Finitely partitioned forests (work in progress)

Paul Blain Levy

University of Birmingham

July 21, 2026

Outline

Three kinds of state

- Ground state means that the cells in memory can store integers and booleans.
- General references means that the cells can store anything—even code.

Three kinds of state

- Ground state means that the cells in memory can store integers and booleans.
- General references means that the cells can store anything—even code.
- Full ground state means that the cells can store cell locations, allowing linked data structures. No storage of code.

Three kinds of state

- Ground state means that the cells in memory can store integers and booleans.
- General references means that the cells can store anything—even code.
- Full ground state means that the cells can store cell locations, allowing linked data structures. No storage of code.

This talk is about full ground state.

Models of full ground state

Two models of full ground state:

- Game semantics [Murawski, Tzevelekos 2012]—fully abstract.
- Presheaf semantics [Kammar, Levy, Moss, Staton 2017]

Models of full ground state

Two models of full ground state:

- Game semantics [Murawski, Tzevelekos 2012]—fully abstract.
- Presheaf semantics [Kammar, Levy, Moss, Staton 2017]

New observation

On the first-order fragment, these semantics are the same.

Our goal

For the first-order calculus, we present its semantics explicitly.

Our goal

For the first-order calculus, we present its semantics explicitly.

And show equivalence with the two existing semantics.

Our goal

For the first-order calculus, we present its semantics explicitly.

And show equivalence with the two existing semantics.

A computation will denote a function.

- Not a relation that's merely "functional up to renaming"
- Not a natural transformation defined over all future worlds.

Our goal

For the first-order calculus, we present its semantics explicitly.

And show equivalence with the two existing semantics.

A computation will denote a function.

- Not a relation that's merely "functional up to renaming"
- Not a natural transformation defined over all future worlds.

Hopefully this will help in

- showing the game model is game-enriched [GALOP 2024]
- analysing the presheaf model.

Types

$\mathbf{C}$ is the set of cell-sorts, e.g. red, yellow, blue.
The types are given by

$$A ::= \quad 0 \mid A + A \mid \sum_{i \in \mathbb{N}} A_i \mid 1 \mid A \times A \mid X \mid \mathtt{rec}\, X.\, A \mid \mathtt{ref}_c \; (c \in \mathbf{C})$$

Values of type $\mathtt{ref}_c$ are c-sorted cells.

Types

$\mathbf{C}$ is the set of cell-sorts, e.g. red, yellow, blue.
The types are given by

$$A ::= \quad 0 \mid A + A \mid \sum_{i \in \mathbb{N}} A_i \mid 1 \mid A \times A \mid X \mid \mathtt{rec}\, X.\, A \mid \mathtt{ref}_c \; (c \in \mathbf{C})$$

Values of type $\mathtt{ref}_c$ are c-sorted cells.

Each sort c has a content type $\mathrm{ctype}(c)$,

the type of values that can be stored in a c-sorted cell.

Types

$\mathbf{C}$ is the set of cell-sorts, e.g. red, yellow, blue.
The types are given by

$$A ::= \quad 0 \mid A + A \mid \sum_{i \in \mathbb{N}} A_i \mid 1 \mid A \times A \mid X \mid \mathtt{rec}\, X.\, A \mid \mathtt{ref}_c \; (c \in \mathbf{C})$$

Values of type $\mathtt{ref}_c$ are c-sorted cells.

Each sort c has a content type $\mathrm{ctype}(c)$,

the type of values that can be stored in a c-sorted cell.

The pair $(\mathbf{C}, \mathrm{ctype})$ is a syntactic full ground signature.

Values and computations

A world w is a list of cell-sorts
e.g. [blue, green, blue] has cell 0, cell 1, cell 2.

Values and computations

A world w is a list of cell-sorts
e.g. [blue, green, blue] has cell 0, cell 1, cell 2.

The calculus is fine-grain call-by-value, with two judgements:

- Value $\Gamma \vdash^{\mathsf{v}}_{w} V : A$
- Computation $\Gamma \vdash^{\mathsf{c}}_{w} M : A$

Values and computations

A world w is a list of cell-sorts
e.g. [blue, green, blue] has cell 0, cell 1, cell 2.

