Constructive sheaf models of type theory

Introduction

The goal of this paper is to present a general way to build new models of univalent type theory which can be seen as a generalization of sheaf models of intuitionistic logic.

We present first an abstract version of the notion of compatible descent data, as a special kind of (pseudo)endomorphism on a model of type theory. We explain in what way this is used to build a new model of univalent type theory. (This can be seen as a constructive version of some results in [7].)

We provide then two main examples. The first one is using a *strict* notion of descent data. This can be used to show the consistency of univalent type theory with the Fan Theorem, or with the negation of Markov's Principle. The second one is with a notion of descent data *up to path equality*. This is used to show the consistency of univalent type theory with the negation of countable choice.

The notion of non strict descent data is interesting even for the trivial site over a category: in this case we show that equivalences in the resulting model coincide with pointwise equivalences.

1 Abstract notion of descent data

If B is a type over A we write $\pi_B : \Sigma_A B \to A$ the corresponding projection from the total space of B.

1.1 Lex morphism of models of type theory

We assume given a map of models: we have a strict functor D on types, and if B is a type over A we have a type $\tilde{D}(B)$ over $D(A)$, and if b is a section of B then $\tilde{D}(b)$ is a section of $\tilde{D}(B)$. We require $\tilde{D}(B\sigma) = \tilde{D}(B)D(\sigma)$ and $\tilde{D}(b\sigma) = \tilde{D}(b)D(\sigma)$ if $\sigma : A' \to A$. There is then a canonical map from $D(\Sigma_A B)$ to $\Sigma_{D(A)}\tilde{D}(B)$ and we require this map to be an *isomorphism*. (This notion of map of model is called "weak morphism" in [3].)

Let Id_A be the total space of the identity type over $A \times A$. The commuting diagram

$$\begin{array}{ccc} D(A) & \longrightarrow & D(\mathsf{Id}_A) \\ \downarrow & & \downarrow \\ \mathsf{Id}_{D(A)} & \longrightarrow & D(A \times A) \end{array}$$

has a diagonal filler. We require this diagonal filler to be an *equivalence*.

It follows from this that D preserves (homotopy) commuting diagram, equivalences, and homotopy pullbacks.

1.2 Example

If R is a given type, we define $D(A) = A^R$ and if $B(a)$ family over A, we define $\tilde{D}(B)(u)$ to be $\Pi(x : R)B(u\ x)$ for $u : D(A)$. The canonical map $D(\Sigma_A B) \to \Sigma_{D(A)}\tilde{D}(B)$ is then an isomorphism.

Since the canonical map

$$u =_{D(A)} v \to \Pi(x : R)\ u\ x =_A v\ x$$

is an equivalence, this defines a lex morphism.

1.3 Abstract notion of descent data

An abstract notion of descent data is a lex endomorphism D together with a strict natural transformation $\eta_A : A \to D(A)$ *and* a path equality between $D(\eta_A)$ and $\eta_{D(A)}$. The second condition should be expressed more precisely as giving for each A an element $\epsilon_A : \Pi(a : D(A))\ D(\eta_A)\ a =_{D^2(A)} \eta_{D(A)}\ a$ such that $\mathsf{app}\ (D^2\sigma)\ \epsilon_{A'}$ is path equal to $\epsilon_A D(\sigma)$ if $\sigma : A' \to A$.

We write $\mathsf{isStack}(A)$ the type (proposition) expressing that η_A is an equivalence.

Given η_A we can know relativize the operation D for types over A: we define $D^A(B)$ to be $\tilde{D}(B)\eta_A$ if B is a type over A. We then have a canonical map $\eta^A_B : \Sigma_A B \to D^A(B)$ over A. We define $\mathsf{isStack}^A(B)$ to be the type expressing that this map is an equivalence.

If $\sigma : A' \to A$ we have the strict equalities $D^A(B)\sigma = D^{A'}(B\sigma)$ and $\mathsf{isStack}^A(B)\sigma = \mathsf{isStack}^{A'}(B\sigma)$.

Theorem 1.1 *The proposition* $\mathsf{isStack}^A(B)$ *is equivalent to the proposition expressing that the strict commuting diagram*

$$\begin{array}{ccc} T & \xrightarrow{\eta_T} & D(T) \\ {\scriptstyle \pi_B}\downarrow & & \downarrow{\scriptstyle D(\pi_B)} \\ A & \xrightarrow[\eta_A]{} & D(A) \end{array}$$

where $T = \Sigma_A B$*, is a homotopy pullback diagram.*

1.4 Example

This is a continuation of the example 1.2. If we define $\eta_A : A \to D(A)$ by $\eta_A\ a = \lambda(x : A)a$ then we have $D(\sigma)\eta_A = \eta_B\sigma$ strictly for any map $\sigma : A \to B$.

