CATEGORICAL MODELS OF TYPE THEORY

Chapter 0: INTRODUCTION

§1 OVERVIEW & MOTIVATION

"I hope you leave here and walk out and say, 'What did he say?'"
— George W. Bush (2008)
43rd President of the USA

Some historical notes:

- Type theory came about ~1910 in Whitehead & Russell's Principia Mathematica to give a formalism for the foundations of mathematics which circumvents Russell's Paradox in Frege's earlier attempt by way of "ramification through type hierarchies".

↳ Missed the main-stream as set-theoretical approaches were developed.

It reemerged in terms of Church's λ-calculus ~1930, which is based on the primitive notion of functions (rather than sets).

↳ Motivated by computability theory, found application in Category Theory & Brouwer's intuitionism as observed by Lawvere later (specifically in topos theory).

- The further development of type theory - particularly the development of dependent type theories & Martin-Löf's identity-types ~1980's - led to connection to fibered category theory, and eventually quite spectacularly to connections to homotopy theory (Awodey, Warren, Voevodsky ~2006), and eventually to the semantics in higher category theory, the implementation of automated theorem checkers, and much more.

Plan for this course (subject to the mysteries of the future)

- Untyped λ-Calculi, their classic "environmental" semantics, & Dana Scott's categorical interpretations.
- Simply typed λ-Calculi & their equivalence to cartesian closed cat's.
- Dependently typed theories & their fibrational semantics.

↝ We start out with a minimal framework and add more complexity along the way. Benefits:

* Introduce crucial concepts in isolated environments for clarity.
* Motivate construction step by step.
* Each theory has its own interesting perks which are fun to study!

Indeed, starting out with untyped λ-Calculi highlights the functional character of the development of type theory.

Literature:

Part I:
- H. Barendregt: "The λ-Calculus: Its syntax & semantics"
- J.R. Hindley, J.P. Seldin: "Lambda-Calculus & Combinators: An Introduction".

Part II:
- J. Lambek, P.J. Scott: "Introduction to higher cat'l logic"
- H. Barendregt, W. Dekkers, R. Statman: "Lambda-Calculus w/ Types".

§2 SYNTACTIC CALCULI

Notation 1: A *language* (or an *alphabet*) $\mathcal{L}$ is an arbitrary set whose elements will be referred to as $\mathcal{L}$-symbols or names. A language $\mathcal{L}$ often comes together with a partition into different *sorts* and additional disjoint sets of basic symbols and variables.

Example 2: In 1st order logic, these sorts are the sets

$$S_n = \{n\text{-ary function symbols}\} \sqcup \{n\text{-ary Relation symbols}\}$$

for $n \in \mathbb{N}$. The basic symbols are among the set

$$\mathcal{B} = \{=, \rightarrow, \wedge, \vee, \neg, \forall, \exists, \bot, \leftrightarrow\},$$

the set of variables Var is an arbitrary countably infinite set which is disjoint from $\bigcup_{n \in \mathbb{N}} S_n$ and $\mathcal{B}$. Then $\mathcal{L}^{\text{FOL}} = \coprod_{n \in \mathbb{N}} S_n \sqcup \mathcal{B} \sqcup \text{Var}$.

Example 3: In type theory, these sorts are the sets

$$S_n = \{n\text{-ary term symbols}\} \sqcup \{n\text{-ary type symbols}\}$$

for $n \in \mathbb{N}$. The set of basic symbols varies, but at least contains the symbol "$\equiv$" referred to as *judgemental equality*.

The set of variables Var is again a countably infinite set of new symbols (depending on the presentation, sometimes parametrized over the sets of sorts as well). Then again $\mathcal{L}^{T.T.} = \coprod_{n\in\mathbb{N}} S_n \amalg B \amalg Var$.

Definition 4: Let $\mathcal{L}$ be a language and let

$\mathcal{L}^\infty := \coprod_{n\in\mathbb{N}} \mathcal{L}^n$ be the set of words in $\mathcal{L}$ of finite length.

Given $n\in\mathbb{N}$, a relation $R \subseteq (\mathcal{L}^\infty)^n$ is called a *rule* over $\mathcal{L}$.

