Math. Struct. in Comp. Science (2015), *vol.* 25, *pp.* 1071–1099.
doi:10.1017/S0960129514000504 First published online 20 February 2015

A generalization of the Takeuti–Gandy interpretation

BRUNO BARRAS[†], THIERRY COQUAND[‡] and SIMON HUBER[‡]

[†]*INRIA Saclay - Île de France, 4, rue Jacques Monod, 91893 ORSAY Cedex, France*
Email: bruno.barras@inria.fr
[‡]*Department of Computer Science and Engineering, University of Gothenburg, SE-412 96 Göteborg, Sweden*
Email: thierry.coquand@cse.gu.se, simon.huber@cse.gu.se

Received 2 March 2013; revised 25 June 2014

We present an interpretation of a version of dependent type theory where a type is interpreted by a Kan semisimplicial set. This interprets only a weak notion of conversion similar to the one used in the first published version of Martin-Löf type theory. Each truncated version of this model can be carried out internally in dependent type theory, and we have formalized the first truncated level, which is enough to represent isomorphisms of algebraic structure as equality.

1. Introduction

This paper is part of a general program trying to understand the 'homotopy theoretic' models of type theory (Awodey and Warren 2009) from a constructive point of view, in order to obtain a computational interpretation of the axiom of univalence. This axiom, introduced by Voevodsky has been justified semantically, in a model of type theory where a type is interpreted by a Kan simplicial set (Voevodsky 2010). The situation is however not completely satisfactory since this justification takes place in ZFC, and uses crucially non-effective arguments (a combination of classical logic and the axiom of choice), while one of the ambitions of type theory is to be a language for constructive mathematics (Martin-Löf 1973). It is a weird fact that this unexpected connection between homotopy theory/algebraic topology and type theory (Awodey and Warren 2009; Kapulkin *et al.* 2012; Voevodsky 2010) involves non-effective reasoning since algebraic topology has its historical root in combinatorial topology, which can be thought of as a constructive counterpart of general topology (Dubucs 1988). It is thus natural to try to justify the univalent axiom in a constructive way. Another motivation for this work is a conjecture of Voevodsky about the computational content of the axiom of univalence (Voevodsky 2010); a constructive understanding of the models of univalence ought to bring light to this issue.

The axiom of univalence can be seen as a generalization of one form of the axiom of extensionality in Church's simple type theory (Church 1940), stating that two equivalent propositions are equal. As shown by Voevodsky (2010), this axiom implies also another form of extensionality, *function extensionality*, stating that two pointwise equal functions are equal. Motivated by considerations from proof theory (the problem of consistency of

simple type theory), both[†] Takeuti (1953) and Gandy (1956) provided an 'explanation' of these two forms of extensionality, by giving an interpretation of *extensional* simple type theory (extended with these two axioms of extensionality) in *intensional* simple type theory (formulated without these two axioms). This interpretation consists essentially in *defining* the equality by induction on the types. The equality of propositions is *defined* to be logical equivalence, and the equality at function types will be roughly defined to be pointwise equality. (This is actually one of the first instances of the powerful notion of 'logical relation' in type theory (Statman 1985).)

It is quite natural to try to generalize Takeuti–Gandy's method and see if this can give an explanation of the axiom of univalence. One strong form of this generalization would be an interpretation/translation of dependent type theory extended with the axiom of univalence in dependent type theory. Furthermore, by analogy with Takeuti–Gandy's translation, we would expect that this interpretation can take place even in dependent type theory *without* identity type (in the same way that Aczel's interpretation gives a translation of CZF in dependent type theory without identity types). Our paper explores this possibility, but succeeds only, even for 'truncated version,' in giving an interpretation of a weaker form of type theory (the rules of which are summarized in Figure 3) in dependent type theory (the rules of which are summarized in Figure 2). Even this is not trivial since one needs to interpret judgmental rules in a definitional way, an aspect which is usually not covered (Hofmann 1994; Palmgren 2012) (this issue is discussed, but *not* solved in Hofmann (1994)), while it is *crucial* for stating precisely what type theory has been modelled. Furthermore, this interpretation has been formally checked in the truncated model at level $\leqslant 2$, giving a computational interpretation to transfer structures and properties for isomorphic structures (with the running examples described in Section 4).

This model interprets a *type* by a Kan *semisimplicial* set. There are two motivations for this choice. The first is that it is possible to represent *internally* truncated version of semisimplicial sets, as explained in Section 5, while it is not clear how to do it for truncated version (even at low level) of simplicial sets. The second motivation is that one obstacle for a constructive understanding of the simplicial set model is the (often implicit) use of decidability of the notion of degeneracy[‡]. By looking only at semisimplicial sets, we remove this problem. Also a Kan semisimplicial set contains automatically some simplices that can play the role of degenerate elements, e.g. any point is connected to itself by a line. While this is done only formally in this paper at level $\leqslant 1$, it is expected that this interpretation would generalize at any truncated levels. The last section indicates how it can be generalized to arbitrary levels, but this time in an informal constructive metatheory, by describing a universe of Kan semisimplicial set, and a proof that this universe satisfies the Kan property. We also explain how to transform a weak equivalence between two

[†] The two works seem to be independent, and Gandy's interpretation can already be found in his Ph.D thesis (Gandy 1953).

[‡] A Kripke counter-model shows that, because of this, we *cannot* constructively work with the usual definition of Kan simplicial set, even if we give the filling as explicit operations (Bezem and Coquand 2013).

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

types to an equality of these types. The interpretation of lambda abstraction and identity however gets artificially complicated even at the truncated level, and we choose not to include it in the present version of this paper.

Recently, it was possible to design a model of univalence in a constructive metatheory (not however internally in dependent type theory) (Bezem *et al.* 2014) where a type is interpreted as a Kan cubical set. W.r.t. this work, the present paper contains the following two contributions: one is the fact that the truncated version of the present model can be done internally, the other is that the description of the universe and a proof that it satisfies the Kan property has strong similarity with what happens for cubical sets. There should be a common general picture, and this should be an interesting future work to formulate this common generalization.

The paper is organized as follows. We first present the Takeuti–Gandy interpretation of Church's simple type theory (Church 1940). Roughly speaking, this interprets a type as a type with an equivalence relation, which is reminiscent of Bishop's notion of set in constructive mathematics (Bishop 1967). We present then a first semantics, where a type is interpreted as a truncated Kan semisimplicial set of level $\leqslant 1$. The correctness of this semantics has been formally verified. The system we interpret is close to the first published version of Martin-Löf type theory (Martin-Löf 1973). We give some applications of this semantics, in particular transport along isomorphisms of structures. We then present a universe of (small) Kan semisimplicial sets, which form a semisimplicial set, and we explain why this universe has the Kan filling property and how to transform a weak equivalence between small Kan semisimplicial sets into a path joining them.

2. The Takeuti-Gandy interpretation

The Takeuti–Gandy interpretation (Gandy 1956; Takeuti 1953) was developed for Church's simple type theory (Church 1940). It is natural to analyse the computational interpretation in this case, before considering dependent type theory. In this theory, we have a type o of propositions and function types $A \to B$. The idea is to define (extensional) equality at type A by induction on A. However at the same time, we may need to 'restrict' the type A since it may contain non-extensional elements.

We can describe this interpretation as an internal interpretation in *dependent* type theory. For any simple type A we define a corresponding type $[A]$ and an equality relation $=_A$ on the type $[A]$. Since we are using dependent type theory, where we use Type to represent propositions, this relation is of type $[A] \to [A] \to \mathsf{Type}$. The type $[o]$ is an universe, and $=_o$ is logical equivalence. The type $[A \to B]$ is defined as a sigma type

$$[A \to B] = \sum_{f:[A]\to[B]} \prod_{x:[A]} \prod_{u:[A]} x =_A u \to f\ x =_B f\ u.$$

An element of type $[A \to B]$ is thus a pair f, f' where f is of type $[A] \to [B]$ and f' is a proof that this function is extensional. We define $(f, f') =_{A\to B} (g, g')$ to be

$$\prod_{x:[A]} \prod_{u:[A]} x =_A u \to f\ x =_B g\ u.$$

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

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

$$\frac{\Gamma \vdash t : A \quad \sigma : \Delta \to \Gamma}{\Delta \vdash t\sigma : A}$$

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

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

$$\frac{\Gamma.A \vdash b : B}{\Gamma \vdash \lambda b : A \to B} \qquad \frac{\Gamma \vdash w : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{app}(w, u) : B}$$

$$1\sigma = \sigma \qquad (\sigma\delta)v = \sigma(\delta v) \qquad (\sigma, u)\delta = (\sigma\delta, u\delta)$$

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

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

Fig. 1. Rules of simple type theory with explicit substitution.

A context Γ is then interpreted by a type $[\Gamma]$ with a relation $=_\Gamma$, and a term $\Gamma \vdash t : A$ is interpreted by a function $t\rho : [A]$ for $\rho : [\Gamma]$ *together with* a proof $t\alpha$ of $t\rho_0 =_A t\rho_1$ whenever α is a proof of $\rho_0 =_\Gamma \rho_1$. $\rho_0 =_\Gamma \rho_1$ implies $t\rho_0 =_A t\rho_1$. The equality relation $=_\Gamma$ is an equivalence relation on $[\Gamma]$ and in particular, for each $\rho : [\Gamma]$ we have a proof $1_\rho : \rho =_\Gamma \rho$. The interpretation of $(\lambda t)\rho$, for $\Gamma.A \vdash t : B$, is the pair f, f' where $f\ u = t(\rho, u)$ and $f'\ x\ u\ \omega = t(1_\rho, \omega)$.

Figure 1 presents some key rules of simple type theory.

In general a simple type A is interpreted by a type $[A]$ with a proof relevant relation $=_A$ on this type. In order to motivate the last section describing general Kan semisimplicial sets, we now give an alternative description of this semantics, using informal set theory instead. It should be clear how to go back and forth between the two presentations.

The pair $[A], =_A$ can be also presented as a set $X[0]$, intuitively a set of 'points,' together with a set of 'lines' $X[1]$ and two maps $d_0, d_1 : X[1] \to X[0]$; a proof p of $a_0 =_A a_1$ corresponds to a line p in $X[1]$ with $d_i p = a_i$. Given $X = X[0], X[1]$ and $Y = Y[0], Y[1]$ the function space Y^X is then defined by taking $Y^X[0]$ to be the set of pairs $f, \eta f$ with $f : X[0] \to Y[0]$ and $\eta f : X[1] \to Y[1]$ such that $d_i \eta f = f d_i$ for $i = 0, 1$. We can define $\mathsf{app}((f, \eta f), u) = f\ u : Y[0]$ if $u : X[0]$. We define $Y^X[1]$ to be the set of elements $\lambda, f_0, f_1, \eta f_0, \eta f_1$ with $f_i : X[0] \to Y[0]$ and $\lambda : X[1] \to Y[1]$ such that $d_i \lambda = f_i d_i$ and $d_j \eta f_i = f_i d_j$. We define then $d_i(\lambda, f_0, \eta f_0, f_1, \eta f_1) = f_i, \eta f_i$. We can define $\mathsf{app}((\lambda, f_0, f_1, \eta f_0, \eta f_1), \omega) = \lambda\omega : Y[1]$ if $\omega : X[1]$. With this definition we have $d_i \mathsf{app}(\alpha, \omega) = \mathsf{app}(d_i\alpha, d_i\omega)$. A type A is then interpreted by a pair of sets $A[0], A[1]$ with two maps $d_0, d_1 : A[1] \to A[0]$. Similarly, a context Γ is then interpreted by a pair of sets $\Gamma[0], \Gamma[1]$ with two maps $d_0, d_1 : \Gamma[1] \to \Gamma[0]$. If $\Gamma \vdash t : A$, we should define $t\rho : A[0]$ for $\rho : \Gamma[0]$, and $t\alpha : A[1]$ for $\alpha : \Gamma[1]$ in such a way that $d_i(t\alpha) = t(d_i\alpha)$ for $i = 0, 1$.

