A monad for full ground reference cells

Ohad Kammar
<ohad.kammar@cs.ox.ac.uk>
joint work with
Paul B. Levy, Sean K. Moss, and Sam Staton

http://arxiv.org/abs/1702.04908

University of Oxford
Department of Computer Science
Oxford Advanced Seminar on Informatic Structures (OASIS)
10 March 2017

Dynamic allocation of memory cells

Kinds of storage

$$(* \text{ ground } *)$$
$$\textbf{type}\ \texttt{data} = \textbf{bool}$$

$$(* \text{ full ground: refs to refs, cyclic data } *)$$
$$\textbf{type}\ \texttt{linked\_list} = \mathbf{1} + \textbf{ref}\ \texttt{list\_cell}$$
$$\textbf{type}\ \texttt{list\_cell} = \textbf{ref}\ \texttt{data} * \textbf{ref}\ \texttt{linked\_list}$$

$$(* \text{ general: suspended computation/effectful functions } *)$$
$$\textbf{type}\ \texttt{process} = \mathbf{1} + (\mathbf{1} \to \textbf{ref}\ \texttt{process})$$

This talk

Full ground storage only

A semantic gap

Success stories

- Operational semantics: general
- Relational semantics [Benton, Hofmann, et al., Bohr and Birkedal'06]: general with dangling pointers
- Step-indexing [Ahmed'04, Birkedal et al.'10 etc]: general small print
- Game semantics: general storage with partiality
- Parametricity semantics [Reddy-Yang'04]: full ground

A dead end? [Reynolds and Oles'82, Moggi'90, O'Hearn and Tennent'92, Stark'94, Ghica'97, Plotkin-Power'02, Levy'02]

- Sets-with-structure and structure preserving functions
- Possible-world semantics
- Monad over a bi-CCC
- Avoid dangling pointers

Contribution

Goal

Semantics for full ground storage:

- Sets-with-structure and structure preserving functions
- Possible-world semantics
- Monad over a bi-CCC
- Avoid dangling pointers

Evaluation

- Effect masking
- Adequacy
- Local ground storage equations

Talk structure

- Full ground storage
- Worlds and initialisations
- A monad for hiding/encapsulation
- The monad, explicitly
- Evaluation
- Conclusion

Aspiration

Computational motivation alongside technical development

Full ground storage

Full ground signature $\langle \mathbb{n}, \textbf{type} \rangle$

- Countably many $c \in \mathbb{n}$ **storable-type names**
- A **full ground type interpretation type** $: \mathbb{n} \to \mathbb{G}$

Where the **full ground types** $\gamma \in \mathbb{G}$ are:

$$\gamma ::= \mathbf{0} \mid \gamma_1 + \gamma_2 \mid \mathbf{1} \mid \gamma_1 * \gamma_2 \mid \mathbf{ref}\ c$$

Example

$\mathbb{n} \stackrel{\text{def}}{=} \{\texttt{data}, \texttt{linked\_list}, \texttt{list\_cell}\}$

$$\begin{aligned} \textbf{type}\ \texttt{data} &= \mathbf{bool} \\ \textbf{type}\ \texttt{linked\_list} &= \mathbf{1} + \mathbf{ref}\ \texttt{list\_cell} \\ \textbf{type}\ \texttt{list\_cell} &= \mathbf{ref}\ \texttt{data} * \mathbf{ref}\ \texttt{linked\_list} \end{aligned}$$

A call-by-value calculus: λ_{ref} (syntax)

$\tau ::=$	types
$\mathbf{ref}\ c$	reference
$\mathbf{0}$	empty
$\tau_1 + \tau_2$	binary sum
$\mathbf{1}$	unit
$\tau_1 * \tau_2$	product
$\tau_1 \to \tau_2$	function

$v ::=$	values
ℓ	location ($\ell \in \mathbb{L}$)
x	identifier
$\mathbf{inj}_i^{\tau_1+\tau_2} v$	constructor
$()$	unit
(v_1, v_2)	pair
$\lambda x : \tau.t$	function

Sugar

$\mathbf{let}\ (x : \tau) = t\ \mathbf{in}\ s \equiv (\lambda x : \tau.s)\ t$

$\mathbf{new}_c t \equiv$

$\quad \mathbf{let}\ (x : \mathbf{type}\ c) = t$

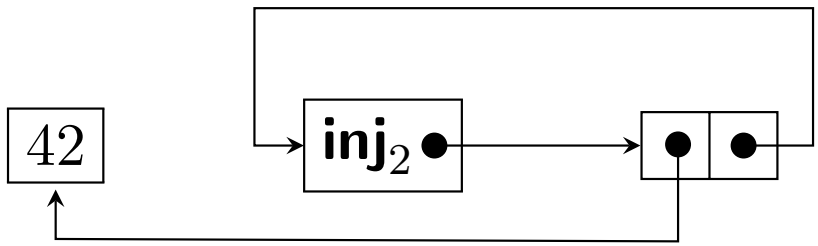
$\quad \mathbf{in}\ \ \mathbf{letref}\ (y : \mathbf{ref}\ c) := x\ \mathbf{in}\ y$

