A setoid model of extensional Martin-Löf type theory in Agda [3]

Erik Palmgren
Stockholm University

Stockholm Logic Seminar
Stockholm, January 30, 2019

[3] Extended version of talk given at the Stockholm-Göteborg Type Theory Seminar December 12, 2018

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)

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

2.1/ Setoids ~ Errett Bishop's notion of set [4]

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.

---

[4] 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/ Type Theory in Type Theory — The Setoid Model

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.

3.2/ 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.

3.3/ 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.

3.4/ 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:

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

3.5/ 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).

3.6/ 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

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

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

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

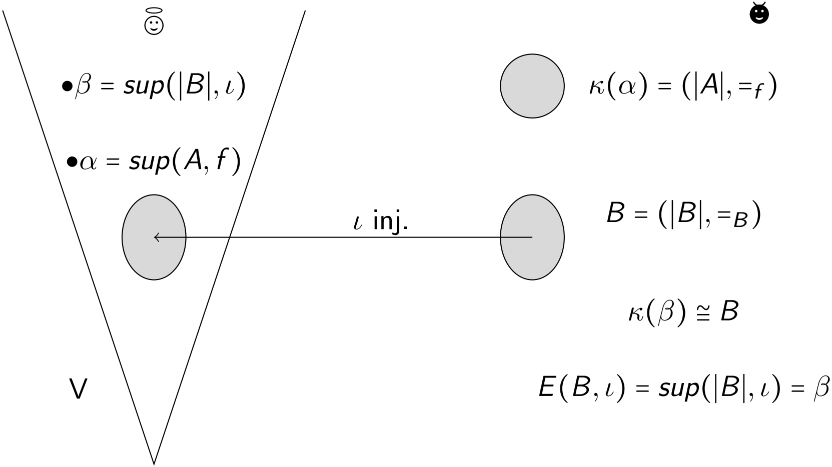
κ extends to a full and faithful E-functor into **Setoids**.

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

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

E.g. as coded in Agda:

```
sigmaV : (a : V) -> (g : setoidmap1 (κ a) VV) -> V
sigmaV a g =
      sup (Σ (# a) (\y -> # (g · y)))
            (\u -> < a ▸ (pj1 u) , (g · (pj1 u)) ▸ (pj2 u) >)
```

```
piV-iV : (a : V) -> (g : setoidmap1 (κ a) VV) -> Set
piV-iV a g =
      Σ ((x : # a) ->  # (g · x))
         (\f -> (x y : # a) ->
             (p : < κ a > x ~ y) ->
              < (κ° g)  § y > (ap (κ° g ± p) (f x))  ~ f y)

piV-bV : (a : V) -> (g : setoidmap1 (κ a) VV)
                 ->  piV-iV a g -> V
piV-bV a g h =
         sup (# a) (\x ->  < a ▸ x , (g · x) ▸ ((pj1 h) x) >)

piV : (a : V) -> (g : setoidmap1 (κ a) VV) -> V
piV a g = sup (piV-iV a g) (piV-bV a g)
```

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.

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

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

4.*/ Summary of interpreted rules

```
E1-tyrefl :  {Γ : ctx}
--
         -> (A : ty Γ)
-- ------------------------
         -> Γ ==> A == A
--
E1-tyrefl =  tyrefl

E2-tysym :  {Γ : ctx} -> {A B : ty Γ}
--
         -> Γ ==> A == B
--   --------------------------
         -> Γ ==> B == A
--
E2-tysym = tysym

E3-tytra :  {Γ : ctx} -> {A B C : ty Γ}
--
         -> Γ ==> A == B  -> Γ ==> B == C
--  -------------------------------------
                -> Γ ==> A == C
--
E3-tytra = tytra
```

4.*/ Summary of interpreted rules (cont.)

```
E4-tmrefl :  {Γ : ctx} -> {A : ty Γ}-> {a : raw Γ}
--
         -> Γ ==> a :: A
--    -----------------------
         -> Γ ==> a == a :: A
--
E4-tmrefl = tmrefl

E5-tmsym :  {Γ : ctx} -> (A : ty Γ) -> (a b : raw Γ)
--
         -> Γ ==> a == b :: A
--    -----------------------
         -> Γ ==> b == a :: A
--
E5-tmsym = tmsym


E6-tmtra :  {Γ : ctx} -> (A : ty Γ) -> (a b c : raw Γ)
--
         -> Γ ==> a == b :: A   -> Γ ==> b == c :: A
--  --------------------------------------------------
         -> Γ ==> a == c :: A
--
E6-tmtra = tmtra
```

4.*/ Summary of interpreted rules (cont.)

```
E10-elttyeq :  {Γ : ctx} ->  {a : raw Γ}  -> {A B : ty Γ}
--
      -> Γ ==> a :: A      -> Γ ==> A == B
--  --------------------------------------------
              -> Γ ==> a :: B
--
E10-elttyeq =  elttyeq

E11-elteqtyeq :  {Γ : ctx} ->  (a b : raw Γ)  -> (A B : ty Γ)
--
      -> Γ ==>  a == b :: A     -> Γ ==> A == B
--  ---------------------------------------------
              -> Γ ==>  a == b :: B
--
E11-elteqtyeq = elteqtyeq


E12-subj-red : {Γ : ctx} -> {A : ty Γ} ->  (a b : raw Γ)
--
     -> Γ ==> a :: A    ->  << Raw Γ >> a ˜ b
--   -------------------------
     -> Γ ==> b :: A
--
E12-subj-red  = subj-red
```

4.*/ Summary of interpreted rules (cont.)

```
S1-sub-id-prop : {Γ : ctx}
--
         -> (a : raw Γ)
--    ---------------------------------
      ->  << Raw Γ >> a [ ids ] ~ a
--
S1-sub-id-prop = sub-id-prop


S2-sub-comp-prop : {Θ  Δ  Γ : ctx}
--
       -> (a : raw Γ)
       ->  (f : subst Δ Γ) -> (g : subst Θ  Δ)
-- -----------------------------------------------------
      -> << Raw Θ >>  a [ f ⌢ g ]  ~ (a [ f ] [ g ])
--
S2-sub-comp-prop = sub-comp-prop
```

4.*/ Summary of interpreted rules (cont.)

```
S3-subst-trp : {Γ Δ : ctx}
--
     ->  (p : << Ctx >> Γ ~ Δ)
-- -------------------------------
     -> subst Γ Δ
S3-subst-trp = subst-trp

S4-sub-trp-prop : {Γ : ctx}
--
    -> (a : raw Γ)   ->  (p : << Ctx >> Γ ~ Γ)
-- ------------------------------------------
    -> << Raw Γ >> a [ subst-trp p ]  ~ a
--
S4-sub-trp-prop = sub-trp-prop
```

