From type theory to setoids and back: a setoid model of extensional Martin-Löf type theory [1]

Erik Palmgren
Stockholm University

PCC 2019
Institute Mittag-Leffler, July 15-19, 2019

[1] Extended version of talks given at HMI, Bonn University, Göteborg University and Leeds University in 2018.

1.1/ Modelling Type Theory

The term model (P. Martin-Löf).

Intensional and extensional TT: Realizability models (P. Aczel, M. Beeson and J. Smith (1978)) – use type-free structures

Intensional and extensional TT: Category-theoretic models (J. Cartmell, P. Dybjer, M. Hofmann, T. Streicher and others)

Partial type theory: Domain models (P. Martin-Löf, E.P)

Set-theoretic models of type theory: A. Salvesen 1986 in ZF, B. Werner model of Coq in ZF, B. Barras model of Coq in IZF — both formalized in Coq.

HoTT : Simplicial models (V. Voedovsky), Cubical models (M. Bezem, T. Coquand, S. Huber, and others)

1.2/ Type Theory in Type Theory

The problem of modelling intensional Martin-Löf type theory within itself is a long standing issue and whether the proposed solutions are "natural" is debated. There are various proof-theoretic interpretations via CZF by Aczel, Rathjen and others, designed for determination of proof strength

$$\mathrm{ML} \longrightarrow^{*} \mathrm{CZF} \longrightarrow^{*} \mathrm{ML}^{+}$$

M. Hofmann (1994): conservativity of extensional ML over intensional ML for ML w/o universes.

The pioneering work of Dybjer Internal Type Theory 1995 attempted a direct interpretation, and revealed that there were many equational problems to solve, that in category theory are known as coherence problems. It also made clear that families of setoids must be defined in a careful way to solve these problems.

1.2/ Categorical models in Type Theory

A very general form of models of type theory is given by Categories with Attributes, CwA, (equivalently categories with families), which consists of a category $\mathbb{C}$ intended to give the semantics of contexts and substitutions. The types in contexts and the action of substitutions is given by a contravariant functor

$$\mathrm{Ty} : \mathbb{C}^{\mathrm{op}} \longrightarrow \mathrm{Set}$$

All type-theoretic notions may then axiomatized in defined in terms of contexts, their extensions and projections

$$\downarrow_A : \Gamma \triangleright A \longrightarrow \Gamma.$$

These categorical model live in set theory (which may be constructive). A challenge is to construct them directly in type theory, using setoids instead of sets. (E.P. ECwA, BCwA, Coq formalizations by E.P, Peter LeFanu Lumsdaine, Chaitanya L. Subramanian.)

2.1/ Setoids ~ Errett Bishop's notion of set [2]

The most common notion of general set in proof-assistants based on Martin-Löf type theory (Coq, Agda) is the setoid.

- A **setoid** $A = (|A|, =_A)$ is a type $|A|$ together with an equivalence relation $=_A$.
- An **(extensional) function** $f : A \to B$ between setoids is a function (operation) $|A| \to |B|$ together with a proof that the operation respects the equalities $=_A$ and $=_B$.

When based on Martin-Löf type theory this forms a good category of sets for constructive mathematics, supporting several choice principles: *Axiom of Unique Choice, Dependent Choice* and *Aczel's Presentation Axiom*.
And moreover allowing many set-theoretic construction, and the crucial *quotient construction* that is missing from MLTT.

---

[2] P. : Bishop-style constructive mathematics in type theory — a tutorial. *Constructive Mathematics: Foundations and Practice* held in Nis, Serbia, June 24-28, 2013. See: http://staff.math.su.se/palmgren

2.2/ Quotient construction

Let $X = (|X|, =_X)$ be a setoid and let $\sim$ be a reflexive relation on this setoid. Then by the extensionality of the relation

$$x =_X y \Longrightarrow x \sim y. \tag{1}$$

Thus if $\sim$ is an equivalence relation on X

$$X/\sim = (|X|, \sim)$$

is a setoid, and $q : X \to X/\sim$ defined by $q(x) = x$ is surjective.
Extension property: If $f : X \to Y$ is a function with

$$x \sim y \Longrightarrow f(x) =_Y f(y), \tag{2}$$

then there is a unique function $\overline{f} : X/\sim \to Y$ with

$$\overline{f}(i(x)) =_Y f(x) \qquad (x \in X).$$

We have constructed **the quotient of X with respect to $\sim$**:
$q : X \to X/\sim$

Remark. Every set is a quotient of a choice set (Presentation Ax.)