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

We get the following operational semantics, reading each equality as a reduction rule

$$(t\sigma)\rho = t(\sigma\rho) \quad (\sigma\delta)\rho = \sigma(\delta\rho) \quad 1\rho = \rho \quad (t\sigma)\alpha = t(\sigma\alpha) \quad (\sigma\delta)\alpha = \sigma(\delta\alpha) \quad 1\alpha = \alpha$$

$$(\sigma, t)\rho = \sigma\rho, t\rho \qquad \mathsf{p}(\rho, u) = u \qquad \mathsf{q}(\alpha, \omega) = \omega$$

$$\mathsf{app}(t_1, t_0)\rho = \mathsf{app}(t_1\rho, t_0\rho) \qquad \mathsf{app}(t_1, t_0)\alpha = \mathsf{app}(t_1\alpha, t_0\alpha)$$

$$\mathsf{app}((\lambda t)\rho, u) = t(\rho, u) \qquad \mathsf{app}((\lambda t)\alpha, \omega) = t(\alpha, \omega) \qquad \mathsf{app}(\eta(\lambda t)\rho, \omega) = t(1_\rho, \omega)$$

$$d_i(\eta f) = f \qquad d_i\mathsf{app}(\lambda, \omega) = \mathsf{app}(d_i\lambda, d_i\omega).$$

A logically equivalent definition of $(f, f') =_{A\to B} (g, g')$ would be

$$\forall x : [A].f\ x =_B g\ x$$

however, this definition *would not provide the right definitional equality*. In particular it would not validate β-conversion

$$\Gamma \vdash t(1, u) = \mathsf{app}(\lambda t, u) : B$$

for $\Gamma.A \vdash t : B$ and $\Gamma \vdash u : A$. With both definitions we get the equality

$$\mathsf{app}(\lambda t, u)\rho = t(1, u)\rho = t(\rho, u\rho) : [B]$$

for $\rho : \Gamma$, while it is only with our definition that we get the equality

$$\mathsf{app}(\lambda t, u)\alpha = t(1, u)\alpha = t(\alpha, u\alpha) : \quad t(\rho_0, u\rho_0) =_B t(\rho_1, u\rho_1)$$

for $\alpha : \rho_0 =_\Gamma \rho_1$.

We do not interpret all the laws of cartesian closed category: the law

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

is *not* valid in this model. This is because the second component of $(\lambda t)\sigma\rho$ and $(\lambda t(\sigma\mathsf{p}, \mathsf{q}))\rho$ do not coincide in general since $1_{\sigma\rho}$ may not coincide with $\sigma 1_\rho$. However all the equations of Figure 1 are valid in this model. (We explain later why this set of laws is satisfactory for representing one version of dependent type theory.) An important point is that to require such an equality is *not* expressible in dependent type theory, if one wants to express this as a conversion (and not only as a propositional equality).

The original motivation of this interpretation for both Takeuti and Gandy was the consistency problem of higher-order arithmetic (Gandy 1956; Takeuti 1953). Simple type theory *without* function extensionality is a simpler system, and it was expected that the consistency of this simpler system would be easier to analyse. This has been confirmed later, and the intuitionistic version of this system, presented in natural deduction, has good proof theoretic properties (Martin-Löf 1971), which lead eventually to a proof-theoretic analysis of the problem of consistency of higher-order arithmetic (Girard 1971; Martin-Löf 1971).

3. Effectivity problems with the Kan simplicial set model

One can see the Kan simplicial set model of type as a generalization of the previous interpretation of simple type theory, where a simplicial set generalizes the notion of set

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

with a relation, and a Kan simplicial set generalizes the notion of set with an equivalence relation.

When analyzing the Kan simplicial set model of type theory (Awodey and Warren 2009; Kapulkin *et al.* 2012; Streicher 2011; Voevodsky 2010), one effectivity problem relies on the use of the *decidability of the notion of degeneracy* and the fact that simplicial maps have to commute with the degeneracy functions (this issue appears already above in the semantics of simple type theory). Let us write $[n]$ for the linear poset $\{0,\dots,n\}$. Let Δ be the category of such linear posets $[n]$ with morphisms all monotone maps. The category of *simplicial sets* if the presheaf category $[\Delta^{op}, \mathsf{Set}]$. A simplicial set is thus a sequence of sets $X[n]$ together with maps $X[n] \to X[m]$, $u \longmapsto uf$ for $f : [m] \to [n]$ satisfying $u1 = u$ and $(uf)g = u(fg) : X[p]$ another notation for uf would be $X(f)(u)$. if $g : [p] \to [m]$. We write $\epsilon_i : [n-1] \to [n]$ for the injective map that omits i; this is the ith face map, and we may write $d_i u$ instead of $u\epsilon_i$. An element $u : X[n]$ is called *degenerate* if, and only if, there is a non-trivial surjective map $g : [n] \to [m]$ and an element $v : X[m]$ such that $u = vg$. In a constructive setting, to be degenerate is not in general a decidable property. (It is decidable in the case of simple examples of spaces, like the spheres, but it is not decidable as soon as we consider simplicial sets obtained by exponentiation.) However the theory of simplicial sets and of Kan simplicial sets *uses this decidability at crucial points.*

We give here a simple example of the use of this decidability. If $p : B \to A$ is a Kan fibration, given two points $a\ u : A[0]$ with a path $\omega : A[1]$ connecting a and u, one expects the fibers $B(a)$ and $B(u)$ to be equivalent Kan simplicial sets. In order to define the map $f_0 : B(a)[0] \to B(u)[0]$ one simply uses the Kan condition: given a point $b : B[0]$ such that $p(b) = a : A[0]$ one can lift the path ω to a path ω' in B such that $d_0\omega' = b$ and one may define $f_0\ b = d_1\omega'$. But in order to define $f_1 : B(a)[1] \to B(u)[1]$ it seems necessary to define $f_1\alpha$ by case whether α is degenerate or not. Moreover, it can be shown, by a suitable Kripke counter-model (Bezem and Coquand 2013), that this fact is *not* intuitionistically provable if we use the ordinary notion of Kan filling condition, *even* if these fillings are explicitly given.

Similar problems are found when analyzing various proofs (Goerss and Jardine 1997; May 1967) that B^A is a Kan simplicial set if B is a Kan simplicial set. We conjecture that this result is also not valid intuitionistically.

These effectivity problems make it impossible to use the Kan simplicial set model for a computational interpretation of the axiom of univalence. There are various directions that we can explore to overcome this problem. The direction we explore in this paper is to interpret a type not as a Kan *simplicial* set, but as a Kan *semisimplicial* set. As we shall show, this gives indeed an interpretation of one formulation of dependent type theory. One important feature of this interpretation is furthermore that any truncated version of the model can be carried out *internally* in dependent type theory. We do it in this paper only for the first level, but it is possible to show that this can be done at all level. Another option is to analyse further the notion of degeneracy. This has been done recently (Bezem *et al.* 2014), and gives a satisfactory interpretation of dependent type theory (also with substitution under abstraction), but it does not seem possible to describe this interpretation internally in dependent type theory.

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

4. The Takeuti–Gandy interpretation for dependent type theory

4.1. *Identity type*

The first predicative version 1972 of type theory (Martin-Löf 1972) did not have identity types. Over the type of natural numbers for instance, equality was defined recursively using a universe. This version (Martin-Löf 1972) stayed unpublished for some time and the identity type was introduced in the first published 1973 version of type theory (Martin-Löf 1973). Since this version has not been described in the literature so far (for instance the reference (Troelstra and van Dalen 1988) gives a presentation using another version of the identity type), we present some remarks about it.

Proposition 4.1. In the 1973 version of type theory (Martin-Löf 1973), *function extensionality* stating that $\mathsf{Id}_{A\to B}\ f\ g$ follows from

$$\prod_{x:A} \mathsf{Id}_B\ \mathsf{app}(f,x)\ \mathsf{app}(g,x)$$

is *not* provable.

Proof. This follows from the fact that if $\mathsf{Id}_T\ a\ u$ is provable in the empty context, then a and u are convertible (Martin-Löf 1973), and the fact that we can have two functions that are not convertible but pointwise equal e.g. the functions $\lambda n.n+0$ and $\lambda n.0+n$ on natural numbers. □

It is remarkable that the explanation of CZF in type theory (Aczel 1978), interpreting a *set* as a well-founded tree up to bisimulation, does not use the identity type, so it is an interpretation of CZF in the 1972 version of type theory (Martin-Löf 1972).

4.2. *Type theory as a formal system and definitional equality*

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. The method to produce a canonical proof is quite uniform: given a term t of type A, we unfold the definitions until we reach a canonical proof. Using the terminology of functional programming, a canonical proof is represented as a term starting with a constructor, and the method of computation is head reduction.

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

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

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 (Martin-Löf 1973). To decide this relation reduces to the problem of comparing two given terms of the same type. 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

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

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. This means that if t is of type $F\ 2$ and u is of type $\neg\ A$ then $\mathsf{app}(t, u)$ is well-typed.

This notion of definitional equality is analysed in Martin-Löf (1975). An early use of this notion can be found in the paper (Martin-Löf 1971). It appeared also before in the work on Automath (De Bruijn 1980) and in Tait's analysis of Gödel's Dialectica interpretation (Tait 1967).

Figure 3 presents the rules of a version of type theory using explicit substitution. One can argue that the conversion rule $(\lambda t)\sigma = \lambda t(\sigma \mathsf{p}, \mathsf{q})$, which expresses the law of substitution under abstraction, is not compatible with this idea of unfolding definition (Martin-Löf 1973, 1975). On the other hand, the rules in Figure 3 can be seen as a formal description of basic rules of definitional equality.

4.3. *The rules of type theory and weak type theory*

The rules of type theory which we are using to formalize the proofs, and that we would like to represent internally are summarized in Figure 2.

The type theory we interpret is a variation of the one presented in the references (Martin-Löf 1973, 1975; Troelstra and van Dalen 1988). Besides the usual judgment $\Gamma \vdash$, $\Gamma \vdash A$ and $\Gamma \vdash a : A$, we also have the judgment $\Gamma \vdash F : (A)\mathsf{Type}$ for families of types over a given type. (This judgment has been introduced by Martin-Löf in his, unpublished, substitution calculus.)

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

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

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

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)

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

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

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

$$\frac{\Gamma \vdash A \quad \sigma : \Delta \to \Gamma}{\Delta \vdash A\sigma} \qquad \frac{\Gamma \vdash t : A \quad \sigma : \Delta \to \Gamma}{\Delta \vdash t\sigma : A\sigma}$$

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

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

$$\frac{\Gamma.A \vdash B}{\Gamma \vdash \Pi\ A\ B} \qquad \frac{\Gamma.A \vdash B \quad \Gamma.A \vdash b : B}{\Gamma \vdash \lambda b : \Pi\ A\ B}$$

$$\frac{\Gamma \vdash w : \Pi\ A\ B \quad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{app}(w, u) : B[u]}$$

$$1\sigma = \sigma 1 = \sigma \qquad (\sigma\delta)v = \sigma(\delta v) \qquad [u] = (1, u)$$

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

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

$$(\Pi\ A\ B)\sigma = \Pi\ (A\sigma)\ (B(\sigma\mathsf{p}, \mathsf{q})).$$

Fig. 2. Rules of MLTT.

