1. The Type Theory of Sett

We denote variables x, constants c, expressions e, f, α, β, and contexts Γ.

$$e, f, \alpha, \beta ::= x \mid c \mid \Pi x : \alpha.\, e \mid \lambda x : \alpha.\, e \mid f\ e \mid \mathsf{Type}$$
$$\Gamma ::= \cdot \mid \Gamma, x : e$$

There are four kinds of judgement:

$$\Gamma\ \mathsf{ctx} \qquad \Gamma \vdash \alpha\ \mathsf{sort} \qquad \Gamma \vdash e : \alpha \qquad \Gamma \vdash e \equiv e'$$

The empty context is well-formed, and contexts may be extended with variables:

$$\frac{}{\cdot\ \mathsf{ctx}} \qquad \frac{\Gamma \vdash \alpha\ \mathsf{sort}}{(\Gamma, x : \alpha)\ \mathsf{ctx}}\ (\text{where } x \notin \Gamma)$$

Every type is a sort, as is Type itself:

$$\frac{\Gamma \vdash \alpha : \mathsf{Type}}{\Gamma \vdash \alpha\ \mathsf{sort}} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathsf{Type}\ \mathsf{sort}}$$

Typing judgements are invariant under context extensions, and may be derived from Γ and $\equiv$.

$$\frac{(\Gamma, x : \alpha)\ \mathsf{ctx} \quad \Gamma \vdash e : \beta}{\Gamma, x : \alpha \vdash e : \beta} \qquad \frac{\Gamma \vdash \alpha\ \mathsf{sort}}{\Gamma, x : \alpha \vdash x : \alpha} \qquad \frac{\Gamma \vdash e : \alpha \quad \Gamma \vdash \alpha \equiv \beta}{\Gamma \vdash e : \beta}$$

Judgemental equality is defined inductively for (valid) expressions.

$$\frac{\Gamma \vdash x : \alpha}{\Gamma \vdash x \equiv x} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash c \equiv c} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathsf{Type} \equiv \mathsf{Type}} \qquad \frac{\Gamma \vdash f \equiv f' \quad \Gamma \vdash e \equiv e' \quad \Gamma \vdash f\ e : \alpha}{\Gamma \vdash f\ e \equiv f'\ e'}$$

$$\frac{\Gamma \vdash \alpha \equiv \alpha' \quad \Gamma, x : \alpha \vdash e \equiv e' \quad \Gamma \vdash \Pi x : \alpha.\, e\ \mathsf{sort}}{\Gamma \vdash \Pi x : \alpha.\, e \equiv \Pi x : \alpha'.\, e'} \qquad \frac{\Gamma \vdash \alpha \equiv \alpha' \quad \Gamma, x : \alpha \vdash e \equiv e' \quad \Gamma \vdash \lambda x : \alpha.\, e : \beta}{\Gamma \vdash \lambda x : \alpha.\, e \equiv \lambda x : \alpha'.\, e'}$$

$$\frac{\Gamma \vdash (\lambda x : \alpha.\, e)\ e' : \beta}{\Gamma \vdash (\lambda x : \alpha.\, e)\ e' \equiv e[e'/x]} \qquad \frac{\Gamma \vdash (\lambda x : \alpha.\, e)\ e' : \beta}{\Gamma \vdash e[e'/x] \equiv (\lambda x : \alpha.\, e)\ e'}$$

Hence, judgemental equality is reflexive, symmetric and transitive. The last two rules define our only reduction rule in this type theory, β-reduction.

