Types and categories I

George McNinch

2022-09-28

Types and categories

- goal of these talks:

  use simple ("algebraic") type theory to motivate & explain idea behind something called *categorical semantics*

- we follow (Crole 1993)

Overview - "simple" or "algebraic" type theory

- A (simple, algebraic) *type theory* encompasses *types*, *terms* and *equalities*.
- for example, you might have a type   for *rational nubmers*, a type   for *real numbers*, a type   for *complex numbers*, a type *Fun_* for complex valued functions on , and a type *Fun_* for real valued functions on .
- terms have types: a: or z: or f:*Fun_*
- and you can formulate rules for equality e.g. of terms a: and b: .
- now, the basic idea behind the formal notion of a *theory* is this:
  - there are basic assumptions - "axioms"
  - together with rules for deducing "theorems" from the axioms

---

- we aim to give a "theory" in "algebraic type theory"
- begin with a given collection of types, terms, equalities
- we view the given equalities as the *axioms* of a theory
- the theorems of the theory are the equalities that can be deduced using the rules of our algebraic type theory.
- these rules amount to: manipulation of equations using the properties of reflexivity, transitivity, symmetry.

Algebraic Type Theories: syntax overview

- we introduce the notion of *algebraic signature* for algebraic type theory
- I believe the term is supposed to recollect the notion of *type signature* from programming languages. E.g.

  ```
  addPossibility :: forall a. List a -> Maybe a -> List a
  addPossibility current (Just x) = x:current
  addPossibility current Nothing = current
  ```

  (the type signature here is the "first line")
- of course, this example involves *type constructors* and *type variables*; the algebraic type theory we are going to describe is simpler than that.

Algebraic type theories: signature

- for us, a *signature* **Sg** will be the following data:
  - a collection of *types*
  - a collection of *function symbols*, each with a given *arity* a - thus a 0
  - a *sorting* for each function symbols – i.e. a list of $a + 1$ types $[\alpha_1, \cdots, \alpha_a, \alpha]$.

    We notate this sorting in symbols as follows:

    $$f : \alpha_1 \times \alpha_2 \times \cdots \times \alpha_a \to \alpha$$
  - if a function symbol k has arity 0, we write $k : \alpha$ and refer to k as a *constant* function symbol.
- we assume given a countable collection of *variables* `Var = {x,y,...}`

- the *raw terms* of the type theory are then defined by the following BNF grammar

  `M ::= x | k | f(M, ,M)`

  where `x` is a variable, `k` a constant, `f` a function symbol of arity `a` (and there are exactly `a` listed arguments of `f`)
- notice that the notion of *raw term* ignores the given sortings.
- *raw terms* = possibly not well-typed expressions.

Example

- let's consider the types mentioned earlier: , , , *Fun__* and *Fun__*
- consider an "inclusion function" i: $\to$ and an "inclusion function" j: $\to$ (so i and j both have arity 1 and their respective sortings are [,] and [,]
- a product function $p_{\mathbb{C}}$: $\times$ $\to$ with arity 2 and sorting [, ,].
- an evaluation function ev:*Fun__* $\times$ $\to$.
- if x, y, z are variables, an example of a (well-typed) raw term is

$$p_{\mathbb{C}}(j(i(x)), y)$$

- and another (not-well-typed) raw term is

$$ev(x, i(y))$$

Substitution of raw terms for free variables

- roughly, the free variables of a raw term M are the variables which "appear in" M
- more formally, the collection of free variables $\text{fv}(M)$ of a raw term M is defined by structural induction:
  - $\text{fv}(x) = x$ when x is a variable,
  - $\text{fv}(k) = \emptyset$ when k is a constant function symbol, and
  - $\text{fv}(f(M_1, M_2, \cdots, M_a)) = \text{fv}(M_1) \cup \text{fv}(M_2) \cup \cdots \cup \text{fv}(M_a)$ when f is a function symbol with positive arity a
- now define the *substitution* $M[x/N]$.
  - M is a raw term,
  - x is a variable, and
  - N is a second raw term.
  - $M[x/N]$ is to be "the substitution of N for x in M"