The substitution rules are then

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

We can add rules for sigma types. The typing rules are

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash F : (A)\mathsf{Type}}{\Gamma \vdash \mathsf{Sum}\ A\ F} \qquad \frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : \mathsf{App}(F, a)}{\Gamma \vdash (a, b) : \mathsf{Sum}\ A\ F}$$

$$\frac{\Gamma \vdash c : \mathsf{Sum}\ A\ F}{\Gamma \vdash \mathsf{p}c : A} \qquad \frac{\Gamma \vdash c : \mathsf{Sum}\ A\ F}{\Gamma \vdash \mathsf{q}c : \mathsf{App}(F, \mathsf{p}c)}$$

and the computation rules

$$\mathsf{p}(a, b) = a \qquad \mathsf{q}(a, b) = b \qquad (\mathsf{Sum}\ A\ F)\sigma = \mathsf{Sum}\ A\sigma\ F\sigma$$

This version of type theory is called weak type theory, by analogy with the notion of weak conversion in lambda-calculus (Martin-Löf 1975), since we do not include the conversion rule

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

The first published version of type theory (Martin-Löf 1973) did not have this rule. (The type theory of Figure 3 is actually an extension of the theory presented in Martin-Löf (1973), which used a system of combinators.) What are the consequences of not having this rule is not clear. On one hand, this actually simplifies type checking since the conversion

$$(\lambda B)\sigma = (\lambda C)\delta$$

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

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

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

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

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

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

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

$$1\sigma = \sigma = \sigma 1 \qquad (\sigma\delta)v = \sigma(\delta v) \qquad 1 = (\mathsf{p}, \mathsf{q})$$

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

$$(F\sigma)\delta = F(\sigma\delta) \qquad F1 = F$$

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

$$\mathsf{app}(w, u)\delta = \mathsf{app}(w\delta, u\delta) \quad \mathsf{App}(F, u)\delta = \mathsf{App}(F\delta, u\delta) \quad (\mathsf{Fun}\ A\ F)\sigma = \mathsf{Fun}(A\sigma)(F\sigma)$$

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

Fig. 3. Rules of WMLTT.

is only possible if $B = C$ and $\sigma = \delta$, while with the rule of substitution under abstraction, this may happen because $B(\sigma\mathsf{p}, \mathsf{q}) = C(\delta\mathsf{p}, \mathsf{q})$. On the other hand, no precise conservativity theorem is known about the two versions of type theory. We conjecture that it is not possible to have an internal representation of the full theory.

It was argued in Martin-Löf (1975) that the weak version of type theory is actually better behaved then the version allowing substitution under abstractions. This was not however explored further and we think that it would be interesting to analyse more in detail the difference between these two versions of type theory.

5. A first version of the model

In this section, we present a model where a type is interpreted by a Kan semisimplicial set of level $\leqslant 1$. The collection of all such types is interpreted by a Kan semisimplicial set of level $\leqslant 2$. We interpret contexts as Kan semisimplicial sets of level $\leqslant 2$. In this way, we can interpret contexts of the form $X : U, a : X, f : X \to X$ where X varies over the collection of all (small) Kan semisimplicial set of level $\leqslant 1$. We give a model of *weak* type theory in type theory, which furthermore interprets an extensional version of the identity type. *This model has been formally verified in the system Coq 8.4.*

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

In the usual (set-based) presentation of semisimplicial sets, there is a single set for each level (points, edges, etc.), and there are face maps that, for instance, return the three edges forming the boundary of a given triangle. It is not clear how this presentation could be internalized in a type-theoretical setting. It would make use of propositional equality and one would need to state coherence conditions between provably but not definitionally equal types. For *truncated* versions however, dependent types can be used to definitionally express the relation between a semisimplicial set and its faces. In this settings, points are the objects of a type. Let us call this type X_0. Edges are represented by a type X_1 parameterized by two points: given $a, b : X_0$, the type $X_1\, a\, b$ is the type of edges between a and b. At level 3, we need to give three points $a_0, a_1, a_2 : X_0$ and three edges $a_{01} : X_1\, a_0\, a_1$, $a_{02} : X_1\, a_0\, a_2$ and $a_{12} : X_1\, a_1\, a_2$ to form the type of triangles $X_2\, a_0\, a_1\, a_2\, a_{01}\, a_{02}\, a_{12}$. We generally omit to mention the points since they can be recovered from the edges types, and simply write $X_2\, a_{01}\, a_{02}\, a_{12}$.

5.1. *Kan completion*

First we define Kan completion operations at each level. At level n, given n faces of level $n-1$ forming a 'horn,' they produce the face of level $n-1$ omitted in the horn.

Definition 5.1. At level 1, given two types A and B, we write $A \leftrightarrow B$ for the type of pair of functions $(comp^1_0, comp^1_1)$ such that

$$comp^1_0 : A \to B \qquad \text{and} \qquad comp^1_1 : B \to A.$$

At level 2, given three types A_0, A_1, A_2, and three heterogeneous relations R_{01}, R_{02} and R_{12},[†] we write $R_{01} \leftrightarrow R_{02} \leftrightarrow R_{12}$ for the type of the following three operations:

$$\begin{aligned} comp^2_0 &: R_{01}\, a_0\, a_1 \to R_{02}\, a_0\, a_2 \to R_{12}\, a_1\, a_2 \\ comp^2_1 &: R_{01}\, a_0\, a_1 \to R_{12}\, a_1\, a_2 \to R_{02}\, a_0\, a_2 \\ comp^2_2 &: R_{02}\, a_0\, a_2 \to R_{12}\, a_1\, a_2 \to R_{01}\, a_0\, a_1 \end{aligned}$$

for all $a_0 : A_0$, $a_1 : A_1$ and $a_2 : A_2$.

At level 3, given four types $A_i\, (0 \leqslant i < 4)$, six relations $R_{ij}\, (0 \leqslant i < j < 4)$ and four types of triangles $T_{ijk}\, (0 \leqslant i < j < k < 4)$, we write $T_{012} \leftrightarrow T_{013} \leftrightarrow T_{023} \leftrightarrow T_{123}$ for the type of the following four operations:

$$\begin{aligned} comp^3_0 &: T_{012}\, a_{01}\, a_{02}\, a_{12} \to T_{013}\, a_{01}\, a_{03}\, a_{13} \to T_{023}\, a_{02}\, a_{03}\, a_{23} \to T_{123}\, a_{12}\, a_{13}\, a_{13} \\ comp^3_1 &: T_{012}\, a_{01}\, a_{02}\, a_{12} \to T_{013}\, a_{01}\, a_{03}\, a_{13} \to T_{123}\, a_{12}\, a_{13}\, a_{13} \to T_{023}\, a_{02}\, a_{03}\, a_{23} \\ comp^3_2 &: T_{012}\, a_{01}\, a_{02}\, a_{12} \to T_{023}\, a_{02}\, a_{03}\, a_{23} \to T_{123}\, a_{12}\, a_{13}\, a_{13} \to T_{013}\, a_{01}\, a_{03}\, a_{13} \\ comp^3_3 &: T_{013}\, a_{01}\, a_{03}\, a_{13} \to T_{023}\, a_{02}\, a_{03}\, a_{23} \to T_{123}\, a_{12}\, a_{13}\, a_{13} \to T_{012}\, a_{01}\, a_{02}\, a_{12} \end{aligned}$$

for all $a_i : A_i$ and $a_{ij} : R_{ij}\, a_i\, a_j$.

Next, we define a Kan filler operation that, given the same input as the Kan completion above, returns a simplex which boundary is the completed horn described in the previous paragraph.

[†] The indices suggest the domain and range type of the relations.

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

Definition 5.2. At level 1, given two types A and B, the *coherence* between a relation R on A and B, and *completion operations* $(comp_0^1, comp_1^1) : A \leftrightarrow B$, written $Coh(R, comp^1)$ are defined by the following operations:

$$Comp_0^1 : \forall x : A.\ R\, x\, (comp_0^1\, x) \qquad \text{and} \qquad Comp_1^1 : \forall y : B.\ R\, (comp_1^1\, y)\, y.$$

At level 2, given three types, and three types of edges R_{01}, R_{02} and R_{12} between these types, the coherence between a type of triangles T and $comp^2$ a completion operation at level 2 ($R_{01} \leftrightarrow R_{02} \leftrightarrow R_{12}$), written $Coh(T, comp^2)$ is defined as:

$$\begin{aligned} Comp_0^2 &: \forall a_{01} : R_{01}\, a_0\, a_1.\, \forall a_{02} : R_{02}\, a_0\, a_2.\ T\, a_{01}\, a_{02}\, (comp_0^2\, a_{01}\, a_{02}) \\ Comp_1^2 &: \forall a_{01} : R_{01}\, a_0\, a_1.\, \forall a_{12} : R_{12}\, a_1\, a_2.\ T\, a_{01}\, (comp_1^2\, a_{01}\, a_{12})\, a_{12} \\ Comp_2^2 &: \forall a_{02} : R_{02}\, a_0\, a_2.\, \forall a_{12} : R_{12}\, a_1\, a_2.\ T\, (comp_2^2\, a_{02}\, a_{12})\, a_{02}\, a_{12} \end{aligned}$$

Note that the conjunction of these two operations at a given level can be reformulated. To have both $comp^1$ and $Comp^1$ is equivalent to the statement: for any point of A, there exists a point in B that is related by a relation R to the former point, and conversely for any point of B, there exists a point in A related by R to the former. At level 2, it says that for every pair of connected edges, there exists a third edge forming a triangle. And so on at higher levels.

However, our formulation makes it clear that we have actual operations that builds the witnesses of the existential statements. The reason for splitting this condition will appear in the definitions of truncated Kan semisimplicial sets below.

5.2. *Small types*

Definition 5.3 (small types). A small type A is a Kan semisimplicial set of level $\leqslant 1$. It consists of the following types and operations:

— a small type of points written simply A when not ambiguous,
— a small type of edges $\eta A\, a_0\, a_1$ for any $a_0, a_1 : A$,
— Kan edge completion $comp^1 : A \leftrightarrow A$,
— Kan edge filling operation at level 1 $Comp^1 : Coh(\eta A, comp^1)$.
— Kan triangle completion $comp^2 : \eta A \leftrightarrow \eta A \leftrightarrow \eta A$.

Note that this truncated version does not require the Kan filling operation at level 2.

Such a structure can be seen as another presentation of the notion of 'proof-relevant' equivalence relation on a type. This can also be seen as a type-theoretic representation of Bishop's notion of set (Bishop 1967; Mines *et al.* 1988). Let us make more precise how this definition is equivalent to setoids. First, setoids can be derived from a Kan semisimplicial set of level $\leqslant 1$:

— ηA is a proof-relevant relation, but none of the requirements discriminate between witnesses of $\eta A\, x\, y$; it can be thought of as the equality on the set A, in the sense of Bishop;
— $comp^2$ implies symmetry and transitivity of ηA, and
— further assuming $comp^1$ and $Comp^1$, we can derive reflexivity of ηA.

Conversely, setoids allow to derive the completion operations above.

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

However, even though the two notions are mutually derivable, we believe that the Kan semisimplicial approach provides more uniform notation and generalizes better to higher dimensions.

From now on, to alleviate the overloading of the term *type*, setoid will refer to the structure given in Definition 5.3.

Setoid morphisms are functions from one setoid to another preserving edges.

Definition 5.4 (type morphisms). Let A and B be two setoids. A morphism from A to B is a pair of functions $(f, \eta f)$ such that

— $f\ a : B$ for all $a : A$ and
— $\eta f\ a_{01} : \eta B\ (f\ a_0)\ (f\ a_1)$ for all $a_0, a_1 : A$ and $a_{01} : \eta A\ a_0\ a_1$.

We proceed to define the notion of equality on setoids. Again, as in Bishop's interpretation, it can be described as the graph of isomorphisms between the (Bishop) sets A and B.