The calculus is fine-grain call-by-value, with two judgements:

- Value $\Gamma \vdash^{\mathsf{v}}_{w} V : A$
- Computation $\Gamma \vdash^{\mathsf{c}}_{w} M : A$

The syntax includes reading, writing, generating and equality-testing.

Values and computations

A world w is a list of cell-sorts
e.g. [blue, green, blue] has cell 0, cell 1, cell 2.

The calculus is fine-grain call-by-value, with two judgements:

- Value $\Gamma \vdash^{\mathsf{v}}_{w} V : A$
- Computation $\Gamma \vdash^{\mathsf{c}}_{w} M : A$

The syntax includes reading, writing, generating and equality-testing.

We can also include iteration; then evaluation can diverge.

Extra judgements

Helpful for thinking about values and computations.

Extra judgements

Helpful for thinking about values and computations.

- Extended value $\Gamma \vdash^{\mathsf{xv}}_{w} (x, V) : A$

Extra judgements

Helpful for thinking about values and computations.

- Extended value $\Gamma \vdash^{\mathsf{xv}}_{w} (x, V) : A$
- Extended state $\vdash^{\mathsf{xs}}_{w} (x, s)$

Extra judgements

Helpful for thinking about values and computations.

- Extended value $\Gamma \vdash^{\mathsf{xv}}_{w} (x, V) : A$
- Extended state $\vdash^{\mathsf{xs}}_{w} (x, s)$
- Stateful value $\vdash^{\mathsf{xsv}}_{w} (x, s, V) : A$

Extra judgements

Helpful for thinking about values and computations.

- Extended value $\Gamma \vdash^{\mathsf{xv}}_{w} (x, V) : A$
- Extended state $\vdash^{\mathsf{xs}}_{w} (x, s)$
- Stateful value $\vdash^{\mathsf{xsv}}_{w} (x, s, V) : A$
- Stateful computation $\vdash^{\mathsf{xsc}}_{w} (x, s, M) : A$

Platforms

A platform is a finite set R with a sort function $R \to \mathbf{C}$.

Each world w denotes a platform $[\![w]\!]$.

Platforms

A platform is a finite set R with a sort function $R \to \mathbf{C}$.

Each world w denotes a platform $[\![w]\!]$.

Category Inj of platforms and injections.
Category pInj of platforms and partial injections.

Platforms

A platform is a finite set R with a sort function $R \to \mathbf{C}$.

Each world w denotes a platform $[\![w]\!]$.

Category Inj of platforms and injections.
Category pInj of platforms and partial injections.

A type denotes a platform family.

Ultimate pattern-matching

Each closed value $\vdash_w V : A$ decomposes into:

- ultimate pattern + filling
- ultimate pattern + partition + injection
- partitioned ultimate pattern + injection

Ultimate pattern-matching

Each closed value $\vdash_w V : A$ decomposes into:

- ultimate pattern + filling
- ultimate pattern + partition + injection
- partitioned ultimate pattern + injection

Example

Consider $\langle \texttt{inl fold inr}\ l, \langle l', l \rangle \rangle$ where $l \neq l'$.

Ultimate pattern-matching

Each closed value $\vdash_w V : A$ decomposes into:

- ultimate pattern + filling
- ultimate pattern + partition + injection
- partitioned ultimate pattern + injection

Example

Consider $\langle \texttt{inl fold inr } l, \langle l', l \rangle \rangle$ where $l \neq l'$.

The ultimate pattern is $\langle \texttt{inl fold inr } -, \langle -, - \rangle \rangle$.
The filling is l, l', l.

Ultimate pattern-matching

Each closed value $\vdash_w V : A$ decomposes into:

- ultimate pattern + filling
- ultimate pattern + partition + injection
- partitioned ultimate pattern + injection

Example

Consider $\langle \texttt{inl fold inr } l, \langle l', l \rangle \rangle$ where $l \neq l'$.