- we define the new raw term $M[x/N]$ by induction on the structure of M
  - $x[N/x] = N$,
  - $y[N/x] = y$ when y is a variable distinct from x,
  - $k[N/x] = k$ when k is a constant function symbol,
  - $f(M_1, M_2, \cdots, M_a)[N/x] = f(M_1[N/x], M_2[N/x], \cdots, M_a[N/x])$ when f is a function symbol of arity a

- Similarly, given $\vec{x} = [x_1, \dots, x_n]$ a finite list of n distinct variables and $\vec{N} = [N_1, \dots, N_n]$ a list of n raw terms, can define the *simultaneous substitution*

  $$M[\vec{N}/\vec{x}] \quad \text{or} \quad M[N_1/x_1, \cdots, N_n/x_n]$$

  in a similar manner.

- **danger:** in general,

  $$M[N/x, N'/y] \quad \text{is not syntactically identical to} \quad M[N/x][N'/y]$$

- **but:**

  $$M[N/x, N'/y] \quad \text{is identical to} \quad M[N[z/y]/x][N'/y][y/z]$$

  provided that $z \notin \mathrm{fv}(M) \cup \mathrm{fv}(N) \cup \mathrm{fv}(N')$.

type assignment: contexts

- our next goal is to define a *type assignment system* which generates *well-typed* raw terms.

  we first introduce some new notions:

- a *context* is a finite list of (variable,type) pairs

  $$\Gamma = [\quad x_1 : \alpha_1 \quad , \quad x_2 : \alpha_2 \quad , \quad \cdots \quad , \quad x_n : \alpha_n \quad]$$

  where the variables x_i are required to be *distinct*

- we denote *concatenation* of contexts using commas, e.g.

  $$\Gamma_1, x : \alpha, \Gamma_2$$

- a syntactic expression ("judgment") of the form

  $$\Gamma \vdash M : \alpha$$

  is called a *term-in-context*

  here Γ is a context, M a raw term, and α a type

- informally $\Gamma \vdash M : \alpha$ means that M is a program whose result has type α and that Γ is an environment for M

- We are going to define a class of judgments for our signature **Sg** which we call *proved terms*; they are indicated by the notation

$$\mathbf{Sg} \triangleright \Gamma \vdash M : \alpha$$

- there are three families of rules defining proved terms. The first two concern function symbols:
  - if $k : \alpha$ is a constant function symbol (i.e. the sorting of k is $[\alpha]$) then

    $$\mathbf{Sg} \rhd \Gamma \vdash k : \alpha$$

    is a proved term.

    We denote this symbolically as follows

    $$\frac{}{\mathbf{Sg} \rhd \Gamma \vdash k : \alpha} \quad (k : \alpha)$$

  - we just give the second rule symbolically

    $$\frac{\mathbf{Sg} \rhd \Gamma \vdash M_1 : \alpha_1 \quad \cdots \quad \mathbf{Sg} \rhd \Gamma \vdash M_n : \alpha_n}{\mathbf{Sg} \rhd \Gamma \vdash f(M_1, \cdots, M_n) : \alpha} \quad (f : \alpha_1, \cdots, \alpha_n \to \alpha)$$

  - and the third rule concerns variables:

    $$\frac{}{\mathbf{Sg} \rhd \Gamma, x : \alpha, \Gamma' \vdash x : \alpha}$$

- **Proposition:** Suppose that Γ is a context and M is a raw term. If both

  $$\mathbf{Sg} \rhd \Gamma \vdash M : \alpha \quad \text{and} \quad \mathbf{Sg} \rhd \Gamma \vdash M : \alpha'$$

  are proved terms, then the types of α and α' are identical.
- **Proof/sketch:** induction on the terms of the derivation of $\mathbf{Sg} \rhd \Gamma \vdash M : \alpha$
- in the setting of the Proposition, we informally refer to α as the *type* of the raw term M.