4.*/ Summary of interpreted rules (cont.)

```
S5-Sub-id-prop : {Γ : ctx}
--
        -> (A : ty Γ)
-- -----------------------------------
      ->  << Ty Γ >> A [[ ids ]] ~ A
--
S5-Sub-id-prop  = Sub-id-prop

S6-Sub-comp-prop : {Θ Δ Γ : ctx}
--
       -> (A : ty Γ)
       ->  (f : subst Δ Γ) -> (g : subst Θ Δ) ->
-- -----------------------------------------------------
       << Ty Θ >>  A [[ f ⌢ g ]]  ~ (A [[ f ]] [[ g ]])
--
S6-Sub-comp-prop = Sub-comp-prop
```

4.*/ Summary of interpreted rules (cont.)

```
S7-tyeq-subst :  {Δ Γ : ctx} -> {A B : ty Γ} -> (f : subst Δ Γ)
--
                  -> Γ ==> A == B
--      --------------------------------------------------
         -> Δ ==> A [[ f ]] ==  B [[ f ]]
--
S7-tyeq-subst = tyeq-subst

S8-elt-subst :  {Δ Γ : ctx} -> {a : raw Γ} -> {A : ty Γ} -> (f : subst Δ Γ)
--
          -> Γ ==> a :: A
--   --------------------------------------------------------
         -> Δ ==> a [ f ] ::  A [[ f ]]
--
S8-elt-subst = elt-subst


S9-elteq-subst :  {Δ Γ : ctx} -> {a b : raw Γ} -> {A : ty Γ}
--
          -> (f : subst Δ Γ)   -> Γ ==> a == b :: A
--   --------------------------------------------------------------------------
         -> Δ ==> a [ f ] == b [ f ] :: A [[ f ]]
--
S9-elteq-subst = elteq-subst
```

4.*/ Summary of interpreted rules (cont.)

```
S10-tyeq-subst2 :  {Δ Γ : ctx}
--
       -> (A : ty Γ)   -> (f g : subst Δ Γ)    -> < Subst Δ Γ > f ~ g
--     --------------------------------------------------
        -> Δ ==> A [[ f ]] ==  A [[ g ]]
--
S10-tyeq-subst2 = tyeq-subst2
```

4.*/ Summary of interpreted rules (cont.)

```
S11-tysubst-id : {Γ : ctx}
--
        -> (A  : ty Γ)
--   -------------------------
     -> Γ ==> (A [[ ids ]]) == A
--
S11-tysubst-id = tysubst-id

S12-tysubst-com : {Θ Δ Γ : ctx}
--
    -> (A : ty Γ)    -> (f : subst Δ Γ)  -> (g : subst Θ Δ)
--   -------------------------
     -> Θ ==> (A [[ f ⌢ g ]]) == (A [[ f ]] [[ g ]])
--
S12-tysubst-com = tysubst-com
```

4.*/ Summary of interpreted rules (cont.)

```
S13-eltsubst-id : {Γ : ctx}
--
    -> (A  : ty Γ)  ->  (a : raw Γ)  -> Γ ==> a :: A
--   ---------------------------
     -> Γ ==> (a [ ids ]) == a :: A
--
S13-eltsubst-id = eltsubst-id


S14-eltsubst-com : {Θ Δ Γ : ctx}
--
    -> (A : ty Γ)
    -> (f : subst Δ Γ)  -> (g : subst Θ Δ)
    -> (a : raw Γ)
    -> Γ ==> a :: A
--   -------------------------
     -> Θ ==> (a [ f ⌢ g ]) == (a [ f ] [ g ]) :: (A [[ f ⌢ g ]])
--
S14-eltsubst-com = eltsubst-com
```

4.*/ Summary of interpreted rules (cont.)

```
S15-subst-trp-id :  {Γ : ctx}
--
       ->  (p : << Ctx >> Γ ˜ Γ)
-- -------------------------------------------
      -> < Subst Γ Γ > subst-trp p ˜ ids {Γ}
--
S15-subst-trp-id = subst-trp-id

S16-subst-trp-irr :  {Γ Δ : ctx}
--
      ->  (p q : << Ctx >> Γ ˜ Δ)
-- ----------------------------------------------
      -> < Subst Γ Δ > subst-trp p ˜ subst-trp q
--
S16-subst-trp-irr  = subst-trp-irr

S17-subst-trp-fun :  {Γ Δ  Θ : ctx}
--
    ->  (p : << Ctx >> Γ ˜ Δ)   ->  (q : << Ctx >> Δ ˜ Θ)
    ->  (r : << Ctx >> Γ ˜ Θ)
-- ----------------------------------------------------------------
    -> < Subst Γ Θ > ((subst-trp q) ⌢ (subst-trp p)) ˜ subst-trp r
--
S17-subst-trp-fun = subst-trp-fun
```

4.*/ Summary of interpreted rules (cont.)

```
C1-↓ : {Γ : ctx}
--
      -> (A : ty Γ)
-- ---------------------
     -> subst (Γ ▷ A) Γ
--
C1-↓ = ↓

C2-asm : {Γ : ctx}
--
      -> (A : ty Γ)
--   --------------------------------------------------------
      -> Γ ▷ A ==> vv A :: A [[ ↓ A ]]
--
C2-asm = asm

C3-ext : {Δ Γ : ctx}
--
      -> (A : ty Γ)
      -> (f : subst Δ Γ) -> (a : raw Δ)
      -> Δ ==> a :: A [[ f ]]
-- ------------------------------------
      -> subst Δ (Γ ▷ A)
--
C3-ext = ext
```

4.*/ Summary of interpreted rules (cont.)

```
C4-ext-irr : {Δ Γ : ctx}
--
      -> (A : ty Γ)
      -> (f : subst Δ Γ) -> (a : raw Δ)
      -> (p : Δ ==> a :: A [[ f ]])
      -> (q : Δ ==> a :: A [[ f ]])
-- ----------------------------------------
      -> < Subst Δ (Γ ▷ A) >  (ext A f a p) ~  (ext A f a q)
--
C4-ext-irr = ext-irr

C5-ext-prop1 : {Δ Γ : ctx}
--
      -> (A : ty Γ)
      -> (f : subst Δ Γ) -> (a : raw Δ)
      -> (p : Δ ==> a :: A [[ f ]])
-- ----------------------------------------------------
      -> < Subst Δ Γ >  ((↓ A) ⌢ (ext A f a p)) ~ f
--
C5-ext-prop1 = ext-prop1
```

4.*/ Summary of interpreted rules (cont.)

```
C6-ext-prop2 : {Δ Γ : ctx}
--
       -> (A : ty Γ)
      ->  (f : subst Δ Γ) -> (a : raw Δ)
      -> (p : Δ ==> a :: A [[ f ]])
-- ----------------------------------------------------
      -> << Raw Δ >>  (vv A) [ ext A f a p ] ~ a
C6-ext-prop2 = ext-prop2


C7-ext-prop3 : {Γ : ctx}
--
       -> (A : ty Γ)
-- --------------------------------------------------------------------------
     ->   < Subst (Γ ▷ A) (Γ ▷ A) >  (ext A (↓ A) (vv A) (asm A)) ~ ids {Γ ▷ A}
--
C7-ext-prop3 = ext-prop3
```

