

# 1 Untyped Syntax

value types	$A$	$::=$	Unit   Bool   Case $A$   OSum   $A \times A$   $A * A$   $U \underline{B}$	
computation types	$B$	$::=$	$A \rightarrow \underline{B}$   $A \multimap \underline{B}$   $F A$	
values	$V$	$::=$	tt   true   false   $x$   $\text{inj}_V V$   $(V, V)$   $\pi_1 V$   $\pi_2 V$   $(V * V)$   $\rho_1 V$   $\rho_2 V$   $\text{thunk } M$	variable injection into the open sum  separating product
computations	$M$	$::=$	$\mathcal{O}$   $\text{force } V$   $\text{ret } V$   $x \leftarrow M; N$   $M N$   $M @ N$   $\lambda x: A. M$   $\alpha x: A. M$   $\text{newcase}_A x; M$   $\text{match } V \text{ with } V \{ \text{inj } x. M \mid N \}$   $\text{if } V \text{ then } M \text{ else } N$	need monad for this  separating function "allocate" a new case in the open sum match on open sum
value typing context	$\Gamma$	$::=$	$\emptyset$   $x : A$   $\Gamma; \Gamma'$   $\Gamma \mathbin{\text{\tiny ;}} \Gamma'$	cartesian product monoidal product
stoup	$\Theta$	$::=$	$\cdot$   $\bullet : B$	

stoup forms?

## 2 Typed Syntax

This version of the syntax is a combination of the  $\nu$ -calculus [?] [?] of Pitts and Stark and the affine  $\alpha\lambda$ -calculus of O’Hearn [?] expressed in CBPV and with the addition of an extensible sum type. [for \(rule airity > 1\) non separating connectives, contexts don’t need to be appended\( \$\Gamma\$ ;  \$\Delta\$ \)?](#)

### 2.1 Structural Rules

; admits weakening, contraction, and exchange (cartesian closed structure)  
 $\S$  admits weakening and exchange (semicartisian symmetric monoidal structure)

### 2.2 Value Typing

#### 2.2.1 Unit

$$\frac{}{\Gamma \vdash_v \text{tt} : \text{Unit}} \text{Intro}$$

#### 2.2.2 Bool

$$\frac{}{\Gamma \vdash_v \text{true} : \text{Bool}} \text{Intro}_1$$

$$\frac{}{\Gamma \vdash_v \text{false} : \text{Bool}} \text{Intro}_2$$

$$\frac{\Gamma \vdash_v V : \text{Bool} \quad \Gamma \mid \cdot \vdash_c M_1 : B \quad \Gamma \mid \cdot \vdash_c M_2 : B}{\Gamma \mid \cdot \vdash_c \text{if } V \text{ then } M_1 \text{ else } M_2 : B} \text{Elim}$$

#### 2.2.3 Var

$$\frac{}{x : A \vdash_v x : A} \text{Intro}$$

#### 2.2.4 OSum

$$\frac{\Gamma \vdash_v V_c : \text{Case A} \quad \Gamma \vdash_v V : A}{\Gamma \vdash_v \text{inj}_{V_c} V : \text{OSum}} \text{Intro}$$

$$\frac{\Gamma \vdash_v V_T : \text{Case A} \quad \Gamma \vdash_v V : \text{OSum} \quad \Gamma; (x : A) \mid \cdot \vdash_c M : B \quad \Gamma \mid \cdot \vdash_c N : B}{\Gamma \mid \cdot \vdash_c \text{match } V_T \text{ with } V \{ \text{inj } x.M \mid N \}} \text{Elim}$$

### 2.2.5 Value Product

$$\frac{\Gamma \vdash_v N : A_1 \quad \Gamma \vdash_v M : A_2}{\Gamma \vdash_v (N, M) : A_1 \times A_2} \text{Intro}$$

$$\frac{\Gamma \vdash_v P : A_1 \times A_2}{\Gamma \vdash_v \pi_1 P : A_1} \text{Elim}_1$$

$$\frac{\Gamma \vdash_v P : A_1 \times A_2}{\Gamma \vdash_v \pi_2 P : A_2} \text{Elim}_2$$

