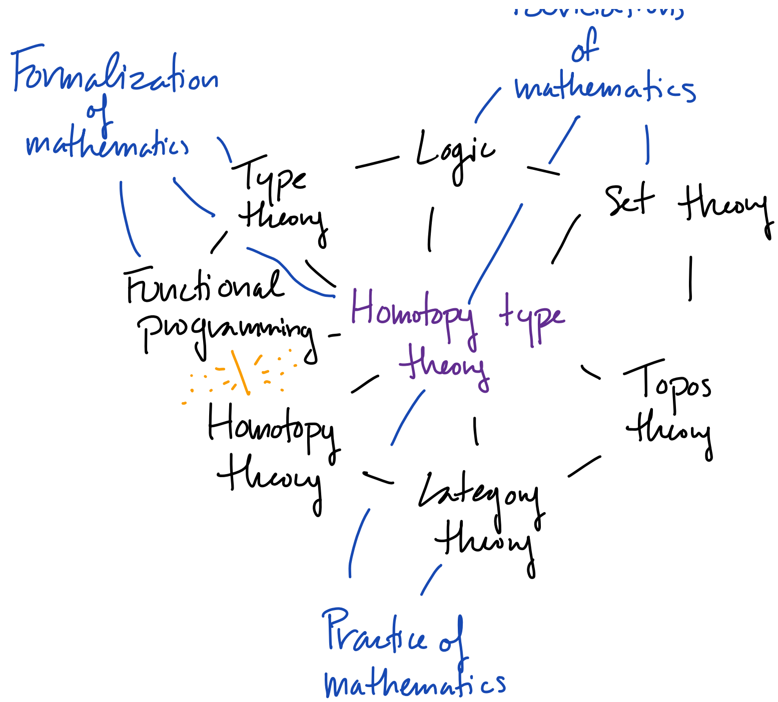
Map :



Logic (natural deduction)

(see Logic and Proof by Avigad et al.)

In natural deduction, we prove statements about propositions using proof trees built out of rules.

Ex. We can prove that Q follows from $P \wedge (P \to Q)$ for any propositions P, Q.

$$\dfrac{\wedge\text{-elim-l}\ \dfrac{P \wedge (P \to Q)}{P} \qquad \dfrac{P \wedge (P \to Q)}{P \to Q}\ \wedge\text{-elim-r}}{Q}\ \to\text{-elim}$$

Rules for $\land$:

$\land$-intro: $$\frac{P \quad Q}{P \land Q}$$

$\land$-elim-ℓ: $$\frac{P \land Q}{P}$$ $\land$-elim-r: $$\frac{P \land Q}{Q}$$

Rules for $\to$:

$\to$-intro: $$\frac{\begin{array}{c}\overline{P}^{\,1} \\ \vdots \\ Q\end{array}}{P \to Q}\,1$$

$\to$-elim: $$\frac{P \quad P \to Q}{Q}$$

<u>Introduction rules</u> tell you how to prove something.
<u>Elimination rules</u> tell you how to use something.

<u>Ex</u>. $(P \land Q) \to P$

$$\dfrac{\dfrac{\overline{P \land Q}^{\,1}}{P}\ \land\text{-elim-}\ell}{(P \land Q) \to P}\ 1\ \to\text{-intro}$$

Simply-typed λ calculus

(*Proofs and types* by Girard)

If we make natural deduction proof relevant, we get the STΛC.

In ND, we write "P" to mean "P holds".

In STΛC, we write "p:P" to mean "p is a proof/witness of P" or "p holds/is inhabited by P".

We call p a proof/witness/term/element of P.

We call P a proposition or type.

Ex. Q follows from $P \wedge (P \to Q)$

$$\dfrac{\dfrac{a : P \wedge (P \to Q)}{pr_1 a : P}\ \wedge\text{-elim-}\ell \qquad \dfrac{a : P \wedge (P \to Q)}{pr_2 a : P \to Q}\ \wedge\text{-elim-}r}{(pr_1 a)(pr_2 a) : Q}\ \to\text{-elim}$$

Rules for ∧:

$\wedge$-form: $\dfrac{P \text{ type} \quad Q \text{ type}}{P \wedge Q \text{ type}}$

$\wedge$-intro: $\dfrac{\Gamma \vdash p : P \quad \Gamma \vdash q : Q}{\Gamma \vdash (p,q) : P \wedge Q}$