4.*/ Summary of interpreted rules (cont.)

```
C8-ext-prop4-lm2 : {Θ Δ Γ : ctx}
--
      -> (A : ty Γ)
      -> (f : subst Δ Γ) -> (a : raw Δ)
      -> (p : Δ ==> a :: A [[ f ]])
      -> (h : subst Θ Δ)
      -> (q : Θ ==> a [ h ] :: A [[ f ⌢ h ]])
--    --------------------------------------------
      -> < Subst Θ (Γ ▷ A) > ((ext A f a p) ⌢ h) ~ (ext A (f ⌢ h) (a [ h ]) q)
--
C8-ext-prop4-lm2  =  ext-prop4-lm2

C9-ext-prop4 : {Θ Δ Γ : ctx}
--
      -> (A : ty Γ)
      -> (f : subst Δ Γ) -> (a : raw Δ)
      -> (p : Δ ==> a :: A [[ f ]])
      -> (h : subst Θ Δ)
--  --------------------------
      -> < Subst Θ (Γ ▷ A) > ((ext A f a p) ⌢ h) ~ (ext A (f ⌢ h) (a [ h ]) (ext-prop4-lm  A f a p h ))
--
C9-ext-prop4 = ext-prop4
```

4.*/ Summary of interpreted rules (cont.)

```
C10-ext-eq : {Γ : ctx}
--
      -> (A  A' : ty Γ)
      -> (Γ ==> A == A')
-- -------------------------
      -> (Γ ▷ A) ≐ (Γ ▷ A')
--
C10-ext-eq = ext-eq
```

```
C11-els  :
       {Γ : ctx}
--
    -> {A : ty Γ}
    -> {a : raw Γ}
    -> (Γ ==> a :: A)
-- ------------------------
    -> subst Γ (Γ ▷ A)
--
C11-els = els
```

4.*/ Summary of interpreted rules (cont.)

```
C12-↑ :  {Δ Γ : ctx}
--
    -> (A : ty Γ)    -> (h : subst Δ Γ)
-- ------------------------------------
     -> subst (Δ ▷ (A [[ h ]])) (Γ ▷ A)
--
C12-↑ = ↑
```

4.*/ Summary of interpreted rules (cont.)

```
P1-Π-f :  {Γ : ctx}
--
       -> (A : ty Γ)   -> (B : ty (Γ ▷ A))
--     -------------------------------------
       -> ty Γ
--
P1-Π-f = Π-f

P2-Π-f-rcong :  {Γ : ctx}
--
       -> (A : ty Γ)   -> (B B’ : ty (Γ ▷ A))
       ->  (Γ ▷ A ==> B == B’)
--     -------------------------------------
       -> (Γ  ==>  Π-f A B == Π-f A B’)
P2-Π-f-rcong = Π-f-rcong

P3-Π-i  :  {Γ : ctx}
--
       -> (A : ty Γ)   -> {B : ty (Γ ▷ A)}
       -> {b : raw (Γ ▷ A)}
       -> Γ ▷ A ==> b :: B
--  -----------------------------------------
        -> Γ ==> lambda A B b :: Π-f A B
--
P3-Π-i = Π-i
```

4.*/ Summary of interpreted rules (cont.)

```
P4-Π-xi  :  {Γ : ctx}
       -> (A : ty Γ)   -> {B : ty (Γ ▷ A)}
       -> {b : raw (Γ ▷ A)}
       -> {b' : raw (Γ ▷ A)}
       -> (r : Γ ▷ A ==> b == b' :: B)
--  -----------------------------------------
        -> Γ ==> lambda A B b ==  lambda A B b' :: Π-f A B
--
P4-Π-xi = Π-xi


P5-Π-e :  {Γ : ctx}
    -> (A : ty Γ)   -> (B : ty (Γ ▷ A))
    -> (c : raw Γ)
    -> (p : Γ ==> c :: Π-f {Γ} A B)
    -> (a : raw Γ)
    -> (q : Γ ==> a :: A)
--  -----------------------------------------
    ->  Γ ==> app A B c p a q :: B [[ els q ]]
--
P5-Π-e = Π-e
```

4.*/ Summary of interpreted rules (cont.)

```
P6-Π-e-cong :  {Γ : ctx}
--
    -> (A : ty Γ)   -> (B : ty (Γ ▷ A))
    -> (c c’ : raw Γ)
    -> (p : Γ ==> c :: Π-f {Γ} A B)
    -> (p’ : Γ ==> c’ :: Π-f {Γ} A B)
    -> (Γ ==> c == c’ :: Π-f {Γ} A B)
    -> (a a’ : raw Γ)
    -> (q : Γ ==> a :: A)
    -> (q’ : Γ ==> a’ :: A)
    -> (Γ ==> a == a’ :: A)
--  -----------------------------------------
    ->  Γ ==> app A B c p a q == app A B c’ p’ a’ q’  :: B [[ els q ]]
--
P6-Π-e-cong = Π-e-cong

P7-Π-beta :  {Γ : ctx}
--
       -> (A : ty Γ)   -> (B : ty (Γ ▷ A))
       -> (b : raw (Γ ▷ A))
       -> (p : Γ ▷ A ==> b :: B)
       -> (a : raw Γ)
       -> (q : Γ ==> a :: A)
--  -----------------------------------------
       ->  Γ ==> app A B (lambda A B b) (Π-i A p) a q
             ==  b [ els q ] :: B [[ els q ]]
--
P7-Π-beta = Π-beta
```

4.*/ Summary of interpreted rules (cont.)

```
P8-Π-beta-gen :  {Γ : ctx}
      -> (A : ty Γ)   -> (B : ty (Γ ▷ A))
      -> (b : raw (Γ ▷ A))
      -> (p : Γ ▷ A ==> b :: B)
      -> (r : Γ  ==> (lambda A B b) :: Π-f {Γ} A B)
      -> (a : raw Γ)
      -> (q : Γ ==> a :: A)
--  -----------------------------------------
      ->  Γ ==> app A B (lambda A B b) r a q
           ==  b [ els q ] :: B [[ els q ]]
--
P8-Π-beta-gen = Π-beta-gen
```

4.*/ Summary of interpreted rules (cont.)

```
P9-Π-f-sub :  {Δ Γ : ctx}
--
       -> (A : ty Γ)   -> (B : ty (Γ ▷ A))  -> (h : subst Δ Γ)
--     ---------------------------------------
       -> Δ ==>  (Π-f A B) [[ h ]] ==  Π-f (A [[ h ]]) (B [[ ↑ A h ]])
--
P9-Π-f-sub = Π-f-sub

P10-lambda-sub  :  {Δ Γ : ctx}
--
       -> (A : ty Γ)   -> {B : ty (Γ ▷ A)}
       -> {b : raw (Γ ▷ A)}
       -> (h : subst Δ Γ)
       -> Γ ▷ A ==> b :: B
--  -----------------------------------------
       -> Δ ==> (lambda A B b) [ h ] ==
                (lambda (A [[ h ]]) (B [[ ↑ A h ]]) (b [ ↑ A h ]))  :: (Π-f A B) [[ h ]]
--
P10-lambda-sub = lambda-sub
```

