Reduction Free Normalization for a proof-irrelevant type of propositions

Thierry Coquand

Computer Science Department, University of Gothenburg

Introduction

We show normalization and decidability of conversion for dependent type theory with a cumulative sequence of universes $\mathsf{U}_0, \mathsf{U}_1 \dots$ with η-conversion *and* where the type U_0 is an impredicative universe of proof irrelevant propositions. One interest of such a system is that it is very close to the type system used by the proof assistant Lean [12].

Such a system with a hierarchy of universes, with the lowest level impredicative, was introduced in [3]. It was conjectured there that this system is stronger than Zermelo set theory (without even introducing primitive data types). This conjecture was solved by A. Miquel in [11] encoding a non well-founded version of set theory where a set is interpreted as a pointed graph up to bissimulation. The notion of proof-irrelevant propositions goes back to de Bruijn [6].

Our proof is a direct adaptation of the normalisation argument presented in [5]. We recall three features of this approach

1. we never need to consider a *reduction* relation
2. we only define a reducibility *predicate* and there is no need to define a *relation*, and this reducibility predicate is *proof relevant*[1]
3. the reducibility predicate is not defined by an inductive-recursive relation

One goal of this note is thus also to illustrate further the flexibility of this "reduction free" approach. One difference with [5] is that we realized that there is no need in the argument to introduce a notion of "neutral" and "normal" expressions. Roughly speaking, to each type A we associate a set of syntactical expressions $\mathsf{Term}(A)$ and a set $\mathsf{Elem}(A)$ of expressions *modulo conversion*. We have a quotient map $\mathsf{Term}(A) \to \mathsf{Elem}(A)$ and the main result (Theorem 2.1) is to show that this map has a section.

The metatheory used in the present note is the impredicative intuitionistic set theory IZFu_ω, introduced by P. Aczel [1]. (Essentially the same argument works in a predicative version CZFu_ω for a predicative universe of proof irrelevant propositions.)

As in the previous work [5], the approach is *algebraic*. We first define what is a model of our type theory, and explain how to build a *normalisation model* starting from the initial model.

1 What is a model of type theory

1.1 Definition

We present a formal system, which at the same time can be thought of describing the syntax of basic dependent type theory, with *explicit substitutions* and a *name-free* (de Bruijn index) presentation, and defining what is a model of type theory.

A model of type theory consists of one set Con of *contexts*. If Γ and Δ are in Con they determine a set $\Delta \to \Gamma$ of *substitutions*. If Γ is in Con, it determines a set $\mathsf{Type}(\Gamma)$ of *types* in the context Γ. Finally,

[1] A key point is to define reducibility as a *structure* and not only as a *property*. It is only for the lowest impredicative universe U_0 that reducibility is a property.

if Γ is in Con and A is in $\mathsf{Type}(\Gamma)$ then this determines a set $\mathsf{Elem}(\Gamma, A)$ of elements of type A in the context Γ.

This describes the *sort* of type theory. We describe now the *operations* and the equations they have to satisfy. For any context Γ we have an identity substitution $\mathsf{id} : \Gamma \to \Gamma$. We also have a composition operator $\sigma\delta : \Theta \to \Gamma$ if $\delta : \Theta \to \Delta$ and $\sigma : \Delta \to \Gamma$. The equations are

$$\sigma\ \mathsf{id} = \mathsf{id}\ \sigma = \sigma \qquad (\theta\sigma)\delta = \theta(\sigma\delta)$$

We have a terminal context 1 and for any Γ a map $() : \Gamma \to 1$. Furthermore $\sigma = ()$ if $\sigma : \Gamma \to 1$.