### 2.2.6 Separating Product

we have these because the monoidal product is semicartesian

$$\frac{\Gamma \vdash_v M : A_1 \quad \Delta \vdash_v N : A_2}{\Gamma \circ \Delta \vdash_v (M * N) : A_1 * A_2} \text{Intro}$$

$$\frac{\Gamma \circ \Delta \vdash_v P : A_1 * A_2}{\Gamma \vdash_v \rho_1 P : A_1} \text{Elim}_1$$

$$\frac{\Gamma \circ \Delta \vdash_v P : A_1 * A_2}{\Delta \vdash_v \rho_2 P : A_2} \text{Elim}_2$$

### 2.2.7 Thunk

$$\frac{\Gamma \mid \cdot \vdash_c M : B}{\Gamma \vdash_v \text{thunk } M : UB} \text{Intro}$$

$$\frac{\Gamma \vdash_v V : UB}{\Gamma \mid \cdot \vdash_c \text{force } V : B} \text{Elim}$$

## 2.3 Computation Typing

### 2.3.1 Function

$$\frac{\Gamma; (x : A) \mid \cdot \vdash_c V : B}{\Gamma \mid \cdot \vdash_c (\lambda x : A. V) : A \rightarrow B} \text{Intro}$$

$$\frac{\Gamma \mid \Theta \vdash_c M : A \rightarrow B \quad \Delta \vdash_v N : A}{\Gamma; \Delta \mid \Theta \vdash_c MN : B} \text{Elim}$$

### 2.3.2 Separating Function

$$\frac{\Gamma \circ (x : A) \mid \cdot \vdash_c V : B}{\Gamma \mid \cdot \vdash_c (\alpha x : A. V) : A \multimap B} \text{Intro}$$

$$\frac{\Gamma \mid \Theta \vdash_c M : A \multimap B \quad \Delta \vdash_v N : A}{\Gamma \circ \Delta \mid \Theta \vdash_c MN : B} \text{Elim}$$

### 2.3.3 Return

$$\frac{\Gamma \vdash_v V : A}{\Gamma \mid \cdot \vdash_c \text{ret } V : FA} \text{Intro}$$

$$\frac{\Gamma \mid \Theta \vdash_c M : FA \quad \Gamma; (x : A) \mid \cdot \vdash_c N : B}{\Gamma \mid \Theta \vdash_c x \leftarrow M; N : B} \text{Elim}$$

### 2.3.4 Error

$$\frac{}{\Gamma \mid \cdot \vdash_c \text{O} : B} \text{Intro}$$

## 2.4 Equations

for effects (but also computation rules)

### 2.4.1 Allocation

The following are from Ian Stark's Categorical Models for Local Names [?]

1. swap - allocation swap
2. drop - (our monad might not support this)
3. Fresh - (a kind of congruence)

### 2.4.2 Dynamic Type

the prism laws here

- 1.

## 3 Semantics

### 3.1 Worlds

Normally, the value types in a CBPV would be interpreted as Sets. However, in our case this is insufficient as the interpretation of the extensible sum type, Osum, is dependent on what Cases have been "allocated" via newcase. Thus we interpret value types in a presheaf category on a category of worlds where objects are a map from a finite set to syntactic types. Let SynTy be a set of syntactic types that can be used in the OSum Case store.

$$\begin{array}{lcl} \text{SynTy} & ::= & \text{Unit} \\ & | & \text{Bool} \\ & | & \text{SynTy} \times \text{SynTy} \end{array}$$

The denotation of these syntactic types is independent of what Cases have been allocated. Limiting the extensible sum type to contain only base types and products of base types allows us to temporarily avoid two separate issues.

- The first issue has to do with circularity of definitions. If we say a world is a mapping from a finite set to semantic types and that semantic types are presheaves on the category of worlds, then we have tied ourselves in a knot. This issue is pointed out in, among other places, [?].

$$\text{world} \cong X \rightarrow \text{SemTy} \quad \text{SemTy} \cong [\mathbf{World}, \mathbf{Set}]$$

- The second issue is well-foundedness of the interpretation of OSum. Consider if we allow the extensible sum type to store the type (OSum  $\rightarrow$  OSum) and try to recursively interpret OSum.