The ultimate pattern is $\langle \texttt{inl fold inr } -, \langle -, - \rangle \rangle$.
The filling is l, l', l.

The partition is $0 \equiv 2 \not\equiv 1$.
The injection sends $\{0, 2\} \mapsto l$ and $\{1\} \mapsto l'$.

Platform families

A type A denotes a platform family $(R_i)_{i \in I}$.

Platform families

A type A denotes a platform family $(R_i)_{i \in I}$.

Idea

I is the set of partitioned ultimate patterns.

R_i is the quotient platform for i.

Platform families

A type A denotes a platform family $(R_i)_{i \in I}$.

Idea

I is the set of partitioned ultimate patterns.

R_i is the quotient platform for i.

Example

This platform family $[\![\mathtt{ref}_c \times \mathtt{ref}_c]\!]$
consists of a doubleton and a singleton.

Product type

A partition of platforms R and S is
an equivalence relation E on $R + S$ that's equality on each component.

Product type

A partition of platforms R and S is
an equivalence relation E on $R + S$ that's equality on each component.

Corresponds to a partial injection.

Product type

A partition of platforms R and S is
an equivalence relation E on $R + S$ that's equality on each component.

Corresponds to a partial injection.

The quotient platform is written $\mathrm{quot}(E)$.

Product type

A partition of platforms R and S is
an equivalence relation E on $R + S$ that's equality on each component.

Corresponds to a partial injection.

The quotient platform is written $\mathrm{quot}(E)$.

The set of partitions is written $\mathrm{Partit}(R, S)$.

Product type

A partition of platforms R and S is
an equivalence relation E on $R + S$ that's equality on each component.

Corresponds to a partial injection.

The quotient platform is written $\mathsf{quot}(E)$.

The set of partitions is written $\mathsf{Partit}(R, S)$.

The product of platform families:

$$(R_i)_{i \in I} \times (S_j)_{j \in J} \stackrel{\text{def}}{=} (\mathsf{quot}(E))_{i \in I, j \in J, E \in \mathsf{Partit}(R_i, S_j)}$$

Product type

A partition of platforms R and S is
an equivalence relation E on $R + S$ that's equality on each component.

Corresponds to a partial injection.

The quotient platform is written $\mathsf{quot}(E)$.

The set of partitions is written $\mathsf{Partit}(R, S)$.

The product of platform families:

$$(R_i)_{i \in I} \times (S_j)_{j \in J} \stackrel{\text{def}}{=} (\mathsf{quot}(E))_{i \in I, j \in J, E \in \mathsf{Partit}(R_i, S_j)}$$

Values and extended values: closed

Let $[\![w]\!] = T$ and $[\![A]\!] = S = (S_j)_{j \in J}$.

Values and extended values: closed

Let $[\![w]\!] = T$ and $[\![A]\!] = S = (S_j)_{j \in J}$.

A closed value $\vdash^{\mathsf{v}}_{w} V : A$ denotes an element of

$$\mathsf{Val}_T(S) \quad \stackrel{\text{def}}{=} \quad \sum_{j \in J} \mathsf{Inj}(S_j, T)$$

Values and extended values: closed

Let $[\![w]\!] = T$ and $[\![A]\!] = S = (S_j)_{j \in J}$.

A closed value $\vdash^{\mathsf{v}}_w V : A$ denotes an element of

$$\mathsf{Val}_T(S) \stackrel{\text{def}}{=} \sum_{j \in J} \mathsf{Inj}(S_j, T)$$

A closed extended value $\vdash^{\mathsf{xv}}_w (x, V) : A$ denotes an element of

$$\mathsf{xVal}_T(S) \stackrel{\text{def}}{=} \sum_{j \in J} \mathsf{pInj}(S_j, T)$$

Values and extended values: closed

Let $[\![w]\!] = T$ and $[\![A]\!] = S = (S_j)_{j \in J}$.