If A in $\mathsf{Type}(\Gamma)$ and $\sigma : \Delta \to \Gamma$ we should have $A\sigma$ in $\mathsf{Type}(\Delta)$ Furthermore

$$A\ \mathsf{id} = A \qquad (A\sigma)\delta = A(\sigma\delta)$$

If a in $\mathsf{Elem}(\Gamma, A)$ and $\sigma : \Delta \to \Gamma$ we should have $a\sigma$ in $\mathsf{Elem}(\Delta, A\sigma)$. Furthermore

$$a\ \mathsf{id} = a \qquad (a\sigma)\delta = a(\sigma\delta)$$

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

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

If a in $\mathsf{Elem}(\Gamma, A)$ we write $[a] = (\mathsf{id}, a) : \Gamma \to \Gamma.A$. Thus if B in $\mathsf{Type}(\Gamma.A)$ and a in $\mathsf{Elem}(\Gamma, A)$ we have $B[a]$ in $\mathsf{Type}(\Gamma)$. If furtermore b in $\mathsf{Elem}(\Gamma.A, B)$ we have $b[a]$ in $\mathsf{Elem}(\Gamma, B[a])$.

If $\sigma : \Delta \to \Gamma$ and A in $\mathsf{Type}(\Gamma)$ we define $\sigma^+ : \Delta.A\sigma \to \Gamma.A$ to be $(\sigma\mathsf{p}, \mathsf{q})$.

The extension operation can then be defined as $(\sigma, u) = [u]\sigma^+$. Thus instead of the extension operation, we could have chosen the operations $[u]$ and σ^+ as primitive. (This was the choice followed in [8].)

We suppose furthermore one operation $\Pi\ A\ B$ such that $\Pi\ A\ B$ in $\mathsf{Type}(\Gamma)$ if A in $\mathsf{Type}(\Gamma)$ and B in $\mathsf{Type}(\Gamma.A)$. We should have $(\Pi\ A\ B)\sigma = \Pi\ (A\sigma)\ (B\sigma^+)$.

We have an abstraction operation λb in $\mathsf{Elem}(\Gamma, \Pi\ A\ B)$ for b in $\mathsf{Elem}(\Gamma.A, B)$ and an application operation $c\ a$ in $\mathsf{Elem}(\Gamma, B[a])$ for c in $\mathsf{Elem}(\Gamma, \Pi\ A\ B)$ and a in $\mathsf{Elem}(\Gamma, A)$. These operations should satisfy the equations

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

We assume each set $\mathsf{Type}(\Gamma)$ to be stratified in $\mathsf{Type}_0(\Gamma) \subseteq \mathsf{Type}_1(\Gamma) \subseteq \dots$.

Furthermore each subset $\mathsf{Type}_n(\Gamma)$ is closed by dependent product, and we have U_n in $\mathsf{Type}_{n+1}(\Gamma)$ such that $\mathsf{Elem}(\Gamma, \mathsf{U}_n) = \mathsf{Type}_n(\Gamma)$.

We use *explicit* subsitutions but it should be clear that any element can be described using the following λ-calculus syntax[2]

$$A, B, t \quad ::= \quad \mathsf{q}\mathsf{p}^n \mid \mathsf{U}_n \mid t\ t \mid \lambda\ t \mid \Pi\ K\ L$$

For instance, the element $(\lambda\mathsf{qp})[\mathsf{q}]$ is equal to $\lambda(\mathsf{qp}[\mathsf{q}]^+) = \lambda(\mathsf{q}[\mathsf{q}]\mathsf{p}) = \lambda(\mathsf{qp})$.

Finally we assume U_0 to be *impredicative* and types in U_0 to be *proof-irrelevant*. This means that $\Pi\ A\ B$ is in $\mathsf{Type}_0(\Gamma)$ if B is in $\mathsf{Type}_0(\Gamma.A)$ where A can be *any* type, and that $a_0 = a_1 : \mathsf{Elem}(\Gamma, A)$ whenever A is in $\mathsf{Type}_0(\Gamma)$ and a_0 and a_1 are in $\mathsf{Elem}(\Gamma, A)$.

We think of types in $\mathsf{Type}_0(\Gamma)$ as proof-irrelevant propositions.