3.1/ Stratified setoids

Martin-Löf type theory (and derivative proof assistants, Agda, Coq) features an infinite hierarchy of type universes

$$U_0 \subseteq U_1 \subseteq U_2 \subseteq \cdots$$

each closed under the standard constructions Σ, Π and certain inductive types. This gives a natural stratification of setoids. A setoid A is an (m, n)-setoid if

$$|A| : U_m \qquad\qquad =_A : |A| \to |A| \to U_n.$$

- m-setoid $=_{\mathrm{def}}$ (m, m)-setoid
- m-classoid $=_{\mathrm{def}}$ $(m+1, m)$-setoid
- ("Replacement") $f : A \to B$, A m-setoid, B m-classoid $\Longrightarrow$ $\mathrm{Im}(f)$ m-setoid. — justification for the name classoid.

3.2/ Examples of stratified setoids

- $\mathbb{N} = (N, \mathrm{Id}(N, \cdot, \cdot))$ is a 0-setoid.
- Aczel's model of CZF: $\mathbb{V} = (V, =_V)$ forms a 0-classoid in ML type theory (if built from the universe U_0).
- $\mathrm{Sub}(A)$ the n-subsetoids of a n-classoid A forms a n-classoid.
- $\Omega_n = (U_n, \leftrightarrow)$ propositions of level n with logical equivalence constitute an n-classoid.
- For an n-setoid A, the setoid of extensional propositional functions of level n

$$P_n(A) = [A \to \Omega_n]$$

is an n-classoid.

3.3/ Families of setoids

Let A and X be setoids. Let $F : A \to \mathrm{Sub}(X)$ be an extensional function.

Then $F(x) = (\delta(F(x)), \iota_{F(x)})$ with $\iota_{F(x)} : \delta F(x) \to X$ injective, and for $p : x =_A y$, there is a unique isomorphism $\phi_p : \delta(F(x)) \to \delta(F(y))$ such that

$$\iota_{F(x)} = \iota_{F(y)}\phi_p. \tag{3}$$

Thus we obtain family F^* of setoids over A with proof-irrelevant transport functions $F^*(p)$ by letting:

$$F^*(x) := \delta(F(x)) \qquad F^*(p) := \phi_p.$$

3.4/ Families of setoids (cont.)

Abstracting on the properties of F^* one can arrive at the definition:

Definition

Let A be a setoid. A **(proof-irrelevant) setoid-family** *consists of a family $F(a)$ of setoids indexed by $a \in A$, with extensional transport functions $F(p) : F(a) \to F(b)$ for each proof $p : a =_A b$, satisfying*

- $F(p) =_{\mathrm{ext}} F(q)$ *for each pair of proofs $p, q : a =_A b$ (proof-irrelevance)*
- $F(\mathrm{r}_a) = \mathrm{id}_{F(a)}$ *where $\mathrm{r}_a : a =_A a$ is the standard proof of reflexivity.*
- $F(p \odot q) = F(p) \circ F(q)$ *if $q : a =_A b$ and $p : b =_A c$, and where $p \odot q : a =_A c$, using the standard proof $\odot$ of transitivity.*

Alternatively $F : A^{\#} \to$ **Setoids** is an E-functor (where $A^{\#}$ is the discrete category of A).

3.5/ Dependent Setoid Constructions

With this notion of family, we can start making type-theoretic constructions towards the setoid model.

For a F a family on A, we can form the dependent sum $\Sigma(A, F)$ and the dependent product setoid $\Pi(A, F)$ as follows
$\Sigma(A, F) = ((\Sigma x : |A|)|F(x)|, \sim)$ where

$$(x, y) \sim (u, v) := (\exists p : x =_A u)[F(p)(y) =_{B(u)} v]$$

$\Pi(A, F) = (P, \sim)$ where

$$\begin{aligned} P :=& (\Sigma f : (\Pi x : |A|)|F(x)|) \\ & (\forall x, y : A)(\forall p : x =_A y)[F(p)(f(x)) =_{B(y)} f(y)] \end{aligned}$$

and the equality is extensional:

$$(f, e) \sim (g, e') := (\forall x : A)[f(x) =_{B(x)} g(y)].$$

The operations Π, Σ, Ex act on families of setoids to produce new families of setoids.

4.1/ Judgement Forms of ML Type Theory

The basic judgement forms of Martin-Löf Type Theory (1984) are displayed to the left

$$\begin{array}{ll} \Gamma \Longrightarrow A\ \mathrm{type} & A \in \mathrm{Fam}(\Gamma) \\ \Gamma \Longrightarrow A = B & ? \\ \Gamma \Longrightarrow a : A & a \in \Pi(\Gamma, A) \\ \Gamma \Longrightarrow a = b : A & a = b \in \Pi(\Gamma, A) \end{array}$$

We may now try to interpret the forms as the statements about setoids to the right above. But we do not yet have any obvious interpretation of the type equality.

