Theoretical Computer Science 777 (2019) 184–191

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Canonicity and normalization for dependent type theory

Thierry Coquand

Computer Science Department, University of Gothenburg, Sweden

ARTICLE INFO

Article history:
Received 22 July 2018
Received in revised form 1 January 2019
Accepted 8 January 2019
Available online 15 January 2019

Keywords:
Dependent type theory
Normalisation proof
Higher-order abstract syntax

0. Introduction

We show canonicity and normalization for dependent type theory with a cumulative sequence of universes $U_0 : U_1 \dots$ *with* η-conversion. We give the argument in a constructive set theory CZFu$_{<\omega}$, designed by P. Aczel [2]. We provide a purely algebraic presentation of a canonicity proof, as a way to build new (algebraic) models of type theory. We then present a normalization proof, which is technically more involved, but is based on the same idea. We believe our argument to be a simplification of existing proofs [16,17,1,8], in the sense that we never need to introduce a reduction relation, and the proof theoretic strength of our meta theory is as close as possible to the one of the object theory [2,10].

Let us expand these two points. If we are only interested in *canonicity*, i.e. to prove that a closed Boolean is convertible to 0 or 1, one argument for simple type theory (as presented e.g. in [20]) consists in defining a “reducibility”[1] predicate by induction on the type. For the type of Boolean, it means exactly to be convertible to 0 or 1, and for function types, it means to send a reducible argument to a reducible value. It is then possible to show by induction on the typing relation that any closed term is reducible. In particular, if this term is a Boolean, we obtain canonicity. The problem of extending this argument for a dependent type system with universes is in the definition of what should be the reducibility predicate for universes. It is natural to try an inductive–recursive definition; this was essentially the way it was done in [16], which is an early instance of an inductive–reductive definition. We define when an element of the universe is reducible, and, by induction on this proof, what is the associated reducibility predicate for the type represented by this element. However, there is a difficulty in this approach: it might well be *a priori* that an element is both convertible for instance to the type of Boolean or of a product type, and if this is the case, the previous inductive–recursive definition is ambiguous.

In [16], this problem is solved by considering first a reduction relation, and then showing this reduction relation to be confluent, and defining convertibility as having a common reduct. This does *not* work however when conversion is defined as a *judgement* (as in [17,1]). This is an essential difficulty, and a relatively subtle and complex argument is involved in [1,8]

E-mail address: coquand@chalmers.se.

[1] The terminology for this notion varies: in [13], where it was first introduced, it is called “berechenbarkeit”, which can be translated by “computable”, in [22] it is called “convertibility”, and in [20] it is called “reducibility”.

https://doi.org/10.1016/j.tcs.2019.01.015
0304-3975/

to solve it: one defines first an *untyped* reduction relation and a reducibility *relation*, which is used to establish a confluence property.

The main point of this paper is that this essential difficulty can be solved, in a seemingly magical way, by considering *proof-relevant* reducibility. We define reducibility as a *structure* and not only as a *property*. Such an approach is hinted in the reference [17], but [17] still introduces a reduction relation, and also presents a version of type theory with a restricted form of conversion (no conversion under abstraction, and no η-conversion; this restriction is motivated in [18]).

Even for the base type, reducibility is a structure: the reducibility structure of an element t of Boolean type contains either 0 (if t and 0 are convertible) or 1 (if t and 1 are convertible) and this might a priori contains both 0 and 1. Another advantage of our approach, when defining reducibility in a proof-relevant way, is that the required meta-language is weaker than the one used for a reducibility relation (where one has to do proofs by induction on this reducibility relation).

Yet another aspect that was not satisfactory in previous attempts [1,8] is that it involved essentially a *partial equivalence relation model*. While one expects that this would be needed for a type theory with an extensional equality, this should not be necessary for the present, intensional, version of type theory. This issue disappears here: we only consider *predicates* (that are proof-relevant).

A more minor contribution of this paper is its *algebraic* character. For both canonicity and decidability of conversion, one considers first a general model construction and one obtains then the desired result by instantiating this general construction to the special instance of the initial (term) model, using in both cases only the abstract characteristic property of the initial model.

1. Informal presentation

We first give an informal presentation of the canonicity proof by first giving the rules of type theory and then explaining the reducibility argument.

1.1. Type system

We use conversion as judgements [1]. Note that it is not clear a priori that subject reduction holds.

$$\frac{\Gamma \vdash A : U_n}{\Gamma, x : A \vdash} \qquad \frac{}{() \vdash} \qquad \frac{\Gamma \vdash}{\Gamma \vdash x : A}\ (x{:}A \text{ in } \Gamma)$$

$$\frac{\Gamma \vdash A : U_n \qquad \Gamma, x : A \vdash B : U_n}{\Gamma \vdash \Pi(x : A)B : U_n} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda(x : A)t : \Pi(x : A)B} \qquad \frac{\Gamma \vdash t : \Pi(x : A)B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B(u)}$$

$$\frac{\Gamma \vdash A : U_n}{\Gamma \vdash A : U_m}\ (n \leqslant m) \qquad \frac{}{\Gamma \vdash U_n : U_m}\ (n < m) \qquad \frac{}{\Gamma \vdash N_2 : U_n}$$