Note that, in an arbitrary model we may have some equality of the form[3] $\Pi\ A\ B = \mathsf{U}_0$ and the operations, like product operations, don't need to be injective.

[2] This syntax is simplified, omitting arguments for readibility; as in any generalized algebraic theory [7] the terms are first-order terms and $\lambda\ t$ is for instance a simplified notation for the first-order term $\lambda(\Gamma, A, B, t)$ while $\Pi\ A\ B$ a simplified notation for the term $\Pi(\Gamma, A, B)$.

[3] This can even be the case a priori in the term model, though it follows from our proof that this is not the case.

1.2 Examples of Models

Like for equational theories, there is always the *terminal* model where all sorts are interpreted by a singleton.

P. Aczel in [1] provides a model in in a impredicative intuitionistic set theory ZFu_ω, with intuitionistic versions of Grotendieck universes $\mathcal{U}_0, \mathcal{U}_1, \ldots, \mathcal{U}_\omega$.

A context is interpreted as a set in $\mathcal{U}_\omega$, and $\mathsf{Type}(\Gamma)$ is interpreted by $\Gamma \to \mathcal{U}_\omega$. The lowest universe U_0 is interpreted by the set of *truth values* $\mathcal{U}_0$: the set of subsets of $\{0\}$. In order to interpret the fact that U_0 is closed by arbitrary products, P. Aczel introduces a non standard encoding of dependent products, see [1], which also plays a crucial role for building our normalisation model.

M. Hofmann [9] shows how to refine a presheaf model over an arbitrary small category to a model of type theory. It models universes, and if we use Aczel's encoding of dependent products, we also get a model where the lowest universe is interpreted by the presheaf of sieves. We write $\mathcal{V}_0,\ \mathcal{V}_1,\ \ldots$ the universes corresponding to $\mathcal{U}_0,\ \mathcal{U}_1,\ \ldots$

From now on, we will work with the *initial* or *term* model M_0. This is the model where elements are syntactical expressions *modulo* equations/conversion rules. One important result which follows from the "normalisation model" we present in the next section, is that equality is *decidable* for the initial model, and that *constructors are injective*; this means in particular that we cannot have an equality of the form $\mathsf{U}_0 = \Pi\ A\ B$ and that $\Pi\ A_0\ B_0 = \Pi\ A_1\ B_1$ in $\mathsf{Type}(\Gamma)$ implies $A_0 = A_1$ in $\mathsf{Type}(\Gamma)$ and $B_0 = B_1$ in $\mathsf{Type}(\Gamma.A_0)$.

2 Normalisation Model

We present a variation of the model used in [5], where we don't need the notion of normal and neutral terms. As in [5], we work in a suitable *presheaf* topos, but with a slight variation for the choice of the base category.

2.1 Base category of syntactic substitutions

For defining the base category, we introduce, for A in $\mathsf{Type}(\Gamma)$, the set $\mathsf{Term}(\Gamma, A)$. This set is a set of *syntactical expressions* but contrary to the set $\mathsf{Elem}(\Gamma, A)$ these expressions are *not* quotiented up to conversion. Also the syntactical expressions don't use explicit substitutions and can be thought of as annotated λ-expressions.

The syntactical expressions are described by the following grammar

$$K, L, k \quad ::= \quad v_n \mid \mathsf{U}_n \mid \mathsf{app}\ K\ L\ k\ k \mid \lambda\ K\ K\ k \mid \Pi\ K\ L \mid 0$$

where v_n are de Bruijn index. We have v_0 in $\mathsf{Term}(\Gamma.A, A)$ and $\langle v_n \rangle = \mathsf{q}\mathsf{p}^n$.

