A Realizability Model for Impredicative Hoare Type Theory

Lars Birkedal

IT University of Copenhagen

Joint work with R.L. Petersen, A. Nanevski, G. Morrisett

Dependent type theory

- Type theory is a program logic:
  - types can express and enforce precise program properties
- Doubles up as a programming language.
- Prototypical higher-order language (e.g, polymorphism, inductive/recursive types, etc.)
- Problem: must be purely functional
  - recursion allowed, if you prove termination
  - effects like state, IO, etc., usually second class

Hoare Logic

- Logic for imperative programs.
- Specifies partial correctness via Hoare triple $\{P\}\ E\ \{Q\}$:
  - if P holds, then E diverges or terminates in a state Q
  - P: precondition
  - Q: postcondition
- Usually targets first-order languages
  - but recent advances in the higher-order case
- Reasoning about state and aliasing very streamlined
  - Separation Logic by O'Hearn, Pym, Reynolds, Yang...

Type theory for imp. programs

- We integrate Hoare Logic into a Type Theory.
- Benefits:
  - types can enforce correct use of effectful programs
  - add effects to type theory
  - preserves equational reasoning about pure programs
- Idea: follow specifications-as-types principle
  - Type of Hoare triples $\{P\}x : \tau\{Q\}$
  - precondition P, postcondition Q, return result of type τ.
  - Dependencies allow P and Q to talk about program data.
- In this talk: a model for Impredicative Hoare Type Theory (HTT)
  - for reasoning about state and aliasing

Impredicative HTT

- Full Higher-order Dependent Type Theory:
  - $\Gamma \vdash \tau : \mathrm{Type}$
  - $\Gamma \vdash A : \mathrm{Kind}$
  - Impredicative: Type is a kind
- with Dependent Predicate Logic on top:
  - $\Gamma \vdash P : \mathrm{Prop}$
  - $\Gamma \mid P_1, \ldots, P_n \vdash P$
  - Higher-order: Prop is a kind.

Impred. HTT: new features, I

- Hoare Type for computations:
  - $\Gamma \vdash \{P\}\ x : \tau\ \{Q\} : \mathrm{Type}$
  - computations can operate on the heap
  - computations can diverge (fixed-point term)
- Challenges:
  - combining effects with type theory
  - types must be modelled as some kind of domains

Impred. HTT: new features, II

- Classical Separation Logic of heaps:
  - $\Gamma \vdash emp : \mathrm{Prop}$
  - $\Gamma \vdash P \ * \ Q : \mathrm{Prop}$
  - $\Gamma \vdash P \ \mathrel{-\!\!*} \ Q : \mathrm{Prop}$
  - Frame rule for local reasoning:

$$\frac{\Gamma \vdash M : (\Delta).\{P\}x : \tau\{Q\} \quad \Gamma, \Delta \vdash R : \mathrm{Prop}}{\Gamma \vdash M : (\Delta).\{P \ * \ R\}x : \tau\{Q \ * \ R\}} \ frame$$

- Challenges:
  - Impredicative type theory + classical logic means that consistency of the system is non-trivial
  - Extending separation logic to work over dependent type theory

Example, Mutable ADT, I

$$
\begin{array}{l}
\mathsf{stacktype} = \\
\quad \Pi\alpha : \mathrm{Type}.\Sigma\beta : \mathrm{Type}.\Sigma inv : \beta \times \alpha\,\mathrm{list} \to \mathrm{Prop}. \\
/*\,\texttt{new}\,*/ \quad (-).\{\mathrm{emp}\}s : \beta\{inv(s,[])\} \times \\
/*\,\texttt{push}\,*/ \quad \Pi s : \beta.\Pi x : \alpha. \\
\qquad\qquad (l : \alpha\,\mathrm{list}).\{inv(s,l)\}u : 1\{inv(s, x :: l)\} \times \\
/*\,\texttt{pop}\,*/ \quad \Pi s : \beta. \\
\qquad\qquad (x : \alpha, l : \alpha\,\mathrm{list}). \\
\qquad\qquad\quad \{inv(s, x :: l)\}y : \alpha\{inv(s,l) \wedge y =_\alpha x\} \times \\
/*\,\texttt{del}\,*/ \quad \Pi s : \beta. \\
\qquad\qquad (l : \alpha\,\mathrm{list}).\{inv(s,l)\}u : 1\{\mathrm{emp}\}
\end{array}
$$