Here we define the category **World**<sup>1 2</sup>. A world  $X \rightarrow \text{SynTy}$  is a map from a finite set into the set of syntactic types. Morphisms  $f : X \rightarrow Y$  in **World** are injective functions  $i : Y \hookrightarrow X$  such that the following triangle commutes. The finite sets here represent a finite collection of case symbols and the injective map  $i$  represents *renaming*. We may denote a world  $X \xrightarrow{w} \text{SynTy}$  by  $w$ .

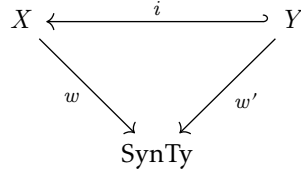


Figure 1: Morphisms in **World**

### 3.1.1 Monoidal Structure

We need to introduce additional structure on **World** to interpret the separating connectives  $A * B$  and  $A \multimap B$ . For this, we will pull from existing literature on categorical semantics of bunched typing, specifically [?] and section 3.1 of [?]. We use the Day convolution [?] to get a monoidal structure on **World**<sup>3</sup> from a monoidal structure on **World**. First, we define an operation  $w \otimes_w w'$  on *worlds* by taking the set coproduct of their domains<sup>4</sup> and using the recursor  $X + Y \xrightarrow{\text{rec } w \ w'} \text{SynTy}$  to define a map into syntactic types. The action on

<sup>1</sup>Similar categories: *HeapV* [?], *IFam<sub>X</sub>* [?], polymorphic algebraic theories [?], [?]

<sup>2</sup>Formalized here as  $\mathbf{World} := (\text{Inc} \downarrow \text{SynTy})^{op}$ , where *Inc* is the inclusion functor from the category of finite sets and injections into the category **Set** and *SynTy* is a constant functor.

<sup>3</sup> $\mathbf{Set}(\mathbf{World}^{op})$ ; contravariant presheaves on **World**

<sup>4</sup>This is not the coproduct in *FinSet<sub>mono</sub>* [?]

morphisms is given by defining the top injection via  $map : (A \rightarrow C) \rightarrow (B \rightarrow D) \rightarrow A + B \rightarrow C + D$  and the commuting triangles still hold. The unit for this monoid structure is the empty map  $\lambda()$ <sup>5</sup>.

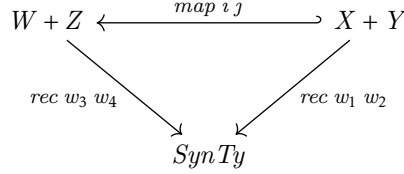


Figure 2:  $\otimes_w$  action on morphisms

Using this operation, we define the monoidal product for  $\widehat{\mathbf{World}}$ . The action on objects being:

$$(A \otimes B)X = \int^{Y,Z} A(Y) \times B(Z) \times \mathbf{World}[X, Y \otimes_w Z]$$

The monoidal unit is given by:

$$I(X) = \mathbf{World}[X, \lambda()]$$

The separating function is defined by:

$$(A \star B)X = \int_Z \mathbf{Set}[A(Z), B(X \otimes_w Z)] \cong \widehat{\mathbf{World}}[A, B(X \otimes_w -)]$$

Finally, we have that:

$$\mathrm{Hom}(A \otimes B, C) \cong \mathrm{Hom}(A, B \star C)$$

These definitions are standard in the literature on bunched implication with the exception that our *resources* are finite maps specifying typing of allocated case symbols. The bicartesian stucture on  $\widehat{\mathbf{World}}$  along with the monoidal structure ( $\otimes$ ) and its right adjoint ( $\star$ ) bundled together gives us the structure of a bicartesian doubly closed category or BiDCC.

### 3.1.2 Affine BiDCC

The concrete model that we are working with has the property that  $\mathbf{1} \cong I$ <sup>6</sup>. That is, the cartesian unit is isomorphic to the monoidal unit. This property validates weakening for  $\circledast$ . Note that:

$$\begin{aligned} \mathbf{1}(X) &= \{*\} \\ I(X) &= \mathbf{World}[X, \lambda()] \end{aligned}$$

$\lambda()$  is terminal in  $\mathbf{World}$  and the cardinality of  $|\mathbf{World}[-, \lambda()]| = 1$ .

<sup>5</sup>The empty map is also the terminal object in  $\mathbf{Worlds}$

<sup>6</sup>See section 3.3 of [?]