A *calculus* $\mathcal{C}$ over $\mathcal{L}$ is a set of rules over $\mathcal{L}$.

Notation 5: A rule $R = \{(w_1^i, \dots, w_n^i) \mid i\in I\}$ over a language $\mathcal{L}$ is denoted by

$$(R)\ \frac{w_1^i \quad \dots \quad w_{n-1}^i}{w_n^i}\ ,\ i\in I$$

Definition 6: Let $\mathcal{C}$ be a calculus over a language $\mathcal{L}$. For a rule $R\in\mathcal{C}$, $R\subseteq(\mathcal{L}^\infty)^n$, and a subset $X\subseteq\mathcal{L}^\infty$, we define the *closure*

$$R[X] := \{w\in\mathcal{L}^\infty \mid \exists x_1,\dots,x_{n-1}\in X : (x_1,\dots,x_{n-1},w)\in R\}$$

of X wrt. R. The smallest subset of $\mathcal{L}^\infty$ which is closed under the rules of $\mathcal{C}$ is called the *product of $\mathcal{C}$* and denoted by

$$\mathrm{Prod}(\mathcal{C}) := \bigcap\{X\subseteq\mathcal{L}^\infty \mid \forall R\in\mathcal{C} : R[X]\subseteq X\},$$

A derivation in $\mathcal{C}$ is a finite sequence $(w_1, \dots, w_k) \in (\mathcal{L}^\infty)^k$ s.th. for all $1 \le i \le k$ there is a rule $R \in \mathcal{C}$, $R \subseteq (\mathcal{L}^\infty)^n$, together with integers $i_0, \dots, i_{n-1} < i$, s.th. $(w_{i_1}, \dots, w_{i_{n-1}}, w_i) \in R$.

Lemma 7: Let $\mathcal{C}$ be a calculus over a language $\mathcal{L}$. Then

$$\mathrm{Prod}(\mathcal{C}) = \{ w \in \mathcal{L}^\infty \mid \text{There is a derivation } (w_1, \dots, w_k) \text{ in } \mathcal{C} \text{ with } w_k = w \}$$

<u>Proof</u>: Exercise. $\square$

Example 8: The calculi of first order logic (over $\mathcal{L}^{FOL}$) are

* the term calculus T^{fol} over $\mathcal{L}^{FOL}$ (or $\coprod_{n \in \mathbb{N}} S_n \sqcup Var$ specifically)
* the formula calculus F^{fol} over $\mathrm{Prod}(T^{FOL}) \sqcup B \sqcup Var$.
* the sequent calculus $\vdash$ over $\mathrm{Prod}(F^{FOL})$.

Contain the "well-formed" terms & formulas, respectively.

↳ Three consecutively performed recursions.

We will see that non-dependent type theories are built similarly.

Dependent type theories are more complicated.

Remark 9: We implicitly work in a background theory, which for conventional reasons is naive set theory. In particular, the semantics we will consider relates type theoretical syntax to set-theoretical syntax. & Common move due to the strength of, familiarity with, and faith in ZFC. It is possible however to formalize type theory with respect to other background theories, such as itself. That particularly applies to categorical models.

CHAPTER I: THE UNTYPED λ-CALCULUS

§1 SYNTAX

Idea: Just like Set theory is a formalization of the notion of a set, the λ-calculus is a formalization of functions / processes in an isolated environment. That means every term is a synthetic function, and as such can be applied to any other function (including itself!).

- Intuition from computability theory: Programs are just data, and so can be applied to other programs as well (see e.g. Pascal, LISP).
- Just like set-theory bases all math'l constructions on the primitive notion of set, the λ-Calculus bases math'l construction on the primitive notion of function.

Surprising features: The λ-calculus

* is consistent
* has real mathematical models
* is expressive enough to formalize various standard math'l structures!

Notation 1: The language of the λ-calculus is given by

$$\mathcal{L}_\lambda^{\mathbb{C}} := \mathcal{L} :\equiv \{ \sqcup \langle \lambda ; App ; (;) ; . ; , \} \sqcup Var \sqcup \mathbb{C}$$

Definition 2: The λ-term calculus $T_\lambda^{\mathbb{C}}$ is given by the following rules.