Example, Mutable ADT, II

Client C:

$$
\begin{array}{l}
C = \lambda S : \texttt{stacktype}. \\
\quad \texttt{do}\ S_{\mathrm{Nat}} \leftarrow \texttt{ret}\ S(\mathrm{Nat})\ \texttt{in} \\
\quad\quad \texttt{unpack}\ S_{\mathrm{Nat}}\ \texttt{as}\ (\beta, inv, \texttt{new}, \texttt{push}, \texttt{pop}, \texttt{del})\ \texttt{in} \\
\quad\quad\quad \texttt{do}\ s \leftarrow \texttt{new}\ \texttt{in} \\
\quad\quad\quad\quad \texttt{do}\ s_4 \leftarrow \texttt{push}(s)(4)\ \texttt{in} \\
\quad\quad\quad\quad\quad \texttt{do}\ n_4 \leftarrow \texttt{pop}(s_4)\ \texttt{in} \\
\quad\quad\quad\quad\quad\quad \texttt{del}(s_4); \texttt{ret}\ n_4
\end{array}
$$

C has type $\Pi S : \texttt{stacktype}.(-).\{\mathrm{emp}\}n : \mathrm{Nat}\{\mathrm{emp} \wedge n =_{\mathrm{Nat}} 4\}.$

Example: recursion

$$fac : T = \Pi n : \mathrm{Nat}.(-).\{\mathrm{emp}\} m : \mathrm{Nat}\{\mathrm{emp} \wedge m =_{\mathrm{Nat}} n!\}:$$

$$\begin{aligned} fac = \ & \texttt{fix}\ f(n)\ \texttt{in}\ \texttt{case}\ n\ \texttt{of} \\ & \texttt{zero} \Rightarrow \texttt{ret}\ 1\ \texttt{or} \\ & \texttt{succ}\ y \Rightarrow \texttt{do}\ m \leftarrow f(y)\ \texttt{in}\ \texttt{ret}\ m \times \texttt{succ}\ y \end{aligned}$$

Summary of Challenges

- Combining effects with dependent type theory
- Types must be a kind of domains, to model non-termination and fixed points
- Impredicative type theory + classical logic means that consistency of the system is non-trivial
- Extending separation logic to work over dependent type theory

Grammar, I

$$
\begin{array}{lrl}
\text{Types} & \tau, \sigma, \rho ::= & \mathrm{Nat} \mid 1 \mid \Pi^T x : A.\tau \mid \Sigma^T x : A.\tau \mid \\
 & & (\Gamma).\{P\}x : \tau\{P\} \\
\text{Kinds} & A, B ::= & \tau \mid \mathrm{Type} \mid \mathrm{Prop} \mid \Pi^K x : A.A \mid \Sigma^K x : A.A \\
\text{Prop's} & P, Q, R ::= & \top \mid \bot \mid M =_A M \mid P \wedge P \mid P \vee P \mid \\
 & & P \supset P \mid \neg P \mid \forall x : A.P \mid \exists x : A.P \mid \\
 & & \mathrm{emp} \mid M \mapsto_\tau M \mid P \ * \ P \mid P \ \mathrel{-\!\!*} \ P
\end{array}
$$