We also consider the interpretation/quotient map

$$k \mapsto \langle k \rangle \qquad \mathsf{Term}(\Gamma, A) \to \mathsf{Elem}(\Gamma, A)$$

If K is in $\mathsf{Term}(\Gamma, \mathsf{U}_n)$ and L in $\mathsf{Term}(\Gamma.\langle K \rangle, \mathsf{U}_n)$ then $\Pi\ K\ L$ is in $\mathsf{Term}(\Gamma, \mathsf{U}_n)$ and $\langle \Pi\ K\ L \rangle = \Pi\ \langle K \rangle\ \langle L \rangle$. If furthermore k' is in $\mathsf{Term}(\Gamma, \langle \Pi\ K\ L \rangle)$ and k in $\mathsf{Term}(\Gamma, \langle K \rangle)$ then $\mathsf{app}\ K\ L\ k'\ k$ is in $\mathsf{Term}(\Gamma, \langle L \rangle[\langle k \rangle])$ and then $\langle \mathsf{app}\ K\ L\ k'\ k \rangle = \langle k' \rangle\ \langle k \rangle$.

One *key* addition to this notion of syntactical expressions, introduced in order to deal with proof-irrelevant propositions, is the special constant 0. We have 0 in $\mathsf{Term}(\Gamma, A)$ whenever A is in $\mathsf{Type}_0(\Gamma)$ and $\mathsf{Elem}(\Gamma, A)$ is *inhabited*.

Since $\mathsf{Elem}(\Gamma, A)$ is a subsingleton we can define $\langle 0 \rangle$ to be any element u of $\mathsf{Elem}(\Gamma, A)$.

If u is in $\mathsf{Elem}(\Gamma, A)$ we write $\mathsf{Term}(\Gamma, A)|u$ the subset of terms k such that $\langle k \rangle = u$.

The set of *syntactic substitutions* $\Delta \to_S \Gamma$ is recursively defined. At the same time, we define for such a syntactic substitution α its interpretation $\langle \alpha \rangle$ which is a substitution in $\Delta \to \Gamma$.

The empty syntactic substitution is in $\Delta \to_S 1$ and has for interpretation $() : \Delta \to 1$. If α is in $\Delta \to_S \Gamma$ and k is in $\mathsf{Term}(\Delta, A\langle \alpha \rangle)$ then α, k is in $\Delta \to_S \Gamma.A$ with $\langle \alpha, k \rangle = \langle \alpha \rangle, \langle k \rangle$.

We can define p_S in $\Gamma.A \to_S \Gamma$ such that $\langle \mathsf{p}_S \rangle = \mathsf{p}$.

If k is in $\mathsf{Term}(\Gamma, A)$ and α is in $\Delta \to_S \Gamma$ we can apply the substitution operation and get the element $k\alpha$ in $\mathsf{Term}(\Delta, A\langle\alpha\rangle)$. By induction on Γ we can then define $\alpha_\Gamma : \Gamma \to_S \Gamma$ such that $\langle \alpha_\Gamma \rangle = \mathsf{id}$ and we have $k\ \alpha_\Gamma = k$ (with a strict equality) if k is in $\mathsf{Term}(\Gamma, A)$.

We also can define the composition operation $\alpha\beta$ in $\Theta \to_S \Gamma$ for α in $\Delta \to_S \Gamma$ and β in $\Delta_1 \to_S \Delta$.

Proposition 2.0.1. *We have $(k\alpha)\beta = k(\alpha\beta)$, if $\alpha : \Delta \to_S \Gamma$ and $\beta : \Delta_1 \to \Delta$ and $k\mathsf{id}_S = k$. This implies $(\alpha\beta)\gamma = \alpha(\delta\gamma)$ if γ is in $\Delta_2 \to_S \Delta_1$.. Since furthermore we have $\mathsf{id}_S\alpha = \alpha = \alpha\mathsf{id}_S$.*

We can use this Proposition to define a category of *syntactic substitutions*. This category of syntactic substitutions will be the base category $\mathcal{C}$ for the presheaf topos $\hat{\mathcal{C}}$ in which we define the normalisation model[4]. As in [9, 5], we freely use the notations of type theory for operations in this presheaf topos. In this presheaf models we have a cumulative sequence of universe $\mathcal{V}_n$, for $n = 0, 1, \ldots, \omega$. Furthermore, as noticed above, $\mathcal{V}_0$ inherits from $\mathcal{U}_0$ the fact that it is closed by arbitrary products.