4.*/ Summary of interpreted rules (cont.)

```
P13-Π-eta-eq :  {Γ : ctx}
--
    -> (A : ty Γ)   -> (B : ty (Γ ▷ A))
    -> (c : raw Γ)
    -> (p : Γ ==> c :: Π-f {Γ} A B)
-- ------------------------------------------------------
    -> Γ ==> lambda A B (app (A [[ ↓ A ]])
                             (B [[ ↑ A (↓ A) ]])
                             (c [ ↓ A ])
                             (Π-eta-left3 {Γ} A B c p)
                             (vv A)
                             (Π-eta-left2 {Γ} A B c p))
             == c ::  Π-f {Γ} A B
--
P13-Π-eta-eq = Π-eta-eq
```

4.*/ Summary of interpreted rules (cont.)

```
P14-Π-eta-eq-gen :  {Γ : ctx}
    -> (A : ty Γ)   -> (B : ty (Γ ▷ A))
    -> (c : raw Γ)
    -> (p : Γ ==> c :: Π-f {Γ} A B)

    -> (q1 : (Γ ▷ A) ==> vv A ::  A [[ ↓ A ]])
    -> (q2 : (Γ ▷ A) ==> (c [ ↓ A ]) ::   (Π-f {Γ ▷ A} (A [[ ↓ A ]]) (B [[ ↑ A (↓ A) ]])))
    -> Γ ==> lambda A B (app (A [[ ↓ A ]])
                             (B [[ ↑ A (↓ A) ]])
                             (c [ ↓ A ])
                             q2
                             (vv A)
                             q1)
             == c ::  Π-f {Γ} A B
P14-Π-eta-eq-gen {Γ} A B c p q1 q2 = Π-eta-eq-gen {Γ} A B c p q1 q2
```

4.*/ Summary of interpreted rules (cont.)

```
P15-Π-f-cong : {Γ : ctx}
--
      -> (A  A' : ty Γ)
      -> (p : Γ ==> A == A')
      -> (B : ty (Γ ▷ A))
      -> (B' : ty (Γ ▷ A'))
      -> (Γ ▷ A ==> B == (B' [[  subst-trp (ext-eq A A' p) ]]))
-- ---------------------
      -> Γ ==> Π-f A B == Π-f A' B'
--
P15-Π-f-cong = Π-f-cong
```

4.*/ Summary of interpreted rules (cont.)

```
I1-ID : {Γ : ctx}
--
     -> (A : ty Γ)
     -> (a : raw Γ)
     -> (p : Γ ==> a :: A)
     -> (b : raw Γ)
     -> (q : Γ ==> b :: A)
-- -----------------------------
     -> ty Γ

I1-ID = ID

I2-ID-i : {Γ : ctx}
--
     -> (A : ty Γ)
     -> (a : raw Γ)
     -> (p : Γ ==> a :: A)
-- --------------------------------------
     -> Γ ==> (rr {Γ} a) :: (ID A a p a p)
--
I2-ID-i = ID-i
```

4.*/ Summary of interpreted rules (cont.)

```
I3-ID-e : {Γ : ctx}
--
     -> (A : ty Γ)
     -> (a b t : raw Γ)
     -> (p : Γ ==> a :: A)
     -> (q : Γ ==> b :: A)
     -> (Γ ==> t :: (ID A a p b q))
-- -----------------------------------------
     -> Γ ==> a == b :: A
--
I3-ID-e = ID-e

I4-ID-uip : {Γ : ctx}
--
     -> (A : ty Γ)
     -> (a b t : raw Γ)
     -> (p : Γ ==> a :: A)
     -> (r : Γ ==> a :: A)
     -> (Γ ==> t :: (ID A a p a r))
-- ------------------------------------------
     -> Γ ==> t == (rr {Γ} a) :: (ID A a p a r)
--
I4-ID-uip = ID-uip
```

4.*/ Summary of interpreted rules (cont.)

```
I5-rr-sub :  {Δ Γ : ctx}
--
     -> (h : subst Δ Γ)
     -> (a : raw Γ)
-- ---------------------------------------------
     -> << Raw Δ >> (rr a) [ h ] ~ rr (a [ h ])
--
I5-rr-sub = rr-sub

I6-rr-cong :  {Γ : ctx}
--
     ->  (a  b : raw Γ)
     -> (<< Raw Γ >> a ~ b)
-- --------------------------------
     -> << Raw Γ >> rr a ~ rr b
--
I6-rr-cong = rr-cong
```

4.*/ Summary of interpreted rules (cont.)

```
I7-ID-sub :  {Δ Γ : ctx}
--
     -> (h : subst Δ Γ)
     -> (A : ty Γ)
     -> (a : raw Γ)
     -> (pa : Γ ==> a :: A)
     -> (b : raw Γ)
     -> (pb : Γ ==> b :: A)
-- ---------------------------------------------------------------------------
     -> << Ty Δ >> (ID A a pa b pb) [[ h ]]
             ~ (ID (A [[ h ]]) (a [ h ]) (elt-subst h pa) (b [ h ]) (elt-subst h pb))
--
I7-ID-sub = ID-sub

-- More general rule where RHS doesn't depend on elt-subst:


I8-ID-sub-gen :  {Δ Γ : ctx}
     -> (h : subst Δ Γ)
     -> (A : ty Γ)
     -> (a : raw Γ)
     -> (pa : Γ ==> a :: A)
     -> (b : raw Γ)
     -> (pb : Γ ==> b :: A)
     -> (q : Δ ==> a [ h ] :: A [[ h ]])
     -> (r : Δ ==> b [ h ] :: A [[ h ]])
     -> << Ty Δ >> (ID A a pa b pb) [[ h ]] ~ (ID (A [[ h ]]) (a [ h ]) q (b [ h ]) r)
I8-ID-sub-gen = ID-sub-gen
```

4.*/ Summary of interpreted rules (cont.)

```
I9-ID-cong :  {Γ : ctx}
--
     -> (A A' : ty Γ)
     -> (a b a' b' : raw Γ)
     -> (<< Ty Γ >> A ~ A')
     -> (<< Raw Γ >> a ~ a')
     -> (<< Raw Γ >> b ~ b')
     -> (pa : Γ ==> a :: A)
     -> (pa' : Γ ==> a' :: A')
     -> (pb : Γ ==> b :: A)
     -> (pb' : Γ ==> b' :: A')
-- ------------------------------------------
     -> << Ty Γ >> ID A a pa b pb ~  ID A' a' pa' b' pb'
--
I9-ID-cong = ID-cong
```

4.*/ Summary of interpreted rules (cont.)

```
Sg1-Σ-f :  {Γ : ctx}
--
       -> (A : ty Γ)   -> (B : ty (Γ ▷ A))
-- ------------------------------------------
       -> ty Γ
--
Sg1-Σ-f = Σ-f

Sg2-Σ-f-cong : {Γ : ctx}
--
      -> (A  A' : ty Γ)
      -> (p : Γ ==> A == A')
      -> (B : ty (Γ ▷ A))
      -> (B' : ty (Γ ▷ A'))
      -> (Γ ▷ A ==> B == (B' [[  subst-trp (ext-eq A A' p) ]]))
-- -------------------------------------------------------------
      -> Γ ==> Σ-f A B == Σ-f A' B'
--
Sg2-Σ-f-cong = Σ-f-cong
```