1. (Var-Intro.) $\dfrac{}{x}$ for all $x \in Var$
2. (Cnst-Intro) $\dfrac{}{c}$ for all $c \in \mathbb{C}$
3. (λ-Abstraction) $\dfrac{t}{\lambda x . t}$ for all $x \in Var$
4. (λ-Application) $\dfrac{s \quad t}{App(s,t)}$

The λ-calculus over $\mathcal{L}_\lambda^{\emptyset}$ is called pure. Elements of the product $\Lambda^{\mathbb{C}} := Prod(T_\lambda^{\mathbb{C}})$ are called $\lambda^{\mathbb{C}}$-terms (or just λ-terms if $\mathbb{C}$ is implicit from context).

Examples 3:

1. $I = \lambda x . x$ is the synthetic "identity"-operator $(\overset{?}{=} \lambda y . y)$
2. $K = \lambda x . \lambda y . x$ " " " "projection"-operator. $\left[\begin{matrix} X \to X^X \\ x \mapsto cnst_x \end{matrix} \wedge \begin{matrix} X \times X \xrightarrow{\pi_1} X \\ (x,y) \mapsto x \end{matrix} \right]$
3. $S = \lambda x . \lambda y . \lambda z . App(App(x,z), App(y,z))$

   is the "parametrized composition"-operator. $\left[\begin{matrix} X^{X \times X} \times X^X \times X \to X \\ (f,g,z) \mapsto f(z,g(z)) = f_z \circ g(z) \end{matrix} \right]$

Observation 4: The λ-term calculus is "uniquely readable" (as are all calculi that we will consider in this course). That means, every λ-term $t \in \Lambda^{\mathbb{C}}$ arises by application of exactly one distinguished rule to exactly one set of λ-terms.

It follows that we can perform induction and recursion along the complexity of terms.

Definition 5: The set $FV(t)$ of free variables of a λ-term t is defined recursively as follows.

1. $FV(x) = \{x\}$ f.a. $x \in Var$.
2. $FV(c) = \emptyset$ f.a. $c \in \mathbb{C}$.
3. $FV(\lambda x.t) = FV(t) \setminus \{x\}$.
4. $FV(App(s,t)) = FV(s) \cup FV(t)$.

A $\lambda^{\mathbb{C}}$-term t is said to be closed (a "combinator") if $FV(t) = \emptyset$. The set of closed λ-terms is denoted by $\Lambda_0^{\mathbb{C}} \subset \Lambda^{\mathbb{C}}$.

Definition 6: The substitution $s[t/x]$ of a λ-term t in a λ-term s for a free variable $x \in FV(s)$ is defined recursively as follows.

1. $x[t/x] = t$.
2. $a[t/x] = a$ f.a. $c \in \mathbb{C} \cup Var \setminus \{x\}$.

3. $(\lambda x.s)[t/x] = \lambda x.s$.

4. $(\lambda y.s)[t/x] = \lambda y.\, s[t/x]$ if $y \in Var \setminus (\{x\} \cup FV(t))$ or $x \notin FV(s)$.

5. $(\lambda y.s)[t/x] = \lambda z.\, s[z/y][t/x]$ if $y \in (Var \setminus \{x\}) \cap FV(t)$ and $x \in FV(s)$

6. $App(s,u)[t/x] = App(s[t/x], u[t/x])$.

In 5., $z \in Var \setminus (FV(s) \cup FV(t))$ minimal.

For every $t \in \Lambda^c$, $x \in Var$ we hence obtain a substitution function $\_[t/x] : \Lambda^c \longrightarrow \Lambda^c$.

Definition 7: For a λ-term s to contain an occurrence of another λ-term t is again recursively defined as follows.

1. t contains an occurrence of t f.a. λ-terms t.
2. $\lambda x.s$ contains an occurrence of t if s contains an occurrence of t, or $t = x$.
3. $App(s,u)$ contains an occurrence of t if s or u contain an occurrence of t.