### 3.1.3 Concrete Separating Connectives

It may be possible for us to have an alternative interpretation of the separating product if we work in a full subcategory of  $\widehat{\mathbf{World}}$ . O'Hearn has affine models [?] [?] for his  $\alpha\lambda$ -calculus<sup>7</sup> using a presheaf category on  $\mathbf{FinSet}_{\text{mono}}$ . He shows that (section 4, [?]) the full subcategory of  $\mathbf{Set}^{\mathbf{FinSet}_{\text{mono}}^{op}}$  which consists of pullback preserving functors lets you ask for the *smallest* world that any  $a \in A(X)$  comes from, aka  $\text{support}(a)$  (Section 9.2, [?]). With this notion of support, you can define non-interference as:

$$a \# b \Leftrightarrow (\text{support}(a) \cap \text{support}(b) = \emptyset)$$

With non-interference,  $A * B$  can be directly interpreted as:

$$\begin{aligned} (A * B)X &= \{(a, b) \in A(X) \times B(X) \mid a \# b\} \\ (A * B)f(a, b) &= (A(f)(a), B(f)(b)) \end{aligned}$$

It should be noted that the full subcategory mentioned above is equivalent to the Shanel topos (Remark 6.9 [?]). It seems this would extend to our *multi-sorted* scenario<sup>8</sup> [?] [?].

## 3.2 Allocation Monad

### 3.3 Interpreting Types

We will take  $\widehat{\mathbf{World}}$  to be the value category in our model.

With all the categorical structure in place, we now give an interpretation of syntax. We first give an interpretation to syntactic types  $SynTy$  via  $\llbracket - \rrbracket_{el} : SynTy \rightarrow \widehat{\mathbf{World}}$ . We recall that the interpretation of these types is not dependent on what cases have been allocated.  $\mathbf{1}$  is the terminal presheaf in  $\widehat{\mathbf{World}}$  and  $+$ ,  $\times$  come from the bicartesian structure of  $\widehat{\mathbf{World}}$ .

$$\begin{aligned} \llbracket Unit \rrbracket_{el} &= \mathbf{1} \\ \llbracket Bool \rrbracket_{el} &= \mathbf{1} + \mathbf{1} \\ \llbracket A \times B \rrbracket_{el} &= \llbracket A \rrbracket_{el} \times \llbracket B \rrbracket_{el} \end{aligned}$$

Next, we interpret the remaining value types in  $\widehat{\mathbf{World}}$ . The separating product is interpreted as the monoidal product from the Day convolution and  $| \llbracket B \rrbracket_c |$  is the forgetful functor which returns the carrier of the algebra  $\llbracket B \rrbracket_c$ . The main definitions of interest are the interpretation of  $OSum$  and  $Case$ .  $OSum$  is an extensible sum type where the current world determines the number and type of cases in the extensible sum.  $Case A$  is interpreted to be the set of allocated

<sup>7</sup>And an extension with commands, memory cells, and assignment called he calls SCI+ which is based on Reynold's Syntactic Control of Interference

<sup>8</sup>The Schanuel topos models classical HOL, is this problematic?

case symbols for type  $A$  in the current world. Note that syntactic type equality is used in the set comprehension. This means that  $\sigma : \text{Case}(A \times (B \times C))$  and  $\sigma' : \text{Case}((A \times B) \times C)$  are distinct elements of two distinct sets. It may be nice to have a more flexible definition  $\{\sigma \mid \sigma \in \text{dom}(\rho) \wedge \llbracket \rho(\sigma) \rrbracket_{el} \cong \llbracket A \rrbracket_{el}\}$ . We would have to be more careful when constructing and destructing elements of  $\text{OSum}$ . Consider  $\sigma : \text{Case } A$  and  $\text{inj}_\sigma(*, a) : \text{OSum}$ . Should this be allowed with  $\llbracket A \rrbracket \cong \llbracket \text{Unit} \times A \rrbracket$ ?