We now assume that R is a *proposition*. The two maps $D(\eta_A)$ and $\eta_{D(A)}$ are path equals, and D defines a notion of descent data.

1.5 Closure properties

Proposition 1.2 *If A is a stack and B a family of stacks over A then $\Sigma_A B$ is a stack.*

Proposition 1.3 *If A is a stack then Id_A is a family of stacks over $A \times A$.*

So far we have not used the second condition on D.

Proposition 1.4 *For the map $\eta_A : A \to D(A)$ to be an equivalence, it is enough to have a patch function $p_A : D(A) \to A$ such that $p_A\eta_A$ is path equal to the identity of A.*

Proof. The condition implies that $D(p_A)D(\eta_A)$ is path equal to the identity. This is path equal to $D(p_A)\eta_{D(A)}$ which is strictly equal to $\eta_A p_A$. So, we have that p_A is both left and right path inverse to η_A and hence that η_A is an equivalence. □

Proposition 1.5 *If B is a family of stacks over A then $T = \Pi(x : A)B$ is a stack.*

Proof. We have a family of patch functions $p_B : D(B) \to B$. We can then define a patch function $p_T : D(T) \to T$ by taking $p_T(u)(a) = p_{B(a)}(D(e_a)u)$ where $e_a : T \to B(a)$ is the evaluation map $v \longmapsto v\ a$. If $u = \eta_T v$ we have $D(e_a)u = \eta_{B(a)}\ e_a(v)$ and hence $p_T(\eta_T v)(a) = p_{B(a)}(\eta_{B(a)}(v\ a))$ which is path equal to $v\ a$ as required. □

Proposition 1.6 *If B is a family of stacks over A then $\tilde{D}(B)$ is a family of stacks over $D(A)$.*

Proof. Let T be the type $\Sigma_A B$. We have to prove that the following strictly commuting diagram

$$\begin{array}{ccc} D(T) & \xrightarrow{\eta_{D(T)}} & D^2(T) \\ {\scriptstyle D(\pi_B)}\downarrow & & \downarrow{\scriptstyle D^2(\pi_B)} \\ D(A) & \xrightarrow[\eta_{D(A)}]{} & D^2(A) \end{array}$$

is a homotopy pullback diagram. By the second condition on D, this is equivalent to the fact that the strictly commuting diagram

$$\begin{array}{ccc} D(T) & \xrightarrow{D(\eta_T)} & D^2(T) \\ {\scriptstyle D(\pi_B)}\downarrow & & \downarrow{\scriptstyle D^2(\pi_B)} \\ D(A) & \xrightarrow[D(\eta_A)]{} & D^2(A) \end{array}$$

is a homotopy pullback diagram. This is the case since D is lex and B is a family of stacks over A. □

Corollary 1.7 *The type* $\Sigma(X : U)\mathsf{isStack}(El\ X)$ *is a stack.*

Proof. Let A be the type $\Sigma(X : U)\mathsf{isStack}(El\ X)$. We have a family of stacks $B(X, p) = El\ X$ over A. By the previous proposition $\tilde{D}(B)$ is a family of small stacks over $D(A)$. So we have defined a map $p_A : D(A) \to A$. Also $\tilde{D}(B)\eta_A(X, p) = D(El\ X)$ which is equivalent to $El\ X$. Since U is univalent and being a stack is a proposition, p_A is a patch function for A. □

1.6 Connections with the notion of modality

In general D may not be a modal operation in the sense of [8, 5, 6] since $D(A)$ may not be a stack. However, we can define the following type $M(A)$ as a higher inductive type with constructors

$$\begin{array}{lcl} \mathsf{inc} & : & A \to M(A) \\ \mathsf{patch} & : & D(M(A)) \to M(A) \\ \mathsf{linv} & : & \Pi(x : M(A))\ \mathsf{patch}(\eta_{M(A)} x) =_{M(A)} x \end{array}$$

The pair M, $\mathsf{isStack}$ define then a left exact modality in the sense of [8, 5, 6].

1.7 Model associated to a notion of descent data

We can now define an internal translation which provides a new model of univalent type theory, following [5]. We take the same notion of context, but a type of the new model is now a type *together with* a proof that this type is a stack.