$t, s ::=$	terms
ℓ	location ($\ell \in \mathbb{L}$)
x	identifier
$\mathbf{inj}_i^{\tau_1+\tau_2} t$	sum constructor
$()$	unit
(t, s)	pair
$\lambda x : \tau.t$	function abstraction
$\mathbf{match}\ t\ \mathbf{with}\ \{\}^{\tau}$	pattern matching
$\mathbf{match}\ t\ \mathbf{with}$ $\quad \{\mathbf{inj}_1 x_1 \mapsto s_1$ $\quad \mid \mathbf{inj}_2 x_2 \mapsto s_2\}$	
$\mathbf{prj}_i t$	projections
$t\ s$	function application
$t := s$	assignment
$!\,t$	dereferencing
$\mathbf{letref}$ $\quad (x_1 : \mathbf{ref}\ c_1) := v_1,$ $\quad \vdots$ $\quad (x_n : \mathbf{ref}\ c_n) := v_n$ $\mathbf{in}\ t$	allocation

A call-by-value calculus: λ_{ref} (allocation syntax)

Example

$$\begin{aligned}
&\textbf{letref}\ (\texttt{payload} &&: \textbf{ref}\ \texttt{data}) &&:= 42,\\
&\qquad(\texttt{cyclic_list} &&: \textbf{ref}\ \texttt{linked_list}) &&:= \textbf{inj}_2\texttt{head},\\
&\qquad(\texttt{head} &&: \textbf{ref}\ \texttt{list_cell}) &&:= (\texttt{payload}, \texttt{cyclic_list})\\
&\textbf{in}\ \texttt{cyclic_list}
\end{aligned}$$



A call-by-value calculus: λ_{ref} (type system)

Typing judgements $\boxed{\Gamma \vdash_w t : \tau}$

Type contexts: $\Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n$

Heap layouts: $w = \{\ell_1 : c_1, \ldots, \ell_n : c_n\}$

$\underline{w} \stackrel{\mathsf{def}}{=} \{\ell_1, \ldots, \ell_n\}$

$$\frac{(\ell : c) \in w}{\Gamma \vdash_w \ell : \mathbf{ref}\ c}$$

$$\frac{\Gamma \vdash_w t : \mathbf{ref}\ c \qquad \Gamma \vdash_w s : \mathbf{type}\ c}{\Gamma \vdash_w t := s : \mathbf{1}} \qquad \frac{\Gamma \vdash_w t : \mathbf{ref}\ c}{\Gamma \vdash_w\ !t : \mathbf{type}\ c}$$

$$\frac{\begin{array}{c}\text{for } i = 1, \ldots, n\text{: } \Gamma, x_1 : \mathbf{ref}\ c_1, \ldots, x_n : \mathbf{ref}\ c_n \vdash_w v_i : \mathbf{type}\ c_i \\ \Gamma, x_1 : \mathbf{ref}\ c_1, \ldots, x_n : \mathbf{ref}\ c_n \vdash_w t : \tau\end{array}}{\begin{array}{l}\Gamma \vdash_w \mathbf{letref}(x_1 : \mathbf{ref}\ c_1) := v_1, \\ \qquad \vdots \\ \qquad (x_n : \mathbf{ref}\ c_n) := v_n\ \mathbf{in}\ t : \tau\end{array}}$$

A call-by-value calculus: λ_{ref} (operational semantics)

$$\boxed{\eta = \langle v_\ell \rangle_{\ell \in \underline{w}_\eta}}$$

$$\frac{}{\langle \ell, \eta \rangle \Downarrow \langle \ell, \eta \rangle}$$

$$\frac{\langle t, \eta \rangle \Downarrow \langle \lambda x : \tau.t', \eta_1 \rangle \quad \langle s, \eta_1 \rangle \Downarrow \langle v', \eta_2 \rangle \quad \langle t'[x \mapsto v'], \eta_2 \rangle \Downarrow \langle v, \eta' \rangle}{\langle t\ s, \eta \rangle \Downarrow \langle v, \eta' \rangle}$$