A closed value $\vdash^{\mathsf{v}}_w V : A$ denotes an element of

$$\mathsf{Val}_T(S) \stackrel{\text{def}}{=} \sum_{j \in J} \mathsf{Inj}(S_j, T)$$

A closed extended value $\vdash^{\mathsf{xv}}_w (x, V) : A$ denotes an element of

$$\mathsf{xVal}_T(S) \stackrel{\text{def}}{=} \sum_{j \in J} \mathsf{pInj}(S_j, T)$$

Each element $r = \langle j, E \rangle$ has quotient platform $\mathsf{quot}(r) \stackrel{\text{def}}{=} \mathsf{quot}(E)$.

Values and extended values: open

Let $[\![w]\!] = T$ and $[\![\Gamma]\!] = R = (R_i)_{i \in I}$ and $[\![A]\!] = S = (S_j)_{j \in J}$.

Values and extended values: open

Let $[\![w]\!] = T$ and $[\![\Gamma]\!] = R = (R_i)_{i \in I}$ and $[\![A]\!] = S = (S_j)_{j \in J}$.

A value $\Gamma \vdash^{\mathsf{v}}_{w} V : A$ denotes an element of

$$\mathsf{Val}_T(R, S) \stackrel{\text{def}}{=} \prod_{r \in \mathsf{x}\mathsf{Val}_T(R)} \mathsf{Val}_{\mathsf{quot}(r)}(S)$$

Values and extended values: open

Let $[\![w]\!] = T$ and $[\![\Gamma]\!] = R = (R_i)_{i \in I}$ and $[\![A]\!] = S = (S_j)_{j \in J}$.

A value $\Gamma \vdash^{\mathsf{v}}_w V : A$ denotes an element of

$$\mathsf{Val}_T(R, S) \stackrel{\text{def}}{=} \prod_{r \in \mathsf{xVal}_T(R)} \mathsf{Val}_{\mathsf{quot}(r)}(S)$$

An extended value $\Gamma \vdash^{\mathsf{xv}}_w (x, V) : A$ denotes an element of

$$\mathsf{xVal}_T(R, S) \stackrel{\text{def}}{=} \prod_{r \in \mathsf{xVal}_T(R)} \mathsf{xVal}_{\mathsf{quot}(r)}(S)$$

We obtain categories Val_T and xVal_T.

Signatures

A syntactic signature $(\mathbf{C}, \mathrm{ctype})$ assigns to each cell sort c a closed type $\mathrm{ctype}(c)$.

The language depends on a syntactic signature.

Signatures

A syntactic signature $(\mathbf{C}, \mathrm{ctype})$ assigns to each cell sort c a closed type $\mathrm{ctype}(c)$.

The language depends on a syntactic signature.

A semantic signature $(\mathbf{C}, \mathrm{cpf})$ assigns to each cell sort c a platform family $\mathrm{cpf}(c)$.

Signatures

A syntactic signature $(\mathbf{C}, \mathrm{ctype})$ assigns to each cell sort c a closed type $\mathrm{ctype}(c)$.

The language depends on a syntactic signature.

A semantic signature $(\mathbf{C}, \mathrm{cpf})$ assigns to each cell sort c a platform family $\mathrm{cpf}(c)$.

The model depends on a semantic signature.

Semantics of an extended state

Let $[\![w]\!] = T$.

Semantics of an extended state

Let $[\![w]\!] = T$.

What does an extended state $\vdash^{\mathsf{h}}_{w} (x, s)$ denote?

Semantics of an extended state

Let $[\![w]\!] = T$.

What does an extended state $\vdash^{\mathsf{h}}_{w} (x, s)$ denote?

Of the x-cells, only the reachable ones should be in the semantics.

Frugality condition in [MT12].

Semantics of an extended state

Let $[\![w]\!] = T$.

What does an extended state $\vdash^{\mathsf{h}}_{w} (x, s)$ denote?

Of the x-cells, only the reachable ones should be in the semantics.

Frugality condition in [MT12].

Furthermore, they should have no name and no order.

The forest, syntactically

Given the syntactic signature $(\mathbf{C}, \mathrm{ctype})$,

The forest, syntactically

Given the syntactic signature $(\mathbf{C}, \mathrm{ctype})$,

what does an extended state $\vdash^{\mathsf{h}}_{w} (x, s)$ denote?