Next we define the typing rules of functions: small Π-types live in Type, but large ones are only sorts; they are introduced by λ, and eliminated by application.

$$\frac{\Gamma \vdash \alpha : \mathsf{Type} \quad \Gamma, x : \alpha \vdash e : \mathsf{Type}}{\Gamma \vdash \Pi x : \alpha.\, e : \mathsf{Type}} \qquad \frac{\Gamma, x : \alpha \vdash e\ \mathsf{sort}}{\Gamma \vdash \Pi x : \alpha.\, e\ \mathsf{sort}}$$

$$\frac{\Gamma, x : \alpha \vdash e : \beta}{\Gamma \vdash \lambda x : \alpha.\, e : \Pi x : \alpha.\, \beta} \qquad \frac{\Gamma \vdash f : \Pi x : \alpha.\, \beta \quad \Gamma \vdash e : \alpha}{\Gamma \vdash f\ e : \beta[e/x]}$$

Now we introduce the 22 constants of the language.

$$\begin{aligned} c ::=\ & \mathsf{Eq} \mid \mathsf{refl} \mid \mathsf{elim}_{=} \mid \mathsf{elim\_refl} \mid \mathsf{K} \mid \mathsf{funext} \\ & \mid \mathsf{Sigma} \mid \mathsf{pair} \mid \mathsf{elim}_{\Sigma} \mid \mathsf{elim\_pair} \\ & \mid \mathbb{P} \mid \mathsf{El}_{\mathbb{P}} \mid \mathsf{propresize} \\ & \mid \mathsf{Tree} \mid \mathsf{node} \mid \mathsf{branch} \mid \mathsf{elim}_{\mathrm{T}} \mid \mathsf{elim\_node} \mid \mathsf{elim\_branch} \\ & \mid \mathsf{large\_elim}_{\mathrm{T}} \mid \mathsf{large\_elim\_node} \mid \mathsf{large\_elim\_branch} \end{aligned}$$

We add some notation for convenience:

- $\Pi(x_0 : \alpha_0)...(x_n : \alpha_n).\ \beta := \Pi x_0 : \alpha_0....\Pi x_n : \alpha_n.\ \beta$;
- $\Pi(x_0...x_n : \alpha).\ \beta := \Pi(x_0 : \alpha)...(x_n : \alpha).\ \beta$;
- $\Sigma x : \alpha.\ \beta := \mathsf{Sigma}\ \alpha\ (\lambda x : \alpha.\ \beta)$ and $\mathrm{T}x : \alpha.\ \beta := \mathsf{Tree}\ \alpha\ (\lambda x : \alpha.\ \beta)$;
- $\alpha \to \beta := \Pi x : \alpha.\ \beta$, and likewise $\alpha \times \beta := \Sigma x : \alpha.\ \beta$, where x is not free in β;
- $e = e' := \mathsf{Eq}\ \alpha\ e\ e'$, where $e : \alpha$;
- we omit function parameters in applications when they can be inferred;
- the types of binders may similarly be omitted when inferrable.

Each constant has a typing rule. For brevity, they are expressed below as a list; each entry can be taken as a judgement true for the empty context. We begin with the axioms of equality, with the type former Eq, its reflexivity, the principle of based path induction, and its reduction rule; we postulate axiom K (the unicity of identity proofs) and well as function extensionality.

- $\mathsf{Eq} : \Pi\alpha : \mathsf{Type}.\ \alpha \to \alpha \to \mathsf{Type}$
- $\mathsf{refl} : \Pi(\alpha : \mathsf{Type})(a : \alpha).\ a = a$
- $\mathsf{elim}_= : \Pi(\alpha : \mathsf{Type})(a : \alpha)(C : \Pi b.\ a = b \to \mathsf{Type})(h : C\ a\ (\mathsf{refl}\ a))(b : \alpha)(t : a = b).\ C\ b\ t$
- $\mathsf{elim\_refl} : \Pi(\alpha : \mathsf{Type})(a : \alpha)(C : \Pi b.\ a = b \to \mathsf{Type})(h : C\ a\ (\mathsf{refl}\ a)).\ \mathsf{elim}_=\ h\ (\mathsf{refl}\ a) = h$
- $\mathsf{K} : \Pi(\alpha : \mathsf{Type})(a\ b : \alpha)(h\ h' : a = b).\ h = h'$
- $\mathsf{funext} : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type})(f\ g : \Pi a.\ \beta\ a)(h : \Pi x.\ f\ x = g\ x).\ f = g$

Next is the Σ-type, defined for simplicity in terms of an eliminator rather than projections.