$$\frac{\langle t, \eta \rangle \Downarrow \langle v, \eta' \rangle}{\left\langle \mathbf{inj}_i^{\tau_1+\tau_2} t, \eta \right\rangle \Downarrow \left\langle \mathbf{inj}_i^{\tau_1+\tau_2} v, \eta' \right\rangle}$$

$$\frac{\langle t, \eta \rangle \Downarrow \langle \ell, \hat{\eta} \rangle \quad \langle s, \hat{\eta} \rangle \Downarrow \langle v, \eta' \rangle}{\langle t := s, \eta \rangle \Downarrow \langle (), \eta'[\ell \mapsto v] \rangle} (\ell \in \underline{w}_{\eta'})$$

$$\frac{}{\langle (), \eta \rangle \Downarrow \langle (), \eta \rangle}$$

$$\frac{\langle t_1, \eta \rangle \Downarrow \langle v_1, \hat{\eta} \rangle \quad \langle t_2, \hat{\eta} \rangle \Downarrow \langle v_2, \eta' \rangle}{\langle (t_1, t_2), \eta \rangle \Downarrow \langle (v_1, v_2), \eta' \rangle}$$

$$\frac{\langle t, \eta \rangle \Downarrow \langle \ell, \eta' \rangle}{\langle !\, t, \eta \rangle \Downarrow \langle \eta'(\ell), \eta' \rangle} (\ell \in w_{\eta'})$$

$$\frac{}{\langle \lambda x : \tau.t, \eta \rangle \Downarrow \langle \lambda x : \tau.t, \eta \rangle}$$

$$\frac{\langle t, \eta \rangle \Downarrow \left\langle \mathbf{inj}_i^{\tau_1+\tau_2} \hat{v}, \hat{\eta} \right\rangle \quad \langle s_i[x_i \mapsto \hat{v}], \hat{\eta} \rangle \Downarrow \langle v, \eta' \rangle}{\left\langle \begin{array}{l} \textbf{match}\, t\ \textbf{with} \\ \{\mathbf{inj}_1 x_1 \mapsto s_1 \\ \mid \mathbf{inj}_2 x_2 \mapsto s_2\} \end{array}, \eta \right\rangle \Downarrow \langle v, \eta' \rangle}$$

$$\frac{\left\langle t[\theta], \eta \left[\ell_i \mapsto v_i[\theta]\right]_{i=1}^n \right\rangle \Downarrow \langle v, \eta' \rangle \quad \text{where } \theta \stackrel{\text{def}}{=} [x_i \mapsto \ell_i]_{i=1}^n}{\textbf{letref}\ \left\langle \begin{array}{l} (x_1 : \mathbf{ref}\ c_1) := v_1, \\ \vdots \\ (x_n : \mathbf{ref}\ c_n) := v_n\ \textbf{in}\ t \end{array}, \eta \right\rangle \Downarrow \langle v, \eta' \rangle} (\#_{\underline{w}_\eta} \langle \ell_1, \ldots, \ell_n \rangle)$$

$$\frac{\langle t, \eta \rangle \Downarrow \langle (v_1, v_2), \eta' \rangle}{\langle \mathbf{prj}_i t, \eta \rangle \Downarrow \langle v_i, \eta' \rangle}$$

A call-by-value calculus: λ_{ref} (meta-theory)

Functoriality of types

If $w \leq w'$ as partial functions, then $\Gamma \vdash_w t : \tau \implies \Gamma \vdash_{w'} t : \tau$. So

$$\tau^{\mathrm{v}} w \subseteq \tau^{\mathrm{v}} w' \qquad \text{and} \qquad \tau^{\mathrm{t}} w \subseteq \tau^{\mathrm{t}} w' \qquad \text{where}$$

$$\tau^{\mathrm{v}} w \stackrel{\text{def}}{=} \{v | \vdash_w v : \tau\} \qquad \text{and} \qquad \tau^{\mathrm{t}} w \stackrel{\text{def}}{=} \{t | \vdash_w t : \tau\}$$

Non-functoriality of heaps

For $\mathbb{H}w := \left\{\eta \middle| \underline{w_\eta} = \underline{w}, \forall(\ell : c) \in w. \vdash_w \eta(\ell) : \textbf{type}\ c\right\}$ and $w \leq w'$ we **lack** functions

$$\mathbb{H}w \to \mathbb{H}w' \qquad \text{and} \qquad \mathbb{H}w \leftarrow \mathbb{H}w'$$

A call-by-value calculus: λ_{ref} (meta-theory)