The conversion rules are

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash A \mathsf{\ conv\ } B : U_n}{\Gamma \vdash t : B} \qquad \frac{\Gamma \vdash t \mathsf{\ conv\ } u : A \qquad \Gamma \vdash A \mathsf{\ conv\ } B : U_n}{\Gamma \vdash t \mathsf{\ conv\ } u : B}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t \mathsf{\ conv\ } t : A} \qquad \frac{\Gamma \vdash t \mathsf{\ conv\ } v : A \qquad \Gamma \vdash u \mathsf{\ conv\ } v : A}{\Gamma \vdash t \mathsf{\ conv\ } u : A}$$

$$\frac{\Gamma \vdash A \mathsf{\ conv\ } B : U_n}{\Gamma \vdash A \mathsf{\ conv\ } B : U_m}\ (n \leqslant m) \qquad \frac{\Gamma \vdash A_0 \mathsf{\ conv\ } A_1 : U_n \qquad \Gamma, x : A_0 \vdash B_0 \mathsf{\ conv\ } B_1 : U_n}{\Gamma \vdash \Pi(x : A_0)B_0 \mathsf{\ conv\ } \Pi(x : A_1)B_1 : U_n}$$

$$\frac{\Gamma \vdash t \mathsf{\ conv\ } t' : \Pi(x : A)B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u \mathsf{\ conv\ } t'\ u : B(u)} \qquad \frac{\Gamma \vdash t : \Pi(x : A)B \qquad \Gamma \vdash u \mathsf{\ conv\ } u' : A}{\Gamma \vdash t\ u \mathsf{\ conv\ } t\ u' : B(u)}$$

$$\frac{\Gamma, x : A \vdash t : B \qquad \Gamma \vdash u : A}{\Gamma \vdash (\lambda(x : A)t)\ u \mathsf{\ conv\ } t(u) : B(u)}$$

We consider type theory with η-rules

$$\frac{\Gamma \vdash t : \Pi(x : A)B \qquad \Gamma \vdash u : \Pi(x : A)B \qquad \Gamma, x : A \vdash t\ x \mathsf{\ conv\ } u\ x : B}{\Gamma \vdash t \mathsf{\ conv\ } u : \Pi(x : A)B}$$

Finally we add $N_2 : U_0$ with the rules

$$\frac{}{\Gamma \vdash 0 : N_2} \qquad \frac{}{\Gamma \vdash 1 : N_2} \qquad \frac{\Gamma, x : N_2 \vdash C : U_n \qquad \Gamma \vdash a_0 : C(0) \qquad \Gamma \vdash a_1 : C(1)}{\Gamma \vdash \mathsf{brec}\ a_0\ a_1 : \Pi(x : N_2)C}$$

with computation rules $\mathsf{brec}\ a_0\ a_1\ 0 \mathsf{\ conv\ } a_0 : C(0)$ and $\mathsf{brec}\ a_0\ a_1\ 1 \mathsf{\ conv\ } a_1 : C(1)$.

1.2. Reducibility proof

The informal reducibility proof consists of associating to each closed expression a of type theory (treating types and terms equally) an abstract object a' which represents a "proof" that a is reducible. If A is a (closed) type, then A' is a family of sets over the set $\mathsf{Term}(A)$ of closed expressions of type A *modulo conversion*. If a is of type A then a' is an element of the set $A'(a)$.

The metatheory is a (constructive) set theory with a cumulative hierarchy of universes $\mathcal{U}_n$ [2].

This is defined by structural induction on the expression as follows

- $(c\ a)'$ is $c'\ a\ a'$
- $(\lambda(x:A)b)'$ is the function which takes as arguments a closed expression a of type A and an element a' in $A'(a)$ and produces $b'(a,a')$
- $(\Pi(x:A)B)'(w)$ for w closed expression of type $\Pi(x:A)B$ is the set $\Pi(a:\mathsf{Term}(A))(a':A'(a))B'(a,a')(w\ a)$
- $N_2'(t)$ is the set $\{0 \mid t \text{ conv } 0\} \cup \{1 \mid t \text{ conv } 1\}$
- $U_n'(A)$ is the set $\mathsf{Term}(A) \to \mathcal{U}_n$

It can then be shown[2] that if $a:A$ then a' is an element of $A'(a)$ and furthermore that if a conv $b:A$ then $a'=b'$ in $A'(a)=A'(b)$. In particular, if $a:N_2$ then a' is 0 or 1 and we get that a is convertible to 0 or 1.

One feature of this argument is that the required meta theory, here constructive set theory, is known to be of similar strength as the corresponding type theory; for a term involving n universes, the meta theory will need $n+1$ universes [10]. This is to be contrasted with the arguments in [16,1,8] involving induction recursion which is a much stronger principle.

We believe that the mathematical purest way to formulate this argument is an algebraic argument, giving a (generalized) algebraic presentation of type theory. We then use only of the *term* model the fact that it is the *initial* model of type theory. This is what is done in the next section.

2. Model and syntax of dependent type theory with universes

2.1. Cumulative categories with families

We present a slight variation (for universes) of the notion of "category" with families [11].[3] A model is given first by a class of *contexts*. If Γ, Δ are two given contexts we have a set $\Delta \to \Gamma$ of *substitutions* from Δ to Γ. These collections of sets are equipped with operations that satisfy the laws of composition in a category: we have a substitution id in $\Gamma \to \Gamma$ and a composition operator $\sigma\delta$ in $\Theta \to \Gamma$ if δ is in $\Theta \to \Delta$ and σ in $\Delta \to \Gamma$. Furthermore we should have $\sigma\mathsf{id} = \mathsf{id}\sigma = \sigma$ and $(\sigma\delta)\theta = \sigma(\delta\theta)$ if $\theta : \Theta_1 \to \Theta$.

We assume to have a "terminal" context (): for any other context, there is a unique substitution, also written (), in $\Gamma \to ()$. In particular we have $()\sigma = ()$ in $\Delta \to ()$ if σ is in $\Delta \to \Gamma$.