The forest, syntactically

Given the syntactic signature $(\mathbf{C}, \mathrm{ctype})$,

what does an extended state $\vdash^{\mathsf{h}}_{w} (x, s)$ denote?

Game:

- O asks about l_0 in w, of sort $c_0 \mapsto A_0$.
- P gives the partitioned ultimate pattern p_0 of the contents there.
- O asks about l_1 in that pattern, of sort $c_1 \mapsto A_1$.
- P gives the partitioned ultimate pattern p_1 of the contents.
- And so forth.

The forest, syntactically

Given the syntactic signature $(\mathbf{C}, \mathrm{ctype})$,

what does an extended state $\vdash^{\mathsf{h}}_{w} (x, s)$ denote?

Game:

- O asks about l_0 in w, of sort $c_0 \mapsto A_0$.
- P gives the partitioned ultimate pattern p_0 of the contents there.
- O asks about l_1 in that pattern, of sort $c_1 \mapsto A_1$.
- P gives the partitioned ultimate pattern p_1 of the contents.
- And so forth.

A forest (strategy for this game) gives the tags and the local aliasing.

The forest, syntactically

Given the syntactic signature $(\mathbf{C}, \mathrm{ctype})$,

what does an extended state $\vdash^{\mathsf{h}}_{w} (x, s)$ denote?

Game:

- O asks about l_0 in w, of sort $c_0 \mapsto A_0$.
- P gives the partitioned ultimate pattern p_0 of the contents there.
- O asks about l_1 in that pattern, of sort $c_1 \mapsto A_1$.
- P gives the partitioned ultimate pattern p_1 of the contents.
- And so forth.

A forest (strategy for this game) gives the tags and the local aliasing.

Then we have to describe the global aliasing.

The forest, semantically

Given the semantic signature $(\mathbf{C}, \mathrm{ctype})$,

and $[\![w]\!] = T$,

The forest, semantically

Given the semantic signature $(\mathbf{C}, \mathrm{ctype})$,

and $[\![w]\!] = T$,

what does an extended state $\vdash^{\mathsf{h}}_{w} (x, s)$ denote?

The forest, semantically

Given the semantic signature $(\mathbf{C}, \mathrm{ctype})$,

and $[\![w]\!] = T$,

what does an extended state $\vdash^{\mathsf{h}}_{w} (x, s)$ denote?

Game:

- O asks about $l_0 \in T$ of sort $c_0 \mapsto (R_i)_{i \in I}$.
- P gives $i_0 \in I$.
- O asks about $l_1 \in R_{i_0}$ of sort $c_1 \mapsto (S_j)_{j \in J}$.
- P gives $i_1 \in J$.
- And so forth.

The forest, semantically

Given the semantic signature $(\mathbf{C}, \mathrm{ctype})$,

and $[\![w]\!] = T$,

what does an extended state $\vdash^{\mathsf{h}}_{w} (x, s)$ denote?

Game:

- O asks about $l_0 \in T$ of sort $c_0 \mapsto (R_i)_{i \in I}$.
- P gives $i_0 \in I$.
- O asks about $l_1 \in R_{i_0}$ of sort $c_1 \mapsto (S_j)_{j \in J}$.
- P gives $i_1 \in J$.
- And so forth.

A play is a sequence $(l_0{:}c_0), i_0, (l_1{:}c_1), i_1, \ldots$

The forest, semantically

Given the semantic signature $(\mathbf{C}, \mathrm{ctype})$,

and $[\![w]\!] = T$,

what does an extended state $\vdash^{\mathsf{h}}_{w} (x, s)$ denote?

Game:

- O asks about $l_0 \in T$ of sort $c_0 \mapsto (R_i)_{i \in I}$.
- P gives $i_0 \in I$.
- O asks about $l_1 \in R_{i_0}$ of sort $c_1 \mapsto (S_j)_{j \in J}$.
- P gives $i_1 \in J$.
- And so forth.