Definition 8: (α-Congruence: Change of bound variables)

If a λ-term t contains an occurrence of a λ-term $\lambda x.s$, and $y \in Var \setminus FV(s)$, then we can define a term t' containing an occurrence of

$\lambda y.s[y/x]$ by replacing the occurrences of $\lambda x.s$ in t by $\lambda y.s[y/x]$.

This is called a change of bound variables. Two λ-terms s,t are *α-congruent* if s can be obtained from finitely many changes of bound variables.

Exercise 9: * α-congruence defines an equivalence relation '$\equiv_\alpha$' on $\Lambda^{\mathbb{C}}$.

* Definitions I.1.4 and I.1.5 are well-defined on α-quotients:

$$FV : \Lambda^{\mathbb{C}}/_{\sim\alpha} \to Var \quad ,$$

$$\_[t/x] : \Lambda^{\mathbb{C}}/_{\sim\alpha} \to \Lambda^{\mathbb{C}}/_{\sim\alpha} .$$

Henceforth, we may work in the quotient $\Lambda^{\mathbb{C}}/_\alpha$, or impose it structurally as an axiom or a rule.

Example 10: * $\lambda x.x = \lambda y.y$ f.a. $x,y \in Var$.

* $\lambda x.y \neq \lambda x.x \neq \lambda y.x$ ".

Definition 11: The *$\lambda^{(\mathbb{C})}$-formula calculus* $\mathcal{F}_\lambda^{(\mathbb{C})}$ is given by the following rules.

$$\frac{}{s := t} \quad \text{for } \lambda\text{-terms } s,t.$$

Elements of the product $\Phi_\lambda^{\mathbb{C}} := \mathrm{Prod}(\mathcal{F}_\lambda^{\mathbb{C}})$ are called *λ-formulas*.

A *$\lambda^{\mathbb{C}}$-theory* (or again, just a *λ-theory*) is a set of λ-formulas.

Example 12:

1. $\alpha = \{ \lambda x.t :\equiv \lambda y.t[y/x] \mid x \in Var, t \in \Lambda^{\mathbb{C}}, y \in Var \setminus FV(t) \}$
2. $\beta = \{ App(\lambda x.t, s) :\equiv t[s/x] \mid x \in Var, s,t \in \Lambda^{\mathbb{C}} \}$
3. $\eta = \{ \lambda x. App(t, x) :\equiv t \mid t \in \Lambda^{\mathbb{C}}, x \in Var \setminus FV(t) \}$
4. $\lambda\beta = \alpha \cup \beta$
5. $\lambda\beta\eta = \lambda\beta \cup \eta$

These λ-theories define β-congruence (a computation rule), and η-congruence (a uniqueness rule) often referred to as function extensionality.

Definition 13: In the following, the meta-variables s, t, u, v range over λ-terms, x ranges over Var, and Γ is a finite set of λ-formulas.

The calculus of "structural rules" is given as follows.

1. (Monotonicity) $\dfrac{}{\Gamma, t :\equiv s \vdash t :\equiv s}$

2. (Weakening) $\dfrac{\Gamma \vdash t :\equiv s}{\Gamma, u :\equiv v \vdash t :\equiv s}$

3. (App-Congruence 1) $\dfrac{\Gamma \vdash s :\equiv t}{\Gamma \vdash App(u, s) :\equiv App(u, t)}$

4. (App-Cong. 2) $\dfrac{\Gamma \vdash s :\equiv t}{\Gamma \vdash App(s, u) :\equiv App(t, u)}$

5. (λ-Congruence (or ξ)) $\dfrac{\Gamma \vdash s :\equiv t}{\Gamma \vdash \lambda x.s :\equiv \lambda x.t}$

6. (Reflexivity) $\dfrac{}{\Gamma \vdash s :\equiv s}$

7. (Symmetry) $\dfrac{\Gamma \vdash s :\equiv t}{\Gamma \vdash t :\equiv s}$

8. (Transitivity) $\dfrac{\Gamma \vdash s :\equiv t \quad \Gamma \vdash t :\equiv u}{\Gamma \vdash s :\equiv u}$