$$\begin{aligned}
\llbracket \text{Unit} \rrbracket_v &= \llbracket \text{Unit} \rrbracket_{el} \\
\llbracket \text{Bool} \rrbracket_v &= \llbracket \text{Bool} \rrbracket_{el} \\
\llbracket A \times B \rrbracket_v &= \llbracket A \times B \rrbracket_{el} \\
\llbracket \text{OSum} \rrbracket_v(\rho) &= \sum_{\sigma \in \text{dom}(\rho)} \llbracket \rho(\sigma) \rrbracket_{el}(\rho) \\
\llbracket \text{Case } A \rrbracket_v(\rho) &= \{\sigma \mid \sigma \in \text{dom}(\rho) \wedge \rho(\sigma) = A\} \\
\llbracket A * B \rrbracket_v &= \llbracket A \rrbracket_v \otimes \llbracket B \rrbracket_v \\
\llbracket U B \rrbracket_v &= \mid \llbracket B \rrbracket_c \mid
\end{aligned}$$

Lastly, we interpret the computation types as their respective algebras in **TAlg**.

$$\begin{aligned}
\llbracket A \rightarrow B \rrbracket_c &= \text{expAlg} \llbracket A \rrbracket_v \llbracket B \rrbracket_c \\
\llbracket A * B \rrbracket_c &= \text{sepAlg} \llbracket A \rrbracket_v \llbracket B \rrbracket_c \\
\llbracket FA \rrbracket_c &= \text{freeAlg} \llbracket A \rrbracket_v
\end{aligned}$$

### 3.4 Interpreting Contexts

In *regular* type theory, contexts have the structure of a list and are interpreted as an n-ary product in the category used to interpret types. In the bunched type theory we are working with here, there are two connectives for constructing contexts (" $;$ " and " $\mathbin{\text{\textcircled{;}}}$ ") and the structure of contexts are "tree-like". We use  $;$  to denote the usual context construction operation that is interpreted as a cartesian product and  $\mathbin{\text{\textcircled{;}}}$  to denote the new context construction operation that is interpreted as the tensor product.

$$\begin{aligned}
\llbracket x : A \rrbracket_{ctx} &= \llbracket A \rrbracket_v \\
\llbracket \phi \rrbracket_{ctx} &= \mathbf{1} \\
\llbracket \Gamma ; \Gamma \rrbracket_{ctx} &= \llbracket \Gamma \rrbracket_{ctx} \times \llbracket \Gamma \rrbracket_{ctx} \\
\llbracket \varphi \rrbracket_{ctx} &= I \\
\llbracket \Gamma \mathbin{\text{\textcircled{;}} } \Gamma \rrbracket_{ctx} &= \llbracket \Gamma \rrbracket_{ctx} \otimes \llbracket \Gamma \rrbracket_{ctx}
\end{aligned}$$



The cartesian fragment of the context supports weakening and contraction. The monoidal fragment supports weakening<sup>9</sup>.

### 3.5 Interpreting Terms

Value terms  $\Gamma \vdash M : A$  are interpreted as morphisms  $\llbracket \Gamma \rrbracket_{ctx} \xrightarrow{M} \llbracket A \rrbracket_v$  in  $\widehat{\mathbf{World}}$  which are natural transformations from  $\mathbf{World}^{\text{op}}$  to  $\mathbf{Set}$ . So for any term  $\Gamma \vdash M : A$ , we need to provide a family of maps  $\forall w, \alpha_w : \llbracket \Gamma \rrbracket(w) \rightarrow \llbracket A \rrbracket(w)$  such that given any morphism  $f : w \rightarrow w'$  in  $\mathbf{World}$  we have:

$$\begin{array}{ccc} \llbracket \Gamma \rrbracket(w) & \xleftarrow{\llbracket \Gamma \rrbracket(f)} & \llbracket \Gamma \rrbracket(w') \\ \downarrow \alpha_w & & \downarrow \alpha_{w'} \\ \llbracket A \rrbracket(w) & \xleftarrow{\llbracket A \rrbracket(f)} & \llbracket A \rrbracket(w') \end{array}$$

Here we define the terms as natural transformations by giving the components. **check: do we get naturality via parametric polymorphism in this setting?**  $w$  is the current world and  $\gamma : \llbracket \Gamma \rrbracket(w)$ . The base terms are straightforward.

$$\begin{aligned} \llbracket \text{tt} \rrbracket_{vtn}(w, \gamma) &= * \\ \llbracket \text{true} \rrbracket_{vtn}(w, \gamma) &= \text{inl} * \\ \llbracket \text{false} \rrbracket_{vtn}(w, \gamma) &= \text{inr} * \end{aligned}$$