A play is a sequence $(l_0{:}c_0), i_0, (l_1{:}c_1), i_1, \ldots$

A forest on T is a set of passive-ending (i.e. even-length) plays that is prefix-closed, contains ε and is total.

Partition of a forest σ

For each passive-ending play $s \in \sigma$, we have a platform M_s consisting of the possible next O-moves.

Partition of a forest σ

For each passive-ending play $s \in \sigma$, we have a platform M_s consisting of the possible next O-moves.

A partition of $(M_s)_{s \in \sigma}$ is a sort-respecting equivalence relation E on $\sum_{s \in \sigma} M_s$ that's equality on each component.

Partition of a forest σ

For each passive-ending play $s \in \sigma$, we have a platform M_s consisting of the possible next O-moves.

A partition of $(M_s)_{s \in \sigma}$ is a sort-respecting equivalence relation E on $\sum_{s \in \sigma} M_s$ that's equality on each component.

Partition of a forest σ

For each passive-ending play $s \in \sigma$, we have a platform M_s consisting of the possible next O-moves.

A partition of $(M_s)_{s \in \sigma}$ is a sort-respecting equivalence relation E on $\sum_{s \in \sigma} M_s$ that's equality on each component.

Requirements:

- Let $\langle s, l \rangle \equiv \langle s', l' \rangle$ of sort $c \mapsto (R_i)_{i \in I}$.
  The next move $i \in I$ of $s.l$ is also that of $s'.l'$,
  and for any $k \in R_i$, we have $\langle s.l.i, k \rangle \equiv \langle s'.l'.i, k \rangle$.

Partition of a forest σ

For each passive-ending play $s \in \sigma$, we have a platform M_s consisting of the possible next O-moves.

A partition of $(M_s)_{s \in \sigma}$ is a sort-respecting equivalence relation E on $\sum_{s \in \sigma} M_s$ that's equality on each component.

Requirements:

- Let $\langle s, l \rangle \equiv \langle s', l' \rangle$ of sort $c \mapsto (R_i)_{i \in I}$.
  The next move $i \in I$ of $s.l$ is also that of $s'.l'$,
  and for any $k \in R_i$, we have $\langle s.l.i, k \rangle \equiv \langle s'.l'.i, k \rangle$.
- The quotient set quot(E) must be finite, i.e. a platform,
  as only finitely many cells are listed in x.

Write FPForest(T) for the set of all finitely partitioned forests $t = \langle \sigma, E \rangle$

Stateful value

Let $[\![w]\!] = T$ and $[\![A]\!] = S$.

Stateful value

Let $[\![w]\!] = T$ and $[\![A]\!] = S$.

A stateful value $\vdash^{\mathsf{xsv}}_{w} (x, s, V) : A$
consists of an extended value and a state.

Stateful value

Let $[\![w]\!] = T$ and $[\![A]\!] = S$.

A stateful value $\vdash^{\mathsf{xsv}}_{w} (x, s, V) : A$
consists of an extended value and a state.

So it denotes an element of

$$\mathsf{xsVal}_T(S) \stackrel{\text{def}}{=} \sum_{r \in \mathsf{xVal}_T S} \mathsf{FPForest}(\mathsf{quot}(r))$$

Stateful computation

Let $[\![w]\!] = T$ and $[\![A]\!] = S$.

Stateful computation

Let $[\![w]\!] = T$ and $[\![A]\!] = S$.

A stateful computation $\vdash^{\mathsf{xsc}}_{w} (x, s, M) : A$
either diverges or returns a stateful value,

Stateful computation

Let $[\![w]\!] = T$ and $[\![A]\!] = S$.

A stateful computation $\vdash^{\mathsf{xsc}}_{w} (x, s, M) : A$
either diverges or returns a stateful value,

So it denotes an element of $\mathsf{xsVal}_T(S)_\perp$.

Computation

Let $[\![w]\!] = T$ and $[\![\Gamma]\!] = R$ and $[\![A]\!] = S$.

Computation

Let $[\![w]\!] = T$ and $[\![\Gamma]\!] = R$ and $[\![A]\!] = S$.