Grammar, II

$$\begin{array}{lll} \text{Terms} & M, N ::= & x \mid \texttt{zero} \mid \texttt{succ}\ M \mid \texttt{rec}_{\mathrm{Nat}}(M, M) \mid () \mid \\ & & \lambda^K x : A.M \mid \lambda^T x : A.M \mid M\ M \mid \\ & & (M, M)^K \mid (M, M)^T \mid \texttt{fst}\ M \mid \\ & & \texttt{snd}\ M \mid \texttt{unpack}\ M\ \texttt{as}\ (x, y)\ \texttt{in}\ M \\ & & \texttt{case}\ M\ \texttt{of}\ \texttt{zero} \Rightarrow M\ \texttt{or}\ \texttt{succ}\ x \Rightarrow M \\ & & \texttt{fix}\ f(x)\ \texttt{in}\ M \mid \texttt{ret}\ M \mid \\ & & !_\tau M \mid M :=_\tau M \mid \texttt{do}\ x \leftarrow M\ \texttt{in}\ M \mid \\ & & \texttt{alloc}_\tau\ M \mid \texttt{dealloc}\ M \end{array}$$

Pure Part

- standard kinds and types (weak sums of types over kinds)
- classical predicate logic + separation logic
- standard terms
- external equality rules also standard, include β and η

Some Computation Terms

$$\frac{\Gamma, \Delta \vdash \tau : \mathrm{Type} \quad \Gamma \vdash M : \tau}{\Gamma \vdash \mathtt{ret}\ M : (\Delta).\{\mathrm{emp}\}x : \tau\{\mathrm{emp} \wedge x =_\tau M\}}\ dia$$

$$\frac{\begin{array}{c}\Gamma \vdash M : (\Delta).\{P\}y : \sigma\{S\} \quad \Gamma, \Delta, x : \tau \vdash Q : \mathrm{Prop} \\ \Gamma, y : \sigma \vdash N : (\Delta).\{S\}x : \tau\{Q\}\end{array}}{\Gamma \vdash \mathtt{do}\ y \leftarrow M\ \mathtt{in}\ N : (\Delta).\{P\}x : \tau\{Q\}}\ seq$$

$$\frac{\Gamma \vdash \tau : \mathrm{Type} \quad \Gamma \vdash M : \mathrm{Nat}}{\Gamma \vdash !_\tau M : (y : \tau).\{M \mapsto_\tau y\}x : \tau\{M \mapsto_\tau y \wedge x =_\tau y\}}\ lookup$$

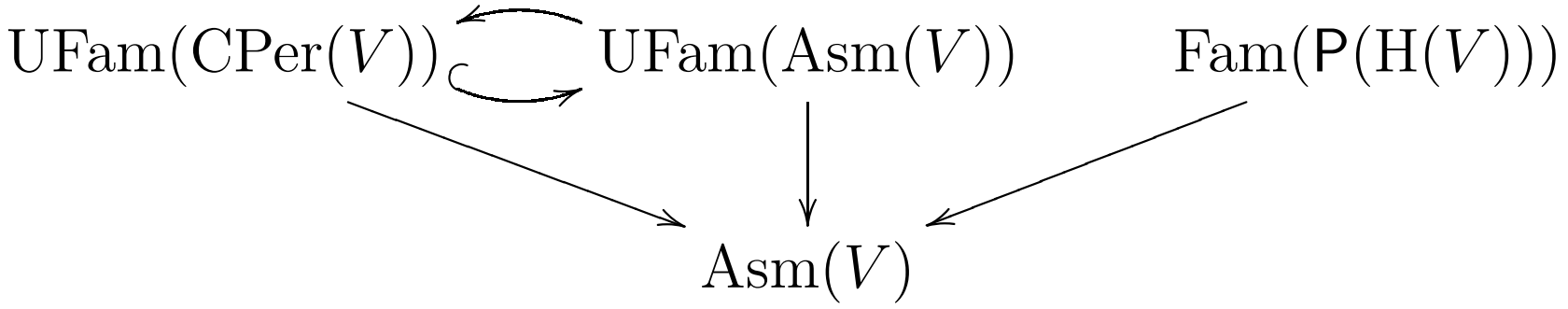
$$\frac{\Gamma, f : \Pi^T y : A.(\Delta).\{P\}x : \tau\{Q\}, y : A \vdash M : (\Delta).\{P\}x : \tau\{Q\}}{\Gamma \vdash \mathtt{fix}\ f(x)\ \mathtt{in}\ M : \Pi^T y : A.(\Delta).\{P\}x : \tau\{Q\}}\ fix$$