$\wedge$-elim-ℓ: $\dfrac{\Gamma \vdash a : P \wedge Q}{\Gamma \vdash pr_1 a : P}$ $\qquad$ $\wedge$-elim-r: $\dfrac{\Gamma \vdash a : P \wedge Q}{\Gamma \vdash pr_2 a : Q}$

$\wedge$-comp-β-ℓ: $\dfrac{\Gamma \vdash p : P \quad \Gamma \vdash q : Q}{\Gamma \vdash pr_1 (p,q) \doteq p : P}$ $\qquad$ $\wedge$-comp-β-r: $\dfrac{\Gamma \vdash p : P \quad \Gamma \vdash q : Q}{\Gamma \vdash pr_2 (p,q) \doteq q : Q}$

$\wedge$-comp-η : $\dfrac{\Gamma \vdash a : P \wedge Q}{\Gamma \vdash (pr_1 a, pr_2 a) \doteq a : P \wedge Q}$

Notice: If we think of types as sets and terms as elements, then $P \wedge Q$ behaves like the product $P \times Q$ of sets.

Thm (Lambek 1985). There is an interpretation of the STΛC into Set, the category of sets. (Actually there is an equivalence between STΛC and CCC.)

Rules for $\to$:

$\to$-form : $\dfrac{\Gamma \vdash P \text{ type} \qquad \Gamma \vdash Q \text{ type}}{P \to Q \text{ type}}$

$\to$-intro : $\dfrac{\begin{array}{c}\overline{P}^{\,1} \\ \vdots \\ Q\end{array}}{P \to Q}\,1$ $\Bigg\}$ a proof of Q from P $\rightsquigarrow$ $\dfrac{\Gamma, x : P \vdash q : Q}{\Gamma \vdash \lambda x . q : P \to Q}$

$\to$-elim : $\dfrac{\Gamma \vdash p : P \qquad f : P \to Q}{\Gamma \vdash fp : Q}$

...

Ex. How do we prove $P \wedge Q \to P$?

$$\dfrac{\dfrac{\dfrac{}{a : P\wedge Q \vdash a : P\wedge Q}}{a : P\wedge Q \vdash \mathrm{pr}_1 a : P}\ \wedge\text{-elim-}\ell}{\lambda a.\mathrm{pr}_1 a : (P\wedge Q)\to P}\ \to\text{-intro}$$

generic element / projection / variable (see Rijke)

→ add contexts everywhere

So an expression like $a : P\wedge Q \vdash \mathrm{pr}_1 a : P$ corresponds to a proof tree with a hypothesis that $P\wedge Q$ holds and which concludes that P holds. The term $\mathrm{pr}_1 a$ records the shape of the proof tree.

Ex. $x : P\wedge Q \vdash (\mathrm{pr}_2 x, \mathrm{pr}_1 x) : Q \wedge P$

corresponds to

$$\dfrac{\wedge\text{-elim-r}\ \dfrac{P\wedge Q}{Q} \qquad \dfrac{P\wedge Q}{P}\ \wedge\text{-elim-}\ell}{Q\wedge P}\ \wedge\text{-intro}$$

Thm (Howard 1969) (Falls under the umbrella of the 'Curry-Howard correspondence')
The proof trees of natural deduction are in 1-to-1 correspondence with terms of STΛC.

Computation rules for $\to$.

$\to$-comp-β: $$\frac{\Gamma \vdash x : P \vdash q : Q \qquad \Gamma \vdash p : P}{\Gamma \vdash (\lambda x.\, q)\, p \doteq q[p/x] : Q}$$ substitution

$\to$-comp-η: $$\frac{\Gamma \vdash f : P \to Q}{\Gamma \vdash \lambda x.\, f x \doteq f : P \to Q}$$

Under the Howard (logical) interpretation, $\to$ corresponds to implication.
Under the Lambek (set) interpretation, $\to$ corresponds to functions.

We can also interpret types as program specifications
- ex. A type $P \to P$ specifies a program that takes a term of type P as an input and returns a term of type P as an output.

and terms as programs meeting the specification
- ex. We can construct the identity $id_P : P \to P$.

Dependent type theory

- In natural deduction, we have no terms.
- In STλC, terms can depend on terms.
  - ex. $a : P \wedge Q \vdash pr_1\, a : P$
- In dependent type theory, not only terms but types can depend on terms.