We may e.g. need to compare e.g. $\Pi(A, B)$ and $\Pi(A', B')$ as setoid families over Γ. A solution is to embed all dependent families of setoids in to a big universal setoid (classoid).

4.2/ Type-free interpretations of type theory

To obtain a setoid model without coherence problems we may seek inspiration from type-free interpretations of (extensional) type theory, e.g.

Jan Smith, *An Interpretation of Martin-Löf's Type Theory in a Type-Free Theory of Propositions,* Journal of Symbolic Logic 1984

But instead of using a combinators or recursive realizers, use constructive sets.

4.3/ Aczel's iterative sets model

Aczel's type of iterative sets V (Aczel 1978) is a type of well-founded trees where the branching f can be indexed by any type A in a universe U of small types. The introduction rule tells how to build a set $\alpha = \sup(A, f)$ from a family $f(x)$ $(x : A)$ of previously constructed sets

$$\frac{A : U \quad f : A \to V}{\sup(A, f) : V} \ (V \text{ intro})$$

Equality $=_V$ is defined by bisimulation, and then membership is given by

$$x \mathrel{\dot\in} \sup(A, f) := (\exists a : A)(x =_V f(a)).$$

$(V, =_V, \dot\in)$ is a model of CZF + DC (and possibly more, depending on the type theory).

4.4/ Aczel's iterative sets and setoids

Observations:

1. If $U = U_0$, then $\mathbb{V} = (V, =_V)$ is a classoid. Every set $\alpha = \sup(A, f) : V$ gives rise to a canonical setoid

$$\kappa(\alpha) = (A, =_f) \qquad \text{where } a =_f b := f(a) =_V f(b)$$

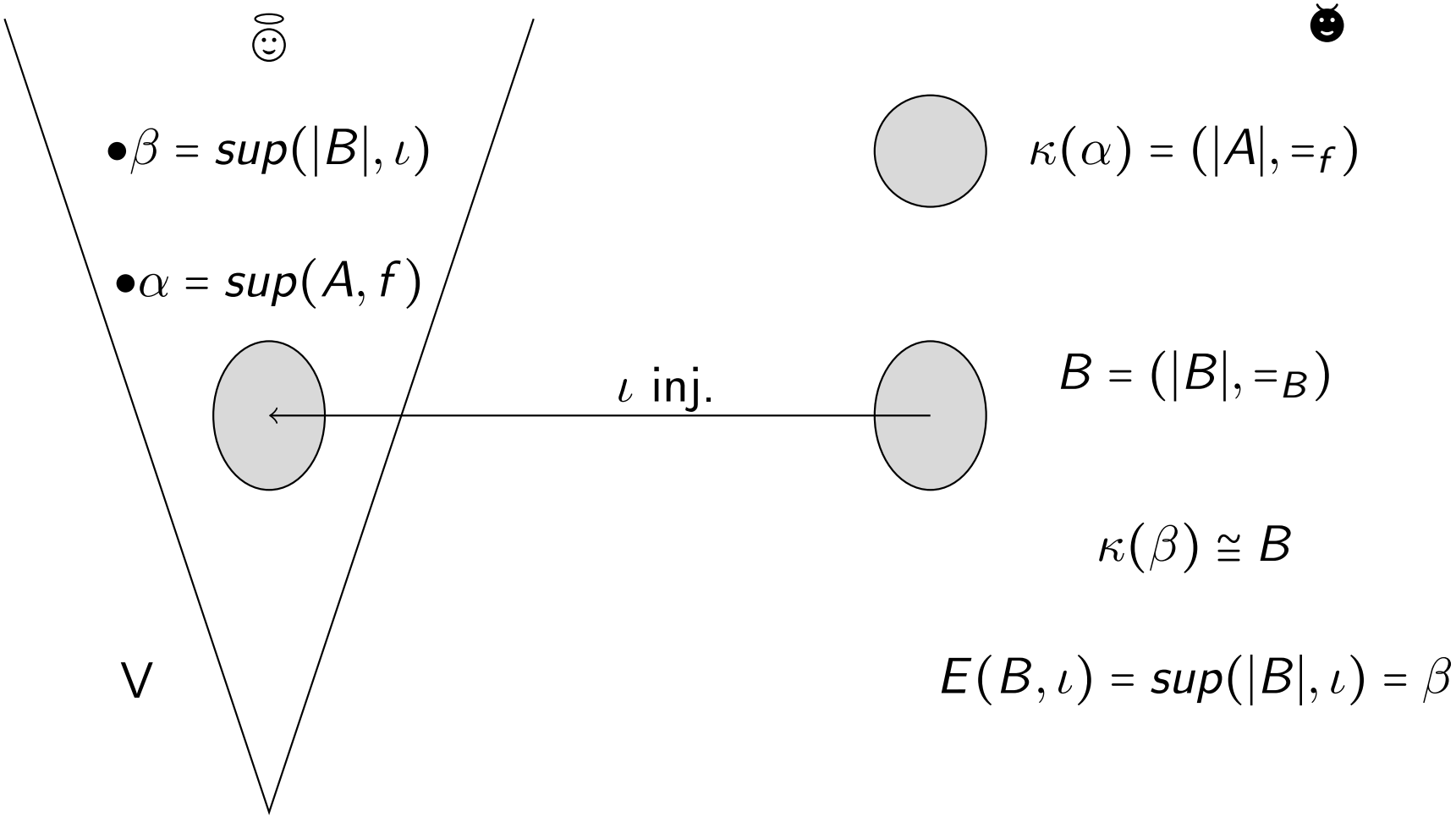
κ extends to a full and faithful E-functor into **Setoids**.