Selected Structural Rules

$$\frac{\Gamma \vdash M : (\Delta).\{R\}x : \tau\{S\} \quad \Gamma, \Delta \vdash P \supset R \quad \Gamma, \Delta \vdash S \supset Q}{\Gamma \vdash M : (\Delta).\{P\}x : \tau\{Q\}}\ consequent$$

$$\frac{\Gamma \vdash M : (\Delta).\{P\}x : \tau\{Q\} \quad \Gamma, \Delta \vdash R : \mathrm{Prop}}{\Gamma \vdash M : (\Delta).\{P\ *\ R\}x : \tau\{Q\ *\ R\}}\ frame$$

Model — Overview, I



- A Realizability Model over universal domain V
- $\mathrm{H}(V)$ is the domain of heaps
- Computations will be realized by continuous functions in $\mathrm{T}(V) = \mathrm{H}(V) \to V \otimes \mathrm{H}(V)_\perp \oplus \mathbb{E}$.
- Note: $\mathrm{Asm}(V)$ aka ω-sets.

Model — Kinds

- Definition of $\mathrm{Asm}(V)$:
  - objects: (X, E), where X is a set, and $E : X \to \mathsf{P}(V)$, such that for all $x \in X$, $E(x) \neq \emptyset$
  - morphisms $f : (X, E) \to (X', E')$, where $f : X \to X'$ is a set-theoretic function, such that there exists a realizer α for it, i.e

$$\exists \alpha : V \to V . \forall x \in X . \forall d \in E(x) . \alpha(d) \in E(f(x)).$$

- An extension of Set: full and faithful functor $\nabla : \mathrm{Set} \to \mathrm{Asm}(V)$

Model — Types

- Definition of $\mathrm{CPer}(V)$:
  - objects: **chain-complete partial equivalence** relations over V
  - morphisms: equivalence classes of equivalence-preserving continuous functions
- Intuition: predomains (complete partial orders)

Model - DTT

- dependency means that we index complete pers and assemblies over assemblies to get fibred categories

Soundness of Pure Part

Lemma 1. *The fibred inclusion of* $\mathrm{UFam}(\mathrm{CPer}(V))$ *into* $\mathrm{UFam}(\mathrm{Asm}(V))$ *has a fibred left adjoint.*

Theorem 2. *The categories and functors in the diagram above constitute a split weak FhoDTT with a fibred natural numbers object in* $\mathrm{UFam}(\mathrm{CPer}(V))$*, which is also a fibred natural numbers object in* $\mathrm{UFam}(\mathrm{Asm}(V))$*.*

Corollary 3. *The pure type and kind fragment (excluding computation types) of HTT is sound wrt. the interpretation in the above FhoDTT.*

Model — Logic, I

- how to model predicates $\Gamma \vdash P : \mathrm{Prop}$?
- recall that a context Γ is interpreted as an assembly (X, E)
- then P will be interpreted as a function $X \to \mathsf{P}(\mathrm{H}(V))$
- entailment $\Gamma \mid P \vdash Q$ modelled by subset inclusion:

$$P \leq Q \text{ iff for all } x \in X. P(x) \subseteq Q(x).$$

Model — Logic, II

- Classical logic (subsets, ordered by inclusion)
- Separation logic connectives in the standard way, e.g.:
  - $[\![\mathrm{emp} : \mathrm{Prop}]\!] = \emptyset$
  - $[\![\Gamma \vdash P * Q : \mathrm{Prop}]\!] =$

$$\{h \mid \exists h_1 \in [\![\Gamma \vdash P : \mathrm{Prop}]\!]_x^{\mathrm{Props}}, h_2 \in [\![\Gamma \vdash Q : \mathrm{Prop}]\!]_x^{\mathrm{Props}}. \\ h = h_1 * h_2\}$$

Model — Logic, III

Theorem 4. *The fibration*

$$\begin{array}{c} \mathrm{Fam}(\mathsf{P}(\mathrm{H}(V))) \\ \downarrow \\ \mathrm{Asm}(V) \end{array}$$

is a BI-hyperdoctrine [Biering, Birkedal, Torp-Smith: 2005] with quantification along all maps in the base category.