2.2 Presheaf model

Each Type_n defines a presheaf over $\mathcal{C}$ and both Term and Elem can be seen as dependent presheaves over Type_n with an interpretation function (natural transformation) $k \mapsto \langle k \rangle$ from $\mathsf{Term}(A)$ to $\mathsf{Elem}(A)$ for A in Type_n.

Each context Γ defines a presheaf $|\Gamma|$ by taking $|\Gamma|(\Delta)$ to be $\Delta \to_S \Gamma$ and each element A in $\mathsf{Type}_n(\Gamma)$ defines then a dependent presheaf $\rho \mapsto A\langle\rho\rangle$ over $|\Gamma|$.

Lemma 2.0.1. *In the presheaf topos $\hat{\mathcal{C}}$, we have the following operations, for A in $\mathsf{Type}(\Gamma)$ and B in $\mathsf{Type}(\Gamma.A)$ and ρ in $|\Gamma|$.*

1. $\Pi_S\ K\ G : \mathsf{Term}(\mathsf{U}_n)|(\Pi\ A\ B)\rho$ *for* $K : \mathsf{Term}(\mathsf{U}_n)|A\rho$ *and* $G : \Pi_{k:\mathsf{Term}(A\rho)}\mathsf{Term}(\mathsf{U}_n)|B(\rho, \langle k \rangle)$

2. $\lambda_S\ g : \mathsf{Term}((\Pi\ A\ B)\rho)|w$ *for* $g : \Pi_{k:\mathsf{Term}(A\rho)}\mathsf{Term}(B(\rho, \langle k \rangle))|(w\ \langle k \rangle)$

3. $\mathsf{app}_S\ K\ G\ k'\ k : \mathsf{Term}(B(\rho, \langle k \rangle))|(w\ \langle k \rangle)$ *for* $K : \mathsf{Term}(\mathsf{U}_n)|A\rho$ *and* $G : \Pi_{k:\mathsf{Term}(A\rho)}\mathsf{Term}(\mathsf{U}_n)|B(\rho, \langle k \rangle)$ *and* $k' : \mathsf{Term}((\Pi\ A\ B)\rho)|w$ *and* $k : \mathsf{Term}(A\rho)$

Proof. We prove the first point, the argument for the two other points being similar.

We have to define $\Pi_S\ K\ G$ in $\mathsf{Term}(\Delta, \mathsf{U}_n)$ such that $\langle \Pi_S\ K\ G \rangle = (\Pi\ A\ B)\langle\rho\rangle$. Here ρ is in $\Delta \to_S \Gamma$ and K is in $\mathsf{Term}(\Delta, \mathsf{U}_n)$ and such that $\langle K \rangle = A\langle\rho\rangle$. Furthermore G is an operation $G\alpha\ k$ is in $\mathsf{Term}(\Delta_1, \mathsf{U}_n)$ such that $\langle G\alpha\ k \rangle = B(\langle\rho\alpha\rangle, \langle k \rangle)$ for $\alpha : \Delta_1 \to_S \Delta$ and k in $\mathsf{Term}(\Delta_1, A\langle\rho\alpha\rangle)$ and such that $(G\alpha\ k)\beta = G(\alpha\beta)\ (k\beta)$ for $\beta : \Delta_2 \to_S \Delta_1$.

We then take $\Pi_S\ K\ G$ to be $\Pi\ K\ (G\mathsf{p}_S\ v_0)$. □

2.3 Normalisation model

We can now define the normalisation model M_0^*, where a context is a pair Γ, Γ' where Γ is a context of M_0 and Γ' is a dependent family over $|\Gamma|$.