In order to interpret the type of natural numbers with the desired computation rules, we need to use the following higher inductive type

$$\begin{array}{lcl} \mathsf{zero} & : & N \\ \mathsf{succ} & : & N \to N \\ \mathsf{patch} & : & D(N) \to N \\ \mathsf{linv} & : & \Pi(x : N)\ \mathsf{patch}(\eta_N x) =_N x \end{array}$$

The same idea applies to the interpretation of other inductive types such as the W type.

1.8 Generalization to a family of notions of descent data

If Cov is a given type, we generalize the notion of descent data by considering a lex morphism $D_c(A)$ $(c : \mathsf{Cov})$ from the given model to the model of types over Cov with maps $\eta^c_A : A \to D_c(A)$. We define then $\mathsf{isStack}(A)$ to be the proposition $\Pi(c : \mathsf{Cov})\mathsf{isEquiv}\ \eta^c_A$.

2 Presheaf model of univalent type theory

We now assume given a small cartesian category $\mathcal{B}$ with an object $\mathbf{I}$ which has two distinct global points 0 and 1. We also assume that $\mathbf{I}$ has a structure of bounded distributive lattice (though we think that our results still hold without this hypothesis and apply also to the cartesian cubical sets).

We write $I, J, K, \dots$ the objects of $\mathcal{B}$ and $X, Y, Z, \dots$ the objects of $\mathcal{C}$. A *sieve* S on I is a set of arrows of codomain I such that fg is in S whenever $f : J \to I$ is in S and $g : K \to J$. If S is a sieve on I and $f : J \to I$ we define a sieve Sf on J by taking the set of arrows g of codomain J such that fg is in S. Furthermore Sf is decidable if S is decidable. We define in this way a presheaf Φ by taking $\Phi_{\mathcal{B}}(J)$ to be the set of decidable sieves on J.

Let $\mathcal{C}$ be another small 1-category. We write $X|I$ the objects of $\mathcal{C} \times \mathcal{B}$ and $f|g$ the morphisms of this category where f is a map of $\mathcal{C}$ and g a map of $\mathcal{B}$.

If we have r and s in $\mathbb{I}_{\mathcal{B}}(I)$ we define a sieve $r = s$ on I by taking the set of maps $f : J \to I$ such that $rf = sf$. We assume that this sieve is decidable.

It is then known, following [1, 4, 2] how to define a model of univalent type theory with higher order inductive types, using for interval the presheaf $\mathbb{I}_{\mathcal{B}}$ represented by $\mathbf{I}$ and for cofibrations the maps classified by $\Phi_{\mathcal{B}}$.

We can also define an interval $\mathbb{I}$ on $\mathcal{C} \times \mathcal{B}$ by $\mathbb{I}(X|I) = \mathbb{I}_{\mathcal{B}}(I)$ and we define $\Phi(X|I)$ to be the set of decidable sieves on $X|I$ (with associated restriction maps). We still get a model of univalent type theory (and HITs) using the interval $\mathbb{I}$ and cofibrations Φ.

In this model, a context Γ is interpreted by a presheaf over $\mathcal{C} \times \mathcal{B}$ so a family of sets $\Gamma(X|I)$ with suitable restriction maps $\rho \mapsto \rho(f|g)$ with $f : Y \to X$ in $\mathcal{C}$ and $g : J \to I$ in $\mathcal{B}$. We may simply write ρf instead of $\rho(f|1_I)$ and ρg instead of $\rho(1_X|g)$.

A dependent type A over Γ is then given by a presheaf over the category of elements of Γ: for any ρ in $\Gamma(X|I)$ we have a set $A\rho$ with suitable restriction maps $A\rho \to A\rho(f|g),\ u \mapsto u(f|g)$ together with a composition [1, 4] operation. We write $\mathsf{Type}(\Gamma)$ the collection of all types over Γ. The set $\mathsf{Elem}(\Gamma, A)$ is then the set of sections: a family $a\rho$ in $A\rho$ such that $(a\rho)(f|g) = a(\rho(f|g))$ for any ρ in $\Gamma(X|I)$ and $(f|g)$ map of codomain $X|I$.

We write $\mathsf{Type}_0(\Gamma)$ the set of small types (such that each set $A\rho$ is small). The presheaf Type_0 is then represented by a fibrant type U which is univalent [1].