Definition 5.5 (isomorphisms). Let A and B be two setoids. An isomorphism between A and B is a structure I formed of a relation (written I) such that we have:

— a Kan completion of level 1, $comp^1 : A \leftrightarrow B$ and $Comp^1 : Coh(I, comp^1)$, and
— two degenerate triangles

$$\eta_0 I : \eta A \leftrightarrow I \leftrightarrow I \qquad \text{and} \qquad \eta_1 I : I \leftrightarrow I \leftrightarrow \eta B.$$

The two triangles ensure that the relation respects the equality on both ends of the relation.

5.3. *Contexts*

Definition 5.6 (contexts). A context Γ is a Kan semisimplicial set of level $\leqslant 2$. It consists of all the fields of small types (with the difference that types are not required to be small), and

— a type of triangles $\eta_1\eta\Gamma\ a_{01}\ a_{02}\ a_{12}$, for all $a_i : \Gamma$ and $a_{ij} : \eta\Gamma\ a_i\ a_j$,
— Kan triangle filling operations $Comp^2 : Coh^2(\eta_1\eta\Gamma, comp^2)$, and
— Kan tetrahedron completion $comp^3 : \eta_1\eta\Gamma \leftrightarrow \eta_1\eta\Gamma \leftrightarrow \eta_1\eta\Gamma \leftrightarrow \eta_1\eta\Gamma$.

Note that this truncated version does not require the Kan tetrahedron filling operation.

Context morphisms are functions from one context to another preserving edges and triangles.

Definition 5.7 (context morphism). Let Δ and Γ be two contexts. A morphism from Δ to Γ is a triple $(f, \eta f, \eta_1\eta f)$ such that:

— $f\ \rho : \Gamma$ for all $\rho : \Delta$ and
— $\eta f\ \rho_{01} : \eta\Gamma\ (f\ \rho_0)\ (f\ \rho_1)$ for all $\rho_0, \rho_1 : \Delta$, $\rho_{01} : \eta\Delta\ \rho_0\ \rho_1$.
— $\eta_1\eta f\ \theta_{012} : \eta_1\eta\Gamma\ (\eta f\ \rho_{01})\ (\eta f\ \rho_{02})\ (\eta f\ \rho_{12})$ for all $\theta_{012} : \eta_1\eta\Delta\ \rho_{01}\ \rho_{02}\ \rho_{12}$.

Lemma 5.8. Any setoid can be turned into a context by introducing exactly one triangle for each triple of connected edges.

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

5.4. *Interpretation of the universe*

Theorem 5.9. The semisimplicial set U of level $\leqslant 2$ where

— the points of U are setoids,
— $\eta U\, A\, B$ is the set of isomorphisms between A and B,
— $\eta_1 \eta U\, I_{01}\, I_{02}\, I_{12}$ is $I_{01} \leftrightarrow I_{02} \leftrightarrow I_{12}$,

satisfies the Kan extension property.

Proof. At level 1, we use the fact that a setoid is isomorphic to itself. Level 2 completions involve the composition of isomorphisms. Given three setoids A_0, A_1 and A_2, and two isomorphisms I_{01}, I_{02}, it suffices to compose them and obtain an isomorphism between A_1 and A_2. The third level is tedious but straightforward. Given four types, six morphisms and three triangles, we can form the fourth triangle of the tetrahedron. □

5.5. *Interpretation of the judgments*

The judgment $\Gamma \vdash$ of Section 4 is represented in type theory as an expression Γ of the type of structures of Definition 5.6. The other judgments are described below.

5.5.1. *Types.* Types of a context Γ (written $Ty(\Gamma)$) are mappings from Γ to Kan semisimplicial types of level $\leqslant 1$ (setoids), with additional requirements ensuring that equal contexts yield isomorphic setoids, and similarly for triangles. In other words, an element of $Ty(\Gamma)$ is simply a context morphism between Γ and U.

Then, a judgment of the form $\Gamma \vdash A$ is represented in type theory as an expression A of type $Ty(\Gamma)$.

5.5.2. *Elements.* Given a context Γ and a type $A : Ty(\Gamma)$, an element t of A is a function that returns an element of $A\rho$ for each element ρ of the context. This function is also required to map equal contexts to equal elements.

Definition 5.10. Given a context Γ and $A : Ty(\Gamma)$, an element of A is given by two functions t and ηt such that:

— $t\rho : A\rho$ for any $\rho : \Gamma$,
— $\eta t \rho_{01} : \eta A \rho_{01}\, (t\rho_0)\, (t\rho_1)$ for any $\rho_i : \Gamma$ and $\rho_{01} : \eta\Gamma\, \rho_0\, \rho_1$.

We define $Elt(\Gamma, A)$ to be the type of elements of A.

Formally, a judgment of the form $\Gamma \vdash t : A$ is represented by an expression t of type $Elt(\Gamma, A)$.

5.5.3. *Substitutions.* Substitutions are represented by context morphisms. A judgment $\sigma : \Delta \to \Gamma$ is encoded in type theory as a term σ which is a morphism from context Δ to context Γ.

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

The construction of the identity morphism and the composition of morphisms justify the rules

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

It is also straightforward to derive the rules

$$\frac{\Gamma \vdash A \quad \sigma : \Delta \to \Gamma}{\Delta \vdash A\sigma} \qquad \frac{\Gamma \vdash t : A \quad \sigma : \Delta \to \Gamma}{\Delta \vdash t\sigma : A\sigma}$$

as a form of composition of A (resp. t) with σ.

Definition 5.11 (context extension $\Gamma.A$). Given Γ a context and A a type of Γ, we can build a context $\Gamma.A$, defined by:

— the type of points of $\Gamma.A$ is $\Sigma\rho : \Gamma.\, A\rho$;
— the edges between two points (ρ_0, a_0) and (ρ_1, a_1) is a dependent pair of edges, of type $\Sigma\omega : \eta\Gamma\, \rho_0\, \rho_1.\, \eta A\omega\, a_0\, a_1$;
— a triangle between three edges (ω_{01}, a_{01}), (ω_{02}, a_{02}) and (ω_{12}, a_{12}) is simply a triangle between ω_{01}, ω_{02} and ω_{12}. This follows the idea that small types are injected in contexts by equipping them with trivial triangles.
— The Kan operations are defined straightforwardly.

This definition interprets the rule

$$\frac{\Gamma \vdash \quad \Gamma \vdash A}{\Gamma.A \vdash}$$

The definition of $\Gamma.A$ suggests that an element of this context can be projected to obtain either an element of Γ or an element of A, and conversely, an element of $\Gamma.A$ can be formed from a context of Γ and an element of A. Hence the definition of p, q and $(\_, \_)$ is justifying the rules

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

5.5.4. *Definitional equality.* The definitional equality of WMLTT is interpreted by the definitional equality of underlying type theory (here Coq) in which we represent this system. This is the main feature of our interpretation. It is actually difficult to state precisely in what sense one has an internal model of type theory if these definitional equalities are not present.

Note that equalities justified by our model are typed. Although the rules given in Section 4 are presented in an untyped style, they should actually be presented as a judgment. More precisely, the equality judgment $t = u$ omits the context and type of this judgment in the informal notation, but they do appear in the formal presentation, and some typing premises are required, as we shall see below.

The definitional equality of type theory can be represented within type theory using the following meta-theoretical result: if $t = u$ is provable by reflexivity, then t and u are definitionally equal.

The rule $\mathsf{p}(\sigma, u) = \sigma$ is represented in type theory by the fact that the lemma $\mathsf{p}(\sigma, u) = \sigma$ (equation between morphisms from $\Delta \to \Gamma$) for all $\sigma : \Delta \to \Gamma$ and $u : Elt(\Gamma, A\sigma)$ is proved

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

by reflexivity. In the same way, the following equations are proved

$$\frac{\sigma : \Delta \to \Gamma}{1\sigma = \sigma : \Delta \to \Gamma \quad \sigma 1 = \sigma : \Delta \to \Gamma} \qquad \frac{\sigma : \Delta \to \Gamma \quad \delta : \Theta \to \Delta \quad v : \Psi \to \Theta}{(\sigma\delta)v = \sigma(\delta v) : \Psi \to \Gamma}$$

$$\frac{\sigma : \Delta \to \Gamma \quad \Gamma \vdash u : A\sigma}{\mathsf{p}(\sigma, u) = \sigma : \Delta \to \Gamma \quad \Delta \vdash \mathsf{q}(\sigma, u) = u : A\sigma}$$

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

The latter rule needs $(A\sigma)\delta = A(\sigma\delta)$ to type-check because $u\delta$ has $(A\sigma)\delta$ while it is expected to have type $A(\sigma\delta)$.

We have now fully defined a type theory with explicit substitutions. Next, we introduce the usual type constructors of MLTT: product and sum types.

5.5.5. *Type families.* This section is dedicated to what corresponds to the notion of family of sets used in constructive mathematics (Bishop 1967; Mines *et al.* 1988; Palmgren 2012), where objects $A : U$ correspond to Bishop sets. A family of setoids indexed by a type A is written $(A)\mathsf{Type}$.

However, since we need to model expressions with free variables, we also need to define families in a context Γ. In our model, this needs further definitions, as we need to explain when two families are isomorphic, and also when three family isomorphisms form a triangle.

As suggested above, a setoid family indexed by A is simply a context morphism from A (a setoid viewed as a context) to U.

Definition 5.12 (setoid family isomorphism). Given an isomorphism $I : \eta U\, A_0\, A_1$ and two setoid families F_0 and F_1 with F_i indexed by A_i, an isomorphism J between F_0 and F_1 consists of a triple $(J, \eta_0 J, \eta_1 J)$ specified by:

— Ja_{01} is an isomorphism from $F_0 a_0$ to $F_1 a_1$ for all $a_i : A_i$ and $a_{01} : I\, a_0\, a_1$;
— $\eta_0 J(a_{00'}, a_{01}, a_{0'1}) : \eta F_0 a_{00'} \leftrightarrow Ja_{01} \leftrightarrow Ja_{0'1}$ for all $a_i : A_i$, $a_{0'} : A_0$, $a_{00'} : \eta A_0\, a_0\, a_{0'}$ and $a_{ij} : I\, a_i\, a_j$;
— $\eta_1 J(a_{01}, a_{01'}, a_{11'}) : Ja_{01} \leftrightarrow Ja_{01'} \leftrightarrow \eta F_1 a_{11'}$ for all $a_i : A_i$, $a_{1'} : A_1$, $a_{11'} : \eta A_1\, a_1\, a_{1'}$ and $a_{ij} : I\, a_i\, a_j$.

Quite naturally, three setoid family isomorphisms form a triangle if any triangle in the index types can be mapped to a triangle in U between the resulting setoid isomorphisms.

Definition 5.13 (type family triangle). Given three setoids, three isomorphisms A_{01}, A_{02} and A_{12} between them, three families F_0, F_1 and F_2 (with F_i indexed by A_i), a triple of type family isomorphisms F_{01}, F_{02} and F_{12} forms a triangle when we have $F_{01}a_{01} \leftrightarrow F_{02}a_{02} \leftrightarrow F_{12}a_{12}$ for all $a_i : A_i$ and $a_{ij} : A_{ij}\, a_i\, a_j$.

This condition is indeed equivalent to the fact that two isomorphisms $\eta F\, a_{01}$ and $\eta F\, a'_{01}$ are extensionally equal for any two proofs a_{01}, a'_{01} of $\eta A\, a_0\, a_1$.

Informally, a type family (in a context) should simply be a morphism from the context to the structure for which we have define the points, edges and triangles. However, since

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

the index type may depend on its context, we cannot reuse as is the notion of context morphism. Instead we make a similar definition.