Equations in context and algebraic theories

- given two proved terms $\mathbf{Sg} \rhd \Gamma \vdash M : \alpha$ and $\mathbf{Sg} \rhd \Gamma \vdash M' : \alpha$

  (note that the context Γ and the type   are the same in these terms!)

  an *equation in context* has the form

  $$\Gamma \vdash M = M' : \alpha$$

- by an **algebraic theory** *Th* we mean a pair $(\mathbf{Sg}, \mathbf{Ax})$ where **Sg** is a *signature* as before, and **Ax** is a collection of equations in context formed using **Sg**.
- the equations **Ax** are the *axioms* of the theory.
- the *theorems* of the theory *Th* are the collection of statements ("judgments") generated by rules we record on the next slide.

Generating Theorems

- Axioms

$$\frac{\mathbf{Ax} \rhd \Gamma \vdash M = M' : \alpha}{\mathbf{Th} \rhd \Gamma \vdash M = M' : \alpha}$$

- Equational reasoning

$$\frac{\mathbf{Sg} \rhd \Gamma \vdash M : \alpha}{\mathbf{Th} \rhd \Gamma \vdash M = M : \alpha} \qquad \frac{\mathbf{Th} \rhd \Gamma \vdash M = M' : \alpha}{\mathbf{Th} \rhd \Gamma \vdash M' = M : \alpha}$$

$$\frac{\mathbf{Th} \rhd \Gamma \vdash M = M' : \alpha \quad \mathbf{Th} \rhd \Gamma \vdash M' = M'' : \alpha}{\mathbf{Th} \rhd \Gamma \vdash M = M'' : \alpha}$$

- Permutation

$$\frac{\mathbf{Th} \rhd \Gamma \vdash M = M' : \alpha}{\mathbf{Th} \rhd \pi\Gamma \vdash M = M' : \alpha} \quad \text{where } \quad \text{is a permutation}$$

Generating Theorems, contd

- Weakening

$$\frac{\mathbf{Th} \rhd \Gamma \vdash M = M' : \alpha}{\mathbf{Th} \rhd \Gamma' \vdash M = M' : \alpha} \quad \text{where } \Gamma \subseteq \Gamma'$$

- Substitution

$$\frac{\mathbf{Th} \rhd \Gamma, x : \alpha \vdash N = N' : \beta \quad \mathbf{Th} \rhd \Gamma \vdash M = M' : \alpha}{\mathbf{Th} \rhd \Gamma \vdash N[M/x] = N'[M'/x] : \beta}$$

Example

Let's describe the algebraic signature of the theory of semi-groups.

- recall that a *semigroup* is a set equipped with an associated binary operation.
- so, there is a type, S; terms represent "elements" of the semigroup
- there is a function symbol $\mu : S \times S \to S$
- the associativity axiom is given by the *equation-in-context*

  \$$x : S, y : S, z : S \vdash \mu(\mu(x, y), z) = \mu(x, \mu(y, z)) : S$

semantics, initially

- consider an algebraic theory $(\mathbf{Sg}, \mathbf{Ax})$ as before.
- a first version of "modeling" our theory is this:
- a proved term $(*)$ $\quad x_1 : \alpha_1, \cdots, x_n : \alpha_n \vdash M : \beta$ may be modelled by a function

  $$f : A_1 \times \cdots \times A_n \to B$$

  where we think of the n input variables of M as modelled by an element of the cartesian product $A_1 \times \cdots \times A_n$.
- the next step is rather than consider *sets*, consider instead a category $\mathcal{C}$ which has *finite products*
- now we can model $(*)$ using objects of the category: A_i for the type α_i and B for the type β. And we want to represent the proved term $(*)$ be a *morphism* in the category.

- crucially, our proved terms are built using substitution. e.g. $f(M)$ is precisely $f(x)[M/x]$.
- consider proved terms $(*)$ $\quad x : \alpha \vdash M : \beta$ and $(**)$ $\quad y : \beta \vdash N : \gamma$
- there is a "derived" proved term $x : \alpha \vdash N[M/y] : \gamma$. How should it be modelled?

