Dependent type theory and Curry Howard

Watt Seng Joe   Abdul Haliq S/O Abdul Latiff

April 5, 2021

What more can we do with types?

Formalize mathematics, ie encode theorems and proofs in the computer.

Why?

Use computers to help us prove theorems and verify the correctness of our programs.
This forms the foundation for interactive and automated theorem proving.

How?

Curry Howard says we can encode intuitionistic logic in typed lambda calculi

* Propositional logic $\iff$ Simple types
* Predicate logic $\iff$ Dependent types

Intuitionistic logic

What is it

A logic of positive evidence. To say something is true means to exhibit evidence in the form of a proof.

What we give up

* No excluded middle, ie cannot say $\varphi \vee \neg\varphi$ in general, so no double negation elimination $\neg\neg\varphi \to \varphi$
* No Axiom of Choice

What we gain

Proofs with computational interpretation, ie functions transforming evidence of assumptions to that of conclusion.

Natural deduction – Propositional logic

Conjunction

$$\frac{\Gamma \vdash \varphi \qquad \Gamma \vdash \psi}{\Gamma \vdash \varphi \land \psi} \land\text{-intro}$$

$$\frac{\Gamma \vdash \varphi \land \psi}{\Gamma \vdash \varphi} \land\text{-elim left}$$

$$\frac{\Gamma \vdash \varphi \land \psi}{\Gamma \vdash \psi} \land\text{-elim right}$$

Implication

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi} \to\text{-intro}$$

$$\frac{\Gamma \vdash \varphi \to \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi} \to\text{-elim}$$

Natural deduction – Propositional logic

Disjunction

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi} \ \vee\text{-intro left}$$

$$\frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi} \ \vee\text{-intro right}$$

$$\frac{\Gamma \vdash \varphi \vee \psi \qquad \Gamma \vdash \varphi \to \eta \qquad \Gamma \vdash \psi \to \eta}{\Gamma \vdash \eta} \ \vee\text{-elim}$$

Curry Howard, aka Propositions as Types

The big idea

$$\overbrace{\Gamma}^{\text{Typing env}} \vdash \overbrace{1+2}^{\text{Term}} : \overbrace{\text{Int}}^{\text{Type}}$$

$$\underbrace{\Gamma}_{\text{Assumptions}} \vdash \underbrace{p}_{\text{Proof}} : \underbrace{P}_{\text{Prop}}$$

* Proving a proposition $\iff$ constructing a term of the corresponding type
* Checking a proof for correctness $\iff$ type checking
* Simply typed lambda calculus $\iff$ Intuitionistic propositional logic
* Dependent types $\iff$ First/higher order intuitionistic logic

Typing judgments

Function type
($\cong$ Implication)

* Intro

$$\frac{\Gamma, x : \sigma \vdash e : \tau}{\Gamma \vdash (\lambda\, x,\ e) : \sigma \to \tau}$$

* Elim

$$\frac{\Gamma \vdash e_1 : \sigma \to \tau \qquad \Gamma \vdash e_2 : \sigma}{\Gamma \vdash e_1\, e_2 : \tau}$$

Product/Pair type
($\cong$ conjunction)

* Intro

$$\frac{\Gamma \vdash a : \sigma \qquad \Gamma \vdash b : \tau}{\Gamma \vdash (a,\ b) : \sigma \land \tau}$$

* Elim left

$$\frac{\Gamma \vdash (a,\ b) : \sigma \land \tau}{\Gamma \vdash a : \sigma}$$

* Elim right

$$\frac{\Gamma \vdash (a,\ b) : \sigma \land \tau}{\Gamma \vdash b : \tau}$$

Typing judgments

Sum/tagged union type ($\cong$ disjunction)

* Intro left

$$\frac{\Gamma \vdash a : \sigma}{\Gamma \vdash \text{inl } a : \sigma \vee \tau}$$

* Intro right

$$\frac{\Gamma \vdash b : \tau}{\Gamma \vdash \text{inr } b : \sigma \vee \tau}$$

* Elim

$$\frac{\Gamma \vdash s : \sigma \vee \tau \qquad \Gamma, a : \sigma \vdash c : \eta \qquad \Gamma, b : \tau \vdash d : \eta}{\Gamma \vdash (\text{match } s \text{ with } \mid \text{inl } a \Rightarrow \text{c} \mid \text{inr } b \Rightarrow \text{d end}) : \eta}$$

Key point

The pattern for elim must be complete, ie you must consider both possible cases.

Towards dependent type theory

Limitation of simple type theory

Cannot express quantified formulae like $\forall x\, \varphi(x)$ since types cannot contain variables or other expressions.

Dependent types to the rescue

* Dependent types offer an elegant solution.
* Key idea is to allow types to *depend on values*.

Intuitionistic natural deduction – Quantifiers

Universal quantifier

$$\frac{\Gamma \vdash \varphi(x)}{\Gamma \vdash \forall x\, \varphi(x)}\ \forall\text{-intro}$$

provided x does not occur free in any hypothesis on which φ depends.

$$\frac{\Gamma \vdash \forall x\, \varphi(x)}{\Gamma \vdash \varphi(t)}\ \forall\text{-elim}$$

Existential quantifier

$$\frac{\Gamma \vdash \varphi(t)}{\Gamma \vdash \exists x\, \varphi(x)}\ \exists\text{-intro}$$

$$\frac{\Gamma \vdash \exists x\, \varphi(x) \qquad \Gamma,\ \varphi(x) \vdash \psi}{\psi}\ \exists\text{-elim}$$

provided x is not free in ψ and Γ.

Dependent Pi type (WIP)

Pi type
(generalizes simple function type)

* Intro

$$\frac{\Gamma, x : \sigma \vdash e : \tau}{\Gamma \vdash (\lambda\, x,\ e) : \sigma \to \tau}$$

* Elim

$$\frac{\Gamma \vdash e_1 : \sigma \to \tau \qquad \Gamma \vdash e_2 : \sigma}{\Gamma \vdash e_1\, e_2 : \tau}$$

Product/Pair type
($\cong$ conjunction)

* Intro

$$\frac{\Gamma \vdash a : \sigma \qquad \Gamma \vdash b : \tau}{\Gamma \vdash (a,\ b) : \sigma \wedge \tau}$$

* Elim left

$$\frac{\Gamma \vdash (a,\ b) : \sigma \wedge \tau}{\Gamma \vdash a : \sigma}$$

* Elim right

$$\frac{\Gamma \vdash (a,\ b) : \sigma \wedge \tau}{\Gamma \vdash b : \tau}$$