4.*/ Summary of interpreted rules (cont.)

```
Sg3-Σ-i : {Γ : ctx}
--
      -> (A  : ty Γ)
      -> (B : ty (Γ ▷ A))
      -> (a  : raw Γ)
      -> (p : Γ ==> a :: A)
      -> (b : raw Γ)
      -> (Γ ==> b :: (B [[ els p ]]))
-- --------------------------------------
      -> Γ ==> pr a b :: Σ-f A B
--
Sg3-Σ-i = Σ-i
```

4.*/ Summary of interpreted rules (cont.)

```
Sg4-Σ-e-1 : {Γ : ctx}
--
      -> (A  : ty Γ)
      -> (B : ty (Γ ▷ A))
      -> (c  : raw Γ)
      -> (p : Γ ==> c :: Σ-f A B)
-- -------------------------------------------
      -> Γ ==> pr1 A B c p :: A
--
Sg4-Σ-e-1  = Σ-e-1

Sg5-Σ-e-2 : {Γ : ctx}
--
      -> (A  : ty Γ)
      -> (B : ty (Γ ▷ A))
      -> (c  : raw Γ)
      -> (p : Γ ==> c :: Σ-f A B)
      -> (q : Γ ==> pr1 A B c p :: A)
-- --------------------------------------------
      -> Γ ==> pr2 A B c p :: B [[ els q ]]
--
Sg5-Σ-e-2 = Σ-e-2
```

4.*/ Summary of interpreted rules (cont.)

```
Sg6-Σ-c-1 : {Γ : ctx}
--
      -> (A  : ty Γ)
      -> (B : ty (Γ ▷ A))
      -> (a  : raw Γ)
      -> (p : Γ ==> a :: A)
      -> (b : raw Γ)
      -> (Γ ==> b :: (B [[ els p ]]))
      -> (q : Γ ==> pr a b :: Σ-f A B)
-- ----------------------------------------------
      -> Γ ==> pr1 A B (pr a b) q == a :: A
--
Sg6-Σ-c-1 = Σ-c-1

Sg7-Σ-c-2 : {Γ : ctx}
--
      -> (A  : ty Γ)
      -> (B : ty (Γ ▷ A))
      -> (a  : raw Γ)
      -> (p : Γ ==> a :: A)
      -> (b : raw Γ)
      -> (Γ ==> b :: (B [[ els p ]]))
      -> (q : Γ ==> pr a b :: Σ-f A B)
-- ----------------------------------------------------------------
      ->  Γ ==> pr2 A B (pr a b) q == b  :: (B [[ els p ]])
--
Sg7-Σ-c-2 = Σ-c-2
```

4.*/ Summary of interpreted rules (cont.)

```
Sg8-Σ-c-eta : {Γ : ctx}
--
      -> (A  : ty Γ)
      -> (B : ty (Γ ▷ A))
      -> (c  : raw Γ)
      -> (p : Γ ==> c :: Σ-f A B)
-- -----------------------------------------------------------------
      ->  Γ ==> c == pr (pr1 A B c p) (pr2 A B c p) :: Σ-f A B
--
Sg8-Σ-c-eta = Σ-c-eta
```

4.*/ Summary of interpreted rules (cont.)

```
-- Natural numbers


N1-Nat-i-0 : (Γ : ctx) ->
--  -------------------------------------------
          Γ ==> zero Γ :: Nat Γ
--
N1-Nat-i-0 =  Nat-i-0



N2-Nat-i-s : (Γ : ctx)
        -> (a : raw Γ)
        -> (Γ ==> a :: Nat Γ)
--  -------------------------------------------
       ->  Γ ==> succ Γ a :: Nat Γ
--
N2-Nat-i-s = Nat-i-s
```

4.*/ Summary of interpreted rules (cont.)

```
N3-Nat-e : {Γ : ctx}
      -> (C : ty (Γ ▷ Nat Γ))
      -> (d : raw Γ)
      -> (p : Γ ==> d :: C [[ els (Nat-i-0 Γ) ]])
      -> (e : raw ((Γ ▷ Nat Γ) ▷ C))
      -> (q : (Γ ▷ Nat Γ) ▷ C  ==> e :: C [[ step-sub Γ ]] [[ ↓ C ]])
      -> (c : raw Γ)
      -> (r : Γ ==> c :: Nat Γ)
      -> Γ ==> Rec C d p e q c r :: C [[ els r ]]
N3-Nat-e = Nat-e
```

4.*/ Summary of interpreted rules (cont.)

```
N4-Nat-c-0 : {Γ : ctx}
      -> (C : ty (Γ ▷ Nat Γ))
      -> (d : raw Γ)
      -> (p : Γ ==> d :: C [[ els (Nat-i-0 Γ) ]])
      -> (e : raw ((Γ ▷ Nat Γ) ▷ C))
      -> (q : (Γ ▷ Nat Γ) ▷ C  ==> e :: C [[ step-sub Γ ]] [[ ↓ C ]])
      -> Γ ==> Rec C d p e q (zero Γ) (Nat-i-0 Γ) == d :: C [[ els (Nat-i-0 Γ)]]
N4-Nat-c-0 = Nat-c-0

N5-Nat-c-s : {Γ : ctx}
      -> (C : ty (Γ ▷ Nat Γ))
      -> (d : raw Γ)
      -> (p : Γ ==> d :: C [[ els (Nat-i-0 Γ) ]])
      -> (e : raw ((Γ ▷ Nat Γ) ▷ C))
      -> (q : (Γ ▷ Nat Γ) ▷ C  ==> e :: C [[ step-sub Γ ]] [[ ↓ C ]])
      -> (a : raw Γ)
      -> (r : Γ ==> a :: Nat Γ)
      -> Γ ==> Rec C d p e q (succ Γ a) (Nat-i-s Γ a r)
           == e [  ext C (els r) (Rec C d p e q a r) (Nat-e {Γ} C d p e q a r) ] :: C [[ els (Nat-i-s Γ a
N5-Nat-c-s = Nat-c-s
```

