A computational interpretation of topos theory

Introduction

We give a computational interpretation of topos theory. A particular feature of our approach is that we use as a metalanguage an inconsistent theory, namely a type system with a type of all types. (One could relativize our interpretation to consistent systems if one is only interested in a predicative version of topos theory.) This interpretation can be seen as a variation on Gandy's interpretation of extensional simple type theory in intensional simple type theory.

We motivate informally our semantics. An object of the topos will be interpreted by a setoid, i.e. a type with a (type valued) equivalence relation, written $\rightsquigarrow$. A morphism $C \to A$ will be interpreted by an ordinary type theoretic function $a : C \to A$ together with a function of type

$$\prod_{\rho_0\ \rho_1 : C} \rho_0 \rightsquigarrow_C \rho_1 \quad \longrightarrow \quad a\rho_0 \rightsquigarrow_A a\rho_1$$

The object of truth values Ω is interpreted by the type of all types with logical equivalence as equivalence relation. If $\varphi : C \to \Omega$ the judgement $C \vdash u : \varphi$ will be interpreted by a section of type

$$\prod_{\rho : C} \varphi\rho$$

without requiring any condition on the equality on C (on the other hand, we should have a proof of

$$\prod_{\rho_0\ \rho_1 : C} \rho_0 \rightsquigarrow_C \rho_1 \quad \longrightarrow \quad \varphi\rho_0 \leftrightarrow \varphi\rho_1$$

since $\varphi : C \to \Omega$ has to send equal elements of C to equal, i.e. equivalent, elements of Ω).

In this simple way, we give a computational interpretation of the axioms of topos theory. Definitional equality of two objects, terms, proofs are interpreted by definitional equality of the corresponding interpretation (in type theory, where we have a notion of definitional equality). An important point is that in this interpretation, we justify the rule of weak conversion[1]

$$\mathsf{app}((\lambda b)\sigma, u) = b(\sigma, u)$$

but we don't justify substitution under abstraction

$$(\lambda b)\sigma = \lambda b(\sigma\mathsf{p}, \mathsf{q})$$

This is because our notion of morphism between setoid does not require definitional preservation of the reflexivity rule (contrary to what happens for morphisms of simplicial sets which have to commute with degeneracy maps).

[1]For this we have to take as definition of equality at function type $(a, a') \rightsquigarrow (b, b')$ to be

$$\prod_{\rho_0\ \rho_1 : C} \rho_0 \rightsquigarrow_C \rho_1 \quad \longrightarrow \quad a\rho_0 \rightsquigarrow_A b\rho_1$$

and *not* the following definition, like in [2]

$$\prod_{\rho : C} a\rho \rightsquigarrow_A b\rho$$

We have a natural interpretation of equality $\mathsf{eq} : A \times A \to \Omega$ by taking $\mathsf{eq}(\rho_0, \rho_1)$ to be the type $\rho_0 \rightsquigarrow_A \rho_1$.

The reason why we can interpret topos theory, despite having only weak conversion, is because our notion of equality of morphisms is strong: two maps $a_0\ a_1 : C \to A$ are equal iff we have a section p of type

$$\prod_{\rho : C} a_0\rho \rightsquigarrow a_1\rho$$

i.e. $C \vdash p : \mathsf{eq}(a_0, a_1)$. With this notion of equality all rules of topos theory (even with natural number objects) are satisfied.

It is interesting to compare our interpretation of topos theory with the one in Lambek and Scott [4] which is an interpretation in simple type theory. This interprets a function as a functional *relation*, why we interpret a function as a functional term of a dependent type theory. (For instance the successor function is interpreted here by itself, while it is interpreted by the functional relation $m = n + 1$ in [4].)

1 Syntax

Type formations

$$A, B \ ::= \ \mathsf{N} \mid \Omega \mid A \to B \mid A \times B \mid \{A|\varphi\}$$

Term formations

$$t, u \ ::= \ \mathsf{eq} \mid \mathsf{p} \mid \mathsf{q} \mid t, u \mid \lambda t \mid \mathsf{app}(t, u) \mid 1 \mid tu \mid \iota t \mid \mathsf{E}(t, t) \mid \mathsf{Ref} \mid \varphi$$

$$\varphi \ ::= \ \forall\varphi \mid \exists\varphi \mid \varphi \wedge \varphi \mid \varphi \Rightarrow \varphi$$

We have two kinds of judgement $t : A \to B$ and $A \vdash t : \varphi$ if $\varphi : A \to \Omega$