- $\mathsf{Sigma} : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type}).\ \mathsf{Type}$
- $\mathsf{pair} : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type})(a : \alpha)(b : \beta\ a).\ \mathsf{Sigma}\ \alpha\ \beta$
- $\mathsf{elim}_\Sigma : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type})(C : \mathsf{Sigma}\ \alpha\ \beta \to \mathsf{Type})(f : \Pi a\ b.\ C\ (\mathsf{pair}\ a\ b))$
  $(t : \mathsf{Sigma}\ \alpha\ \beta).\ C\ t$
- $\mathsf{elim\_pair} : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type})(C : \mathsf{Sigma}\ \alpha\ \beta \to \mathsf{Type})(f : \Pi a\ b.\ C\ (\mathsf{pair}\ a\ b))$
  $(a : \alpha)(b : \beta\ a).\ \mathsf{elim}_\Sigma\ f\ (\mathsf{pair}\ a\ b) = f\ a\ b$

We postulate a small Tarski universe $\mathbb{P}$, intended as a universe of propositions.

- $\mathbb{P} : \mathsf{Type}$
- $\mathsf{El}_\mathbb{P} : \mathbb{P} \to \mathsf{Type}$

For the next few typing rules, we first must make some standard definitions.

- $\mathsf{IsProp}\ \alpha := \Pi(a\ b : \alpha).\ a = b$; a proposition is a type with no more than one element.
- $\mathsf{Prop} := \Sigma P : \mathbb{P}.\ \mathsf{IsProp}\ (\mathsf{El}_\mathbb{P}\ P) \times (\Pi(Q : \mathbb{P}).\ \mathsf{IsProp}\ (\mathsf{El}_\mathbb{P}\ Q) \to$
  $(\mathsf{El}_\mathbb{P}\ P \to \mathsf{El}_\mathbb{P}\ Q) \to (\mathsf{El}_\mathbb{P}\ Q \to \mathsf{El}_\mathbb{P}\ P) \to Q = P)$
  As given by the axioms, the basic $\mathbb{P}$ type is "too large": there is no guarantee that any element of it is *actually* a proposition. Hence, we restrict the set to its elements we care about: the type of propositions is those elements of $\mathbb{P}$ that are extensional propositions.
- $[\alpha] := \Pi P : \mathsf{Prop}.\ (\alpha \to \mathsf{El}_\mathbb{P}\ P) \to \mathsf{El}_\mathbb{P}\ P$
  The bracket type $[\alpha]$, also known as the *propositional truncation* of α, is inhabited precisely when α is, and has at most one element. Propositional resizing enables us to use a Church encoding.
- $\exists x : \alpha.\ \beta\ a := [\Sigma x : \alpha.\ \beta\ a]$
- $\exists! x : \alpha.\ \beta\ a := (\Sigma x : \alpha.\ \beta\ a) \times \mathsf{IsProp}\ (\Sigma x : \alpha.\ \beta\ a)$
- $\mathsf{Injective}\ f := \Pi x\ y.\ f\ x = f\ y \to x = y$
- $\mathsf{Surjective}\ f := \Pi b.\ \exists a.\ f\ a = b$
- $\mathsf{Bijective}\ f := \mathsf{Injective}\ f \times \mathsf{Surjective}\ f$
- $\alpha \leftrightarrow \beta := \Sigma f : \alpha \to \beta.\ \mathsf{Bijective}\ f$

We are now ready to state the propositional resizing axiom, which gives $\mathbb{P}$ its power:

- $\mathsf{propresize} : \Pi(\alpha : \mathsf{Type}).\ \mathsf{IsProp}\ \alpha \to \exists! P : \mathbb{P}.\ \mathsf{El}_{\mathbb{P}}\ P \leftrightarrow \alpha$

In words, this states that for all propositions α, there is a unique member of $\mathbb{P}$ of the same cardinality. This tells us that the number of propositions is *small*, enabling impredicative constructs like the powerset.

We postulate the Tree type, which enables well-founded recursion. It is simiar in spirit to the well-known W-type, but is additionally allowed to be a node, which terminates the tree and stores no data. This bundles booleans and W-types into one, so we only need one large eliminator.