We write $|\Gamma|$ the set of substitutions $() \to \Gamma$.

If Γ is a context we have a cumulative sequence of sets $\mathsf{Type}_n(\Gamma)$ of *types over* Γ at level n (where n is a natural number). If A in $\mathsf{Type}_n(\Gamma)$ and σ in $\Delta \to \Gamma$ we should have $A\sigma$ in $\mathsf{Type}_n(\Delta)$. Furthermore $A\mathsf{id} = A$ and $(A\sigma)\delta = A(\sigma\delta)$. If A in $\mathsf{Type}_n(\Gamma)$ we also have a collection $\mathsf{Elem}(\Gamma, A)$ of *elements of type* A. If a in $\mathsf{Elem}(\Gamma, A)$ and σ in $\Delta \to \Gamma$ we have $a\sigma$ in $\mathsf{Elem}(\Delta, A\sigma)$. Furthermore $a\mathsf{id} = a$ and $(a\sigma)\delta = a(\sigma\delta)$. If A is in $\mathsf{Type}_n()$ we write $|A|$ the set $\mathsf{Elem}((), A)$.

We have a *context extension operation*: if A is in $\mathsf{Type}_n(\Gamma)$ then we can form a new context $\Gamma.A$. Furthermore there is a projection p in $\Gamma.A \to \Gamma$ and a special element q in $\mathsf{Elem}(\Gamma.A, A\mathsf{p})$. If σ is in $\Delta \to \Gamma$ and A in $\mathsf{Type}_n(\Gamma)$ and a in $\mathsf{Elem}(\Delta, A\sigma)$ we have an extension operation (σ, a) in $\Delta \to \Gamma.A$. We should have $\mathsf{p}(\sigma, a) = \sigma$ and $\mathsf{q}(\sigma, a) = a$ and $(\sigma, a)\delta = (\sigma\delta, a\delta)$ and $(\mathsf{p}, \mathsf{q}) = \mathsf{id}$.

A *global* type of level n is given by an element C in $\mathsf{Type}_n()$. We write simply C instead of $C()$ in $\mathsf{Type}_n(\Gamma)$ for () in $\Gamma \to ()$. Given such a global element C, a global element of type C is given by an element c in $\mathsf{Elem}((), C)$. We then write similarly simply c instead of $c()$ in $\mathsf{Elem}(\Gamma, C)$.

Models are sometimes presented by giving a class of special maps (fibrations), where a type is modelled by a fibration and elements by a section of this fibration. In our case, the fibrations are the maps p in $\Gamma.A \to \Gamma$, and the sections of these fibrations correspond exactly to elements in $\mathsf{Elem}(\Gamma, A)$. Any element a in $\mathsf{Elem}(\Gamma, A)$ defines a section $(\mathsf{id}, a) : \Gamma \to \Gamma.A$ and any such section is of this form.

2.2. Dependent product types

A category with families has *product types* if we furthermore have one operation $\Pi\ A\ B$ in $\mathsf{Type}_n(\Gamma)$ for A is in $\mathsf{Type}_n(\Gamma)$ and B is in $\mathsf{Type}_n(\Gamma.A)$. We should have $(\Pi\ A\ B)\sigma = \Pi\ (A\sigma)\ (B\sigma^+)$ where $\sigma^+ = (\sigma\mathsf{p}, \mathsf{q})$. We have an abstraction operation

[2] We prove this statement by induction on the derivation and consider a more general statement involving a context; we don't provide the details in this informal part since this will be covered in the next section.

[3] As emphasized in this reference, these models should be more exactly thought of as *generalized algebraic structures* rather than *categories*; e.g. the initial model is defined up to isomorphism and not up to equivalence). This provides a generalized algebraic notion of model of type theory.

λb in $\mathsf{Elem}(\Gamma, \Pi\ A\ B)$ given b in $\mathsf{Elem}(\Gamma.A, B)$. We have an application operation such that $\mathsf{app}(c, a)$ is in $\mathsf{Elem}(\Gamma, B(\mathsf{id}, a)$ if a is in $\mathsf{Elem}(\Gamma, A)$ and c is in $\mathsf{Elem}(\Gamma, \Pi\ A\ B)$. These operations should satisfy the equations

$$\mathsf{app}(\lambda b, a) = b(\mathsf{id}, a) \qquad c = \lambda(\mathsf{app}\ (c\mathsf{p}, \mathsf{q})) \qquad (\lambda b)\sigma = \lambda(b\sigma^+) \qquad \mathsf{app}(c, a)\sigma = \mathsf{app}(c\sigma, a\sigma)$$

where we write $\sigma^+ = (\sigma\mathsf{p}, \mathsf{q})$.

2.3. Cumulative universes

We assume to have global elements U_n in $\mathsf{Type}_{n+1}(\Gamma)$ such that $\mathsf{Type}_n(\Gamma) = \mathsf{Elem}(\Gamma, U_n)$.

2.4. Booleans

Finally we add the global constant N_2 in $\mathsf{Type}_0(\Gamma)$ and global elements 0 and 1 in $\mathsf{Elem}(\Gamma, N_2)$. Given T in $\mathsf{Type}_n(\Gamma.N_2)$ and a_0 in $\mathsf{Elem}(\Gamma, T(\mathsf{id}, 0))$ and a_1 in $\mathsf{Elem}(\Gamma, T(\mathsf{id}, 1))$ we have an operation $\mathsf{brec}(a_0, a_1)$ producing an element in $\mathsf{Elem}(\Gamma, \Pi\ N_2\ T)$ satisfying the equations $\mathsf{app}(\mathsf{brec}(a_0, a_1), 0) = a_0$ and $\mathsf{app}(\mathsf{brec}(a_0, a_1), 1) = a_1$.