3 Strict notion of descent data

3.1 Negation of Markov's Principle

3.2 Model of the Fan Theorem

4 Descent data up to path equality

To any A is in $\mathsf{Type}(\Gamma)$ we can associate the type $C_n(A) = A^{\mathbb{I}^n}$ of n-cubes in A: we take $C_n(A)\rho = A\rho\mathsf{p}^n$ where $\mathsf{p}^n : J \times \mathbf{I}^n \to J$ We will use the $n+1$ maps $\sigma_k : \mathbb{I}^{n-1} \to \mathbb{I}^n$ given by

$$\sigma_k(\vec{i}, i_k, \vec{j}) = (\vec{i}, i_k, i_k, \vec{j})$$

for $0 < k < n$ and $\sigma_0\vec{i} = (0, \vec{i})$ and $\sigma_n\vec{i} = (\vec{i}, 1)$. Corresponding to these maps we have $n+1$ maps $\sigma_k : C_n(A) \to C_{n-1}(A)$.

We will use the notation $v = u(i_1, \dots, i_n)$ for elements in $C_n(A)\rho$, where $i_1, \dots, i_n$ are purely formal symbols. If for instance $v = u(i_1, i_2, i_3)$ we can then write $u(i_1, i_1, i_2)$ for $\sigma_1 v$.

We write $\alpha, \beta, \dots$ lists of composable arrows $f_1 : X_2 \to X_1,\ f_2 : X_3 \to X_2, \dots,\ f_n : X_{n+1} \to X_n$.
If $\alpha = f_1, \dots, f_n$ then $\langle\alpha\rangle$ denotes the composition $f_1 \dots f_n : X_{n+1} \to X_1$.

An element u of $D(A)(X|I)$ is given by a family of elements $u(f, \alpha)$ in $A(X_{n+1}|I \times \mathbf{I}^n)$. This family should satisfy the *compatibility conditions*

1. $\sigma_0 u(f, f_1, \gamma) = u(ff_1, \gamma)$

2. $\sigma_k u(f,\gamma,f_k,f_{k+1},\gamma') = u(f,\gamma,f_kf_{k+1},\gamma')$ for $0<k<n$

3. $\sigma_n u(f,\gamma,f_n) = u(f,\gamma)f_n$

that we can also express as

1. $u(f,f_1,\gamma)(0,\vec{i}) = u(ff_1,\gamma)(\vec{i})$

2. $u(f,\gamma,f_k,f_{k+1},\gamma')(\vec{i},i_k,i_k,\vec{i'}) = u(f,\gamma,f_kf_{k+1},\gamma')(\vec{i},i_k,\vec{i'})$ for $0<k<n$

3. $u(f,\gamma,f_n)(\vec{i},1) = u(f,\gamma)(\vec{i})f_n$

For instance, for $n=2$ this means

$$\sigma_0 u(f,f_1,f_2) = u(ff_1,f_2),\ \sigma_1 u(f,f_1,f_2) = u(f,f_1f_2),\ \sigma_2 u(f,f_1,f_2) = u(f,f_1)f_2$$

If $f: Y \to X$ and $g: J \to I$ we define the restriction $u(f|g)$ in $D(A)(Y|J)$ by $u(f|g)(f_1,\alpha) = u(ff_1,\alpha)g$.

We define a map $\eta_A : A \to D(A)$ by $(\eta_A a)(f,\alpha) = a(f\langle\alpha\rangle|\mathsf{p}^n)$.

If B is a type over A, we define $\tilde{D}(B)$ type over $D(A)$. If ρ is in $D(A)(X|I)$, then $\tilde{D}(B)\rho$ is a set of families $v(f,\alpha)$ in $B\rho(f,\alpha)$ satisfying the compatibiliy conditions.

Lemma 4.1 *If B has a composition operation over A then so does $\tilde{D}(B)$ over $D(A)$.*

Proof. Given ρ in $D(A)(X|J\times\mathbf{I})$ and ψ in $\Phi(X|J)$ and a family of elements $u_{f|g}$ of elements in $\tilde{D}(B)\rho(f|g)$ for $(\psi\mathsf{p}\vee\delta_0)(f|g)=1$ and such that $u_{f|g}(f'|g') = u_{ff'|gg'}$ we explain how to find an element v in $\tilde{D}(B)\rho\delta_1$ such that $v(f|g) = u_{f|\delta_1 g}$ if $\psi(f|g)=1$.

To simplify the presentation we explain the case $n=3$.