Definition 5.14 (type families). Given a context Γ, and $A : Ty(\Gamma)$, a type family over A is a tuple of functions $(F, \eta F, \eta_1\eta F)$ such that:

— $F\rho$ is a context morphism from $A\rho$ to U for any $\rho : \Gamma$,
— $\eta F\rho_{01}$ is a setoid family isomorphism from $F\rho_0$ to $F\rho_1$ for all $\rho_i : \Gamma$ and $\rho_{01} : \eta\Gamma\,\rho_0\,\rho_1$; the isomorphism between index setoids $A\rho_0$ and $A\rho_1$ is, without surprise, $\eta A\rho_{01}$;
— $\eta_1\eta F\rho_{012}$ is a setoid family triangle between $\eta F\rho_{01}$, $\eta F\rho_{02}a_{02}$ and $\eta F\rho_{12}a_{12}$ for all $\rho_i : \Gamma$, $\rho_{ij} : \eta\Gamma\,\rho_i\,\rho_j$ and $\rho_{012} : \eta_1\eta\Gamma\,\rho_{01}\,\rho_{02}\,\rho_{12}$.

We define $Fam(\Gamma, A)$ as the type of families over A.

The reader should be careful about the ambiguity there may be between the field η_1 of ηF and $\eta_1\eta F$. This will be addressed by writing simply $\eta_1 F$ for the former (and for consistency of notations, we will also write $\eta_0 F$).

In the current setting, a judgment of the form $\Gamma \vdash F : (A)\mathsf{Type}$ is interpreted by an expression F of type $Fam(\Gamma, A)$.

Ideally, following the informal explanation above, type families should be defined independently of the ambient context Γ: a context F of all setoid families $(A)\mathsf{Type}$ would be defined, and then $Fam(\Gamma, A)$ would be the set of context morphisms f from Γ to F such that the index type of $f\rho$ is $A\rho$. This lends itself better to a generalization, as explained in Section 8. Dealing with the dependency of the index on the context is the key difficulty.

Definition 5.15 (type application). Given a family of types F indexed by A in context Γ and a an element of A, we can define $\mathsf{App}(F, a)$, a type which corresponds to the element of the family at a, by:

$$\mathsf{App}(F,a)\rho = F\rho\,a\rho \quad \eta\mathsf{App}(F,a)\rho_{01} = \eta F\rho_{01}\,\eta a\rho_{01}$$
$$\eta_1\eta\mathsf{App}(F,a)\rho_{012} = \eta_1\eta F\rho_{012}(\eta a\rho_{01}, \eta a\rho_{02}, \eta a\rho_{12})$$

with the usual typing convention.

This definition interprets the rule

$$\frac{\Gamma \vdash F : (A)\mathsf{Type} \quad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{App}(F, a)}$$

Substitution of type families is defined in the same style as before, and we can prove that substitution commutes with application:

$$\frac{\Gamma \vdash F : (A)\mathsf{Type} \quad \sigma : \Delta \to \Gamma}{\Delta \vdash F\sigma : (A\sigma)\mathsf{Type}} \qquad \mathsf{App}(F,a)\sigma = \mathsf{App}(F\sigma, a\sigma).$$

Again, this equation is a simplification of the formal result, which requires well-typedness conditions.

The introduction rule for type families is more tedious and less canonical as it depends on auxiliary definitions that may be implemented in different ways.

Definition 5.16 (type-level λ-abstraction). Given a type A in context Γ and a type B in $\Gamma.A$, then we can define λB, a family indexed by A in Γ by:

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

— $\lambda B \rho a = B(\rho, a)$
— $\eta(\lambda B \rho) a_{01} = \eta B(\varphi(\rho), \psi(\rho, a_{01}))$
— $\eta_1 \eta(\lambda B \rho) a_{012} = \eta_1 \eta B(\varphi'(\rho), (\psi(\rho, a_{01}), \psi(\rho, a_{02}), \psi(\rho, a_{12})))$
— $\eta(\lambda B) \rho_{01} a_{01} = \eta B(\rho_{01}, a_{01})$
— $\eta_0(\lambda B) \rho_{01}(a_{00'}, a_{01}, a_{0'1}) = \eta_1 \eta B(\varphi_0(\rho_{01}), (\psi(\rho_0, a_{00'}), a_{01}, a_{0'1}))$
— $\eta_1(\lambda B) \rho_{01}(a_{01}, a_{01'}, a_{11'}) = \eta_1 \eta B(\varphi_1(\rho_{01}), (a_{01}, a_{01'}, \psi(\rho_1, a_{11'})))$
— $\eta_1 \eta(\lambda B) \rho_{012}(a_{01}, a_{02}, a_{12}) = \eta_1 \eta B(\rho_{123}, (a_{01}, a_{02}, a_{12}))$

with the following auxiliary definitions

— $\varphi(\rho) : \eta\Gamma\, \rho\, \rho$ (reflexivity)
— $\varphi_0(\rho_{01}) : \eta_1 \eta\Gamma\, \varphi(\rho_0)\, \rho_{01}\, \rho_{01}$ (degenerate triangle where one edge is the reflexivity)
— $\varphi_1(\rho_{01}) : \eta_1 \eta\Gamma\, \rho_{01}\, \rho_{01}\, \varphi(\rho_1)$ (idem)
— $\varphi'(\rho) : \eta_1 \eta\Gamma\, \varphi(\rho)\, \varphi(\rho)\, \varphi(\rho)$ (degenerate triangle where all three edges are reflexivity)
— $\psi(\rho, a_{ij}) : \eta A\, \varphi(\rho)\, a_i\, a_j$ (remember that $a_{ij} : \eta(A\rho)\, a_i\, a_j$).

The auxiliary definitions are easily derivable from the Kan completion operations of Γ up to level 3 (tetrahedron completion), and those of A.

Substitution does not commute with λ-abstraction for reasons similar to what is explained in Section 2. Nevertheless, the type level β-reduction property can be derived:

$$\mathsf{App}((\lambda B)\sigma, a) = B(\sigma, a).$$

5.6. *Interpretation of the product*

The goal of this section is to define the interpretation of $\mathsf{Fun}\, A\, F$, given a context Γ, a type $A : Ty(\Gamma)$ and a type family $F : Fam(\Gamma, A)$. We first deal with the case when there is no ambient context Γ and define a setoid $\mathsf{Fun}\, A\, F : U$ given a setoid $A : U$ and a family of setoids F indexed by A. Once we establish that this morphism preserves isomorphisms and triangles, we can extend the definition of $\mathsf{Fun}\, A\, F$ in a context Γ.

Consider $A : U$ and F a context morphism from A (seen as a context) to U. The type of dependent functions from A to F does not always form a setoid: reflexivity fails for functions that do not map equal objects of A to equal images in F. The obvious fix is to consider only functions that respect equality.

Lemma 5.17 (product of setoids). Given A a setoid and F a setoid family indexed by A, there exists a setoid $\mathsf{Fun}\, A\, F : U$ such that:

— $\mathsf{Fun}\, A\, F = \Sigma(f : \Pi a : A.\, Fa).\, \Pi a_0, a_1 : A. \Pi a_{01} : \eta A\, a_0\, a_1. \eta F a_{01}\, (f\, a_0)\, (f\, a_1)$,
— $\eta(\mathsf{Fun}\, A\, F)\, (f_0, f_0')\, (f_1, f_1')$ holds when $\eta F a_{01}\, (f_0\, a_0)\, (f_1\, a_1)$ for all $a_0, a_1 : A$ and $a_{01} : \eta A\, a_0\, a_1$.

Proof. Completing the setoid definition is straightforward. The most noticeable fact is that level 1 composition derives from the second component of the elements of $\mathsf{Fun}\, A\, F$. □

Lemma 5.18 (isomorphic products). Given an isomorphism $A_{01} : \eta U\, A_0\, A_1$ and a setoid family isomorphism F_{01} (with the usual indexing conventions), there exists an isomorphism

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

$\eta\mathsf{Fun}\, A_{01}\, F_{01}$ between $\mathsf{Fun}\, A_0\, F_0$ and $\mathsf{Fun}\, A_1\, F_1$ such that

$$\eta\mathsf{Fun}\, A_{01}\, F_{01}\, (f_0, f_0')\, (f_1, f_1')$$

produces an object of $F_{01} a_{01}\, (f_0\, a_0)\, (f_1\, a_1)$ for all $a_i : A_i$ and $a_{01} : A_{01}\, a_0\, a_1$.

Proof. We refer to the formal development. □

Lemma 5.19. Given three isomorphisms A_{01}, A_{02}, A_{12} and $A_{012} : A_{01} \leftrightarrow A_{02} \leftrightarrow A_{12}$ (a triangle of U), and a setoid family isomorphism triangle F_{012} between F_{01}, F_{02} and F_{12}, then there exists

$$\eta_1\eta\mathsf{Fun}\, A_{012}\, F_{012} : \mathsf{Fun}\, A_{01}\, F_{01} \leftrightarrow \mathsf{Fun}\, A_{02}\, F_{02} \leftrightarrow \mathsf{Fun}\, A_{12}\, F_{12}.$$

Proof. We refer to the formal development. □

Definition 5.20 (product). Given a context Γ, a type $A : Ty(\Gamma)$ and a type family $F : Fam(\Gamma, A)$, we define the type $\mathsf{Fun}\, A\, F : Ty(\Gamma)$ by:

— $(\mathsf{Fun}\, A\, F)\rho = \mathsf{Fun}\ (A\rho)\ (F\rho)$
— $\eta(\mathsf{Fun}\, A\, F)\rho_{01} = \eta\mathsf{Fun}\ (\eta A\rho_{01})\ (\eta F\rho_{01})$
— $\eta_1\eta(\mathsf{Fun}\, A\, F)\rho_{012} = \eta_1\eta\mathsf{Fun}\ (\eta_1\eta A\rho_{012})\ (\eta_1\eta F\eta_{012})$

This definition interprets the rules

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash F : (A)\mathsf{Type}}{\Gamma \vdash \mathsf{Fun}\, A\, F} \qquad (\mathsf{Fun}\, A\, F)\sigma = \mathsf{Fun}\, A\sigma\, F\sigma.$$

As in the case of type level application, the term level application is straightforward:

$$\mathsf{app}(w, u)\rho = \pi_1(w\rho)\, u\rho \qquad \eta\mathsf{app}(w, u)\rho_{01} = \eta w\rho_{01}\, \eta u\rho_{01}$$

where π_1 is the first projection of Σ-types.

The term level λ-abstraction is more interesting: one obviously defines the first component of the Σ-type by

$$\pi_1((\lambda b)\rho)a = b(\rho, a),$$

but the definition of the second component requires the Kan completion operations of the domain and co-domain types. The level 2 part of λb is easy:

$$\eta(\lambda b)\rho_{01} a_{01} = \eta b(\rho_{01}, a_{01}).$$

These definitions interpret the following typing rules:

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

λ-expressions.

5.7. *Interpretation of sum types*

The definition of sum types follows the same scheme as for the product types. It is nonetheless more straightforward, since the dependent sum of a setoid with a family of setoids does form a setoid. An isomorphism between two sum types is defined as a pair of

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

an isomorphism between the first components, and a type family isomorphism between the second components, and similarly for the triangles.

Given a setoid A and a setoid family F in A, the setoid $\mathsf{Sum}\, A\, F : U$ is defined as:

— $\mathsf{Sum}\, A\, F = \Sigma(a : A).Fa$
— $\eta(\mathsf{Sum}\, A\, F)\,(a_0, b_0)\,(a_1, b_1) = \Sigma(a_{01} : \eta A\, a_0\, a_1).\eta F\, a_{01}\, b_0\, b_1$

The isomorphism $\eta\mathsf{Sum}\, A_{01}\, F_{01}$ between $\mathsf{Sum}\, A_0\, F_0$ and $\mathsf{Sum}\, A_1\, F_1$ is defined as:

$$\eta\mathsf{Sum}\, A_{01}\, F_{01}\,(a_0, b_0)\,(a_1, b_1) = \Sigma(a_{01} : A_{01}\, a_0\, a_1).F_{01} a_{01}\, b_0\, b_1$$

All other requirements are fulfilled without surprise.