Furthermore, $\mathsf{brec}(a_0, a_1)\sigma = \mathsf{brec}(a_0\sigma, a_1\sigma)$.

3. Reducibility model

Given a model of type theory M as defined above, we describe how to build a new associated "reducibility" model M*. When applied to the initial/term model M_0, this gives a proof of canonicity which can be seen as a direct generalization of the argument presented in [20] for Gödel system T. As explained in the introduction, the main novelty here is that we consider a proof-relevant notion of reducibility.

A context of M* is given by a context Γ of the model M together with a family of sets $\Gamma'(\rho)$ for ρ in $|\Gamma|$. A substitution in $\Delta, \Delta' \to^* \Gamma, \Gamma'$ is given by a pair σ, σ' with σ in $\Delta \to \Gamma$ and σ' in $\Pi(\nu \in |\Delta|)\Delta'(\nu) \to \Gamma'(\sigma\nu)$.

The identity substitution is the pair $1^* = 1, 1'$ with $1'\rho\rho' = \rho'$.

Composition is defined by $(\sigma, \sigma')(\delta, \delta') = \sigma\delta, (\sigma\delta)'$ with

$$(\sigma\delta)'\alpha\alpha' = \sigma'(\delta\alpha)(\delta'\alpha\alpha')$$

The set $\mathsf{Type}^*_n(\Gamma, \Gamma')$ is defined to be the set of pairs A, A' where A is in $\mathsf{Type}_n(\Gamma)$ and $A'\rho\rho'$ is in $|A\rho| \to \mathcal{U}_n$. We then define $A'(\sigma, \sigma')\nu\nu' = A'(\sigma\nu)(\sigma'\nu\nu')$.

We define $\mathsf{Elem}^*(\Gamma, \Gamma')(A, A')$ to be the set of pairs a, a' where a is in $\mathsf{Elem}(\Gamma, A)$ and $a'\rho\rho'$ is in $A'\rho\rho'(a\rho)$ for each ρ in $|\Gamma|$ and ρ' in $\Gamma'(\rho)$. We define then $(a, a')(\sigma, \sigma') = a\sigma, a'(\sigma, \sigma')$ with $a'(\sigma, \sigma')\nu\nu' = a'(\sigma\nu)(\sigma'\nu\nu')$.

The extension operation is defined by $(\Gamma, \Gamma').(A, A') = \Gamma.A, (\Gamma.A)'$ where $(\Gamma.A)'(\rho, u)$ is the set of pairs ρ', u' with $\rho' \in \Gamma'(\rho)$ and u' in $A'\rho\rho'(u)$.

We define an element $\mathsf{p}^* = \mathsf{p}, \mathsf{p}'$ in $(\Gamma, \Gamma').(A, A') \to^* \Gamma, \Gamma'$ by taking $\mathsf{p}'(\rho, u)(\rho', u') = \rho'$. We have then an element q, q' in $\mathsf{Elem}^*((\Gamma, \Gamma').(A, A'), (A, A')\mathsf{p}^*)$ defined by $\mathsf{q}'(\rho, u)(\rho', u') = u'$.

3.1. Dependent product

We define a new operation $\Pi^*\ (A, A')\ (B, B') = \Pi\ A\ B, (\Pi\ A\ B)'$ where $(\Pi\ A\ B)'\rho\rho'(w)$ is the set

$$\Pi(u \in |A\rho|)\Pi(u' \in A'\rho\rho'(u))B'(\rho, u)(\rho', u')(\mathsf{app}(w, u))$$

If b, b' is in $\mathsf{Elem}^*((\Gamma, \Gamma').(A, A'), (B, B'))$ then $\lambda^*(b, b') = \lambda b, (\lambda b)'$ where $(\lambda b)'$ is defined by the equation

$$(\lambda b)'\rho\rho'uu' = b'(\rho, u)(\rho', u')$$

which is in

$$B'(\rho, u)(\rho', u')(\mathsf{app}((\lambda b)\rho, u)) = B'(\rho, u)(\rho', u')(b(\rho, u))$$

We have an application operation $\mathsf{app}^*((c, c'), (a, a')) = (\mathsf{app}(c, a), \mathsf{app}(c, a)')$ where $\mathsf{app}(c, a)'\rho\rho' = c'\rho\rho'(a\rho)(a'\rho\rho')$.

3.2. Universes

We define $U'_n(A)$ for A in $|U_n|$ to be the set of functions $|A| \to \mathcal{U}_n$. Thus an element A' of $U'_n(A)$ is a family of sets $A'(u)$ in $\mathcal{U}_n$ for u in $|A|$. The universe U^*_n of M* is defined to be the pair U_n, U'_n and we have $\mathsf{Elem}^*((\Gamma, \Gamma'), U^*_n) = \mathsf{Type}^*_n(\Gamma, \Gamma')$.

3.3. Booleans

We define $N_2'(u)$ for u in $|N_2|$ to be the set consisting of 0 if $u=0$ and of 1 if $u=1$. We have N_2' in $U_0'(N_2)$. Note that $N_2'(u)$ may not be a subsingleton if we have $0=1$ in the model. We define $\mathsf{brec}(a_0,a_1)'\rho\rho' uu'$ to be $a_0'\rho\rho'$ if $u'=0$ and to be $a_1'\rho\rho'$ if $u'=1$.