A computation $\Gamma \vdash^{\mathsf{c}}_{w} M : A$ denotes an element of

$$\mathcal{K}_T(R, S) \stackrel{\text{def}}{=} \prod_{t \in \mathsf{xsVal}_T(R)} \mathsf{xsVal}_{\mathsf{quot}(t)}(S)_\perp$$

Computation

Let $[\![w]\!] = T$ and $[\![\Gamma]\!] = R$ and $[\![A]\!] = S$.

A computation $\Gamma \vdash^{\mathsf{c}}_{w} M : A$ denotes an element of

$$\mathcal{K}_T(R, S) \stackrel{\text{def}}{=} \prod_{t \in \mathsf{xsVal}_T(R)} \mathsf{xsVal}_{\mathsf{quot}(t)}(S)_\perp$$

Distributive Freyd category $\mathsf{Val}_T \to \mathcal{K}_T$ with uniform Elgot structure.
Ghalayini and Krishnaswami [GALOP 2024]

The empty platform

For a platform family $R = (R_i)_{i \in I}$ we have

$$\begin{aligned} \mathsf{xVal}_\emptyset R &\cong I \\ \mathsf{xsVal}_\emptyset R &\cong \sum_{i \in I} \mathsf{FPForest}(R_i) \end{aligned}$$

The empty platform

For a platform family $R = (R_i)_{i \in I}$ we have

$$\begin{aligned} \mathsf{xVal}_\emptyset R &\cong I \\ \mathsf{xsVal}_\emptyset R &\cong \sum_{i \in I} \mathsf{FPForest}(R_i) \end{aligned}$$

And we have

$$\begin{aligned} \mathsf{Val}_\emptyset &\cong \mathbf{Fam}(\mathsf{Inj}^{\mathsf{op}}) \simeq \text{Strong nominal sets} \\ \mathsf{xVal}_\emptyset &\cong \mathbf{Fam}(\mathsf{pInj}^{\mathsf{op}}) \end{aligned}$$

The empty platform

For a platform family $R = (R_i)_{i \in I}$ we have

$$\begin{aligned} \mathsf{xVal}_\emptyset R &\cong I \\ \mathsf{xsVal}_\emptyset R &\cong \sum_{i \in I} \mathsf{FPForest}(R_i) \end{aligned}$$

And we have

$$\begin{aligned} \mathsf{Val}_\emptyset &\cong \mathbf{Fam}(\mathsf{Inj}^{\mathsf{op}}) \simeq \text{Strong nominal sets} \\ \mathsf{xVal}_\emptyset &\cong \mathbf{Fam}(\mathsf{pInj}^{\mathsf{op}}) \end{aligned}$$

For a platform T, define $T.R \stackrel{\text{def}}{=} (\mathsf{quot}(E))_{i \in I, E \in \mathsf{pInj}(R_i, T)}$

Then $T.-$ is a comonad on the categories $\mathsf{xVal}_\emptyset \leftarrow \mathsf{Val}_\emptyset \rightarrow \mathcal{K}_\emptyset$

and the coKleisli categories are $\mathsf{xVal}_T \leftarrow \mathsf{Val}_T \rightarrow \mathcal{K}_T$.

Summary (work in progress)

We give a denotational semantics of the first-order language agreeing with [MT12] and [KLMS17].

Summary (work in progress)

We give a denotational semantics of the first-order language agreeing with [MT12] and [KLMS17].

A world denotes a platform, a type a platform family.

Summary (work in progress)

We give a denotational semantics of the first-order language

agreeing with [MT12] and [KLMS17].

A world denotes a platform, a type a platform family.

For each platform T, we get a

distributive Freyd category $\mathrm{Val}_T \to \mathcal{K}_T$ with uniform Elgot structure.

Summary (work in progress)

We give a denotational semantics of the first-order language

agreeing with [MT12] and [KLMS17].

A world denotes a platform, a type a platform family.

For each platform T, we get a

distributive Freyd category $\mathrm{Val}_T \to \mathcal{K}_T$ with uniform Elgot structure.

A computation denotes a function.