~~Deleted whole page, because our monoidal product is affine, we have projections~~ [?](Definition 2.1.ii) We can use this process (call it *lookup*) to construct the proper denotation for variables.

$$\llbracket x \rrbracket_{vtn} = \text{lookup}(x)$$

For case symbols, we'd like a natural transformation  $\llbracket \sigma \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \text{Case } A \rrbracket$  where components are defined as so:

$$\begin{array}{ccc} \llbracket \Gamma \rrbracket(w) & \xrightarrow{\quad} & \{\sigma \mid w(\sigma) = A\} \\ \downarrow ! & \nearrow \text{select}_\sigma & \\ \mathbf{1}(w) & & \end{array}$$

---

<sup>9</sup>consequence of  $\mathbf{1} \cong I$  in our model

which is definable as long as long as  $\sigma \mapsto A \in w$ . Consider the empty map  $\lambda() \in \text{Obj}(\mathbf{Worlds})$ ,  $\alpha_{\lambda()} : \llbracket \Gamma \rrbracket(\lambda()) \rightarrow \emptyset$ . This seems problematic..

$$\llbracket \sigma \rrbracket_{vtn}(w) = \text{select}_{\sigma} \circ !$$

For injection of values into the open sum type, we can assume we already have  $\llbracket \Gamma \rrbracket \xrightarrow{\sigma} \llbracket \text{Case } A \rrbracket$  and  $\llbracket \Gamma \rrbracket \xrightarrow{V} \llbracket A \rrbracket$ . We then want to construct  $\llbracket \text{inj}_{\sigma} V \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \text{OSum} \rrbracket$ . First, lets consider the partially applied term  $\llbracket \text{inj}_{\sigma} \rrbracket : \llbracket A \rrbracket \rightarrow \llbracket \text{OSum} \rrbracket$ . Consider the subset  $\{\alpha_w \mid w \text{ s.t. } \sigma \mapsto A \in w\}$  of components of  $\llbracket \text{inj}_{\sigma} \rrbracket$ . These components will be of the form  $\alpha_w = (\text{inl} \mid \text{inr}) \circ \text{inr}^*$  which will inject a value of type  $\llbracket A \rrbracket$  into the open sum type at world  $w$ .  $\llbracket \Gamma \rrbracket \xrightarrow{\sigma, V} \llbracket \text{Case } A \rrbracket \times \llbracket A \rrbracket \xrightarrow{\text{inj}_{\sigma}, \text{id}} \llbracket \text{OSum} \rrbracket^{\llbracket A \rrbracket} \times \llbracket A \rrbracket \xrightarrow{\text{eval}} \llbracket \text{OSum} \rrbracket$

$$\llbracket \text{inj}_{\sigma} V \rrbracket_{vtn} = ?$$

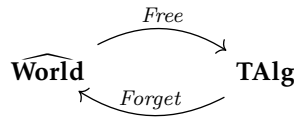
The denotation of product and separating product are both akin to constructing bilinear maps.

$$\begin{aligned} \llbracket (V_1, V_2) \rrbracket_{vtn} &= \llbracket V_1 \rrbracket, \llbracket V_2 \rrbracket \\ \llbracket (V_1 * V_2) \rrbracket_{vtn} &= \llbracket V_1 \rrbracket * \llbracket V_2 \rrbracket \end{aligned}$$

And finally, the interpretation of thunk  $M$  is given by the forgetful functor which projects out the carrier of the algebra  $\llbracket M \rrbracket_{ctm}$ .

$$\llbracket \text{thunk } M \rrbracket_{vtn} = \text{Forget}(\llbracket M \rrbracket_{ctm})$$