2. Moreover, if B is a subsetoid of V, then, for some $\beta : V$, there is a bijection

$$B \cong \kappa(\beta).$$

Indeed, if $\iota : (|B|, =_B) \to (V, =_V)$ is an injection, and we may let $\beta = \sup(|B|, \iota)$.

4.5/ Aczel's iterative sets and setoids (cont.)



Notation: if $\alpha = sup(A, f)$, write $\#\alpha = A$ and $\alpha \blacktriangleright x = f(x)$.

In fact as classoids, we have a bijection:

$$\mathbb{V} \cong \mathrm{Sub}(\mathbb{V}).$$

4.6/ Aczel's iterative sets and setoids (cont.)

Furthermore CZF (and V) admits constructions of dependent sums and products of sets (Aczel 1982). These relate well to the corresponding setoid constructions.

For $\alpha : V$ and $g : \kappa(\alpha) \to V$ we have sets $\sigma(\alpha, g), \pi(\alpha, g) : V$ with

$$\kappa(\sigma(\alpha, g)) \cong \Sigma(\kappa(\alpha), \kappa \circ g) \qquad \kappa(\pi(\alpha, g)) \cong \Pi(\kappa(\alpha), \kappa \circ g)$$

The set-theoretic version of the Σ-construction is, for $a : V$, $g : [\kappa(a) \to \mathbb{V}]$,

$$\sigma(\alpha, g) = \sup((\Sigma y : \#(a))\#(f(y)), \lambda u. < a \triangleright (\pi_1(u)), (g(\pi_1(u))) \triangleright (\pi_2(u)) >)$$

The set-theoretic Π-construction is more involved. For $a : V$ and $g : [\kappa(a) \to \mathbb{V}]$ define

$$\begin{aligned}
\text{piV-iV}(a, g) &= (\Sigma f : (\Pi x : \#(a))\#(g(a))) \\
&\quad (\forall x, y : \#(a))(\forall p : x =_{\kappa(a)} y) \\
&\quad (\kappa \circ g)(p)(f(x)) =_{(\kappa \circ g)(y)} f(y) \\
\text{piV-bV}(a, g) &= \lambda h.\sup(\#(a), (\lambda x. < a \blacktriangleright x, g(x) \blacktriangleright (\pi_1(h)(x)) >)) \\
\pi(a, g) &= \sup(\text{piV-iV}, \text{piV-bV}(a, g))
\end{aligned}$$

The first type piV-iV(a, g) singles out the extensional functions with a Σ-type. The branching function piV-bV then transforms such an extensional function to its graph in terms of set-theoretic pairs.

4.8/ The setoid model – interpretation in V

Contexts : elements $\Gamma, \Delta, \Theta, \ldots$ in classoid $\mathbb{V} = (V, =_V)$.

Context morphisms : setoid maps $\kappa(\Delta) \Rightarrow \kappa(\Gamma)$

Types : setoid maps $\kappa(\Gamma) \Longrightarrow \mathbb{V}$.

Raw terms : setoid maps $\kappa(\Gamma) \Longrightarrow \mathbb{V}$.

4.9/ The setoid model – interpretation in V (cont.)