Typing rules

$$\frac{A \vdash}{1 : A \to A} \qquad \frac{t : A \to B \qquad u : B \to C}{ut : A \to C}$$

$$\frac{\varphi : A \to \Omega}{\{A \mid \varphi\} \vdash} \qquad \frac{\varphi : A \to \Omega}{\mathsf{p} : \{A \mid \varphi\} \to A} \qquad \frac{\varphi : A \to \Omega}{\{A \mid \varphi\} \vdash \mathsf{q} : \varphi\mathsf{p}}$$

$$\frac{t : C \times A \to B}{\lambda t : C \to (A \to B)} \qquad \frac{w : C \to A \times B \qquad u : C \to A}{\mathsf{app}(w, u) : C \to B}$$

$$\frac{t : C \to A \qquad C \vdash u : \psi t}{(t, u) : C \to \{A \mid \psi\}} \qquad \frac{t : C \to A \qquad u : C \to B}{(t, u) : C \to A \times B}$$

$$\frac{C \vdash t_0 : \varphi_0 \qquad C \vdash t_1 : \varphi_1}{C \vdash (t_0, t_1) : \varphi_0 \wedge \varphi_1} \qquad \frac{C \vdash t : \varphi_0 \wedge \varphi_1}{C \vdash \mathsf{p}t : \varphi_0} \qquad \frac{C \vdash t : \varphi_0 \wedge \varphi_1}{C \vdash \mathsf{q}t : \varphi_1}$$

$$\frac{\varphi : C \times A \to \Omega}{\forall\varphi : C \to \Omega} \qquad \frac{\varphi : C \times A \to \Omega}{\exists\varphi : C \to \Omega}$$

$$\frac{C \times A \vdash b : \varphi}{C \vdash \lambda b : \forall\varphi} \qquad \frac{C \vdash w : \forall\varphi \qquad a : C \to A}{C \vdash \mathsf{app}(w, a) : \varphi[a]}$$

where $[a] = (1, a) : C \to C \times A$ if $a : C \to A$

$$\frac{\{C \mid \varphi\} \vdash b : \psi\mathsf{p}}{C \vdash \lambda b : \varphi \Rightarrow \psi} \qquad \frac{C \vdash w : \varphi \Rightarrow \psi \qquad C \vdash u : \varphi}{C \vdash \mathsf{app}(w, u) : \psi}$$

$$\frac{a : C \to A \qquad C \vdash u : \varphi[a]}{C \vdash (a, u) : \exists\varphi} \qquad \frac{C \times A \vdash w : \varphi \Rightarrow \psi\mathsf{p} \qquad C \vdash v : \exists\varphi}{C \vdash \mathsf{E}(v, w) : \psi}$$

The rules for equality are

$$\mathsf{eq} : A \times A \to \Omega \qquad \frac{a : C \to A}{C \vdash \mathsf{Ref}\ a : \mathsf{eq}(a, a)}$$

$$\frac{a_0\ a_1 : C \to A \qquad C \vdash u : \mathsf{eq}(a_0, a_1) \qquad C \vdash v : \varphi[a_0]}{C \vdash \mathsf{J}(u, v) : \varphi[a_1]}$$

We have extensionality and a weak form of univalence

$$\frac{C \vdash u : \forall \mathsf{eq}(b_0, b_1)}{C \vdash \mathsf{ext}(u) : \mathsf{eq}(\lambda b_0, \lambda b_1)}$$

$$\frac{C \vdash u : (\varphi_0 \Rightarrow \varphi_1) \wedge (\varphi_1 \Rightarrow \varphi_0)}{C \vdash \mathsf{univ}(u) : \mathsf{eq}(\varphi_0, \varphi_1)}$$

Equations (definitional equality)

$$t1 = t = 1t \qquad \mathsf{p}(t, u) = t \qquad \mathsf{q}(t, u) = u$$

$$(t_0, t_1)u = (t_0 u, t_1 u) \qquad \mathsf{app}(t_0, t_1)u = \mathsf{app}(t_0 u, t_1 u) \qquad \mathsf{E}(t_0, t_1)u = \mathsf{E}(t_0 u, t_1 u)$$

$$\mathsf{app}((\lambda b)u, v) = b(u, v) \qquad \mathsf{E}((a, u), w) = \mathsf{app}(w[a], u)$$

These equations will be justified by the semantics in the next section. Notice that this semantics *does not* justify the rule

$$(\lambda b)\sigma = \lambda b(\sigma \mathsf{p}, \mathsf{q})$$

2 Semantics

The semantics is really a translation in a type system with a type of all types extended with strong sums. For the translation we call *object* the types of the interpreted theory, while we call types the type of the interpreting theory.

