Some notes for a talk on extensional interpretation of type theory

1 Motivation of the Takeuti-Gandy interpretation

1.1 Identity type

We have different versions of type theory

1972 version [4], with no Identity type

1973 version [6], introduction of Identity type (as we use it now)

Type theory without function extensionality does not seem appropriate to represent the notion of homotopy level. For representing intuitions from homotopy theory, and for having a convenient representation of mathematical notions (e.g. category theory) we need at least function extensionality and sometimes the univalence axiom.

It is remarkable that the explanation of CZF in Type Theory, interpreting a *set* as a well-founded tree up to bissimulation, does not use the Identity type, so it is an interpretation of CZF in the 1972 version of type theory.

We are looking similarly for a translation which *explains* the axiom of function extensionality and the axiom of univalence. The model will be a generalization of Takeuti-Gandy's interpretation [1] of extensional simple type theory in intensional simple type theory. So we are looking for a translation of type theory with the extensionality axiom in the 1972 of type theory.

1.2 Type theory as a formal system

All rules of type theory are justified following the pattern

1. The *introduction rules* give the meaning to the logical connectives (they are represented by *constructors*, following the terminology of functional programming)

2. The *elimination rules* are justified w.r.t. the introduction rules (they are represented by *defined functions*)

3. These justifications take the form of *computation rules* (the function is defined by case analysis)

A *proof* t of a type/proposition A is supposed to be a method to produce a canonical proof of A. This method is quite uniform in type theory: given a term t of type A, we unfold the definitions until we reach a canonical proof. In functional programming terminology, such a proof is represented as a term starting with a constructor, and the method of computation is head reduction.

An important point is that the computation rules can all be seen as *unfolding definitions.* If we have a type N of natural numbers, an empty type N_0 we can define $\neg : U \rightarrow U$ by $\neg A = A \rightarrow N_0$. This *definition* of $\neg$ can be seen as a computation rule (unfolding of definitions).

The situation is similar if we define a function $f : \Pi x : N.C(x)$ by the equations

$$f\ 0 = a : C(0) \qquad f\ (n+1) = g\ n\ (f\ n) : C(n+1)$$

These equations *define* a function f.

A related point is that the typing/provability relation $t : A$ is decidable. To decide this relation is reduced to the problem of comparing two given terms of the same type and this can be done by unfolding definitions (which can be interpreted as "computing" the meaning of the two terms), and comparing the result. For instance, if we define

$$F\ 0 = A, \quad F\ (n+1) = \neg\ A$$

then $F\ 2 = (A \to N_0) \to N_0 : U$ since $F\ 2$ is by definition $\neg\ (F\ 1)$ which is by definition $(F\ 1) \to N_0$ and $F\ 1$ is by definition $\neg\ (F\ 0)$ which is $F\ 0 \to N_0$ and $F\ 0$ is A.

One can argue that the conversion rule $(\lambda t)\sigma = \lambda(t\mathsf{p}, \mathsf{q})$ is not compatible with this idea of unfolding definition [5, 4]. On the other hand, the rules in Figure 1 can be seen as a formal description of basic rules of definitional equality.

Type theory

The rules of the version of Type Theory we interpret are presented in Figure 1. This is a variation of the version of type theory presented in the references [6, 5]. Besides the usual judgement $\Gamma \vdash$, $\Gamma \vdash A$, $\Gamma \vdash a : A$ it is convenient to have the judgement $\Gamma \vdash F : (A)\mathsf{Type}$ for families of types over a given type.

So far, we have formalized only the first level of the semisimplicial set model, giving a computational interpretation of invariance under isomorphisms. We have a formal proof that if A and B are isomorphic types and $T(X)$ is a definable type forming operations then $T(A)$ and $T(B)$ are also isomorphic. We can transport for instance any monoid structure on A to a monoid structure on B, and transport any proof of a property of this monoid to the one of B.

However the definitions are uniform and should extend at all levels, and if we have arbitrary levels, give a formal version of univalence. We have proved formally that the collection of all Kan semisimplicial set truncated at level 1 form a Kan semisimplical set of level 2.

For interpreting definitional equalities, it seems crucial to interpret the equality of two Kan semisimplical sets as a *relation* between these two sets satisfying some properties. Our model is a submodel of the model of logical relation. For function space, the definition of equality used in the setoid and groupoid model (two functions are equal if they are equal on each argument) does not preserve definitional equality while the relational definition (two functions are equal if they send equal elements to equal values) preserves definitional equality.