The basic judgement forms of Martin-Löf Type Theory (1984) are displayed to the left. The setoid interpretation is on the right.

$\Gamma \Longrightarrow A\ \mathrm{type}$	$A : \kappa(\Gamma) \Longrightarrow \mathbb{V}$
$\Gamma \Longrightarrow A = B$	$A =_{\mathrm{ext}} B : \kappa(\Gamma) \Longrightarrow \mathbb{V}$
$\Gamma \Longrightarrow a : A$	$\forall x : \kappa(\Gamma), a(x) \in_V A(x)$
$\Gamma \Longrightarrow a = b : A$	$\forall x : \kappa(\Gamma), a(x) =_V b(x) \in_V A(x)$

On the right, $a, b : \kappa(\Gamma) \Longrightarrow \mathbb{V}$ are raw terms.

4.10/ Interpreted rules : Substitutions

$$\frac{\Gamma \text{ context}}{\Gamma == \Gamma} \qquad \frac{\Gamma == \Delta}{\Delta == \Gamma} \qquad \frac{\Gamma == \Delta \quad \Delta == \Phi}{\Gamma == \Phi}$$

$$\frac{f : \Gamma \longrightarrow \Delta}{f == f : \Gamma \longrightarrow \Delta} \qquad \frac{f == g : \Gamma \longrightarrow \Delta}{g == f : \Gamma \longrightarrow \Delta} \qquad \frac{f == g : \Gamma \longrightarrow \Delta \quad g == h : \Gamma \longrightarrow \Delta}{f == h : \Gamma \longrightarrow \Delta}$$

$$\frac{\Gamma \text{ context}}{\mathrm{id}_\Gamma : \Gamma \longrightarrow \Gamma} \qquad \frac{g : \Gamma \longrightarrow \Delta \quad f : \Delta \longrightarrow \Phi}{f \frown g : \Gamma \longrightarrow \Phi}$$

$$\frac{g : \Gamma \longrightarrow \Delta}{g \frown \mathrm{id}_\Gamma == g : \Gamma \longrightarrow \Delta} \qquad \frac{g : \Gamma \longrightarrow \Delta}{\mathrm{id}_\Delta \frown g == g : \Gamma \longrightarrow \Phi}$$

$$\frac{h : \Gamma \longrightarrow \Delta \quad g : \Delta \longrightarrow \Phi \quad f : \Phi \longrightarrow \Xi}{(f \frown g) \frown h == f \frown (g \frown h) : \Gamma \longrightarrow \Xi}$$

$$\frac{g == g' : \Gamma \longrightarrow \Delta \quad f == f' : \Delta \longrightarrow \Phi}{f \frown g == f' \frown g' : \Gamma \longrightarrow \Phi}$$

$$\frac{p : \Gamma == \Delta}{\phi_p : \Gamma \longrightarrow \Delta}\ (\texttt{subst-trp}) \qquad \frac{p : \Gamma == \Delta \quad q : \Gamma == \Delta}{\phi_p == \phi_q : \Gamma \longrightarrow \Delta}\ (\texttt{subst-trp-irr})$$

$$\frac{p : \Gamma == \Gamma}{\phi_p = \mathrm{id}_\Gamma : \Gamma \longrightarrow \Gamma}\ (\texttt{subst-trp-id}) \qquad \frac{p : \Gamma == \Delta \quad q : \Delta == \Phi \quad r : \Gamma == \Phi}{\phi_q \frown \phi_p == \phi_r : \Gamma \longrightarrow \Phi}\ (\texttt{subst-trp-fun})$$

4.11/ Interpreted rules: Context extension etc

$$\frac{}{\text{context}} \qquad \frac{\Gamma \Longrightarrow A\ \text{type}}{\Gamma \rhd A\ \text{context}}$$

$$\frac{\Gamma \Longrightarrow A\ \text{type} \quad \Delta \Longrightarrow B\ \text{type} \quad p : (\Gamma == \Delta) \quad \Gamma \Longrightarrow A == B[[\phi_p]]}{\Gamma \rhd A == \Delta \rhd B}$$

$$\frac{\Gamma \Longrightarrow A\ \text{type}}{\downarrow A : \Gamma \rhd A \longrightarrow \Gamma}\ (\downarrow) \qquad \frac{\Gamma \Longrightarrow A\ \text{type}}{\Gamma \rhd A \Longrightarrow \mathrm{v}_A :: A[[\downarrow A]]}\ (\texttt{asm})$$