For A in Type_n, we define $\mathsf{Type}'_n(A)$ to be the set of 4-tuples $(A', K, \mathsf{q}_A, \mathsf{r}_A)$ where[5]

1. A' is in $\mathsf{Elem}(A) \to \mathcal{V}_n$

2. K is in $\mathsf{Term}(\mathsf{U}_n)|A$

3. q_A, a "quote" function, is in $\Pi_{u:\mathsf{Elem}(A)}A'u \to \mathsf{Term}(A)|u$

4. r_A, a "reflect" function, is in $\Pi_{k:\mathsf{Term}(A)}A'\langle k \rangle$

[4]The use of context as world for a normalisation argument goes back to [4].

[5]This definition goes back to the unpublished paper [2] for system F; one difference is that we don't introduce any notion of neutral and normal terms.

A type over a context Γ, Γ' is a pair $A, \overline{A}$ where A is in some $\mathsf{Type}_n(\Gamma)$ and $\overline{A}\rho\overline{\rho}$ is an element of $\mathsf{Type}'_n(A\rho)$ for ρ in $|\Gamma|$ and $\overline{\rho}$ in $\Gamma'(\rho)$,

An element of this type is a pair $a, \overline{a}$ where a is in $\mathsf{Elem}(\Gamma, A)$ and $\overline{a}\rho\overline{\rho}$ is an element of $\overline{A}\rho\overline{\rho}.1(a\rho)$.

We define $\mathsf{q}_{\mathsf{U}_n}\ A\ (A', K, \mathsf{q}_A, \mathsf{r}_A) = K$.

For $n > 0$ and K in $\mathsf{Term}(\mathsf{U}_n)$ we define $\mathsf{r}_{\mathsf{U}_n}\ K$ to be $(K', K, \mathsf{q}_K, \mathsf{r}_K)$ where $K'u$ is $\mathsf{Term}(K)|u$ and $\mathsf{q}_K\ u\ \overline{u} = \overline{u}$ and $\mathsf{r}_K\ k = k$.

We define $\mathsf{r}_{\mathsf{U}_0}\ K$ to be $(K', K, \mathsf{q}_K, \mathsf{r}_K)$ where $K'u$ is $\{0\}$ and[6] $\mathsf{q}_K\ u\ \overline{u} = 0$ and $\mathsf{r}_K\ k = 0$.

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 $\overline{A}\rho\overline{\rho}$ is in $\mathsf{Type}'_n(A\rho)$.

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 $\overline{\rho}, \overline{u}$ with $\overline{\rho} \in \Gamma'(\rho)$ and $\overline{u}$ in $\overline{A}\rho\overline{\rho}.1(u)$.

As in [5], we define a new operation $\Pi^*\ (A, \overline{A})\ (B, \overline{B}) = C, \overline{C}$ where $C = \Pi\ A\ B$ and $\overline{C}\rho\overline{\rho}$ is the tuple

- $Z'(w) = \Pi_{u:\mathsf{Elem}(A\rho)}\Pi_{\overline{u}:X'(u)}F'u\overline{u}(wu)$
- $L = \Pi_S\ K\ G$ with $G\ k = F_0\langle k\rangle(\mathsf{r}_X\ k)$
- $\mathsf{q}_Z\ w\ \overline{w} = \lambda_S g$ with $g\ k = \mathsf{q}_F\langle k\rangle(\mathsf{r}_X k)(w\ \langle k\rangle)(\overline{w}\langle k\rangle(\mathsf{r}_X k))$
- $(\mathsf{r}_Z\ k)u\overline{u} = \mathsf{r}_F u\overline{u}(\mathsf{app}_S\ X_0\ G\ k\ (\mathsf{q}_X u\overline{u}))$

where we write $(X', K, \mathsf{q}_X, \mathsf{r}_X) = \overline{A}\rho\overline{\rho}$ in $\mathsf{Type}'_n(A\rho)$ and for each u in $\mathsf{Elem}(A\rho)$ and $\overline{u}$ in $X'(u)$ we write $(F'u\overline{u}, F_0u\overline{u}, \mathsf{q}_F u\overline{u}, \mathsf{r}_F u\overline{u}) = \overline{B}(\rho, u)(\overline{\rho}, \overline{u})$ in $\mathsf{Type}'_n(B(\rho, u))$. We can check that $Z', L, \mathsf{q}_Z, \mathsf{r}_Z$ is an element of $\mathsf{Type}'_n(C\rho)$.