4.*/ Summary of interpreted rules (cont.)

```
N6-Rec-cong : {Γ : ctx}
      -> (C C’ : ty (Γ ▷ Nat Γ))
      -> (d d’ : raw Γ)
      -> (p : Γ ==> d :: C [[ els (Nat-i-0 Γ) ]])
      -> (p’ : Γ ==> d’ :: C’ [[ els (Nat-i-0 Γ) ]])
      -> (e : raw ((Γ ▷ Nat Γ) ▷ C))
      -> (e’ : raw ((Γ ▷ Nat Γ) ▷ C’))
      -> (q : (Γ ▷ Nat Γ) ▷ C  ==> e :: C [[ step-sub Γ ]] [[ ↓ C ]])
      -> (q’ : (Γ ▷ Nat Γ) ▷ C’  ==> e’ :: C’ [[ step-sub Γ ]] [[ ↓ C’ ]])
      -> (c c’ : raw Γ)
      -> (r : Γ ==> c :: Nat Γ)
      -> (r’ : Γ ==> c’ :: Nat Γ)
      -> (Cq : (Γ ▷ Nat Γ)  ==> C == C’)
      -> << Raw Γ >> d ~ d’
      -> << Raw ((Γ ▷ Nat Γ) ▷ C) >> e ~ (e’ [ subst-trp (ext-eq’ {Γ ▷ Nat Γ} C C’ Cq) ])
      -> << Raw Γ >> c ~ c’
      -> << Raw Γ >>  Rec {Γ} C d p e q c r ~ Rec {Γ} C’ d’ p’ e’ q’ c’ r’
N6-Rec-cong  =   Rec-cong
```

4.*/ Summary of interpreted rules (cont.)

```
N7-Nat-sub :  (Δ Γ : ctx) -> (f : subst Δ Γ)
    -> Δ ==> (Nat Γ) [[ f ]] == (Nat Δ)
N7-Nat-sub = Nat-sub

N8-zero-sub : {Δ Γ : ctx} -> (f : subst Δ Γ)
        -> << Raw Δ >>  (zero Γ [ f ]) ~ (zero Δ)
N8-zero-sub = zero-sub

N9-succ-sub : {Δ Γ : ctx} -> (f : subst Δ Γ) -> (a : raw Γ)
        -> << Raw Δ >>  ((succ Γ a) [ f ]) ~ (succ Δ (a [ f ]))
N9-succ-sub  = succ-sub
```

4.*/ Summary of interpreted rules (cont.)

```
N10-Rec-sub : {Δ Γ : ctx}
      -> (f : subst Δ Γ)
      -> (C : ty (Γ ▷ Nat Γ))
      -> (d : raw Γ)
      -> (p : Γ ==> d :: C [[ els (Nat-i-0 Γ) ]])
      -> (p' : Δ ==> (d [ f ]) :: C [[  N-sub f  ]] [[ els (Nat-i-0 Δ) ]])
      -> (e : raw ((Γ ▷ Nat Γ) ▷ C))
      -> (q  : (Γ ▷ Nat Γ) ▷ C  ==> e :: C [[ step-sub Γ ]] [[ ↓ C ]])
      -> (q' : (Δ ▷ Nat Δ ) ▷ (C [[ N-sub f ]])  ==> e [ C-sub f C ] ::
           C [[  N-sub f  ]] [[ step-sub Δ ]] [[ ↓ ( C [[  N-sub f  ]]) ]])
      -> (c : raw Γ)
      -> (r : Γ ==> c :: Nat Γ)
      -> (r' : Δ ==> (c [ f ]) :: (Nat Γ) [[ f ]])
      -> << Raw Δ >> ((Rec {Γ} C d p e q c r) [f ])
                   ~ (Rec {Δ} (C [[  N-sub f  ]]) (d [ f ]) p' (e [ C-sub f C ])  q' (c [ f ]) r')
N10-Rec-sub  = Rec-sub
```

4.*/ Summary of interpreted rules (cont.)

```
Empty1-N0-sub :  (Δ Γ : ctx) -> (f : subst Δ Γ)
    -> Δ ==> (N0 Γ) [[ f ]] == (N0 Δ)
Empty1-N0-sub = N0-sub

Empty2-N0-e : {Γ : ctx}
      -> (C : ty (Γ ▷ N0 Γ))
      -> (c : raw Γ)
      -> (r : Γ ==> c :: N0 Γ)
      -> Γ ==> R0 C c r :: C [[ els r ]]
Empty2-N0-e = N0-e
```

4.*/ Summary of interpreted rules (cont.)

```
Empty3-R0-sub : {Δ Γ : ctx}
      -> (f : subst Δ Γ)
      -> (C : ty (Γ ▷ N0 Γ))
      -> (c : raw Γ)
      -> (r : Γ ==> c :: N0 Γ)
      -> (r' : Δ ==> c [ f ] :: (N0 Γ) [[ f ]])
      -> << Raw Δ >>  ((R0 {Γ} C c r) [[ f ]])  ~  (R0 {Δ} (C [[ ↑ (N0 Γ) f ]]) (c [ f ]) r')
Empty3-R0-sub = R0-sub

Empty4-R0-cong : {Γ : ctx}
      -> (C C' : ty (Γ ▷ N0 Γ))
      -> (c c' : raw Γ)
      -> (r : Γ ==> c :: N0 Γ)
      -> (r' : Γ ==> c' :: N0 Γ)
      -> (Cq : (Γ ▷ N0 Γ)  ==> C == C')
      -> << Raw Γ >> c ~ c'
      -> << Raw Γ >>  R0 {Γ} C c r ~ R0 {Γ} C' c' r'
Empty4-R0-cong = R0-cong
```

4.*/ Summary of interpreted rules (cont.)

```
Sm1-Sum : {Γ : ctx}
--
   -> (A : ty Γ)    -> (B : ty Γ)
-- ------------------------------
   -> ty Γ
--
Sm1-Sum = Sum

Sm2-Sum-cong : {Γ : ctx}
--
   -> (A A' : ty Γ)
   -> (B B' : ty Γ)
   -> Γ ==> A == A'  -> Γ ==> B == B'
-- ------------------------------
   -> Γ ==> Sum A B == Sum A' B'
--
Sm2-Sum-cong = Sum-cong
```

4.*/ Summary of interpreted rules (cont.)

```
Sm3-Sum-sub : {Δ Γ : ctx}
--
   -> (f : subst Δ Γ)
   -> (A : ty Γ)
   -> (B : ty Γ)
-- ---------------------------------------------------------
   -> Δ ==> (Sum A B) [[ f ]] == Sum (A [[ f ]]) (B [[ f ]])
--
Sm3-Sum-sub  = Sum-sub
```

4.*/ Summary of interpreted rules (cont.)

```
Sm4-lf-pf : {Γ : ctx}
--
   -> (A B : ty Γ)
   -> (a : raw Γ)
   -> (p : Γ ==> a :: A)
-- --------------------------------
   -> Γ ==> lf A B a p :: Sum A B
--
Sm4-lf-pf = lf-pf


Sm5-rg-pf : {Γ : ctx}
--
   -> (A B : ty Γ)
   -> (b : raw Γ)
   -> (p : Γ ==> b :: B)
-- --------------------------------
   -> Γ ==> rg A B b p :: Sum A B
--
Sm5-rg-pf = rg-pf
```