$$\frac{f : \Delta \longrightarrow \Gamma \quad \Gamma \Longrightarrow A\ \text{type} \quad p : (\Delta \Longrightarrow a :: A[[f]])}{\langle f, a \rangle_p : \Delta \longrightarrow \Gamma \rhd A}\ (\texttt{ext})$$

$$\frac{f : \Delta \longrightarrow \Gamma \quad \Gamma \Longrightarrow A\ \text{type} \quad p : (\Delta \Longrightarrow a :: A[[f]]) \quad q : (\Delta \Longrightarrow a :: A[[f]])}{\langle f, a \rangle_p == \langle f, a \rangle_q : \Delta \longrightarrow \Gamma \rhd A}\ (\texttt{ext-irr})$$

$$\frac{f : \Delta \longrightarrow \Gamma \quad \Gamma \Longrightarrow A\ \text{type} \quad p : (\Delta \Longrightarrow a :: A[[f]])}{(\downarrow A) \frown \langle f, a \rangle_p == f : \Delta \longrightarrow \Gamma}\ (\texttt{ext-prop1})$$

$$\frac{f : \Delta \longrightarrow \Gamma \quad \Gamma \Longrightarrow A\ \text{type} \quad p : (\Delta \Longrightarrow a :: A[[f]])}{\Delta \Longrightarrow \mathrm{v}_A[\langle f, a \rangle_p] == a :: A[[f]]}\ (\texttt{ext-prop2})$$

$$\frac{\Gamma \Longrightarrow A\ \text{type} \quad p : (\Gamma \Longrightarrow \mathrm{v}_A :: A[[\downarrow A]])}{\langle \downarrow A, \mathrm{v}_A \rangle_p == \mathrm{id}_{\Gamma \rhd A} : \Gamma \rhd A \longrightarrow \Gamma \rhd A}$$

$$\frac{h : \Theta \longrightarrow \Delta \quad f : \Delta \longrightarrow \Gamma \quad \Gamma \Longrightarrow A\ \text{type} \quad p : (\Delta \Longrightarrow a :: A[[f]]) \quad q : (\Delta \Longrightarrow a[h] :: A[[f \frown h]])}{\langle f, a \rangle_p \frown h = \langle f \frown h, a[h] \rangle_q : \Delta \longrightarrow \Gamma \rhd A}$$

4.12/ Interpreted rules: Dependent function space

$$\frac{\Gamma \Longrightarrow A\ \mathrm{type} \quad \Gamma \rhd A \Longrightarrow B\ \mathrm{type}}{\Gamma \Longrightarrow \Pi_{\mathrm{f}}(A, B)\ \mathrm{type}}\ (\Pi\text{-f})$$

$$\frac{\Gamma \Longrightarrow A\ \mathrm{type} \quad \Gamma \rhd A \Longrightarrow B\ \mathrm{type} \quad \Gamma \rhd A \Longrightarrow b :: B}{\Gamma \Longrightarrow \lambda(A, B, b) :: \Pi_{\mathrm{f}}(A, B)}\ (\Pi\text{-i})$$

$$\frac{\Gamma \Longrightarrow A\ \mathrm{type} \quad \Gamma \rhd A \Longrightarrow B\ \mathrm{type} \quad \Gamma \rhd A \Longrightarrow b == b' :: B}{\Gamma \Longrightarrow \lambda(A, B, b) == \lambda(A, B, b') :: \Pi_{\mathrm{f}}(A, B)}\ (\Pi\text{-xi})$$

$$\frac{\Gamma \Longrightarrow A\ \mathrm{type} \quad \Gamma \rhd A \Longrightarrow B\ \mathrm{type} \quad p : (\Gamma \Longrightarrow c :: \Pi_{\mathrm{f}}(A, B)) \quad q : (\Gamma \Longrightarrow a :: A)}{\Gamma \Longrightarrow \mathrm{app}(A, B, c, p, a, q) :: B[[\mathrm{els}(q)]]}\ (\Pi\text{-e})$$

$$\frac{\Gamma \Longrightarrow A\ \mathrm{type} \quad \Gamma \rhd A \Longrightarrow B\ \mathrm{type} \quad r : (\Gamma \Longrightarrow \lambda(A, B, b) :: \Pi_{\mathrm{f}}(A, B)) \quad q : (\Gamma \Longrightarrow a :: A)}{\Gamma \Longrightarrow \mathrm{app}(A, B, \lambda(A, B, b), r, a, q) == b[\mathrm{els}(q)] :: B[[\mathrm{els}(q)]]}\ (\Pi\text{-beta-gen})$$