We define $\overline{\mathsf{U}_n} = \mathsf{U}_n, {\mathsf{Type}_n}', \mathsf{q}_{\mathsf{U}_n}, \mathsf{r}_{\mathsf{U}_n}$ and U^*_n is the pair $\mathsf{U}_n, \overline{\mathsf{U}_n}$.

We get in this way a new model M^*_0 with a projection map $M^*_0 \to M_0$. We have an initial map $M_0 \to M^*_0$ which is a section of this initial map. Hence for any a in $\mathsf{Elem}(A)$ we can compute $\overline{a}$ in $A'(a)$ where $(A', A_0, \mathsf{q}_A, \mathsf{r}_A) = \overline{A}$ and we have $\mathsf{q}_A\ a\ \overline{a}$ in $\mathsf{Term}(A)|a$.

For the two main applications of this normalisation model, we first notice that, by induction on Γ, we can build α_Γ in $|\Gamma|(\Gamma)$ such that $\langle\alpha_\Gamma\rangle = \mathsf{id}$ and $\overline{\alpha_\Gamma}$ in $\Gamma'(\Gamma, \alpha_\Gamma)$.

If A is in $\mathsf{Type}(\Gamma)$ we can compute $\overline{A}\alpha_\Gamma\overline{\alpha_\Gamma} = (X', K, \mathsf{q}_X, \mathsf{r}_X)$ and we define $\mathsf{reify}(A)$ to be $\overline{A}\alpha_\Gamma\overline{\alpha_\Gamma}.2 = K$. We have $\langle\mathsf{reify}(A)\rangle = A$ since $\langle\mathsf{reify}(A)\rangle = A\ \mathsf{id} = A$. If furthermore a is in $\mathsf{Elem}(\Gamma, A)$ we define $\mathsf{reify}(a)$ to be $\mathsf{q}_X a(\overline{a}\alpha_\Gamma\overline{\alpha_\Gamma})$. We have $\langle\mathsf{reify}(a)\rangle = a$ in $\mathsf{Elem}(\Gamma, A)$ and we can summarize this discussion as follows.

Theorem 2.1. *The quotient map $k \mapsto \langle k\rangle$, $\mathsf{Term}(A) \to \mathsf{Elem}(A)$ has a section $a \mapsto \mathsf{reify}(a)$. Furthermore this map satisfies $\mathsf{reify}(\Pi\ A\ B) = \Pi\ \mathsf{reify}(A)\ \mathsf{reify}(B)$.*

Externally, this means that we have a map $\mathsf{Elem}(\Gamma, A) \to \mathsf{Term}(\Gamma, A)$ which commutes with substitution. If $\alpha : \Delta \to_S \Gamma$, we have $\mathsf{reify}(a)\alpha = \mathsf{reify}(a\langle\alpha\rangle)$.

Corollary 2.1.1. *Equality in M_0 is decidable.*

Proof. If a and b are in $\mathsf{Elem}(\Gamma, A)$ we have $\mathsf{reify}(a) = \mathsf{reify}(b)$ in $\mathsf{Term}(\Gamma, A)$ if, and only if, $a = b$ in $\mathsf{Elem}(\Gamma, A)$. The result then follows from the fact that the equality in $\mathsf{Term}(\Gamma, A)$ is decidable. □

We also can prove that Π is one-to-one for conversions, following P. Hancock's argument presented in [10].