  Should depend (only) on how $(*)$ and $(**)$ were modelled.
- we introduce some notation to summarize this:

$$\frac{[[x : \alpha \vdash M : \beta]] = m : A \to B \quad [[y : \beta \vdash N : \gamma]] = n : B \to C}{[[x : \alpha \vdash N[M/y] : \gamma]] = \Box_{m,n} : A \to C}$$

where $\Box_{m,n}$ is some relation ("morphism") depending on n and m.

- we must model $x : \alpha \vdash x : \alpha$ as some morphism $*_A : A \to A$.
- and one then argues that for morphisms $e : E \to A$ and $m : A \to N$ we have $\Box_{*_A,e} = e$ and $\Box(m, *_A) = m$.
- in a similar way, one finds the associativity of $\Box_{*,*}$.
- basic idea is to interpret substitution by composition

$$[[x : \gamma \vdash M[M'/x] : \beta]] = [[x : \alpha \vdash M : \beta]] \circ [[x : \gamma \vdash M' : \alpha]]$$

Categorical semantics, more precisely

- let $\mathcal{C}$ be a category with finite products and let **Sg** be an algebraic signature.
- a structure **M** in $\mathcal{C}$ for **Sg** is specified by giving
  - for each type α of **Sg** an object $[[\alpha]]$ of $\mathcal{C}$.
  - for each constant function symbol $k : \alpha$ a morphism $[[k]] : 1_{\mathcal{C}} \to [[\alpha]]$.
  - for each function symbol $f : \alpha_1, \cdots, \alpha_n \to \beta$ of **Sg** a morphism

$$[[f]] : [[\alpha_1]] \times \cdots \times [[\alpha_n]] \to [[\beta]].$$

- given a context $\Gamma = [x_1 : \alpha_1, \cdots, x_n : \alpha_n]$ we set

$$[[\Gamma]] = [[\alpha_1]] \times \cdots \times [[\alpha_n]].$$

- for every proved term $\Gamma : M : \alpha$ we specify a morphism

$$[[\Gamma \vdash M : \alpha]] : [[\Gamma]] \to [[\alpha]]$$

- we need certain properties to hold:
  -
$$\frac{}{[[\Gamma, x : \alpha, \Gamma' \vdash x : \alpha]] = \pi_2 : [[\Gamma]] \times [[\alpha]] \times [[\Gamma']] \to [[\alpha]]}$$
  -
$$\frac{}{[[\Gamma \vdash k : \alpha]] = [[k]]\circ ! : [[\Gamma]] \to 1_{\mathcal{C}} \to [[\alpha]]} \quad (k : \alpha)$$
  -
$$\frac{[[\Gamma \vdash M_1 : \alpha_1]] = m_1 : [[\Gamma]] \to [[\alpha_1]] \quad \cdots \quad [[\Gamma \vdash M_a : \alpha_a]] = m_a : [[\Gamma]] \to [[\alpha_a]]}{[[\Gamma \vdash f(M_1, \cdots, M_a)]] = [[f]] \circ \langle m_1, \cdots, m_a \rangle : [[\Gamma]] \to (\prod_1^a [[\alpha_i]]) \to [[\alpha]]}$$

Categorical models

- Let **M** be a structure for an algebraic signature **Sg** in a category $\mathcal{C}$ with finite products.
- we say that **M** *satisfies* an equation in context $\Gamma \vdash M = M' : \alpha$ provided that $[[\Gamma \vdash M : \alpha]]$ and $[[\Gamma : M' : \alpha]]$ are equal morphisms in $\mathcal{C}$.
- and we say that **M** is a *model* of the algebraic theory $\mathbf{Th} = (\mathbf{Sg}, \mathbf{Ax})$ if it satisfies all the equations in context found in **Ax**.

  Alternative lingo: the structure **M** satisfies the axioms of **Th**.

Soundness Theorem

Theorem: (*Soundness theorem*) Let **M** be a model of **Th** in $\mathcal{C}$. Then **M** satisfies any theorem of **Th**.

Bibliography

Crole, Roy L. 1993. *Categories for Types.* Cambridge Mathematical Textbooks. Cambridge University Press, Cambridge.