Corollary 5. *The interpretation of the logic in the above BI-hyperdoctrine is sound.*

Computations, I

Intuition:

- $(\Delta).\{P\}x : \tau\{Q\}$ is modeled as an admissible per of realizers in $\mathrm{T}(V) = \mathrm{H}(V) \to V \otimes \mathrm{H}(V)_\perp \oplus \mathbb{E}$
- that given a heap in $[\![P]\!]$, does not produce error, and if it terminates, yields a value v and a heap h satisfying $h \in [\![Q]\!](v)$.
- We "bake-in" the frame rule, essentially by interpreting $(\Delta).\{P\}x : \tau\{Q\}$ as $\forall R \in \mathrm{Prop}.(\Delta).\{P \ * \ R\}x : \tau\{Q \ * \ R\}$.
- We take the admissible closure of the postcondition to guarantee admissibility (as needed for fixed points), so $(\Delta).\{P\}x : \tau\{Q\}$ is interpreted as $\forall R \in \mathrm{Prop}.(\Delta).\{P \ * \ R\}x : \tau\{\overline{Q \ * \ R}\}$.

Computations, II

- Assume $[\![\Gamma]\!]^{\mathrm{Ctxs}} = (X, E)$.
- Assume $[\![\Gamma, \Delta]\!]^{\mathrm{Ctxs}} = (\Sigma_{x \in X} Y_x, F)$.
- Then $[\![\Gamma \vdash (\Delta).\{P\}x : \tau\{Q\} : \mathrm{Type}]\!]^{\mathrm{Types}}$ is the family of pers $(S_x)_{x \in X}$ with fields given by $d \in |S_x|$ iff $d = in_{\mathrm{T}}(f)$ and

$$\forall y \in Y_x.\forall E \in \mathrm{Prop}_{\Gamma,\Delta}.\forall h \in [\![\Gamma, \Delta \vdash (P \ * \ E)]\!]^{\mathrm{Props}}_{(x,y)}.$$
$$(f(h) \neq \mathtt{err}) \wedge \Big(f(h) = (v_f, h_f) \Rightarrow$$
$$v_f \in |[\![\Gamma, \Delta \vdash \tau : \mathrm{Type}]\!]^{\mathrm{Types}}_{(x,y)}| \wedge$$
$$h_f \in \overline{[\![\Gamma, \Delta, x : \tau \vdash (Q \ * \ E)]\!]^{\mathrm{Props}}_{(x,y,v_f)}} \Big)$$

Computations, III

- actual per is given by $in_{\mathrm{T}}(f)\ S_x\ in_{\mathrm{T}}(g)$ iff $in_{\mathrm{T}}(f), in_{\mathrm{T}}(g) \in |S_x|$ and

$$\forall y \in Y_x.\forall E \in \mathrm{Prop}_{\Gamma,\Delta}.\forall h, h' \in [\![\Gamma, \Delta \vdash (P\ *\ E)]\!]^{\mathrm{Props}}_{(x,y)}.$$
$$h \stackrel{\downarrow}{=} h' \Rightarrow$$
$$f(h)\downarrow \Leftrightarrow g(h')\downarrow \wedge \Big(f(h) = (v_f, h_f) \wedge g(h') = (v_g, h_g) \Rightarrow$$
$$v_f\ [\![\Gamma, \Delta \vdash \tau : \mathrm{Type}]\!]^{\mathrm{Types}}_{(x,y)}\ v_g\ \wedge\ h_f \stackrel{\downarrow}{=} h_g \Big)$$

- Note: on two heaps with equal support, the resulting heaps should have equal support. Means that allocation can be modelled by taking the least unallocated address.