We have to describe $v(f,f_1,f_2)(i,j)$ in $A\rho(f,f_1,f_2)(i,j)\delta_1$. We take it to be

$$c_{\tilde{D}(B)}(\psi\to u)(i,j) \ = \ c_B\ (i=0\wedge j=0\wedge\psi_0\to u^0,\ i=0\wedge\psi_1\to u^1,\ \psi_2\to u^2)$$

where ψ_0 is the sieve of $f'|g$ such that $\psi(ff_1f_2|g)=1$ and ψ_1 is the sieve of $f'|g$ such that $\psi(ff_1|g)=1$ and ψ_2 is the sieve of $f'|g$ such that $\psi(f|g)=1$. Furthermore $u^0_{f'|g} = u_{ff_1f_2|g}(1)f'$ and $u^1_{f'|g} = u_{ff_1|g}(1,f_2)f'(j)$ and $u^2_{f'|g} = u_{f|g}(1,f_1,f_2)f'(i,j)$. □

Proposition 4.2 *The two maps $\eta_{D(A)}$ and $D(\eta_A): D(A)\to D^2(A)$ are path equal.*

Proof. An element of $D^2(A)(X|I)$ is given by a family $v(f,\alpha)(g,\beta)$ in $C_{n+m}(A)\rho f\langle\alpha\rangle g\langle\beta\rangle$ satisfying the conditions

1. $u(f,f_1,\gamma)(g,\beta)(0,\vec{i},\vec{j}) = u(ff_1,\gamma)(g,\beta)(\vec{i},\vec{j})$

2. $u(f,\gamma,f_k,f_{k+1},\gamma')(g,\beta)(\vec{i},i_k,i_k,\vec{i'},\vec{j}) = u(f,\gamma,f_kf_{k+1},\gamma')(g,\beta)(\vec{i},i_k,\vec{i'},\vec{j})$ if $0<k<n$

3. $\sigma_n u(f,\gamma,f_n)(g,\beta)(\vec{i},1,\vec{j}) = u(f,\gamma)(f_ng,\beta)(\vec{i},\vec{j})$

4. $u(f,\alpha)(g,g_1,\delta)(\vec{i},0,\vec{j}) = u(f,\alpha)(gg_1,\delta)(\vec{i},\vec{j})$

5. $u(f,\alpha)(g,\delta,g_l,g_{l+1},\delta')(\vec{i},\vec{j},j_l,j_l,\vec{j'}) = u(f,\alpha)(g,\delta,g_lg_{l+1},\delta')(\vec{i},\vec{j},j_l,\vec{j'})$ for $0<l<m$

6. $u(f,\alpha)(g,\delta,g_m)(\vec{i},\vec{j},1) = u(f,\alpha)(g,\delta)(\vec{i},\vec{j})g_m$

We compute, for u in $D(A)(X|I)$

$$(\eta_A u)(f,\alpha)(g,\beta)(\vec{i},\vec{j}) = u(f\langle\alpha\rangle,g,\beta)(\vec{j})$$

and

$$(D(\eta_A)u)(f,\alpha)(g,\beta)(\vec{i},\vec{j}) = u(f,\alpha)(\vec{i})g\langle\beta\rangle$$

We have a homotopy connecting these two maps by defining

$$v_k(f,\alpha)(g,\beta)(\vec{i},\vec{j}) = u(f,\alpha,g,\beta)(\vec{i}\wedge k,k,k\vee\vec{j})$$

since then $v_0(f,\alpha)(g,\beta)(\vec{i},\vec{j}) = u(f\langle\alpha\rangle,g,\beta)\vec{j}$ and $v_1(f,\alpha)(g,\beta)(\vec{i},\vec{j}) = u(f,\alpha)\vec{i}g\langle\beta\rangle$, and each element $v_k(f,\alpha)(g,\beta)(\vec{i},\vec{j})$ satisfies the compatibility conditions. □

Proposition 4.3 *The cobar operation is a notion of compatible descent data.*

More generally, if we have a subpresheaf Cov of the presheaf of sieves, we can define a type $D_c(A)$ ($c : \mathsf{Cov}$). If S is in $\mathsf{Cov}(X)$, an element of $D_c(A)(X|I, S)$ is now a family $u(f, \alpha)$ in $A(X_{n+1}|I \times \mathbf{I}^n)$ *with* f *in* S, satisfying the compatibility conditions.