Last auxiliary definition is the triangle

$$\eta_1\eta\mathsf{Sum}\, A_{012}\, F_{012} : \mathsf{Sum}\, A_{01}\, F_{01} \leftrightarrow \mathsf{Sum}\, A_{02}\, F_{02} \leftrightarrow \mathsf{Sum}\, A_{12}\, F_{12}$$

given two triangles A_{012} and F_{012}.

Definition 5.21 (sum). Given a context Γ, a type $A : Ty(\Gamma)$ and a type family $F : Fam(\Gamma, A)$, we define $\mathsf{Sum}\, A\, F : Ty(\Gamma)$ as:

— $(\mathsf{Sum}\, A\, F)\rho = \mathsf{Sum}\,(A\rho)\,(F\rho)$
— $\eta(\mathsf{Sum}\, A\, F)\rho_{01} = \eta\mathsf{Sum}\ (\eta A\, \rho_{01})\ (\eta F \rho_{01})$
— $\eta_1\eta(\mathsf{Sum}\, A\, F)\rho_{012} = \eta_1\eta\mathsf{Sum}\ (\eta_1\eta A\rho_{012})\ (\eta_1\eta F \rho_{012})$.

The constructors and projections of sum types are defined by

$$\begin{array}{ccc} (a,b)\rho = (a\rho, b\rho) & \mathsf{p}c\rho = \pi_1(c\rho) & \mathsf{q}c\rho = \pi_2(c\rho) \\ \eta(a,b)\rho_{01} = (\eta a\rho_{01}, \eta b\rho_{01}) & \eta\mathsf{p}c\rho_{01} = \pi_1(\eta c\rho) & \eta\mathsf{q}c\rho_{01} = \pi_2(\eta c\rho). \end{array}$$

The following definitional equalities hold:

$$(\mathsf{Sum}\, A\, F)\sigma = \mathsf{Sum}\, A\sigma\, F\sigma \qquad (a,b)\sigma = (a\sigma, b\sigma) \qquad (\mathsf{p}c)\sigma = \mathsf{p}(c\sigma) \qquad (\mathsf{q}c)\sigma = \mathsf{q}(c\sigma).$$

Our model validates both equalities $\mathsf{p}(a,b) = a$ and $\mathsf{q}(a,b) = b$. This is not the case for the model described in Hofmann (1994) (which on the other hand validates the rule of substitution under abstraction, but it is not clear if this model can be formulated in an internal way). Other attempts to explain function extensionality (Altenkirch 1999; Altenkirch *et al.* 2007) use extensions of type theory. Our model is close to Erik Palmgren's representation of Bishop sets in dependent type theory (Palmgren 2012), but we have a different representation of function spaces which interprets more definitional equalities.

6. Applications of the model

We have a direct representation of $(\times) : U \times U \to U$ and $(\to) : U \times U \to U$. For instance $A \to B$ is the type of extensional functions between A and B. If $P : \eta U\ P_0\ P_1$ and $Q : \eta U\ Q_0\ Q_1$ then $P \to Q$ represents the relation $(P \to Q)\ f_0\ f_1$ which holds exactly when $P\ x_0\ x_1$ implies $Q\ (f_0\ x_0)\ (f_1\ x_1)$. This is then a graph of an isomorphism between the sets $P_0 \to Q_0$ and $P_1 \to Q_1$. We can define as well the operations $(\leftrightarrow) : U \times U \to U$.

All the applications we present have been formally verified in the system Coq V8.4.

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

6.1. *A small type of propositions*

This subsection can be seen as a generalization of Russell's work on implication (Russell 1906). We assume that the type theory we are working with has at least two universes $\mathsf{Type}_0, \mathsf{Type}_1$ (as introduced in Martin-Löf (1973)). We define U to be Type_1.

We define $\Omega = \mathsf{Type}_0$ and $\eta\Omega\ X_0\ X_1$ to be the type $X_0 \leftrightarrow X_1$.

We define a semisimplicial map $T : \Omega \to U$ by taking

$$T\ X = X,\ \eta T\ h\ x_0\ x_1 = N_1$$

where N_1 is the unit type. This interprets the rule

$$\frac{\Gamma \vdash a : \Omega}{\Gamma \vdash T\ a}$$

If X is Kan semisimplicial set we define $eq_X : X \times X \to \Omega$. We take $eq_X\ a\ u$ to be the type $\eta\ X\ a\ u$. If $P : \eta U\ X\ Y$ and we have $P\ a\ b$ and $P\ u\ v$ then $eq_X\ a\ u$ and $eq_Y\ b\ v$ are logically equivalent. We define then $Eq_X\ a\ u$ to be the type $T\ (eq_X\ a\ u)$.

It is then possible to show that

$$\prod_{a:X} \prod_{u:X} Eq_X\ a\ u$$

is provable in this model if, and only if, any two elements of X are related by the equality relation ηX. Furthermore, the type Ω satisfies the following weak form of univalence.

Proposition 6.1. In the model, the following type is inhabited

$$\prod_{a:\Omega} \prod_{u:\Omega} (T\ a \leftrightarrow T\ u) \to Eq_\Omega\ a\ u$$

This model interprets also the operation of *quotient*. If X is a type and we have a relation $R : X \times X \to \Omega$ which is an equivalence relation in the model then it is possible to define a new type X/R with an operation $a \longmapsto [a],\ X \to X/R$, such that $\mathsf{Eq}_{X/R}\ [a]\ [u]$ is equal to $R\ a\ u$.

We interpret existential quantification in the following way. The rules are

$$\frac{\Gamma \vdash A\ \mathsf{Type}_0 \qquad \Gamma \vdash \varphi : A \to \Omega}{\Gamma \vdash \exists\varphi : \Omega}$$

with introduction rule

$$\frac{\Gamma \vdash A\ \mathsf{Type}_0 \qquad \Gamma \vdash \varphi : A \to \Omega \qquad \Gamma \vdash a : A \qquad \Gamma \vdash p : T(\mathsf{app}(\varphi, a))}{\Gamma \vdash (a, p) : T(\exists\varphi)}$$

and elimination rule

$$\frac{\Gamma \vdash u : T(\exists\varphi) \qquad \Gamma \vdash \psi : \Omega \qquad \Gamma.A \vdash v : T(\mathsf{app}(\varphi\mathsf{p}, \mathsf{q})) \to T\psi\mathsf{p}}{\Gamma \vdash E\ u\ v : T(\psi)}$$

and computation rule $E\ (a, p)\ v = \mathsf{app}(va, p)$.

Here is the interpretation of $\exists\varphi$. If $\rho : \Gamma[0]$ we define $(\exists\varphi)\rho$ to be the set of pairs u, p with $u : A\rho$ and $p : \mathsf{app}(\varphi\rho, u)$. If $\alpha : \Gamma[1]$ with $d_i\alpha = \rho_i$ we have to show the logical equivalence of $(\exists\varphi)\rho_0$ and $(\exists\varphi)\rho_1$. This follows from the fact that $A\alpha$ is a relation

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

between $A\rho_0$ and $A\rho_1$ satisfying the Kan condition and that $\varphi\alpha$ shows that $\mathsf{app}(\varphi\rho_0, u_0)$ and $\mathsf{app}(\varphi\rho_1, u_1)$ are logically equivalent if $u_0 : A\rho_0$ and $u_1 : A\rho_1$ are related by $A\alpha$.

6.2. *Isomorphisms of setoids*

Using a notation with variables for readability, our model interprets contexts of the form $X : U$ or $X : U, Y : U$ with variable ranging over small Kan semisimplicial types. For instance, we can interpret the judgment

$$X : U \vdash (X \to X) \times X$$

or

$$X : U \vdash \Sigma(f, a) : (X \to X) \times X.Eq_X\ (f\ a)\ a$$

which intuitively represents the structure of having an endofunction with a fixpoint over a set X. If we have such a judgment $X : U \vdash T(X)$ we can use our model and compute for any given set A a corresponding set $T(A)$. In this model $X : U \vdash T(X)$ is interpreted by a function $U \to U$. We can thus use this interpretation to transform any graph of an isomorphism $P : \eta U\ A\ B$ between two sets A and B to a graph of an isomorphism between $T(A)$ and $T(B)$. In particular, in a case like

$$X : U \vdash (X \to X) \times X$$

this allows us to transport any structure on A to a structure on B, and in a case like

$$X : U \vdash \Sigma(f, a) : (X \to X) \times X.Eq_X\ (f\ a)\ a$$

this shows that any proof of a property on a structure on A (to be a fixpoint) can be transported to a proof of the corresponding property on the isomorphic structure B. We can cover in this way examples similar to the ones in Licata and Harper (2012) but also with computations on open terms.

We have another stronger form of univalence in this model, which transforms any isomorphism between two sets to a proof that these two sets are equal.

Proposition 6.2. If $f : A \to B$ is an isomorphism between the (Bishop) set A and the (Bishop) set B, the relation $P(a, b)$ defined by $\eta B\ (f\ a)\ b$ is the graph of the isomorphism and we have that $P : U[1]$.

7. The semisimplicial set model

Before presenting the universe of Kan semisimplicial sets, we describe the (simpler) semisimplicial set model of type theory. This model justifies all the rules of Figure 2.

We let Δ_{mon} be the category of objects of the form $[n]$ and the morphism are injective monotone maps. We have an inclusion $i : \Delta_{mon} \to \Delta$. A *semisimplicial set* is a presheaf on Δ_{mon} i.e. a functor in $[\Delta_{mon}^{op}, \mathsf{Set}]$.

We define a semisimplicial set W, which represents the universe of (small) semisimplicial sets. An element of $\mathsf{W}[n]$ is a family of sets Af indexed by $f : [m] \to [n]$ injective, with maps $Af \to Afg$, $u \longmapsto ug$ for $g : [p] \to [m]$ injective such that $u = u1 : Af$ and

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

$ugh = (ug)h : Afgh$. If $A : \mathsf{W}[n]$ and $g : [m] \to [n]$ is injective we have $Ag : \mathsf{W}[m]$ by $(Ag)h = A(gh)$. If $A : \mathsf{W}[n]$ we may write the set $A1$ simply as A.

A context Γ is interpreted by a semisimplicial set, so we have a collection of sets $\Gamma[n]$ and functions $\Gamma[n] \to \Gamma[m]$, $\rho \longmapsto \rho f$ for any $f : [m] \to [n]$ injective, with $\rho 1 = \rho : \Gamma[n]$ and $\rho fg = \rho(fg) : \Gamma[p]$ if $g : [p] \to [m]$ is injective. A substitution $\sigma : \Delta \to \Gamma$ is interpreted by a function $\sigma\rho : \Gamma[n]$ for $\rho : \Delta[n]$ in such a way that $(\sigma\rho)f = \sigma(\rho f) : \Gamma[m]$ if $f : [m] \to [n]$ is injective.

A judgment $\Gamma \vdash A$ will be interpreted by giving $A\rho : \mathsf{W}[n]$ for any $\rho : \Gamma[n]$ in such a way that $A\rho f = A(\rho f)$ if $f : [m] \to [n]$ is injective. If $\Gamma \vdash A$ we define $(\Gamma.A)[n]$ to be the set of pairs ρ, u with $\rho : \Gamma[n]$ and $u : A\rho$ and $(\rho, u)f = (\rho f, uf)$ if $f : [m] \to [n]$ is injective. A judgment $\Gamma \vdash F : (A)\mathsf{Type}$ is interpreted by giving $F\rho : (A\rho)\mathsf{Type}$ for any $\rho : \Gamma[n]$ in such a way that $\mathsf{App}(F\rho f, uf) = \mathsf{App}(F\rho, u)f : \mathsf{W}[m]$ for any $u : A\rho$ and any $f : [m] \to [n]$ injective.