Computation terms — example:

- Assume $[\![\Gamma]\!]^{\mathrm{Ctxs}} = (X, E)$
- Assume $[\![M]\!]$ is realized by α
- Then

$$[\![\Gamma \vdash !_\tau M : (y : \tau).\{M \mapsto_\tau y\}x : \tau\{M \mapsto_\tau y \wedge x =_\tau y\}]\!]^{\mathrm{Terms}}$$
$$= \lambda e.\lambda h.\texttt{if } h(m(e)) = \bot \texttt{ then err else } (h(m(e)), h)$$

Soundness

Theorem 6. *The interpretation of computations is well-defined.*

Theorem 7. *Impredicative HTT is consistent and* ***well-specified programs do not go wrong.***

Implementation in Coq

- We have a prototype implementation of HTT in Coq.
- The stack example and other examples, including hash tables, have been verified.

Thank You

Judgements

$$\Gamma \vdash A : \mathrm{Kind} \qquad \Gamma \vdash A = A : \mathrm{Kind}$$

$$\Gamma \vdash \tau : \mathrm{Type}$$

$$\Gamma \vdash P : \mathrm{Prop}$$

$$\Gamma \vdash M : A \qquad \Gamma \vdash M = M : A$$

$$\Gamma \mid \Theta \vdash P$$

Kinds

$$\frac{\Gamma \vdash \tau : \mathrm{Type}}{\Gamma \vdash \tau : \mathrm{Kind}}\, ext$$

$$\frac{}{\emptyset \vdash \mathrm{Type} : \mathrm{Kind}}\, \mathrm{Type} \qquad \frac{}{\emptyset \vdash \mathrm{Prop} : \mathrm{Kind}}\, \mathrm{Prop}$$

$$\frac{\Gamma, x : A \vdash B : \mathrm{Kind}}{\Gamma \vdash \Pi^K x : A.B : \mathrm{Kind}}\, \Pi K \qquad \frac{\Gamma, x : A \vdash B : \mathrm{Kind}}{\Gamma \vdash \Sigma^K x : A.B : \mathrm{Kind}}\, \Sigma K$$

Types

$$\frac{}{\emptyset \vdash \mathrm{Nat} : \mathrm{Type}}\,\mathrm{Nat} \qquad \frac{}{\emptyset \vdash 1 : \mathrm{Type}}\,1$$

$$\frac{\Gamma, x : A \vdash \tau : \mathrm{Type}}{\Gamma \vdash \Pi^T x : A.\tau : \mathrm{Type}}\,\Pi T \qquad \frac{\Gamma, x : A \vdash \tau : \mathrm{Type}}{\Gamma \vdash \Sigma^T x : A.\tau : \mathrm{Type}}\,\Sigma T$$

$$\frac{\Gamma, \Delta \vdash \tau : \mathrm{Type} \quad \Gamma, \Delta \vdash P : \mathrm{Prop} \quad \Gamma, \Delta, x : \tau \vdash Q : \mathrm{Prop}}{\Gamma \vdash (\Delta).\{P\}x : \tau\{Q\} : \mathrm{Type}}\,spec$$

Note: weak sums

Logic

- Rules for $\Gamma \vdash P : \mathrm{Prop}$ standard + separation logic connectives.
- Rules for logical entailment $\Gamma \mid \Theta \vdash P$ standard classical predicate logic + separation logic rules, e.g.:

$$\frac{\Gamma \mid \Theta_1 \vdash P \quad \Gamma \mid \Theta_2 \vdash Q}{\Gamma \mid \Theta_1 \ * \ \Theta_2 \vdash P \ * \ Q}$$

$$\frac{\Gamma \mid \Theta \ * \ P \vdash Q}{\Gamma \mid \Theta \vdash P \ \mathrel{-\!\!*} \ Q}$$

Pure Terms

- standard term formation rules
- external equality rules also standard, include β and η