- ex. $n:\mathbb{N} \vdash \mathrm{Vect}(n)$ type

  $n:\mathbb{N} \vdash \mathrm{isEven}(n)$ type

If we interpret types as

- propositions: dependent types are predicates
- sets: dependent types are indexed families of sets
- programs: dependent types are program specifications with a parameter

We have the same rules as before, except the formation rules can also have a context.

$$\wedge\text{-form}: \frac{\Gamma \vdash P \text{ type} \quad \Gamma \vdash Q \text{ type}}{\Gamma \vdash P \wedge Q \text{ type}} \qquad \to\text{-form}: \frac{\Gamma \vdash P \text{ type} \quad \Gamma \vdash Q \text{ type}}{\Gamma \vdash P \to Q \text{ type}}$$

Ex.

$$\frac{n:\mathbb{N} \vdash \mathrm{isEven}(n) \qquad n:\mathbb{N} \vdash \mathrm{isDivFour}(n)}{n:\mathbb{N} \vdash \mathrm{isEven}(n) \wedge \mathrm{isDivFour}(n)}$$

$$n:\mathbb{N} \vdash \mathrm{isDivFour}(n) \to \mathrm{isEven}(n)$$

Dependent functions

Ex. (informal) Let Vect be the set of all vectors (of any length) (i.e., finite lists) in $\mathbb{N}$.

Define $0: \mathbb{N} \to \mathrm{Vect}$ which takes n to the vector of length n whose components are all 0.

<u>But</u> $0(n)$ actually lives in $\mathrm{Vect}(n)$, the set of vectors of length n.
We can encode this by considering 0 as a <u>dependent function</u>

$$0 : \prod_{n:\mathbb{N}} \mathrm{Vect}(n) \quad \text{(sometimes written } 0 : (n:\mathbb{N}) \to \mathrm{Vect}(n)\text{)}$$

The elimination rule gives us $0(n) : \mathrm{Vect}(n)$ for any $n:\mathbb{N}$.

<u>Ex.</u> Consider

$$n:\mathbb{N} \vdash \mathrm{isDivFour}(n) \to \mathrm{isEven}(n).$$

To show this for all n, we construct a term

$$n:\mathbb{N} \vdash t(n) : \mathrm{isDivFour}(n) \to \mathrm{isEven}(n)$$

The introduction rule for dependent functions gives us

$$\vdash \lambda n.t(n) : \prod_{n:\mathbb{N}} \mathrm{isDivFour}(n) \to \mathrm{isEven}(n).$$

In the logical interpretation, we interpret Π as $\forall$.

<u>Rem.</u>
- $\to$ is a special case of Π.

  ex. $\prod_{n:\mathbb{N}} \mathrm{Vect}$ is the same as $\mathbb{N} \to \mathrm{Vect}$ (the rules become the same)

- $\wedge$ is a special case of Π, if we have $\mathbb{B}$ (the type with 2 elements)

  <u>ex.</u> $\mathbb{B} \to \mathrm{Vect}$ (i.e. $\prod_{b:\mathbb{B}} \mathrm{Vect}$) is the same as $\mathrm{Vect} \wedge \mathrm{Vect}$.

  $\prod_{b:\mathbb{B}} \mathrm{Vect}_b$ is the same as $\mathrm{Vect}_0 \wedge \mathrm{Vect}_1$.

Rules for Π-types.

Π-form:
$$\frac{\Gamma, x:P \vdash Q \text{ type}}{\Gamma \vdash \prod_{x:P} Q \text{ type}}$$

Π-intro:
$$\frac{\Gamma, x:P \vdash q : Q}{\Gamma \vdash \lambda x.q : \prod_{x:P} Q}$$

Π-elim:
$$\frac{\Gamma \vdash f : \prod_{x:P} Q \qquad \Gamma \vdash p : P}{\Gamma \vdash fp : Q[p/x]}$$

Π-comp-β:
$$\frac{\Gamma, x:P \vdash q : Q \qquad \Gamma \vdash p : P}{\Gamma \vdash (\lambda x.q)p \doteq q[p/x] : Q[p/x]}$$

Π-comp-η:
$$\frac{\Gamma \vdash f : \prod_{x:P} Q}{\Gamma \vdash \lambda x.fx \doteq f : \prod_{x:P} Q}$$