in Agda)

```
data Parity : Nat -> Set where
  even : (k : Nat) -> Parity (k * two)
  odd : (k : Nat) -> Parity (one + k * two)

parity : (n : Nat) -> Parity n
parity zero = even zero
parity (suc n) with parity n
parity (suc .(k * two)) | even k = odd k
parity (suc .(one + k * two)) | odd k = even (suc k)

half : Nat -> Nat
half n with parity n
half .(k * two) | even k = k
half .(one + k * two) | odd k = k
```

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)

With dependent types, we get predicate logic
(and our types can be fine-grained specifications)

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