The rules for equality that we validate in our formalized model are

$$\frac{\Gamma \vdash A \quad \Gamma \vdash a : A \quad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{Eq}_A\ a\ u} \qquad \frac{\Gamma \vdash A \quad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{ref}\ a : \mathsf{Eq}_A\ a\ a}$$

$$\frac{\Gamma \vdash e : \mathsf{Eq}_A\ a\ u \quad \Gamma \vdash F : (A)\mathsf{Type} \quad \Gamma \vdash p : \mathsf{app}(F, a)}{\Gamma \vdash \mathsf{J}\ e\ p : \mathsf{app}(F, u)}$$

and

$$\frac{\Gamma \vdash F : (A)\mathsf{Type} \quad \Gamma \vdash a : A \quad \Gamma \vdash p : \mathsf{app}(F, a)}{\Gamma \vdash \mathsf{j} : \mathsf{Eq}_{\mathsf{app}(F,a)}\ (\mathsf{J}\ (\mathsf{ref}\ a)\ p)\ p}$$

These rules express the rules of identity type (where the computation rule is expressed as propositional equality). We have also the extensionality rule (formulated in a name-free way

$$\frac{\Gamma \vdash f : \mathsf{Fun}\ A\ F \quad \Gamma \vdash g : \mathsf{Fun}\ A\ F \quad \Gamma \vdash p : \mathsf{Fun}\ A\ (\lambda \mathsf{Id}\ \mathsf{app}(f\mathsf{p}, \mathsf{q})\ \mathsf{app}(g\mathsf{p}, \mathsf{q}))}{\Gamma \vdash \mathsf{ext}\ p : \mathsf{Eq}_{\mathsf{Fun}\ A\ F}\ f\ g}$$

The substitution rules are then

$$(\mathsf{Id}\ A\ a\ u)\sigma = \mathsf{Id}\ A\sigma\ a\sigma\ u\sigma \qquad (\mathsf{ext}\ u)\sigma = \mathsf{ext}\ u\sigma \qquad (\mathsf{J}\ e)\sigma = \mathsf{J}\ e\sigma$$

2 Applications of the model

1. We can add a small type of propositions where equality is defined to be logical equivalence. (This is a weak form of univalence.)

2. It is then possible to justify in the model the description operator for unique existence

$$\frac{\Gamma\vdash}{1:\Gamma\to\Gamma}\qquad\frac{\sigma:\Delta\to\Gamma\quad\delta:\Theta\to\Delta}{\sigma\delta:\Theta\to\Gamma}$$

$$\frac{\Gamma\vdash A\quad\sigma:\Delta\to\Gamma}{\Delta\vdash A\sigma}\qquad\frac{\Gamma\vdash t:A\quad\sigma:\Delta\to\Gamma}{\Delta\vdash t\sigma:A\sigma}\qquad\frac{\Gamma\vdash F:(A)\mathsf{Type}\quad\sigma:\Delta\to\Gamma}{\Delta\vdash F\sigma:(A\sigma)\mathsf{Type}}$$

$$\frac{}{()\vdash}\qquad\frac{\Gamma\vdash\quad\Gamma\vdash A}{\Gamma.A\vdash}\qquad\frac{\Gamma\vdash A}{\mathsf{p}:\Gamma.A\to\Gamma}\qquad\frac{\Gamma\vdash A}{\Gamma.A\vdash\mathsf{q}:A\mathsf{p}}$$

$$\frac{\sigma:\Delta\to\Gamma\quad\Gamma\vdash A\quad\Delta\vdash u:A\sigma}{(\sigma,u):\Delta\to\Gamma.A}$$

$$\frac{\Gamma\vdash A\quad\Gamma.A\vdash B}{\Gamma\vdash\ \lambda B:(A)\mathsf{Type}}\qquad\frac{\Gamma\vdash F:(A)\mathsf{Type}\quad\Gamma\vdash a:A}{\Gamma\vdash\ \mathsf{app}(F,a)}$$

$$\frac{\Gamma\vdash A\quad\Gamma\vdash F:(A)\mathsf{Type}}{\Gamma\vdash\ \mathsf{Fun}\ A\ F}\qquad\frac{\Gamma.A\vdash b:\mathsf{app}(F\mathsf{p},\mathsf{q})}{\Gamma\vdash\lambda b:\mathsf{Fun}\ A\ F}\qquad\frac{\Gamma\vdash w:\mathsf{Fun}\ A\ F\quad\Gamma\vdash u:A}{\Gamma\vdash\mathsf{app}(w,u):\mathsf{app}(F,u)}$$

$$1\sigma=\sigma=\sigma 1\qquad(\sigma\delta)\nu=\sigma(\delta\nu)$$

$$(\sigma,u)\delta=(\sigma\delta,u\delta)\qquad\mathsf{p}(\sigma,u)=\sigma\qquad\mathsf{q}(\sigma,u)=u$$

$$(A\sigma)\delta=A(\sigma\delta)\qquad A1=A\qquad(a\sigma)\delta=a(\sigma\delta)\qquad a1=a$$

$$\mathsf{app}(w,u)\delta=\mathsf{app}(w\delta,u\delta)\qquad\mathsf{app}(F,u)\delta=\mathsf{app}(F\delta,u\delta)$$

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

Figure 1: Rules of WMLTT

3. We can also add quotient (and truncation operations in the general model) without using impredicative definitions, this has been implemented at the first level
4. We may add the fact that the type of propositions is impredicative. We can then give a computational interpretation of topos theory and of unique existence, but the computations are carried out in a type system with a type of all types (we conjecture that normalization still holds)
5. Given two isomorphic setoids, we can transport any structure and any proof of properties of this structure from one setoid to the other in a computational way
6. At the next level, we have programs to transport properties of equivalent groupoids (and categories) from one to the other (but this is not yet implemented)

References

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

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

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

[4] P. Martin-Löf. An intuitionistic theory of types. 1972, published in the volume *25 Years of Type Theory*, G. Sambin and J. Smith, eds, 1996.

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

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

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