Non-functoriality of heaps

For $\mathbb{H}w := \left\{\eta \middle| \underline{w}_\eta = \underline{w}, \forall(\ell : c) \in w.\, \vdash_w \eta(\ell) : \textbf{type}\ c\right\}$ and $w \leq w'$ we **lack** functions

$$\mathbb{H}w \to \mathbb{H}w' \qquad \text{and} \qquad \mathbb{H}w \leftarrow \mathbb{H}w'$$

as we lack **initialisation data** and want to avoid **dangling references**:

$$\begin{aligned} \langle\rangle \in \mathbb{H}\emptyset &\mapsto ? \in \mathbb{H}\left\{\ell \ : \mathtt{ptr}\right\} \\ ? \in \mathbb{H}\left\{\ell_1 : \mathtt{ptr}\right\} &\mapsfrom \left\langle \begin{array}{l} \ell_1 \mapsto \ell_2, \\ \ell_2 \mapsto \ell_1 \end{array} \right\rangle \in \mathbb{H}\left\{\ell_1, \ell_2 : \mathtt{ptr}\right\} \end{aligned}$$

for **type** $\mathtt{ptr} = \textbf{ref}\ \mathtt{ptr}$.

A call-by-value calculus: λ_{ref} (meta-theory)

Theorem (totality)

All well-typed closed programs fully evaluate:

$$\forall \vdash_w t : \tau, w_1 \geq w, \eta_1 \in \mathbb{H}w_1.$$
$$\implies \exists w_2 \geq w_1, \vdash_{w_2} v : \tau, \eta_2 \in \mathbb{H}w_2. \langle t, \eta_1 \rangle \Downarrow \langle v, \eta_2 \rangle$$

Proof highlights

Use Tait's method with Kripke logical predicates, and a w-indexed predicate on $\mathbb{H}w$ that is **not** Kripke.

Allocation case: initialisation data $\vdash_{w \oplus \{\ell_1 : c_n, \ldots, \ell_n : c_n\}} v_1, \ldots, v_n$ needs to extend $\eta \in \mathbb{H}w_1$, $w_1 \geq w$:

$$\begin{array}{ccc} w & \leq & w \oplus \{\ell_1 : c_1, \ldots, \ell_n : c_n\} \\ \mid\!\wedge & & \mid\!\wedge \\ w' & \leq & w' \oplus \{\ell_1 : c_1, \ldots, \ell_n : c_n\} \end{array}$$

Talk structure

- Full ground storage
- Worlds and initialisations
- A monad for hiding/encapsulation
- The monad, explicitly
- Evaluation
- Conclusion

Aspiration

Computational motivation alongside technical development

Worlds

The category $\mathbb{W}$

Objects: layouts w

Morphisms $\rho : w \to w'$: label preserving injections $\rho : \underline{w} \rightarrowtail \underline{w'}$

Interpreting full ground types

Set $\mathbf{V} := [\mathbb{W}, \mathbf{Set}]$ and define:

$$[\![\mathbf{0}]\!] \stackrel{\text{def}}{=} \mathbb{0} \qquad [\![\gamma_1 + \gamma_2]\!] \stackrel{\text{def}}{=} [\![\gamma_1]\!] + [\![\gamma_2]\!]$$

$$[\![\mathbf{1}]\!] \stackrel{\text{def}}{=} \mathbb{1} \qquad [\![\gamma_1 + \gamma_2]\!] \stackrel{\text{def}}{=} [\![\gamma_1]\!] \times [\![\gamma_2]\!]$$

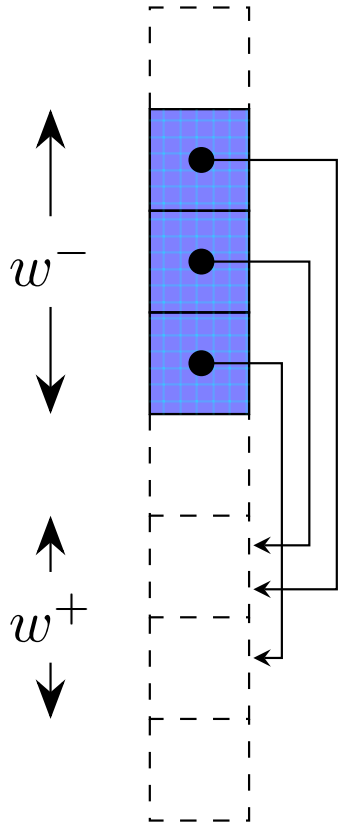
$$[\![\mathbf{ref}\ c]\!]\, w \stackrel{\text{def}}{=} \{\ell \in w | w(\ell) = c\} \qquad [\![\mathbf{ref}\ c]\!]\, \rho(\ell) \stackrel{\text{def}}{=} \rho(\ell)$$