Computation Terms, I

$$\frac{\Gamma, \Delta \vdash \tau : \mathrm{Type} \quad \Gamma \vdash M : \tau}{\Gamma \vdash \mathtt{ret}\ M : (\Delta).\{\mathrm{emp}\}x : \tau\{\mathrm{emp} \wedge x =_\tau M\}}\ dia$$

$$\frac{\begin{array}{c}\Gamma \vdash M : (\Delta).\{P\}y : \sigma\{S\} \quad \Gamma, \Delta, x : \tau \vdash Q : \mathrm{Prop} \\ \Gamma, y : \sigma \vdash N : (\Delta).\{S\}x : \tau\{Q\}\end{array}}{\Gamma \vdash \mathtt{do}\ y \leftarrow M\ \mathtt{in}\ N : (\Delta).\{P\}x : \tau\{Q\}}\ seq$$

$$\frac{\Gamma \vdash \tau : \mathrm{Type} \quad \Gamma \vdash M : \mathrm{Nat}}{\Gamma \vdash !_\tau M : (y : \tau).\{M \mapsto_\tau y\}x : \tau\{M \mapsto_\tau y \wedge x =_\tau y\}}\ lookup$$

$$\frac{\Gamma \vdash \tau : \mathrm{Type} \quad \Gamma \vdash M : \mathrm{Nat} \quad \Gamma \vdash N : \tau}{\Gamma \vdash M :=_\tau N : (-).\{M \mapsto_\sigma -\}x : 1\{M \mapsto_\tau N\}}\ update$$

Computation Terms, II

$$\frac{\Gamma \vdash \tau : \mathrm{Type} \quad \Gamma \vdash M : \tau}{\Gamma \vdash \mathtt{alloc}_\tau\ M : (-).\{\mathrm{emp}\}x : \mathrm{Nat}\{x \mapsto_\tau M\}}\ alloc$$

$$\frac{\Gamma \vdash \tau : \mathrm{Type} \quad \Gamma \vdash M : \mathrm{Nat}}{\Gamma \vdash \mathtt{dealloc}\ M : (-).\{M \mapsto_\tau -\}x : 1\{\mathrm{emp}\}}\ dealloc$$

$$\frac{\begin{array}{c}\Gamma \vdash M_1 : (\Delta).\{P \wedge M =_{\mathrm{Nat}} \mathtt{zero}\}x : \tau\{Q\} \quad \Gamma \vdash M : \mathrm{Nat} \\ \Gamma, y : \mathrm{Nat} \vdash M_2 : (\Delta).\{P \wedge M =_{\mathrm{Nat}} \mathtt{succ}\ y\}x : \tau\{Q\}\end{array}}{\Gamma \vdash \mathtt{case}\ M\ \mathtt{of}\ \mathtt{zero} \Rightarrow M_1\ \mathtt{or}\ \mathtt{succ}\ y \Rightarrow M_2 : (\Delta).\{P\}x : \tau\{Q\}}\ case$$

$$\frac{\Gamma, f : \Pi^T y : A.(\Delta).\{P\}x : \tau\{Q\}, y : A \vdash M : (\Delta).\{P\}x : \tau\{Q\}}{\Gamma \vdash \mathtt{fix}\ f(x)\ \mathtt{in}\ M : \Pi^T y : A.(\Delta).\{P\}x : \tau\{Q\}}\ fix$$

Selected Structural Rules

$$\frac{\Gamma \vdash M : (\Delta).\{R\}x : \tau\{S\} \quad \Gamma, \Delta \vdash P \supset R \quad \Gamma, \Delta \vdash S \supset Q}{\Gamma \vdash M : (\Delta).\{P\}x : \tau\{Q\}}\ consequent$$

$$\frac{\Gamma \vdash M : (\Delta).\{P\}x : \tau\{Q\} \quad \Gamma, \Delta \vdash R : \mathrm{Prop}}{\Gamma \vdash M : (\Delta).\{P\ *\ R\}x : \tau\{Q\ *\ R\}}\ frame$$