3.4. Main result

Theorem 3.1. *The new collection of contexts, with the operations* $\to^*$, $\mathsf{Type}_n^*, \mathsf{Elem}^*$ *and* U_n^* *and* N_2^* *define a new model of type theory.*

The proof consists in checking that the required equalities hold for the operations we have defined. For instance, we have

$$\mathsf{app}^*(\lambda^*(b,b'),(a,a')) = (\mathsf{app}(\lambda b,a),\mathsf{app}(\lambda b,a)') = (b(\mathsf{id},a),\mathsf{app}(\lambda b,a)')$$

and

$$\mathsf{app}(\lambda b,a)'\rho\rho' = (\lambda b)'\rho\rho'(a\rho)(a'\rho\rho') = b'(\rho,a\rho)(\rho',a'\rho\rho')$$

and

$$(b(\mathsf{id},a))'\rho\rho' = b'(\rho,a\rho)(1'\rho\rho',a'\rho\rho') = b'(\rho,a\rho)(\rho',a'\rho\rho')$$

When checking the equalities, we *only use* β,η*-conversions at the metalevel.*

There are of course strong similarities with the parametricity model presented in [5]. This model can also be seen as a constructive version of the *glueing* technique [15,21]. Indeed, to give a family of sets over $|\Gamma|$ is essentially the same as to give a set X and a map $X \to |\Gamma|$, which is what happens in the glueing technique [15,21].

4. The term model

There is a canonical notion of morphism between two models. For instance, the first projection $\mathsf{M}^* \to \mathsf{M}$ defines a map of models of type theory. As for models of generalized algebraic theories [11], there is an *initial* model unique up to isomorphism. We define the *term* model M_0 of type theory to be this initial model. As for equational theories, this model can be presented by first-order terms (corresponding to each operations) modulo the equations/conversions that have to hold in any model.

Theorem 4.1. *In the initial model given u in* $|N_2|$ *we have* $u=0$ *or* $u=1$. *Furthermore we don't have* $0=1$ *in the initial model.*

Proof. We have a unique map of models $\mathsf{M}_0 \to \mathsf{M}_0^*$. The composition of the first projection with this map has to be the identity function on M_0. If u is in $|N_2|$ the image of u by the initial map has hence to be a pair of the form u,u' with u' in $N_2'(u)$. It follows that we have $u=0$ if $u'=0$ and $u=1$ if $u'=1$. Since $0'=0$ and $1'=1$ we cannot have $0=1$ in the initial model M_0. $\square$

5. Presheaf model

We suppose given an arbitrary model M. We define from this the following category $\mathcal{C}$ of "telescopes". An object of $\mathcal{C}$ is a list $A_1,\dots,A_n$ with A_1 in $\mathsf{Type}()$, A_2 in $\mathsf{Type}(A_1)$, A_3 in $\mathsf{Type}(A_1.A_2)$ ... To any such object X we can associate a context $i(X) = A_1.\dots.A_n$ of the model M. If A is in $\mathsf{Type}(i(X))$, we define the set $\mathsf{Var}(X,A)$ of numbers v_k such that qp^k is in $\mathsf{Elem}(i(X),A)$. We may write simply $\mathsf{Elem}(X,A)$ instead of $\mathsf{Elem}(i(X),A)$. Similarly we may write $\mathsf{Type}_n(X) = \mathsf{Elem}(X,U_n)$ for $\mathsf{Type}_n(i(X))$. If v_k is in $\mathsf{Var}(X,A)$ we write $[v_k] = \mathsf{qp}^k$. If $Y = B_1,\dots,B_m$ is an object of $\mathcal{C}$, a map $\sigma : Y \to X$ is given by a list $u_1,\dots,u_n$ such that u_p is in $\mathsf{Var}(Y, A_p([u_1],\dots,[u_{p-1}]))$. We then define $[\sigma] = ([u_1],\dots,[u_p]) : i(Y) \to i(X)$. It is direct to define a composition operation such that $[\sigma\delta] = [\sigma][\delta]$ which gives a category structure on these objects.

We use freely that we can interpret the language of dependent types (with universes) in any presheaf category [14]. A presheaf F is given by a family of sets $F(X)$ indexed by contexts with restriction maps $F(X) \to F(Y)$, $u \mapsto u\sigma$ if $\sigma : Y \to X$, satisfying the equations $u1 = u$ and $(u\sigma)\delta = u(\sigma\delta)$ if $\delta : Z \to Y$. A dependent presheaf G over F is a presheaf over the category of elements of F, so it is given by a family of sets $G(X,\rho)$ for ρ in $F(X)$ with restriction maps.

We write $\mathcal{V}_0,\mathcal{V}_1,\dots$ the cumulative sequence of presheaf universes, so that $\mathcal{V}_n(X)$ is the set of $\mathcal{U}_n$-valued dependent presheaves on the presheaf represented by X.

Type_n defines a presheaf over this category, with Type_n subpresheaf of Type_{n+1}. We can see Elem as a dependent presheaf over Type_n since it determines a collection of sets $\mathsf{Elem}(X,A)$ for A in $\mathsf{Type}_n(X)$ with restriction maps.