A judgment $\Gamma \vdash t : A$ is interpreted by giving an element $t\rho : A\rho$ for each $\rho : \Gamma[n]$ in such a way that $t\rho f = t(\rho f) : A\rho f$ if $f : [m] \to [n]$ is injective.

8. A Universe of Kan semisimplicial sets

8.1. *General lemmas about semisimplicial sets*

We let $I, J, K, \ldots$ denote nonempty finite linear orders. If I is such a nonempty finite linear order, it is isomorphic exactly to a unique $[n]$ in a unique way. We define $\mathsf{W}(I)$ to be $\mathsf{W}[n]$. If P is in $\mathsf{W}(I)$ and $f : J \to I$ is an injection it corresponds to exactly one injection $g : [m] \to [n]$ and we define Pf to be Pg and the map $u \longmapsto uf$, $P \to Pf$ to be the map $u \longmapsto ug$. If f is an inclusion, we may write $P(J)$ instead of Pf and similarly $u(J)$ instead of uf. If a is an element of I and f is the inclusion $(I - a) \to I$ we may write $\partial_a P$ instead of Pf and similarly $\partial_a u$ instead of uf if $u : \mathsf{W}(I)$. Note that we leave I implicit as it can be inferred from the context $u : \mathsf{W}(I)$. This deviates from the usual notation for face maps but is more convenient for what follows. Note that the semisimplicial identities become $\partial_a \partial_b = \partial_b \partial_a$ for $a \neq b$ elements of I. We will sometimes decompose a linear order and write e.g. $I = a, L, M$; by convention, comma binds weaker than minus, so $L - b, M$ is $(L - b), M$.

For instance if $P : \mathsf{W}[1]$ we have three sets $P = P1$ and $P(0)$ and $P(1)$ and two maps $u \longmapsto u(0)$, $P \to P(0)$ and $u \longmapsto u(1)$, $P \to P(1)$. We also have $P(0) = \partial_1 P$ and $P(1) = \partial_0 P$.

If $J \subseteq I$ and a is an element of J, we can define $P(\Lambda^a(J))$ as the set of compatible families $u_b : \partial_b P(J)$ for $b \neq a$, i.e. such that $\partial_c u_b = \partial_b u_c$ for b, c distinct from a. We have a canonical map $P(J) \to P(\Lambda^a(J))$. We say that P is in $\mathsf{V}(I)$ iff each canonical map $P(J) \to P(\Lambda^a(J))$ has a section.

If L is a nonempty subset of I and $P : \mathsf{W}(I)$ we define a L-compatible family of P to be a family of elements $u_b : P(I - b)$ for each b in L such that $\partial_c u_b = \partial_b u_c$ for all b and c in L. We say that P has *compositions* iff for any a not in L and $a, L, M \subseteq I$ we have an operation $\mathsf{comp}\ u : P(L, M)$ which takes a L-compatible family $u_b : P(a, L - b, M)$ and produces an element $u_a = \mathsf{comp}\ u : P(L, M)$ satisfying $\partial_b u_a = \partial_a u_b$ for all b in L. Furthermore, all these

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

operations should be compatible, in the sense that we have $\partial_c(\mathsf{comp}\ u) = \mathsf{comp}\ (\partial_c \circ u)$ for all c in M, where we write $\partial_c \circ u$ for the family $\partial_c u_b : P(a, L-b, M-c)$ with b in L.

A stronger notion is to have *extension operations*. Given a compatible family of elements $u_b : P(L-b, a, M)$ these operations produce an element $u = \mathsf{Comp}(u_b)$ in $P(L, a, M)$. This element should satisfy $\partial_b u = u_b : P(L-b, a, M)$ for all b in L, and we should have $\partial_c u = \mathsf{Comp}(\partial_c u_b) : P(L, a, M-c)$ for all c in M. If P has extension operations, then it also has composition operations by defining $\mathsf{comp} = \partial_a \mathsf{Comp}$. To have extension operations for P in $\mathsf{W}(I)$ is a priori a stronger condition than being in $V(I)$, by taking $L = I - a$. However, the two properties are actually equivalent.

Lemma 8.1. If $P : \mathsf{V}(I)$ then P has extension operations, and hence has compositions.

Proof. Given a compatible family $u_b : P(L-b, a, M),\ b \in L$, we define the extension $u : P(L, a, M)$ by induction on the cardinality of M. By induction we have defined the extensions $u_c : P(L, a, M-c)$ of all the $\partial_c u_b$'s for each c in M in a compatible way. We use then the Kan extension operation to define $u : P(L, a, M)$ such that $\partial_b u = u_b$ for all b in L and $\partial_c u = u_c$ for all c in M. □

Here is a special case of Lemma 8.1.

Corollary 8.2. If $P : \mathsf{V}(I)$ and a, b are two elements of I there exists an operation $\mathsf{Ext}\ u : P(a, b, M)$ which takes an element $u : P(b, M)$ and satisfies $\partial_a \mathsf{Ext}\ u = u$ and $\partial_c \mathsf{Ext}\ u = \mathsf{Ext}\ \partial_c u$ for all c in M.

To have composition operations can be seen as a generalization of the notion of partial equivalence relations, while to have extension operations generalizes the notion of equivalence relations.

One basic example of composition is the composition of two binary relations. The next result generalizes this notion. We say that a semisimplicial set X has composition iff for any nonempty finite linear order I and any a not in L and $a, L \subseteq I$ we have an operation $\mathsf{comp}\ u : X(I-a)$ which takes a L-compatible family $u_b : X(I-b)$ and produces an element $u_a = \mathsf{comp}\ u : X(I-a)$ satisfying $\partial_b u_a = \partial_a u_b$ for all b in L. Furthermore, all these operations should be compatible, in the sense that we have $\partial_c(\mathsf{comp}\ u) = \mathsf{comp}\ (\partial_c \circ u)$ for all c in $I - (a, L)$, where we write $\partial_c \circ u$ for the family $\partial_c u_b : X(I - (b, c))$ with b in L.

Lemma 8.3. The semisimplicial set W has compositions.

Proof. Assume given a, L subset of I and a compatible family $Q_b : \mathsf{W}(I-b)$ for b in L. We define $Q_a = \mathsf{comp}\ (Q_b) : \mathsf{W}(I-a)$. An element of Q_a is a compatible family $v = (u_b)$ where $u_b : Q_b$ for b in L, i.e. a family satisfying $\partial_c u_b = \partial_b u_c$ for all b, c in L. We define then $\partial_c v$ to be the family $\partial_c u_b,\ b : L$ for c in $I - (L, a)$ and $\partial_b v$ to be $\partial_a u_b$ for b in L. □

In the case, $n = 2$ and $I = J = 0, 1, 2$ and $a = 1$ we get back the usual notion of composition of relations.

Lemma 8.4. The semisimplicial set V is closed under the compositions of W.

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

Proof. To simplify the notation, we describe the argument in the case where we compose $P : \mathsf{V}(01M)$ and $Q : \mathsf{V}(02M)$ obtaining a relation $R : \mathsf{W}(12M)$. An element in R is a pair $u(01M), v(02M)$ such that $u(0M) = v(0M)$ and we have

$$\partial_1(u, v) = v(2M) \qquad \partial_2(u, v) = u(1M)$$

and $\partial_b(u, v) = \partial_b u, \partial_b v$ for all b in M. We show that R is in $V(12M)$. We have three cases.

The first case is if we have a in M and we have a compatible family consisting of an element (u_b, v_b) in $R(12M_b)$ for all b in $M_a = M - a$ and we have an element v_0 in $Q(2M)$ and an element u_0 in $P(1M)$. We have a compatible family u_b in $P(01M_b)$ for $b \neq a$ and u_0 in $P(1M)$. By Lemma 8.1 we can find $u : P(01M)$ such that $\partial_0 u = u_0$ and $\partial_b u = u_b$ for all b in $M - a$. The family consisting of $v_b : Q(02M_b)$ for b in $M - a$ and $v_0 : Q(2M)$ and $u(0M) : P(0M) = Q(0M)$ is then compatible and since $Q : \mathsf{V}(02M)$ we can find $v : Q(02M)$ such that $\partial_b v = v_b$ for b in $M - a$ and $\partial_0 v = v_0$ and $v(0M) = u(0M)$. The element u, v in $R(12M)$ is the required filling.

The second case is if we have a compatible family consisting of an element u_b, v_b in $R(12M_b)$ for each b in M and an element u_0 in $P(1M)$. Since the family $u_b : P(01M_b)$ and $u_0 : P(1M)$ is compatible and $P : \mathsf{V}(01M)$ we find a filling $u : P(01M)$. We have then a compatible family $v_b : Q(02M_b)$ and $u(0M) : P(0M) = Q(0M)$. Since $Q : \mathsf{V}(02M)$ we have a filling $v : Q(02M)$. The element u, v in $R(12M)$ is the required filling.

The third case is if we have a compatible family consisting of an element u_b, v_b in $R(12M_b)$ for each b in M and an element v_0 in $Q(2M)$. This case is similar to the second case. □

The inclusion $i : \Delta_{mon} \to \Delta$ defines a functor $i^* : [\Delta^{op}, \mathsf{Set}] \to [\Delta^{op}_{mon}, \mathsf{Set}]$ which has a right adjoint $i_R : [\Delta^{op}_{mon}, \mathsf{Set}] \to [\Delta^{op}, \mathsf{Set}]$. If X is a semisimplicial set, $i_R X[n]$ is the set of all natural transformations $i^*\Delta^n \to X$, where Δ^n is the simplicial set represented by $[n]$. More concretely, $i_R X[n]$ is the set of families $u_f : X[m]$ for $f : [m] \to [n]$ monotone (but not necessarily strictly monotone) such that $u_f g = u_{fg} : X[k]$ whenever $g : [k] \to [m]$ is strictly monotone.

If X is a semisimplicial set, then each restriction $X(I)$ defines an element of $\mathsf{W}(I)$. A *Kan* semisimplicial set is a semisimplicial set Y such that each restriction $Y(I)$ is in $\mathsf{V}(I)$.

The next lemma generalizes what happens in Section 4. The basic case is to define the degenerates of a composition of two lines as a composition of three triangles.

Lemma 8.5. If the semisimplicial set X has compositions then $i_R X$ is a Kan simplicial set.

Proof. To simplify the notations, we consider only the case where we have compatible $u_2 : Y(01)$ and $u_1 : Y(02)$ and we explain how to build the extension $u : Y(012)$. We give an algorithm for computing $uf : X(I)$ for any map $f : I \to 012$ such that we have $ufi = u(fi)$ for strictly monotone $i : J \to I$. We let $z_1 < \cdots < z_n < a_1 < \cdots < a_p < b_1 < \cdots < b_q$ be I, with $f(z_i) = 0$ and $f(a_j) = 1$ and $f(b_k) = 2$. The definition is by induction on $p + q$. We first treat the case $p = 0$ or $q = 0$ separate; if $p = 0$, i.e. 1 is not in the image of f, we can write $f = \partial_1 f'$ for a uniquely determined $f' : I \to 02$, and define $uf = u_1 f'$. Note that by the uniqueness of the decomposition and $fi = \partial_1 f' i$ we have $u(fi) = u_1 f' i$, and thus $u(fi) = ufi$. Similarly, if $q = 0$, we write $f = \partial_2 f'$ and set $uf = u_2 f'$. Note

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

that if both $p = q = 0$, we have $f = \partial_1\partial_2 f''$ for some f'', and then $\partial_1 u_2 = \partial_2 u_1$ yields $u_2(\partial_1 f'') = u_1(\partial_2 f'')$, and hence both definitions of uf coincide.