$$\frac{\begin{array}{l} p : (\Gamma \Longrightarrow c :: \Pi_{\mathrm{f}}(A, B)) \\ q_{\mathbf{1}} : (\Gamma \rhd A \Longrightarrow \mathrm{v}_A :: A[[\downarrow (A)]]) \\ q_{\mathbf{2}} : (\Gamma \rhd A \Longrightarrow c[\downarrow (A)] :: \Pi_{\mathrm{f}}(A[[\downarrow (A)]], B[[\uparrow (A, \downarrow (A))]])) \end{array}}{\lambda(A, B, \mathrm{app}(A[[\downarrow (A)]], B[[\uparrow (A, \downarrow (A))]], c[\downarrow (A)], q_{\mathbf{2}}, \mathrm{v}_A, q_{\mathbf{1}})) == c :: \Pi_{\mathrm{f}}(A, B)}\ (\Pi\text{-eta-eq-gen})$$

$$\frac{\Gamma \Longrightarrow A\ \mathrm{type} \quad \Gamma \rhd A \Longrightarrow B\ \mathrm{type} \quad h : \Delta \longrightarrow \Gamma}{\Delta \Longrightarrow \Pi_{\mathrm{f}}(A, B)[[h]] == \Pi_{\mathrm{f}}(A[[h]], B[[\uparrow (A, h)]])}\ (\Pi\text{-f-sub})$$

...

4.13/ Interpreted rules: the identity type

$$\frac{\Gamma \Longrightarrow A\ \mathrm{type} \quad p : (\Gamma \Longrightarrow a :: A) \quad q : (\Gamma \Longrightarrow a :: A)}{\Gamma \Longrightarrow \mathrm{ID}(A, a, p, b, q)\ \mathrm{type}}\ (\texttt{ID})$$

$$\frac{\Gamma \Longrightarrow A\ \mathrm{type} \quad p : (\Gamma \Longrightarrow a :: A)}{\Gamma \Longrightarrow \mathrm{rr}(a) :: \mathrm{ID}(A, a, p, a, p)}\ (\texttt{ID-i})$$

$$\frac{p : (\Gamma \Longrightarrow a :: A) \quad q : (\Gamma \Longrightarrow a :: A) \quad \Gamma \Longrightarrow t :: \mathrm{ID}(A, a, p, b, q)}{\Gamma \Longrightarrow a == b :: A}\ (\texttt{ID-e})$$

$$\frac{p : (\Gamma \Longrightarrow a :: A) \quad q : (\Gamma \Longrightarrow a :: A) \quad \Gamma \Longrightarrow t :: \mathrm{ID}(A, a, p, a, q)}{\Gamma \Longrightarrow t == \mathrm{rr}(a) :: \mathrm{ID}(A, a, p, a, q)}\ (\texttt{ID-uip})$$

4.14/ Interpreted rules: universes

For each $k \in \mathbb{N}$

$$\frac{\Gamma \text{ context}}{\Gamma \Longrightarrow \mathrm{U}_k \text{ type}} \qquad \frac{\Gamma \Longrightarrow A :: \mathrm{U}_k}{\Gamma \Longrightarrow A \text{ type}}$$

$$\frac{\Gamma \text{ context}}{\Gamma \Longrightarrow \mathrm{Nat} :: \mathrm{U}_k} \qquad \frac{\Gamma \text{ context}}{\Gamma \Longrightarrow \mathrm{N_0} :: \mathrm{U}_k}$$

$$\frac{\Gamma \Longrightarrow A :: \mathrm{U}_k \quad \Gamma \rhd A \Longrightarrow B :: \mathrm{U}_k}{\Gamma \Longrightarrow \Pi_{\mathrm{f}}(A, B) :: \mathrm{U}_k} \qquad \frac{\Gamma \Longrightarrow A :: \mathrm{U}_k \quad \Gamma \rhd A \Longrightarrow B :: \mathrm{U}_k}{\Gamma \Longrightarrow \Sigma_{\mathrm{f}}(A, B) :: \mathrm{U}_k}$$

$$\frac{\Gamma \Longrightarrow A :: \mathrm{U}_k \quad \Gamma \Longrightarrow B :: \mathrm{U}_k}{\Gamma \Longrightarrow \mathrm{Sum}(A, B) :: \mathrm{U}_k}$$

$$\frac{\Gamma \Longrightarrow A :: \mathrm{U}_k \quad p : (\Gamma \Longrightarrow a :: A) \quad q : (\Gamma \Longrightarrow b :: A)}{\Gamma \Longrightarrow \mathrm{ID}(A, a, p, b, q) :: \mathrm{U}_k}$$

$$\frac{\Gamma \text{ context}}{\Gamma \Longrightarrow \mathrm{U}_k :: \mathrm{U}_{s(k)}} \qquad \frac{\Gamma \Longrightarrow A :: \mathrm{U}_k}{\Gamma \Longrightarrow A :: \mathrm{U}_{s(k)}}$$