If A is in $\mathsf{Type}_n(X)$ we let $\mathsf{Norm}(X, A)$ (resp. $\mathsf{Neut}(X, A)$) be the set of all expressions of type A that are in normal form (resp. neutral). As for $\mathsf{Var}(X, A)$ these are sets of symbolic expressions, with decidable equality. We define then inductively and at the same time, we define the value $[u]$ in $\mathsf{Elem}(X, A)$ of the symbolic expression u in $\mathsf{Norm}(X, A)$

- if u is in $\mathsf{Var}(X, A)$ then u is in $\mathsf{Neut}(X, A)$
- if u is in $\mathsf{Neut}(X, A)$ then u is in $\mathsf{Norm}(X, A)$
- U_k is in $\mathsf{Norm}(X, U_n)$ for $k < n$ with value U_k
- N_2 is in $\mathsf{Norm}(X, U_n)$ with value N_2
- 0 (resp. 1) is in $\mathsf{Norm}(X, N_2)$ with value 0 (resp. 1)
- if T is in $\mathsf{Norm}(X, N_2 \to U_n)$ and u_0 (resp. u_1) is in $\mathsf{Norm}(X, \mathsf{app}([T], 0))$ (resp. $\mathsf{Norm}(X, \mathsf{app}([T], 1))$) and v is in $\mathsf{Neut}(X, N_2)$ then $\mathsf{brec}\ T\ u_0\ u_1\ v$ is in $\mathsf{Neut}(X, \mathsf{app}([T], [v]))$ and $[\mathsf{brec}\ T\ u_0\ u_1\ v] = \mathsf{app}(\mathsf{brec}([u_0], [u_1]), [v])$
- if A is in $\mathsf{Norm}(X, U_n)$ and B is in $\mathsf{Norm}(X.[A], U_n)$ then $\Pi\ A\ B$ is in $\mathsf{Norm}(X, U_n)$ and $[\Pi\ A\ B] = \Pi\ [A]\ [B]$
- if A is in $\mathsf{Norm}(X, U_n)$ and B is in $\mathsf{Norm}(X.[A], U_n)$ and v is in $\mathsf{Norm}(X.[A], [B])$ then $\lambda\ A\ B\ v$ is in $\mathsf{Norm}(X, \Pi\ [A]\ [B])$ and $[\lambda\ A\ B\ v] = \lambda\ [v]$
- if A is in $\mathsf{Norm}(X, U_n)$ and B is in $\mathsf{Norm}(X.[A], U_n)$ and w is in $\mathsf{Neut}(X, \Pi\ [A]\ [B])$ and u is in $\mathsf{Norm}(X, [A])$ then $\mathsf{app}\ A\ B\ w\ u$ is in $\mathsf{Norm}(X, [B](\mathsf{id}, [u]))$ and $[\mathsf{app}\ A\ B\ w\ u] = \mathsf{app}([w], [u])$

Note the use of the decorated syntax for application and abstraction, which is crucial for a non ambiguous definition of the value $[u]$ of an expression. (We also use the same overloaded notation N_2 or U_n for an expression and its value.)

As for Elem, we can see Neut and Norm as dependent types over Type_n, and we have

$$\mathsf{Var}(A) \subseteq \mathsf{Neut}(A) \subseteq \mathsf{Norm}(A)$$

We have an evaluation function $[e] : \mathsf{Elem}(A)$ if $e : \mathsf{Norm}(A)$. If a is in $\mathsf{Elem}(A)$ then we let $\mathsf{Norm}(A)|a$ (resp. $\mathsf{Neut}(A)|a$) be the subtypes of $\mathsf{Norm}(A)$ (resp. $\mathsf{Neut}(A)$) of elements e such that $[e] = a$.

Each context Γ defines a presheaf $|\Gamma|$ by letting $|\Gamma|(X)$ be the set of all substitutions $i(X) \to \Gamma$.

Any element A of $\mathsf{Type}_n(\Gamma)$ defines internally a function $|\Gamma| \to \mathsf{Type}_n,\ \rho \mapsto A\rho$.

We have a canonical isomorphism between $\mathsf{Var}(A) \to \mathsf{Type}_n$ and $\mathsf{Elem}(A \to U_n)$. We can then use this isomorphism to build an operation

$$\pi : \Pi(A : \mathsf{Type}_n)(\mathsf{Var}(A) \to \mathsf{Type}_n) \to \mathsf{Type}_n$$

such that $(\Pi\ A\ B)\rho = \pi(A\rho)((\lambda x : \mathsf{Var}(A\rho))B(\rho, [x]))$.

We can also define, given $A : \mathsf{Type}_n$ and $F : \mathsf{Var}(A) \to \mathsf{Type}_n$ an operation $\Lambda f : \mathsf{Elem}(\pi AF)$, for $f : \Pi(x : \mathsf{Var}(A))\mathsf{Elem}(F\ x)$.

Similarly, we can define an operation

$$\pi : \Pi(A : \mathsf{Norm}(U_n))(\mathsf{Var}([A]) \to \mathsf{Norm}(U_n)) \to \mathsf{Norm}(U_n)$$

such that $[\pi AF] = \pi[A](\lambda(x : \mathsf{Var}([A]))[F\ x])$ and given $A : \mathsf{Norm}(U_n)$ and $F : \mathsf{Var}([A]) \to \mathsf{Norm}(U_n)$ and $f : \Pi(x : \mathsf{Var}([A]))\mathsf{Norm}([F\ x])$ an operation $\Lambda AFf : \mathsf{Norm}([\pi AF])$ such that $[\Lambda AFf] = \Lambda(\lambda(x : \mathsf{Var}([A])[f\ x]))$.

While equality might not be decidable in $\mathsf{Var}(A)$ (because we use arbitrary renaming as maps in the base category), the product operation is injective: if $\pi AF = \pi BG$ in $\mathsf{Norm}(U_n)$ then $A = B$ in $\mathsf{Norm}(U_n)$ and $F = G$ in $\mathsf{Var}([A]) \to \mathsf{Norm}(U_n)$.

6. Normalization model

The model is similar to the reducibility model and we only explain the main operations.

As before, a context is a pair Γ, Γ' where Γ is a context of M and Γ' is a dependent family over $|\Gamma|$.