Corollary 2.1.2. *If $\Pi\ A_0\ B_0 = \Pi\ A_1\ B_1$ in $\mathsf{Type}(\Gamma)$ in the term model, we have $A_0 = A_1$ in $\mathsf{Type}(\Gamma)$ and $B_0 = B_1$ in $\mathsf{Type}(\Gamma.A_0)$.*

Proof. We have $\mathsf{reify}(\Pi\ A_0\ B_0) = \Pi\ \mathsf{reify}(A_0)\ \mathsf{reify}(B_0) = \Pi\ \mathsf{reify}(A_1)\ \mathsf{reify}(B_1) = \mathsf{reify}(\Pi\ A_1\ B_1)$ as syntactical expressions, and hence $\mathsf{reify}(A_0) = \mathsf{reify}(A_1)$. This implies $A_0 = A_1$ in $\mathsf{Type}(\Gamma)$. We then have $\mathsf{reify}(B_0) = \mathsf{reify}(B_1)$, which implies similarly $B_0 = B_1$ in $\mathsf{Type}(\Gamma.A_0)$. □

We can define a normal form function $\mathsf{nf} : \mathsf{Term}(\Gamma, A) \to \mathsf{Term}(\Gamma, A)$ by $\mathsf{nf}(k) = \mathsf{reify}(\langle k\rangle)$.

[6] This is well-defined since u is in $\mathsf{Elem}(\Gamma, \langle K\rangle)$ and so 0 is in $\mathsf{Term}(\Gamma, \langle K\rangle)$.

3 Conclusion

Our argument extends to the addition of dependent sum types with surjective pairing, or inductive types. In general inductive types have to be declared in some universe U_n with $n > 0$. Note that it is possible to define the absurd proposition $\perp$ in U_0 as $\Pi_{X:\mathsf{U}_0} X$ and to add the large elimination rule $\perp \rightarrow A$ for *any* type A while preserving decidability of equality.

References

[1] P. Aczel. On Relating Type Theories and Set Theories. *Types for proofs and programs*, 1–18, Lecture Notes in Comput. Sci., 1657, 1999.

[2] Th. Altenkirch, M. Hofmann and Th. Streicher. Reduction-free normalisation for system F. Unpublished note, 1997.

[3] Th. Coquand An analysis of Girard's paradox. LICS 86.

[4] Th. Coquand and J. Gallier. A Proof of Strong Normalisation for the Theory of Constructions using a Kripke-Like Interpretation. Proceeding of first meeting on Logical Framework, 1990.

[5] Th. Coquand. Canonicity and normalisation for type theory. TCS 2018

[6] N. de Bruijn. Some extensions of Automath: The AUT-4 family. In R. Nederpelt, J. Geuvers, and R. de Vrijer, editors, Selected Papers on Automath, volume 133 of Studies in Logic and the Foundations of Mathematics, pages 283–288. Elsevier, 1994.

[7] P. Dybjer. Internal Type Theory. in *Types for Programs and Proofs*, Springer, 1996.

[8] Th. Ehrhard. *Une sémantique catégorique des types dépendents.* PhD thesis, 1988.

[9] M. Hofmann. Syntax and semantics of dependent type theory. In *Semantics of Logic of Computation*, Cambridge University Press, 1997.

[10] P. Martin-Löf. An intuitionistic theory of types: predicative part. Logic Colloquium '73 (Bristol, 1973), pp. 73–118.

[11] A. Miquel. lamda-Z: Zermelo's Set Theory as a PTS with 4 Sorts. TYPES 2004: 232-251.

[12] L. de Moura, S. Kong, J. Avigad, F. van Doorn and J. von Raumer. The Lean Theorem Prover. 25th International Conference on Automated Deduction (CADE-25), Berlin, Germany, 2015.

[13] M. Shulman. Univalence for inverse diagrams and homotopy canonicity. *Mathematical Structures in Computer Science*, 25:05, p. 1203–1277, 2014.