The interpretation of computation terms



$$\begin{aligned} \llbracket \text{U} \rrbracket_{ctm} &= \\ \llbracket \text{print } V \rrbracket_{ctm} &= \\ \llbracket \text{force } V \rrbracket_{ctm} &= \llbracket V \rrbracket_{vtn} \\ \llbracket \text{ret } V \rrbracket_{ctm} &= \text{Free}(\llbracket V \rrbracket_{vtn}) \\ \llbracket x \leftarrow M; N \rrbracket_{ctm} &= \\ \llbracket M \ N \rrbracket_{ctm} &= \langle \text{eval}(\llbracket M \rrbracket_{ctm}) (\llbracket N \rrbracket_{vtn}), \theta_{\llbracket M \rrbracket_{ctm}} \llbracket N \rrbracket_{vtn} \rangle \end{aligned}$$

$$\begin{aligned}
\llbracket M@N \rrbracket_{ctm} &= \langle eval^*(\llbracket M \rrbracket_{ctm}) (\llbracket N \rrbracket_{vtm}), \theta_{\llbracket M \rrbracket_{ctm}} \llbracket N \rrbracket_{vtm} \rangle \\
\llbracket \lambda x : A. M \rrbracket_{ctm}(e : \llbracket \Gamma \rrbracket_v) &= \langle \lambda a : \llbracket A \rrbracket_v. \llbracket M \rrbracket_{ctm}(e, a) \mid, \theta \rangle \\
\llbracket \alpha x : A. M \rrbracket_{ctm}(e : \llbracket \Gamma \rrbracket_v) &= \langle \alpha a : \llbracket A \rrbracket_v. \llbracket M \rrbracket_{ctm}(e * a) \mid, \theta \rangle \\
\llbracket \text{newcase}_A \sigma; M \rrbracket_{ctm} &= \text{updated world and value environment?} \\
\llbracket \text{match } V_1 \text{ with } V_2 \{x.M \mid N\} \rrbracket_{ctm} &= \\
\llbracket \text{let } (x, y) = V; M \rrbracket_{ctm} &= \\
\llbracket \text{let } (x * y) = V; M \rrbracket_{ctm} &= \\
\llbracket \text{if } V \text{ then } M \text{ else } N \rrbracket_{ctm} &=
\end{aligned}$$

## References

- [1] <https://math.stackexchange.com/questions/4797353/does-the-category-of-finite-sets-and-injective-functions-i-e-monomorphisms-h>.
- [2] <https://ncatlab.org/nlab/show/day+convolutionday70thesis>.
- [3] FIORE, M., AND HAMANA, M. Multiversal Polymorphic Algebraic Theories: Syntax, Semantics, Translations, and Equational Logic. In *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science* (New Orleans, LA, USA, June 2013), IEEE, pp. 520–529.
- [4] JACOBS, B. Semantics of weakening and contraction. *Annals of Pure and Applied Logic* 69, 1 (1994), 73–106.
- [5] O’HEARN, P. A Model for Syntactic Control of Interference. *College of Engineering and Computer Science - Former Departments, Centers, Institutes and Projects* (Jan. 1993).
- [6] O’HEARN, P. On bunched typing. *Journal of Functional Programming* 13, 4 (July 2003), 747–796.
- [7] O’HEARN, P. W. Resource Interpretations, Bunched Implications and the  $A\lambda$ -Calculus (Preliminary Version). In *Typed Lambda Calculi and Applications*, G. Goos, J. Hartmanis, J. Van Leeuwen, and J.-Y. Girard, Eds., vol. 1581. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999, pp. 258–279.
- [8] PITTS, A. M. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2013.
- [9] PYM, D. J. *The Semantics and Proof Theory of the Logic of Bunched Implications*, vol. 26 of *Applied Logic Series*. Springer Netherlands, Dordrecht, 2002.

- [10] SIMPSON, A. Category-theoretic structure for independence and conditional independence. *Electronic Notes in Theoretical Computer Science* 336 (2018), 281–297. The Thirty-third Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXIII).
- [11] STARK, I. Categorical models for local names. *Lisp and Symbolic Computation* 9, 1 (Feb. 1996), 77–107.
- [12] STARK, I. Names, equations, relations: Practical ways to reason about *new*. In *Typed Lambda Calculi and Applications: Proceedings of the Third International Conference TLCA '97* (1997), no. 1210 in Lecture Notes in Computer Science, Springer-Verlag, pp. 336–353. A full version appears as [?].
- [13] STERLING, J., GRATZER, D., AND BIRKEDAL, L. Free theorems from univalent reference types: The impact of univalence on denotational semantics.
- [14] STERLING, J., GRATZER, D., AND BIRKEDAL, L. Denotational semantics of general store and polymorphism, Apr. 2023. arXiv:2210.02169 [cs].