4.*/ Summary of interpreted rules (cont.)

```
Sm6-lf-cong : {Γ : ctx}
--
   -> (A B A’ B’ : ty Γ)
   -> (a a’ : raw Γ)
   -> (p : Γ ==> a :: A)
   -> (p’ : Γ ==> a’ :: A’)
   -> (q : Γ ==> A == A’)
   -> (r : Γ ==> B == B’)
   -> << Raw Γ >>  a ~ a’
-- -------------------------------------------------------
   -> << Raw Γ >> (lf {Γ} A B a p) ~ (lf {Γ} A’ B’ a’ p’)
--
Sm6-lf-cong = lf-cong


Sm7-lf-sub : {Δ Γ : ctx}
   -> (f : subst Δ Γ)
   -> (A B : ty Γ)
   -> (a : raw Γ)
   -> (p : Γ ==> a :: A)
   -> (p’ : Δ ==> a [ f ] :: A [[ f ]])
   -> << Raw Δ >> (lf A B a p) [ f ] ~ lf (A [[ f ]])  (B [[ f ]]) (a [ f ]) p’
Sm7-lf-sub = lf-sub
```

4.*/ Summary of interpreted rules (cont.)

```
Sm8-rg-cong : {Γ : ctx}
--
   -> (A B A' B' : ty Γ)
   -> (b b' : raw Γ)
   -> (p : Γ ==> b :: B)
   -> (p' : Γ ==> b' :: B')
   -> (q : Γ ==> A == A')
   -> (r : Γ ==> B == B')
   -> << Raw Γ >>  b ~ b'
-- ---------------------------------------------------------
   -> << Raw Γ >> (rg {Γ} A B b p) ~ (rg {Γ} A' B' b' p')
--
Sm8-rg-cong = rg-cong


Sm9-rg-sub : {Δ Γ : ctx}
   -> (f : subst Δ Γ)
   -> (A B : ty Γ)
   -> (b : raw Γ)
   -> (p : Γ ==> b :: B)
   -> (p' : Δ ==> b [ f ] :: B [[ f ]])
   -> << Raw Δ >> (rg A B b p) [ f ] ~ rg (A [[ f ]])  (B [[ f ]]) (b [ f ]) p'
Sm9-rg-sub = rg-sub
```

4.*/ Summary of interpreted rules (cont.)

```
Sm10-Sum-e : {Γ : ctx}
--
      -> (A B : ty Γ)
      -> (C : ty (Γ ▷ (Sum A B)))
      -> (d : raw (Γ ▷ A))
      -> (p : (Γ ▷ A) ==> d :: C [[ Sum-sub-lf A B ]])
      -> (e : raw (Γ ▷ B))
      -> (q : (Γ ▷ B) ==> e :: C [[ Sum-sub-rg A B ]])
      -> (c : raw Γ)
      -> (r : Γ ==> c :: Sum A B)
-- ---------------------------------------------------------
      -> Γ ==> Sum-rec A B C d p e q c r :: C [[ els r ]]
--
Sm10-Sum-e = Sum-e
```

4.*/ Summary of interpreted rules (cont.)

```
Sm11-Sum-c1 : {Γ : ctx}
--
       -> (A B : ty Γ)
       -> (C : ty (Γ ▷ (Sum A B)))
       -> (d : raw (Γ ▷ A))
       -> (p : (Γ ▷ A) ==> d :: C [[ Sum-sub-lf A B ]])
       -> (e : raw (Γ ▷ B))
       -> (q : (Γ ▷ B) ==> e :: C [[ Sum-sub-rg A B ]])
       -> (a : raw Γ)
       -> (r : Γ ==> a :: A)
       -> (r' : Γ ==> (lf {Γ} A B a r) :: Sum A B)
-- ------------------------------------------------------------------------------------------
       -> Γ ==> Sum-rec A B C d p e q (lf {Γ} A B a r) r' == d [ els r ] :: C [[ els r' ]]
--
Sm11-Sum-c1 = Sum-c1
```

4.*/ Summary of interpreted rules (cont.)

```
Sm12-Sum-c2 : {Γ : ctx}
--
       -> (A B : ty Γ)
       -> (C : ty (Γ ▷ (Sum A B)))
       -> (d : raw (Γ ▷ A))
       -> (p : (Γ ▷ A) ==> d :: C [[ Sum-sub-lf A B ]])
       -> (e : raw (Γ ▷ B))
       -> (q : (Γ ▷ B) ==> e :: C [[ Sum-sub-rg A B ]])
       -> (b : raw Γ)
       -> (r : Γ ==> b :: B)
       -> (r' : Γ ==> (rg {Γ} A B b r) :: Sum A B)
-- ------------------------------------------------------------------------------------------
       -> Γ ==> Sum-rec A B C d p e q (rg {Γ} A B b r) r' == e [ els r ] :: C [[ els r'  ]]
--
Sm12-Sum-c2 =  Sum-c2
```

4.*/ Summary of interpreted rules (cont.)

```
Sm13-Sum-rec-cong : {Γ : ctx}
--
       -> (A B A' B' : ty Γ)
       -> (C : ty (Γ ▷ (Sum A B)))
       -> (C' : ty (Γ ▷ (Sum A' B')))
       -> (d : raw (Γ ▷ A))
       -> (d' : raw (Γ ▷ A'))
       -> (p : (Γ ▷ A) ==> d :: C [[ Sum-sub-lf A B ]])
       -> (p' : (Γ ▷ A') ==> d' :: C' [[ Sum-sub-lf A' B' ]])
       -> (e : raw (Γ ▷ B))
       -> (e' : raw (Γ ▷ B'))
       -> (q : (Γ ▷ B) ==> e :: C [[ Sum-sub-rg A B ]])
       -> (q' : (Γ ▷ B') ==> e' :: C' [[ Sum-sub-rg A' B' ]])
       -> (c : raw Γ)
       -> (c' : raw Γ)
       -> (r : Γ ==> c :: Sum A B)
       -> (r' : Γ ==> c' :: Sum A' B')
       -> (Aq : Γ ==> A == A')
       -> (Bq : Γ ==> B == B')
       -> (Γ ▷ (Sum A B)  ==> C == C' [[  subst-trp (ext-eq' {Γ} (Sum A B) (Sum A' B') (Sum-cong A A' B B
       -> << Raw (Γ ▷ A) >> d ~ (d' [ subst-trp (ext-eq' {Γ} A A' Aq) ])
       -> << Raw (Γ ▷ B) >> e ~ (e' [ subst-trp (ext-eq' {Γ} B B' Bq) ])
       -> << Raw Γ >> c ~ c'
-- ---------------------------------------------------------------------------------------------
       -> << Raw Γ >>  Sum-rec {Γ} A B C d p e q c r ~  Sum-rec {Γ} A' B' C' d' p' e' q' c' r'
Sm13-Sum-rec-cong = Sum-rec-cong
```