- $\mathsf{Tree} : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type}).\ \mathsf{Type}$
- $\mathsf{node} : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type}).\ \mathsf{Tree}\ \alpha\ \beta$
- $\mathsf{branch} : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type})(a : \alpha)(b : \beta\ a \to \mathsf{Tree}\ \alpha\ \beta).\ \mathsf{Tree}\ \alpha\ \beta$
- $\mathsf{elim}_{\mathrm{T}} : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type})(C : \mathsf{Tree}\ \alpha\ \beta \to \mathsf{Type})$
  $(h_1 : C\ \mathsf{node})(h_2 : \Pi a\ b.\ (\Pi i.\ C\ (b\ i)) \to C\ (\mathsf{branch}\ a\ b))(t : \mathsf{Tree}\ \alpha\ \beta).\ C\ t$
- $\mathsf{elim\_node} : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type})(C : \mathsf{Tree}\ \alpha\ \beta \to \mathsf{Type})$
  $(h_1 : C\ \mathsf{node})(h_2 : \Pi a\ b.\ (\Pi i.\ C\ (b\ i)) \to C\ (\mathsf{branch}\ a\ b)).$
  $\mathsf{elim}_{\mathrm{T}}\ h_1\ h_2\ \mathsf{node} = h_1$
- $\mathsf{elim\_branch} : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type})(C : \mathsf{Tree}\ \alpha\ \beta \to \mathsf{Type})$
  $(h_1 : C\ \mathsf{node})(h_2 : \Pi a\ b.\ (\Pi i.\ C\ (b\ i)) \to C\ (\mathsf{branch}\ a\ b))$
  $(a : \alpha)(b : \beta\ a \to \mathsf{Tree}\ \alpha\ \beta).$
  $\mathsf{elim}_{\mathrm{T}}\ h_1\ h_2\ (\mathsf{branch}\ a\ b) = h_2\ a\ b\ (\lambda i.\ \mathsf{elim}_{\mathrm{T}}\ h_1\ h_2\ (b\ i))$

Finally, we add support for large elimination to Tree, enabling the construction of certain large sets.

- $\mathsf{large\_elim}_{\mathrm{T}} : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type})$
  $(h_1 : \mathsf{Type})(h_2 : \Pi a.\ (\beta\ a \to \mathsf{Tree}\ \alpha\ \beta) \to (\beta\ a \to \mathsf{Type}) \to \mathsf{Type})$
  $(t : \mathsf{Tree}\ \alpha\ \beta).\ \mathsf{Type}$
- $\mathsf{large\_elim\_node} : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type})$
  $(h_1 : \mathsf{Type})(h_2 : \Pi a.\ (\beta\ a \to \mathsf{Tree}\ \alpha\ \beta) \to (\beta\ a \to \mathsf{Type}) \to \mathsf{Type}).$
  $\mathsf{large\_elim}_{\mathrm{T}}\ h_1\ h_2\ \mathsf{node} \leftrightarrow h_1$
- $\mathsf{large\_elim\_branch} : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type})$
  $(h_1 : \mathsf{Type})(h_2 : \Pi a.\ (\beta\ a \to \mathsf{Tree}\ \alpha\ \beta) \to (\beta\ a \to \mathsf{Type}) \to \mathsf{Type})$
  $(a : \alpha)(b : \beta\ a \to \mathsf{Tree}\ \alpha\ \beta).$
  $\mathsf{large\_elim}_{\mathrm{T}}\ h_1\ h_2\ (\mathsf{branch}\ a\ b) \leftrightarrow h_2\ a\ b\ (\lambda i.\ \mathsf{large\_elim}_{\mathrm{T}}\ h_1\ h_2\ (b\ i))$

2. Extra Axioms

The following axioms are not part of the base theory, but can be postulated to prove certain theorems. We first make some definitions:

- $\pi_1 \coloneqq \mathsf{elim}_{\Sigma}\ (\lambda a\ b.\ a)$, the first projection for Σ-types.
- $\mathsf{El} \coloneqq \lambda P : \mathsf{Prop}.\ \mathsf{El}_{\mathbb{P}}\ (\pi_1\ P)$
- $\mathbf{0} \coloneqq \Pi P : \mathsf{Prop}.\ \mathsf{El}\ P$, the type with no elements (Church-encoded with the statement “every proposition is true”).
- $\mathbf{1} \coloneqq \mathbf{0} \to \mathbf{0}$
- $\mathbf{2} \coloneqq \mathrm{T} i : \mathbf{1}.\ \mathbf{0}$, the type of booleans.
- $\mathsf{Sum}\ \alpha\ \beta \coloneqq \Sigma b : \mathbf{2}.\ \mathsf{large\_elim}_{\mathbf{2}}\ \alpha\ \beta\ b$ (the definition of $\mathsf{large\_elim}_{\mathbf{2}}$ is not given, but is trivial).
- $\alpha \lor \beta \coloneqq [\mathsf{Sum}\ \alpha\ \beta]$
- $\neg P \coloneqq P \to \mathbf{0}$

The Law of Excluded Middle states that every proposition is either true or false.

- $\mathsf{em} : \forall P : \mathbb{P}.\ P \vee \neg P$

The Axiom of Choice postulates the existence of a choice function for any indexed type family.

- $\mathsf{choice} : \Pi(\alpha : \mathsf{Type})(\beta : \alpha \to \mathsf{Type})(h : \Pi a.\ [\beta\ a]).\ [\Pi a.\ \beta\ a]$

3. Translating Sett to ZFC

Given $\Gamma \vdash e : \alpha$, we define a translation $[\![\Gamma \vdash e]\!]\gamma$, where γ is a list of translated expressions for every variable in Γ. The translation results in an ZFC expression exactly, or any higher-order function on ZFC expressions.

We denote symbols in the language of ZFC in red. Take $\bullet$ to be any set whose identity is unimportant; for example, $\varnothing$. We denote functions in ZFC as $(x \in X) \mapsto f(x)$, as it is necessary to specify the domain to be a function (that the codomain is a set follows from replacement). We denote the definite description operator $\iota(x(\varphi(x)))$, evaluating to the unique x such that $\varphi(x)$ if it exists, and is undefined otherwise.