5.1/ Interpretation of the universes

Use the type universe Set of Agda as a *superuniverse.* Agda's data construct allows building universes inside it via a so-called simultaneous inductive recursive definition.

```
mutual
  data Uo (A : Set) (B : A -> Set) : Set where
     n₀ : Uo A B
     n₁ : Uo A B
     n : Uo A B
     ix : Uo A B
     lft : A -> Uo A B
     σ : (a : Uo A B) -> (To a -> Uo A B) -> Uo A B
     π : (a : Uo A B) -> (To a -> Uo A B) -> Uo A B
     w : (a : Uo A B) -> (To a -> Uo A B) -> Uo A B
  To : {A : Set} {B : A -> Set} ->  Uo A B -> Set
  To n₀               = N₀
  To n₁               = N₁
  To n                = N
  To {A} {B} ix       = A
  To {A} {B} (lft a)   = B a
  To (σ a b)          = Σ (To a) (\x -> To (b x))
  To (π a b)          = (x  : To a) -> To (b x)
  To (w a b)          = W (To a) (\x -> To (b x))


]
```

5.2/ Interpretation of the universes.

To explain the above, we note that the universe

$$\frac{A\ \mathrm{type} \quad x : A \Longrightarrow B\ \mathrm{type}}{\mathrm{Uo}(A, (x)B)} \qquad \frac{a : A}{\mathrm{To}(A, (x)B, a)\ \mathrm{type}}$$

has the same closure rules as type universes à la Tarski. In addition it has constructors for lifting a given family $A, (x)B$ into the universe

$$\frac{}{\mathrm{ix} : \mathrm{Uo}(A, (x)B)} \qquad \frac{}{\mathrm{To}(A, (x)B, \mathrm{ix}) = A}$$

$$\frac{a : A}{\mathrm{lft}(a) : \mathrm{Uo}(A, (x)B)} \qquad \frac{a : A}{\mathrm{To}(A, (x)B, \mathrm{lft}(a)) = B(a/x)}$$

5.3/ Interpretation of the universes.

Considering that the set universe V can be obtained by applying a W-type

```
data W (A : Set) (B : A -> Set) : Set where
   sup : (a : A) -> (b : B a -> W A B) -> W A B
```

to a type universe, we get a method for constructing a hierarchy of Aczel universes. This gives us a set universe $\mathrm{sV}(I, F)$ for each family of types I, F.

```
sV : (I : Set) -> (F : I -> Set)  -> Set
sV I F = W (Uo I F) (To {I} {F})
```

5.4/ Interpretation of the universes.

The elements of the set universe $\mathrm{sV}(I, F)$ are embedded into V

```
emb :   (I : Set) -> (F : I -> Set) -> sV I F -> V
emb I F (sup A f) = sup (To {I} {F} A) (\x -> emb I F (f x))
```

and they form a set $\mathrm{uV}(I, F)$ in V

```
uV : (I : Set) -> (F : I -> Set) -> V
uV I F = sup (sV I F) (emb I F)
```

We think of $\mathrm{uV}(I, F)$ as a constructive version of an inaccessible.

5.5/ Interpretation of the universes.

Iterating the universe building operator

```
mutual

  I- : (k : N) -> Set
  I- O = IO
  I- (s k) = Uo (I- k) (F- k)

  F- : (k : N) -> I- k -> Set
  F- O = FO
  F- (s k) = To {I- k} {F- k}
```

(here I_0, F_0 is an empty family) we then obtain an infinite hierarchy of inaccessibles

$$V_k = \mathrm{uV}(I_k, F_k)$$

in V such that $V_k \in V_{k+1}$. Each is a transitive set so $V_k \subseteq V_{k+1} \subseteq V$. This is the basis for the interpretation of the hierarchy of universes.

6/ Prospects for interpretation of further constructions

In view of the following theoretical results we can expect to interpret further type-constructions:

- The set-theoretic universe V has quotients, constructed by sets of equivalence classes (Aczel and Rathjen 2001)
- The set-theoretic universe V admits inductive definitions using REA (Aczel 1986)
- The set-theoretic universe V admits *transfinitely iterated* internal set-theoretic universes (Rathjen, Griffor and Palmgren 1998).

Current state of Agda development is available at:

`http://staff.math.su.se/palmgren/agda-model.zip`