4.*/ Summary of interpreted rules (cont.)

```
Sm14-Sum-rec-sub : {Δ Γ : ctx}
       -> (f : subst Δ Γ)
       -> (A B : ty Γ)
       -> (C : ty (Γ ▷ (Sum A B)))
       -> (d : raw (Γ ▷ A))
       -> (p : (Γ ▷ A) ==> d :: C [[ Sum-sub-lf A B ]])
       -> (e : raw (Γ ▷ B))
       -> (q : (Γ ▷ B) ==> e :: C [[ Sum-sub-rg A B ]])
       -> (c : raw Γ)
       -> (r : Γ ==> c :: Sum A B)
       -> (p' : Δ ▷ _[[_]] {Δ} {Γ} A f ==>
                  _[_] {Δ ▷ _[[_]] {Δ} {Γ} A f} {Γ ▷ A} d (↑ {Δ} {Γ} A f) ::
                 _[[_]] {Δ ▷ _[[_]] {Δ} {Γ} A f}
                {Δ ▷ Sum {Δ} (_[[_]] {Δ} {Γ} A f) (_[[_]] {Δ} {Γ} B f)}
                  (_[[_]] {Δ ▷ _[[_]] {Δ} {Γ} (Sum {Γ} A B) f} {Γ ▷ Sum {Γ} A B} C
                  (↑ {Δ} {Γ} (Sum {Γ} A B) f))
                 (Sum-sub-lf {Δ} (_[[_]] {Δ} {Γ} A f) (_[[_]] {Δ} {Γ} B f)))
       -> (q' : Δ ▷ _[[_]] {Δ} {Γ} B f ==>
                _[_] {Δ ▷ _[[_]] {Δ} {Γ} B f} {Γ ▷ B} e (↑ {Δ} {Γ} B f) ::
                 _[[_]] {Δ ▷ _[[_]] {Δ} {Γ} B f}
             {Δ ▷ Sum {Δ} (_[[_]] {Δ} {Γ} A f) (_[[_]] {Δ} {Γ} B f)}
               (_[[_]] {Δ ▷ _[[_]] {Δ} {Γ} (Sum {Γ} A B) f} {Γ ▷ Sum {Γ} A B} C
                    (↑ {Δ} {Γ} (Sum {Γ} A B) f))
               (Sum-sub-rg {Δ} (_[[_]] {Δ} {Γ} A f) (_[[_]] {Δ} {Γ} B f)))
       -> (r' : Δ ==> c [ f ] :: Sum (A [[ f ]]) (B [[ f ]]))
       -> << Raw Δ >> ((Sum-rec {Γ} A B C d p e q c r) [ f ])
                    ~  (Sum-rec {Δ} (A [[ f ]]) (B [[ f ]]) (C [[ ↑ {Δ} {Γ} (Sum A B) f ]])
                              (d [ ↑  A f ]) p' (e [ ↑ B f ]) q' (c [ f ]) r')
Sm14-Sum-rec-sub = Sum-rec-sub
```

4.*/ Summary of interpreted rules (cont.)

```
-- Universe à la Russell

U1-T-f-Russell-eq :  {Γ : ctx}
     -> (A : raw Γ)
     -> (p : Γ ==> A :: U-f Γ)
-- -------------------------------
     ->  Γ ==> T-f A p == A
--
U1-T-f-Russell-eq = T-f-Russell-eq


U2-U-f-Russell-nat : (Γ : ctx)
-- -----------------------------------
    -> Γ ==> Nat Γ :: U-f Γ
--
U2-U-f-Russell-nat = U-f-Russell-nat
```

4.*/ Summary of interpreted rules (cont.)

```
U3-U-f-Russell-sigma : {Γ : ctx}
--
    ->  (A : ty Γ)
    ->  (p : Γ ==> A :: U-f Γ)
    ->  (B : ty (Γ ▷ A))
    ->  (q : (Γ ▷ A) ==> B :: U-f Γ [[ ↓ A ]])
-- --------------------------------------------
    ->  Γ ==> Σ-f A B :: U-f Γ
--
U3-U-f-Russell-sigma  = U-f-Russell-sigma
```

4.*/ Summary of interpreted rules (cont.)

```
U4-U-f-Russell-pi : {Γ : ctx}
--
    ->  (A : ty Γ)
    ->  (p : Γ ==> A :: U-f Γ)
    ->  (B : ty (Γ ▷ A))
    ->  (q : (Γ ▷ A) ==> B :: U-f Γ [[ ↓ A ]])
-- -----------------------------------------------
    ->  Γ ==> Π-f A B :: U-f Γ
--
U4-U-f-Russell-pi = U-f-Russell-pi
```

4.*/ Summary of interpreted rules (cont.)

```
U6-U-f-Russell-N0 : {Γ : ctx}
-- ---------------------------
     -> Γ ==> N0 Γ :: U-f Γ
--
U6-U-f-Russell-N0  = U-f-Russell-N0
```

4.*/ Summary of interpreted rules (cont.)

```
U7-U-f-Russell-Sum : {Γ : ctx}
--
    ->  (A B : ty Γ)
    ->  (p : Γ ==> A :: U-f Γ)
    ->  (q : Γ ==> B :: U-f Γ)
-- ------------------------------
    ->  Γ ==> Sum A B :: U-f Γ
--
U7-U-f-Russell-Sum = U-f-Russell-Sum
```

4.*/ Summary of interpreted rules (cont.)

```
U8-U-f-sub :  (Δ Γ : ctx)
      -> (f : subst Δ Γ)
-- -------------------------------------
     ->  Δ ==> (U-f Γ) [[ f ]] == U-f Δ
--
U8-U-f-sub = U-f-sub
```

5.1/ Interpretation of the universe à la Russell

Thanks to its inductive-recursive definitions Agda admits universes (and universe operators, and superuniverses, and ... [P. 1998]).

```
mutual
  data U : Set where
     n₀  : U
     n₁  : U
     _⊕_  : U -> U -> U
     _⊗_  : U -> U -> U
     σ  : (a : U) -> (T a -> U) -> U
     π  : (a : U) -> (T a -> U) -> U
     n  : U
     w  : (a : U) -> (T a -> U) -> U
     id : (a : U) -> T a -> T a -> U

  T : U -> Set
  T n₀         = N₀
  T n₁         = N₁
  T (a ⊕ b)    = T a + T b
  T (a ⊗ b)    = prod (T a) (T b)
  T (σ a b)    = Σ (T a) (\x -> T (b x))
  T (π a b)    = (x  : T a) -> T (b x)
  T n          = N
  T (w a b)    = W (T a) (\x -> T (b x))
  T (id a b c) = Id (T a) b c
```

5.2/ Interpretation of the universe à la Russell

Small version of V based on the TT universe U, T

```
data sV  : Set where
   ssup : (A : U) -> (f : T A -> sV) -> sV

eq-sV : sV -> sV -> U
eq-sV (ssup A f) (ssup B g) =
           (π A (\x -> σ B (\y -> eq-sV (f x) (g y))))
            ⊗
           (π B (\y -> σ A (\x -> eq-sV (f x) (g y))))

emb : sV -> V
emb (ssup A f) = sup (T A) (\x -> emb (f x))
```

5.3/ Interpretation of the universe à la Russell ... (cont.)

```
uV : V
uV = sup sV emb

U-f  :  (Γ : ctx)  -> ty Γ
U-f  Γ = Const Γ uV

T-f : {Γ : ctx}  -> (A : raw Γ)
     -> (Γ ==> A :: U-f Γ)
-- --------------------------
     -> ty Γ
T-f {Γ} A p =  tyy (raw.rawterm A)

... (as in previous section)
```

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 development is available at:

http://staff.math.su.se/palmgren/MLTT-and-setoids.zip