- $[\![\Gamma \vdash x]\!]\gamma$ is the element of γ corresponding to x.
- $[\![\Gamma \vdash \Pi x : \alpha.\ \beta]\!]\gamma = \prod_{x \in [\![\Gamma \vdash \alpha]\!]\gamma} [\![\Gamma, x : \alpha \vdash \beta]\!](\gamma, x)$
- $[\![\Gamma \vdash \lambda x : \alpha.\ e]\!]\gamma = (x \in [\![\Gamma \vdash \alpha]\!]\gamma) \mapsto [\![\Gamma, x : \alpha \vdash e]\!](\gamma, x)$,
  if $\Gamma, x : \alpha \vdash e : \beta$ and $\Gamma \vdash \Pi x : \alpha.\ \beta : \mathsf{Type}$;
- $[\![\Gamma \vdash \lambda x : \alpha.\ e]\!]\gamma = x \mapsto [\![\Gamma, x : \alpha \vdash e]\!](\gamma, x)$ otherwise.
- $[\![\Gamma \vdash f\ e]\!]\gamma = [\![\Gamma \vdash f]\!]\gamma([\![\Gamma \vdash e]\!]\gamma)$, if $\Gamma \vdash f : \Pi x : \alpha.\ \beta$ and $\Gamma \vdash \Pi x : \alpha.\ \beta : \mathsf{Type}$;
- $[\![\Gamma \vdash f\ e]\!]\gamma = [\![\Gamma \vdash f]\!]\gamma([\![\Gamma \vdash e]\!]\gamma)$ otherwise.
- $[\![\Gamma \vdash \mathsf{Eq}]\!]\gamma = \alpha \mapsto \mathrm{Eq}$, where $\mathrm{Eq}(a, b) = \{x \in \{\bullet\} \mid a = b\}$
- $[\![\Gamma \vdash \mathsf{refl}]\!]\gamma = \alpha \mapsto (a \in \alpha) \mapsto \bullet$
- $[\![\Gamma \vdash \mathsf{elim}_=]\!]\gamma = \alpha \mapsto a \mapsto C \mapsto (h \in C(a)(\bullet)) \mapsto (b \in \alpha) \mapsto (t \in \mathrm{Eq}(a, b)) \mapsto h$
- $[\![\Gamma \vdash \mathsf{elim\_refl}]\!]\gamma = \alpha \mapsto a \mapsto C \mapsto (h \in C(\alpha)(\bullet)) \mapsto \bullet$
- $[\![\Gamma \vdash \mathsf{K}]\!]\gamma = \alpha \mapsto (a\ b \in \alpha) \mapsto (h\ h' \in \mathrm{Eq}(a, b)) \mapsto \bullet$
- $[\![\Gamma \vdash \mathsf{funext}]\!]\gamma = \alpha \mapsto \beta \mapsto \left(f\ g \in \prod_{a \in \alpha} \beta(a)\right) \mapsto \left(h \in \prod_{x \in \alpha} \mathrm{Eq}(f(x), g(x))\right) \mapsto \bullet$
- $[\![\Gamma \vdash \mathsf{Sigma}]\!]\gamma = \alpha \mapsto \beta \mapsto \sum_{a \in \alpha} \beta(a)$
- $[\![\Gamma \vdash \mathsf{pair}]\!]\gamma = \alpha \mapsto \beta \mapsto (a \in \alpha) \mapsto (b \in \beta(a)) \mapsto (a, b)$
- $[\![\Gamma \vdash \mathsf{elim}_\Sigma]\!]\gamma = \alpha \mapsto \beta \mapsto C \mapsto \left(f \in \prod_{a \in \alpha} \prod_{b \in \beta(a)} C((a, b))\right) \mapsto \left((a, b) \in \sum_{a \in \alpha} \beta(a)\right) \mapsto f(a)(b)$
- $[\![\Gamma \vdash \mathsf{elim\_pair}]\!]\gamma = \alpha \mapsto \beta \mapsto C \mapsto \left(f \in \prod_{a \in \alpha} \prod_{b \in \beta(a)} C((a, b))\right) \mapsto (a \in \alpha) \mapsto (b \in \beta(a)) \mapsto \bullet$
- $[\![\Gamma \vdash \mathbb{P}]\!]\gamma = \mathcal{P}(\{\bullet\})$; this is equal to $\{\varnothing, \{\bullet\}\}$.
- $[\![\Gamma \vdash \mathsf{El}_\mathbb{P}]\!]\gamma = P \mapsto P$
- $[\![\Gamma \vdash \mathsf{propresize}]\!]\gamma = \alpha \mapsto \left(h : \prod_{a\ b \in \alpha} \mathrm{Eq}(a, b)\right) \mapsto (([\alpha], (\mathrm{untrunc}(\alpha), \mathrm{bij}(\alpha))), \mathrm{isprop}(\alpha))$
  where:
  - $[\alpha] = \{x \in \{\bullet\} \mid \alpha \text{ is inhabited}\}$
  - $\mathrm{untrunc}(\alpha) = (x \in [\alpha]) \mapsto \iota(x(x \in \alpha))$
  - $\mathrm{inj}(\alpha) = (x\ y \in [\alpha]) \mapsto (h \in \mathrm{Eq}(\mathrm{untrunc}(\alpha)(x), \mathrm{untrunc}(\alpha)(y))) \mapsto \bullet$
  - $\mathrm{surj}(\alpha) = (b \in \alpha) \mapsto \bullet$
  - $\mathrm{bij}(\alpha) = (\mathrm{inj}(\alpha), \mathrm{surj}(\alpha))$
  - $\mathrm{isprop}(\alpha) = \left(a\ b \in \sum_{P \in \mathcal{P}(\{\bullet\})} \sum_{f : P \to \alpha} \mathrm{Injective}(f : P \to \alpha) \times \mathrm{Surjective}(f : P \to \alpha)\right) \mapsto \bullet$
  - $\mathrm{Injective}(f : \alpha \to \beta) = \prod_{x y \in \alpha} \mathrm{Eq}(f(x), f(y)) \to \mathrm{Eq}(x, y)$
  - $\mathrm{Surjective}(f : \alpha \to \beta) = \prod_{b \in \beta} \left[\sum_{a \in \alpha} \mathrm{Eq}(f(a), b)\right]$
- $[\![\Gamma \vdash \mathsf{Tree}]\!]\gamma = \alpha \mapsto \beta \mapsto \mathrm{T}_{a \in \alpha} \beta(a)$
  where:

  - $\mathrm{T}_{a\in\alpha}\,\beta(a) = \{x \in V_\lambda \mid \forall T, \bullet \in T \wedge (\forall a\, b, a \in \alpha \wedge b : \beta(a) \to T \Rightarrow (a,b) \in T) \Rightarrow x \in T\}$
  - λ is some ordinal whose cofinality is greater than $\sup_{a\in\alpha}(\beta(a))$.
- $[\![\Gamma \vdash \mathsf{node}]\!]\gamma = \alpha \mapsto \beta \mapsto \bullet$
- $[\![\Gamma \vdash \mathsf{branch}]\!]\gamma = \alpha \mapsto \beta \mapsto (a \in \alpha) \mapsto (b : \beta(a) \to \mathrm{T}_{a\in\alpha}\,\beta(a)) \mapsto (a, b)$
- $[\![\Gamma \vdash \mathsf{elim}_{\mathrm{T}}]\!]\gamma = \alpha \mapsto \beta \mapsto C \mapsto (h_1 \in C(\bullet)) \mapsto$

$$\left(h_2 \in \prod\nolimits_{a\in\alpha} \prod\nolimits_{b:\beta(a)\to \mathrm{T}_{a\in\alpha}\,\beta(a)} \prod\nolimits_{h\in \prod_{i\in\beta(a)} C(b(i))} C((a,b))\right) \mapsto$$

$$(t \in \mathrm{T}_{a\in\alpha}\,\beta(a)) \mapsto \iota(x(\varphi(\alpha, \beta, C, h_1, (a,b,h) \mapsto h_2(a,b,h), t, x)))$$

where

$$\varphi(\alpha, \beta, C, h_1, h_2, x, y) = \forall A \subseteq \sum\nolimits_{t\in T_{a\in\alpha}\beta(a)} C(t), (\bullet, h_1) \in A \wedge$$

$$\Big[\forall (a \in \alpha)(b : \beta(a) \to \mathrm{T}_{a\in\alpha}\,\beta(a)) \Big(h \in \prod\nolimits_{i\in\beta(a)} C(b(i))\Big),$$

$$(\forall i \in \beta(a), (b(i), h(i)) \in A) \Rightarrow ((a,b), h_2(a,b,h)) \in A\Big] \Rightarrow (x, y) \in A$$

- $[\![\Gamma \vdash \mathsf{elim\_node}]\!]\gamma = \alpha \mapsto \beta \mapsto C \mapsto h_1 \mapsto h_2 \mapsto \bullet$
- $[\![\Gamma \vdash \mathsf{elim\_branch}]\!]\gamma = \alpha \mapsto \beta \mapsto C \mapsto h_1 \mapsto h_2 \mapsto (a \in \alpha) \mapsto (b : \beta(a) \to \mathrm{T}_{a\in\alpha}\,\beta(a)) \mapsto \bullet$
- $[\![\Gamma \vdash \mathsf{large\_elim}_{\mathrm{T}}]\!]\gamma = \alpha \mapsto \beta \mapsto h_1 \mapsto h_2 \mapsto (t \in \mathrm{T}_{a\in\alpha}\,\beta(a)) \mapsto \iota(x(\varphi(\alpha, \beta, i \mapsto V_\lambda, h_1, h_2, t, x)))$