A type at level n over this context consists now of a pair $A, \overline{A}$ where A is in $\mathsf{Type}_n(\Gamma)$ and $\overline{A}\rho\rho'$ in $U_n'(A\rho)$ for ρ in $|\Gamma|$ and ρ' in $\Gamma'(\rho)$. An element of $U_n'(T)$ for T in Type_n consists of a 4-uple T', T_0, α, β where the element T_0 is in $\mathsf{Norm}(U_n)|T$, the element T' is in $\mathsf{Elem}(T) \to \mathcal{V}_n$, the element β is in $\Pi(k : \mathsf{Neut}(T))T'([k])$ and α is in $\Pi(u : \mathsf{Elem}(T))\ T'(u) \to \mathsf{Norm}(T)|u$.

An element of this type is a pair $a, \overline{a}$ where a is in $\mathsf{Elem}(\Gamma, A)$ and $\overline{a}\rho\rho'$ is an element of $T'(a\rho)$ where $(T', T_0, \alpha, \beta) = \overline{A}\rho\rho'$.

The intuition behind this definition is that it is a "proof-relevant" way to express the method of reducibility used for proving normalization [12]: a reducibility predicate has to contain all neutral terms and only normalizable terms. The function α (resp. β) is closely connected to the "reify" (resp. "reflect") function used in normalization by evaluation [6], but for a "glued" model.

We redefine $N_2{}'(t)$ to be the set of elements u in $\mathsf{Norm}(N_2)|t$ such that u is 0 or 1 or is neutral. We define $\alpha_{N_2} t \nu = \nu$ and $\beta_{N_2}(k) = k$.

We define $\alpha_{U_n}\ T\ (T', T_0, \alpha_T, \beta_T) = T_0$ and for K neutral $\beta_{U_n}(K) = (K', K, \alpha, \beta)$ where $K'(t)$ is $\mathsf{Neut}([K])|t$ and $\alpha tk = k$ and $\beta(k) = k$.

The set $\mathsf{Type}_n^*(\Gamma, \Gamma')$ is defined to be the set of pairs $A, \overline{A}$ where A is in $\mathsf{Type}_n(\Gamma)$ and $\overline{A}\rho\rho'$ is in $U_n'(A\rho)$.

The extension operation is defined by $(\Gamma, \Gamma').(A, \overline{A}) = \Gamma.A, (\Gamma.A)'$ where $(\Gamma.A)'(\rho, u)$ is the set of pairs ρ', ν with $\rho' \in \Gamma'(\rho)$ and ν in $\overline{A}\rho\rho'.1(u)$.

We define a new operation $\Pi^*\ (A,\overline{A})\ (B,\overline{B}) = C,\overline{C}$ where $C = \Pi\ A\ B$ and $\overline{C}\rho\rho'$ is the tuple

- $C'(w) = \Pi(u : \mathsf{Elem}(A\rho))\Pi(v : T'(u))F'uv(\mathsf{app}(w,u))$
- $\beta(k)uv = \beta_F uv(\mathsf{app}(k, \alpha_T uv))$
- $\alpha\ w\ \xi = \Lambda T_0 Gg$ with $g(x) = \alpha_F[x]\beta_T(x)(\mathsf{app}(w,[x]))(\xi[x]\beta_T(x))$
- $C_0 = \pi T_0 G$ with $G(x) = F_0[x]\beta_T(x)$

where we write $(T', T_0, \alpha_T, \beta_T) = \overline{A}\rho\rho'$ in $U_n'(A\rho)$ and for each u in $\mathsf{Elem}(A\rho)$ and v in $T'(u)$ we write $(F'uv, F_0uv, \alpha_F uv, \beta_F uv) = \overline{B}(\rho,u)(\rho',v)$ in $U_n'(B(\rho,u))$. We can check $[C_0] = (\Pi\ A\ B)\rho$ and we have that C', C_0, α, β is an element of $U_n'((\Pi\ A\ B)\rho)$.

We define $\overline{U_n} = U_n, U_n{}', \alpha_{U_n}, \beta_{U_n}$ and $\overline{N_2} = N_2, N_2{}', \alpha_{N_2}, \beta_{N_2}$.

If we have T in $\mathsf{Type}_n(\Gamma.N_2)$ and a_0 in $\mathsf{Elem}(T(\mathsf{id},0))$ and a_1 in $\mathsf{Elem}(T(\mathsf{id},1))$ and for each $\rho : |\Gamma|$ and $\rho' : \Gamma'(\rho)$ and u in $\mathsf{Elem}(N_2)$ and v in $N_2'(u)$ an element $(T'uv, T_0uv, \alpha_T uv, \beta_T uv)$ in $U_n'(T(\rho,u))$ and $\overline{a_0}$ in $T'00(a_0)$ and $\overline{a_1}$ in $T'11a_1$ we define $f = \overline{\mathsf{brec}(a_0,a_1)}\rho\rho'$ as follows. We take $f\ u\ v = \overline{a_0}$ if $v = 0$ and $f\ u\ v = \overline{a_1}$ if $v = 1$ and finally $f\ u\ v = \beta_T uv(\mathsf{brec}\ (\Lambda\ N_2\ (\lambda(x : \mathsf{Var}(N_2))U_n)\ g))\ (\alpha_T 00 a_0\overline{a_0})\ (\alpha_T 11 a_1 \overline{a_1})\ v)$ where $g(x) = T_0[x]\beta_{N_2}(x)$ if v is neutral.

We thus get, starting from an arbitrary model M, a new model M^* with a projection map $\mathsf{M}^* \to \mathsf{M}$. As for the canonicity model, if we start from the initial model M_0 we have an initial map $\mathsf{M}_0 \to \mathsf{M}_0^*$ which is a section of the projection map. Hence for any a in $\mathsf{Elem}(A)$ we can compute $\overline{a}$ in $A'(a)$ where $(A', A_0, \alpha_A, \beta_A) = \overline{A}$ and we have $\alpha_A\ a\ \overline{a}$ in $\mathsf{Norm}(A)|a$.