In case both p and q are $\neq 0$, we consider the linear order J obtained by adding one element z exactly before a_1. Let $f' : J \to 012$ be the extension of f defined by $f'(z) = 0$. Let f_1 be the restriction of f' on $J_1 = J - a_p$ and f_2 be the restriction of f' on $J_2 = J - b_q$. By induction hypothesis, the elements $uf_1 : X(J_1)$ and $uf_2 : X(J_2)$ are defined and are compatible since $\partial_z(uf_1) = u(f_1\partial_z) = u(f_2\partial_z) = \partial_z(uf_2)$. We define uf to be their composition. In order to check $(uf)i = u(fi)$ for injective i, we distinguish cases: in case 1 or 2 are not in the image of i, say 1, we have that $i = \partial_1 i'$ and thus $(uf)i = (\partial_z(uf_1))i' = u(f_1\partial_z i') = u(f\partial_1 i') = u(fi)$. Otherwise, we can assume $i = \partial_0$; by the compatibility condition for compositions, $\partial_0(uf)$ is the composition of $\partial_0(uf_v)$ (for $v = 0, 1$), and $\partial_0(uf_v) = u(f_v\partial_0)$; using $f_v\partial_0 = (f\partial_0)_v$ yields that this composition is $u(f\partial_0)$ by definition.

It remains to check $\partial_1 u = u_1$ and $\partial_2 u = u_2$. But $(\partial_1 u)f = u(\partial_1 f) = u_1 f$, and similarly for the other face. □

8.2. *Interpretation of the universe*

We define U to be the semisimplicial set $i^* i_R \mathsf{V}$. So an element of $U[n]$ is a natural transformation $i^*\Delta^n \to \mathsf{V}$.

Theorem 8.6. *U is a Kan semisimplicial set.*

Proof. This follows from Lemmas 8.4 and 8.5. □

To give an element of $U[n]$ is to give a family of sets $P = (P_f)$ indexed by $f : [m] \to [n]$ together with restriction maps $P_f \to P_{fg}$, $u \longmapsto ug$ for $g : [p] \to [m]$ injective, satisfying $u1 = u$ and $(ug)h = u(gh)$. Furthermore, for any f the family P_{fg}, $g : [p] \to [m]$ defines an element Pf of $\mathsf{V}[m]$. We write $u : P$ for $u : P_1$. There is a canonical map $i^*U \to \mathsf{V}$.

8.3. *Equivalence and equality*

We explain how to use our interpretation to transform any equivalence $\varphi : A \to B$ between two Kan semisimplicial sets to a proof of equality of A and B in U. More generally we explain how to transform any map $\varphi : A \to B$ between two semisimplicial sets into an element $E(\varphi)$ of $i^* i_R \mathsf{W}[1]$. This element $E(\varphi)$ can be thought of as the graph of the map φ. If A and B are Kan semisimplicial and φ is an equivalence than $E(\varphi)$ is in $i^* i_R \mathsf{V}[1] = U[1]$.

We have to define for each $f : [m] \to [1]$ a set $E(\varphi)f$ together with restriction maps $E(\varphi)f \to E(\varphi)fg$ if $g : [p] \to [m]$ is injective. An element $f : I \to [1]$ is either the constant 0, or the constant 1 or is 0 on an initial segment I_0 and 1 on $I - I_0$. We define $E(\varphi)f$ as follows:

1. if f is the constant function 0 then $E(\varphi)f = A(I)$,
2. if f is the constant function 1 then it is $E(\varphi)f = B(I)$,

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

3. if f is 0 on I_0 and 1 on $I - I_0$ then $E(\varphi)f$ is the set of pairs (u, v) with u in $A(I_0)$ and v in $B(I)$ such that $v(I_0) = \varphi u$.

If we have $g : J \to I$ which is injective and w in $E(\varphi)f$ we define wg in $E(\varphi)fg$. If gf is 0 we have $w = u$ in $A(I)$ or $w = (u, v)$ with u in $A(I_0)$ and we take $wg = u(J)$. If gf is 1 we have $w = v$ in $B(I)$ or $w = (u, v)$ with u in $A(I_0)$ and we take $wg = v(J)$. Otherwise, we can write $g = g_0 + g_1 : J_0 + J_1 \to I_0 + I_1$ with $I_1 = I - I_0$, $J_1 = J - J_0$ and we take $wg = (u(J_0), v(J))$. We define in this way an element $E(\varphi)$ in $i_R\mathsf{W}[1]$.

Definition 8.7. A Kan semisimplicial set A is *contractible* iff for any finite linear order I, given any compatible family of elements $u_b : A(I - b)$ for b in I there exists $u : A(I)$ such that $u(I - b) = u_b$ for all b in I.

This definition implies, in the case where I is empty, that $A[0]$ is nonempty if A is contractible.

Given a map $\varphi : A \to B$ and b in $B[0]$ we define the *homotopy fiber* at b to be the following semisimplicial set F: an element of $F(I)$ is a pair α, β where $\alpha : A(I)$ and $\beta : B(I, u)$ with $\beta(I) = \varphi\alpha$ and $\beta(u) = b$ and $i < u$ for all i in I. A map $\varphi : A \to B$ between Kan semisimplicial set is an equivalence if all its homotopy fibers are contractible.

The following result can be seen as a generalization of Proposition 6.2.

Proposition 8.8. If A and B are Kan semisimplicial set and $\varphi : A \to B$ is an equivalence then $E(\varphi)$ is in $i^* i_R\mathsf{V}[1] = U[1]$.

Proof. We show that any horn in $E(\varphi)f$, $f : [n] \to [1]$ can be filled. If f is the constant function 0 this follows from the fact that A has the Kan filling property. If $f^{-1}(1)$ has more than one element, this follows from the fact that B has the Kan filling property. The remaining case is if $f^{-1}(1)$ is a singleton. For instance, for $n = 3$ we are given $a(0), a(1), a(2)$ in $A[0]$ and $b(3)$ in $B[0]$ and we have $a(i, j)$ in $A[1]$ and $b(i, j, 3)$ in $B[2]$ such that $b(i, j) = \varphi a(i, j)$ for $i < j < 3$. The problem is to find an extension $b(0, 1, 2, 3)$ in $B[3]$ and $a(0, 1, 2)$ in $A[3]$ such that $b(0, 1, 2) = \varphi a(0, 1, 2)$. This follows from the fact that the homotopy fiber at $b(3)$ is contractible and from Definition 8.7. □

9. Conclusion

Using cubical sets instead of simplicial sets, it is possible to give a model of type theory satisfying univalence (Bezem *et al.* 2014). In these two models, the Kan semisimplicial set and cubical set model, the Kan filling operations for the universe and the way to transform an equivalence in an equality between types have very similar justifications. The rough idea is that equalities between types can be seen as graph of 'isomorphisms,' and we can use composition of relations to compose these equalities. Compared to the work on cubical sets, the present work can be carried out internally for truncated levels. However, these partial internal interpretations fail to justify the rule of substitution under an abstraction. The question raised by this work is if it is possible to have an interpretation which justifies this rule as well.

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press

Acknowledgment

The last two authors acknowledge financial support from the ERC: the research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement nr. 247219.

The second author acknowledges financial support from grants from The Ellentuck and The Simonyi Fund.

References

Aczel, P. (1978) The type theoretic interpretation of constructive set theory. *Logic Colloquium* **77** 55–66.

Altenkirch, T. (1999) Extensional equality in intensional type theory. In: *14th Symposium on Logic in Computer Science*, 412–420.

Altenkirch, T., McBride C. and Swierstra, W. (2007) Observational equality, now! In: *PLPV '07: Proceedings of the 2007 Workshop on Programming Languages Meets Program Verification*, ACM 57–68.

Awodey, S. and Warren, M. (2009) Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society* **146** 45–55.

Bezem, M. and Coquand, Th. (2013) A Kripke model for simplicial sets. Preprint.

Bezem, M., Coquand, Th. and Huber, S. (2014) A model of type theory in cubical sets. In: R. Matthes and A. Schubert (eds.) *Proceedings of the 19th International Conference on Types for Proofs and Programs (TYPES 2013)*, Dagstuhl, Germany, 107–128.

Bishop, E. (1967) *Foundations of Constructive Analysis,* Ishi Press International, 2012, reprinted from the original version MacGraw-Hill.

Church, A. (1940) A formulation of the simple theory of types. *Journal of Symbolic Logic* **5** 56–68.

De Bruijn, N.G. (1980) A survey of the project AUTOMATH. In: *H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press 579–606.

Dubucs, J.-P. (1988) Brouwer: Topologie et constructivisme. *Revue d'histoire des Sciences* **41** (2) 133–155.

Gandy, R. (1953) *On Axiomatic Systems in Mathematics and Theories in Physics,* Ph.D. thesis, University of Cambridge.

Gandy, R. (1956) On the axiom of extensionality -Part I. *The Journal of Symbolic Logic* **21** 36–48.

Girard, J.Y. (1971) Une extension de l'interprétation de Gödel à l'analyse et son application à l'élimination des coupures dans l'analyse et la théorie des types. In: J. E. Fenstad (ed.) *Proceedings of the Second Scandinavian Logic Symposium*, North-Holland 63–92.

Goerss, P. J. and Jardine, J. F. (1997) *Simplicial Homotopy Theory*, Birkhauser.

Hofmann, M. (1994) *Extensional Concepts in Intensional Type Theory,* Ph.D. thesis, Edinburgh.

Kapulkin, C., Lumsdaine, P. L. and Voevodsky, V. (2012) The simplicial model of univalent foundations. Preprint, `http://arxiv.org/abs/1211.2851`.

Licata, D. and Harper, R. (2012) Canonicity for 2-dimensional type theory. In: *POPL '12: Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ACM 337–348.

Martin-Löf, P. (1971) Haupsatz for intuitionistic simple type theory. In: *Proceeding of the Fourth International congress for Logic, Methodology, and Philosophy of Science*, Bucharest, 279–290.

Martin-Löf, P. (1972) An intuitionistic theory of types. In: Sambin, G. and Smith, J. (eds.) published in the volume *25 Years of Type Theory*

Martin-Löf, P. (1973) An intuitionistic theory of types: Predicative part. *Logic Colloquium*, 73–118.
Martin-Löf, P. (1975) About models for intuitionistic type theories and the notion of definitional equality. In: *Proceedings of the Third Scandinavian Symposium*, North-Holland.
May, P. (1967) *Simplicial Objects in Algebraic Topology*, Van Norstrand.
Mines, R., Richman, F. and Ruitenburg, W. (1988) *A Course in Constructive Algebra*, Springer.
Palmgren, E. (2012) Proof-relevance of families of setoids and identity in type theory. *Archive for Mathematical Logic* **51** 35–47.
Russell, B. (1906) The theory of implications. *American Journal of Mathematics* **28** (2) 159–202.
Statman, R. (1985) Logical relations and the typed lambda-calculus. *Information and Control* **65** 85–97.
Streicher, T. (2011) A model of type theory in simplicial sets. Unpublished notes available at the author's home page `http://www.mathematik.tu-darmstadt.de/~streicher/sstt.pdf`.
Tait, W. (1967) Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic* **32** 198–212.
Takeuti, G. (1953) On a generalized logic calculus. *Japanese Journal of Mathematics* **23** 39–96.
Troelstra, A. S. and van Dalen, D. (1988) *Constructivism in Mathematics. An Introduction*, volume 2, North-Holland.
Voevodsky, V. (2010) Univalent foundations project. NSF grant application.

https://doi.org/10.1017/S0960129514000504 Published online by Cambridge University Press