An object is interpreted by a setoid i.e. a type A with a relation $(\rightsquigarrow) : A \to A \to \mathsf{Type}$. A term $b : A \to B$ is interpreted by a pair of type theoretic functions $b\rho : B$ for $\rho : A$ with a function $b\alpha : b\rho_0 \rightsquigarrow b\rho_1$ if $\alpha : \rho_0 \rightsquigarrow \rho_1$. The object Ω is interpreted by Type with the relation $A \rightsquigarrow B$ defined as $A \leftrightarrow B = (A \to B) \times (B \to A)$. A term $C \vdash a\varphi$ is then interpreted by a section $a\rho : \varphi\rho$ if $\rho : C$. The setoid $A \to B$ is the setoid of pairs (f, f') where $f : A \to B$ and $f'\alpha : f\rho_0 \rightsquigarrow f\rho_1$ if $\alpha : \rho_0 \rightsquigarrow \rho_1$. The relation $(f, f') \rightsquigarrow (g, g')$ is defined to be

$$\prod_{u_0\ \ u_1 : A} u_0 \rightsquigarrow_A u_1 \quad \longrightarrow \quad f\ u_0 \rightsquigarrow_B g\ u_1$$

This translation can be seen as a computational justification of the rules. For instance we have

$$\mathsf{app}(w, u)\rho = w\rho\ (u\rho)$$

and

$$\mathsf{app}(w, u)\alpha = w\alpha\ (u\rho_0)\ (u\rho_1)\ (u\alpha)$$

if $\alpha : \rho_0 \rightsquigarrow \rho_1$.

It can then be used to check conversion during the type checking: for checking that we have

$$t = u : C \to A$$

we check that we have

$$\rho : C \vdash t\rho = u\rho : A$$

and

$$\rho_0\ \rho_1 : C, \alpha : \rho_0 \rightsquigarrow_C \rho_1 \vdash t\alpha = u\alpha : t\rho_0 \rightsquigarrow_A t\rho_1$$

The last rules concern the description axiom. It gives a computational interpretation of Church's description symbol ι. We have two maps $x_0, x_1 : C \times A \times A \to C \times A$ and we define

$$\exists_{\leqslant 1}\varphi = \forall\forall \varphi x_0 \wedge \varphi x_1 \Rightarrow \mathsf{eq}(x_0, x_1)$$

and $\exists!\varphi = \exists\varphi \wedge \exists_{\leqslant 1}\varphi$. We have thus

$$C \vdash (a, u, v) : \exists!\varphi$$

provided $a : C \to A$ and $C \vdash u : \varphi[a]$ and $C \vdash v : \exists_{\leqslant 1}\varphi$. Given $\psi : A \to \Omega$ the description rule is

$$\frac{C \vdash w : \exists!\psi \mathsf{p}}{\iota(w) : C \to \{A \mid \psi\}}$$

and the interpretation is

$$\iota(w)\rho = (u, u')$$

where $w\rho = (u, u', p)$ with $u : A$ and $u' : \psi u$. Given $\alpha : \rho_0 \rightsquigarrow_C \rho_1$ we compute $w\rho_0 = (u_0, u_0', p_0)$ and $w\rho_1 = (u_1, u_1', p_1)$ where $u_i : A,\ u_i' : \psi u_i$ and

$$p_i : \prod_{x\ y:A} \psi x \times \psi y \to x \rightsquigarrow_A y$$

We can then take

$$\iota(w)\alpha = p_1\ u_0\ u_1\ (u_0', u_1') : \quad u_0 \rightsquigarrow u_1$$

so that we have

$$\iota(w)\alpha : \iota(w)\rho_0 \rightsquigarrow \iota(w)\rho_1$$

as required.

In this type system, it should be possible to define the quotient type using equivalence classes, and for instance, to define $\mathbb{Z}$ as a quotient of $\mathbb{N} \times \mathbb{N}$. Voevodsky had such a development using resizing rules and equivalence and it would be interesting to see if we get a natural computational content of this development in this way.

References

[1] J. Cartmell. Generalised algebraic theories and contextual categories. Ann. Pure Appl. Logic 32 (1986), no. 3, 209–243.

[2] R. Gandy. On The Axiom of Extensionality -Part I. The Journal of Symbolic Logic, Vol. 21, 1956.

[3] M. Hofmann. Extensional concepts in intensional type theory. Ph.D. thesis, Edinburgh, 1994.

[4] J. Lambek and P.J. Scott. *Introduction to higher order categorical logic.* Cambridge studies in advanced mathematics 7, 1986.

[5] P. Martin-Löf. An intuitionistic theory of types: predicative part. Logic Colloquium, 1973.

[6] P. Martin-Löf. About models for intuitionistic type theories and the notion of definitional equality Proceedings of the Third Scandinavian Symposium, North-Holland, 1975.

[7] A. Prouté. On the role of description. JPAA 158 (2001) 295-307.

[8] B. Russell. The Theory of Implications. American Journal of Mathematics, 1906.

[9] V. Voevodsky. Univalent foundations project. NSF grant application, 2010.