Theorem 6.1. *Equality in* M_0 *is decidable.*

Proof. If a and b are of type A we can compute $\overline{A} = (A', A_0, \alpha, \beta)$. We then have $a = b$ in $\mathsf{Elem}(A)$ if, and only if, $\alpha a\overline{a} = \alpha b\overline{b}$ in $\mathsf{Norm}(A)$ since $u = [\alpha u\overline{u}]$ for any u in $\mathsf{Elem}(A)$. The result then follows from the fact that the equality in $\mathsf{Norm}((), A)$ is decidable. $\square$

We also can prove that Π is one-to-one for conversions, following P. Hancock's argument presented in [17].

7. Conclusion

Our argument extends directly to the addition of dependent sum types with surjective pairing, or inductive types such as the type W A B [19].

The proof is very similar to the argument presented in [17], but it covers conversion under abstraction and η-conversion. It is also similar to the arguments in [3,4], which do not, however, cover universes and are not formulated in the present algebraic setting.

Instead of set theory, one could formalize the argument in extensional type theory; presheaf models have been already represented elegantly in NuPrl [7]. As we noticed however, the meta theory only uses the form of extensionality (η-conversion) also used in the object theory, and we should be able to express the normalization proof as a program transformation from one type theory to another. The formulation of the presheaf model as a(n extension of) type theory will be similar to the way cubical type theory [9] expresses syntactically a presheaf model over a base category which is a Lawvere theory. This should essentially amount to work in a type theory with a double context, where substitutions for the first context are restricted to be renamings. We leave this as future work, which, if successful, would refute some arguments in [18] for not accepting η-conversion as definitional equality.

Acknowledgements

This work started as a reading group of the paper [21] together with Simon Huber and Christian Sattler. The discussions we had were essential for this work; in particular Christian Sattler pointed out to me the reference [4]. Thanks also to Ambrus Kaposi for an important question about the definition of $\mathsf{Neut}(X, A)$.

References

[1] A. Abel, G. Scherer, On irrelevance and algorithmic equality in predicative type theory, in: Logical Methods in Computer Science, vol. 8, 2012, pp. 1–36.
[2] P. Aczel, On relating type theories and set theories, in: Types for Proofs and Programs, in: Lecture Notes in Comput. Sci., vol. 1657, 1999, pp. 1–18.
[3] Th. Altenkirch, A. Kaposi, Normalisation by Evaluation for Dependent Type Theory, FSCD, 2016.
[4] Th. Altenkirch, M. Hofmann, Th. Streicher, Reduction-free normalisation for system F, unpublished note, 1997.
[5] J.-Ph. Bernardy, P. Jansson, R. Paterson, Parametricity and dependent types, in: ICFP 2010, 2010, pp. 345–356.
[6] U. Berger, H. Schwichtenberg, An inverse of the evaluation functional for typed lambda-calculus, in: Proceedings of LICS, 1991.
[7] M. Bickford, Formalizing category theory and presheaf models of type theory in Nuprl, preprint, https://arxiv.org/abs/1806.06114, 2018.
[8] Th. Coquand, An algorithm for testing conversion in type theory, in: Logical Frameworks, Cambridge University Press, 1991, pp. 255–279.
[9] C. Cohen, Th. Coquand, S. Huber, A. Mörtberg, Cubical type theory: a constructive interpretation of the univalence axiom, in: Proceeding of the Type Conference, 2015.
[10] L. Crosilla, M.L. Rathjen, Inaccessible set axioms may have little consistency strength, Ann. Pure Appl. Logic 115 (2002) 33–70.

[11] P. Dybjer, Internal type theory, in: Types for Programs and Proofs, Springer, 1996.
[12] S. Fortune, D. Leivant, M. O'Donnell, The expressiveness of simple and second-order type structures, J. ACM 30 (1) (1983) 151–185.
[13] K. Gödel, Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes, Dialectica 12 (1958) 280–287.
[14] M. Hofmann, Syntax and semantics of dependent type theory, in: Semantics of Logic of Computation, Cambridge University Press, 1997.
[15] J. Lambek, P.J. Scott, Introduction to Higher Order Categorical Logic, Cambridge Studies in Advanced Mathematics, vol. 7, 1986.
[16] P. Martin-Löf, An intuitionistic theory of types, preliminary version 1972; published in: 25 Years of Type Theory, Springer, Berlin, 1995.
[17] P. Martin-Löf, An intuitionistic theory of types: predicative part, in: Logic Colloquium'73, Bristol, 1973, pp. 73–118.
[18] P. Martin-Löf, About models for intuitionistic type theories and the notion of definitional equality, in: Proceedings of the Third Scandinavian Logic Symposium, 1975, pp. 81–109.
[19] P. Martin-Löf, Constructive mathematics and computer programming, in: Logic, Methodology and Philosophy of Science, VI, Hannover, 1979, in: Stud. Logic Found. Math., vol. 104, North-Holland, Amsterdam, 1982, pp. 153–175.
[20] J.R. Shoenfield, Mathematical Logic, Addison–Wesley, 1967.
[21] M. Shulman, Univalence for inverse diagrams and homotopy canonicity, Math. Structures Comput. Sci. 25 (05) (2014) 1203–1277.
[22] W.W. Tait, Intensional interpretations of functionals of finite type, part I, J. Symbolic Logic 32 (1967) 198–212.