4.1 Example: a model with the negation of countable choice

5 Equivalences in the stack model

5.1 Equivalences between fibrant types

If Γ is a presheaf over $\mathcal{C} \times \mathcal{B}$ then for any X object of $\mathcal{C}$ we have a cubical set $\Gamma(X)$ by $\Gamma(X)(I) = \Gamma(X|I)$ (and canonical restriction maps). Similarly if A is in $\mathsf{Type}(\Gamma)$ then we get a type $A(X)$ in $\mathsf{Type}(\Gamma(X))$ defined by $A(X)\rho = A\rho$. If A is contractible then each $A(X)$ is contractible. The converse may not hold in general. Similarly, if a map $\sigma : A \to B$ is an equivalence, then each $\sigma : A(X) \to B(X)$ is an equivalence, but the converse may not hold.

A simple example is provided by taking $\mathcal{C}$ the category with one object defined by the group $G = \mathbb{Z}/2\mathbb{Z}$. We take for B the unit type and for A the groupoid with two isomorphic objects swapped by the action of G. Then the canonical map $A \to B$ is pointwise an equivalence but is not an equivalence since A has no global points.

Lemma 5.1 *If* A *in* $\mathsf{Type}(\Gamma)$ *is such that each* $A(X)$ *is contractible then we can build an element of* $\mathsf{Elem}(\Gamma, D(A))$.

Proof. Given ρ in $\Gamma(X|I)$ we define a compatible family $c\rho(f, \alpha)$ in $C_n(A)\rho(f\langle\alpha\rangle|I)$ by induction on the length of α using the contractibility proof. □

Corollary 5.2 *If* A *in* $\mathsf{Type}(\Gamma)$ *is modal and such that each* $A(X)$ *is contractible then we can build an element of* $\mathsf{Elem}(\Gamma, A)$.

Corollary 5.3 *If* A *in* $\mathsf{Type}(\Gamma)$ *is modal and such that each* $A(X)$ *is contractible then* A *is contractible.*

Proof. We can find an element a in $\mathsf{Elem}(\Gamma, A)$ by the previous corollary. The type $\mathsf{Path}\ A\mathsf{p}\ a\mathsf{p}\ \mathsf{q}$ over $\Gamma.A$ is still modal, and is pointwise contractible. So by the previous corollary, it has a section, and A is contractible. □

Theorem 5.4 *If* $\sigma : A \to B$ *is pointwise an equivalence and* A *and* B *are modal then* σ *is an equivalence.*

Proof. We consider the fiber F of σ as a type over B. By hypothesis this type is pointwise contractible. Also it is modal if A and B are modal. Hence it is contractible by Corollary 5.3. □

5.2 Equivalences between general presheaves

We have a fibrant replacement operation $\overline{A}$ with a trivial cofibration $A \to \overline{A}$. Also at each level X, we have a a fibrant replacement $A(X) \to \overline{A(X)}$. There is a canonical map $\overline{A(X)} \to \overline{A}(X)$ and this is an equivalence between two fibrant types at level X.

A map $\sigma : A \to B$ determine a map $\overline{\sigma} : \overline{A} \to \overline{B}$ and σ is an equivalence iff $\overline{\sigma}$ is an equivalence. But Theorem 5.4, this is the case iff each map $\overline{A}(X) \to \overline{B}(X)$ is an equivalence at level X, which in turn holds iff each map $\overline{A(X)} \to \overline{B(X)}$ is an equivalence at level X, and hence iff each map $A(X) \to B(X)$ is an equivalence.

References

[1] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: a constructive interpretation of the univalence axiom. In *21st International Conference on Types for Proofs and Programs, TYPES 2015*, 2015.

[2] Thierry Coquand, Simon Huber, and Anders Mörtberg. On higher inductive types in cubical type theory. In *LICS '18 Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 255–264, 2018.

[3] Clove Newstead. *Algebraic models of dependent type theory.* Phd thesis, Carnegie Mellon University, 2018.

[4] Ian Orton and Andrew M. Pitts. Axioms for modelling cubical type theory in a topos. *Log. Methods Comput. Sci.*, 14(4):Paper No. 23, 33, 2018.

[5] Kevin Quirin and Nicolas Tabareau. Lawvere-tierney sheafification in homotopy type theory. *J. Formalized Reasoning*, 9(2):131–161, 2016.

[6] Egbert Rijke, Michael Shulman, and Bas Spitters. Modalities in homotopy type theory. 2017. preprint. URL: `https://arxiv.org/abs/1706.07526`.

[7] Michael Shulman. All $(\infty, 1)$-toposes have strict univalent universes, 2019. URL: `https://arxiv.org/abs/1904.07004`.

[8] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics.* `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.