Stores

Store fragments

Define $\underline{\mathbb{S}} : \mathbb{W}^{\mathrm{op}} \times \mathbb{W} \to \mathbf{Set}$:

$$\underline{\mathbb{S}}(w^-, w^+) \stackrel{\mathsf{def}}{=} \prod_{(\ell:c)\in w^-} [\![\mathbf{type}\ c]\!]\, w^+$$



Stores

$$\mathbb{S}w \stackrel{\mathsf{def}}{=} \underline{\mathbb{S}}(w, w)$$

Combining store fragments

Independent coproducts [Simpson, unpublished]

Given $\# : \mathbb{N} \xrightarrow{\cong} \mathbb{L}$, define:

$$\#w \stackrel{\text{def}}{=} \min\{n \in \mathbb{N} | \forall i \geq n. \# i \notin w\}$$

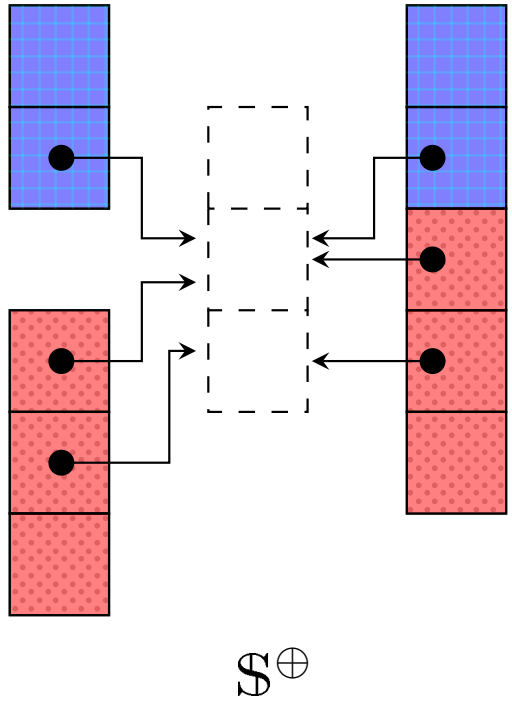
$$\underline{w_1 \oplus w_2} \stackrel{\text{def}}{=} \underline{w}_1 \cup \{\#(\#w + n) | \# n \in w_2\}$$

$$\iota_i^{\oplus} : \; w_i \to \underline{w_1 \oplus w_2}$$

$$\iota_1^{\oplus} : \; \ell_1 \mapsto \ell_1$$

$$\iota_2^{\oplus} : \# n \mapsto \#(\#w + n)$$

$$(w_1 \oplus w_2)(\iota_i^{\oplus} \ell) \stackrel{\text{def}}{=} w_i(\ell)$$



$\mathbb{S}^{\oplus}$

and we have: $w_1 \xrightarrow{\iota_1^{\oplus}} w_1 \oplus w_2 \xleftarrow{\iota_2^{\oplus}} w_2$ in $\mathbb{W}$

Canonical isomorphisms

$$\mathbb{S}^{\oplus} : \underline{\mathbb{S}}(w_1, w) \times \underline{\mathbb{S}}(w_2, w) \xrightarrow{\cong} \underline{\mathbb{S}}(w_1 \oplus w_2, w) \qquad \mathbb{S}^{\emptyset} : \mathbb{1} \xrightarrow{\cong} \underline{\mathbb{S}}(\emptyset, w)$$

Initialisations

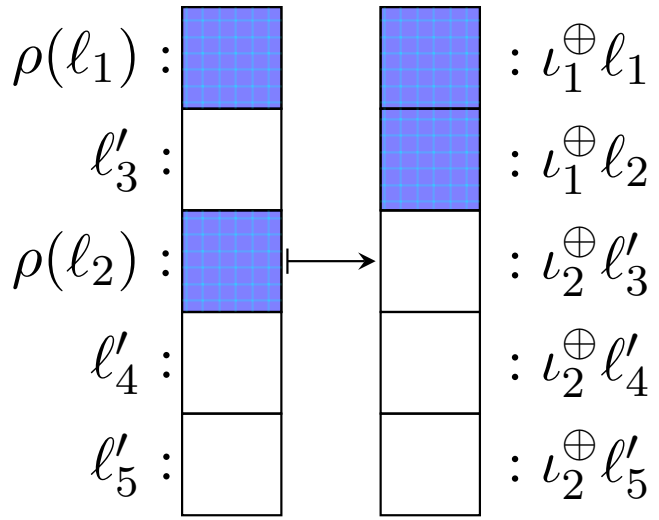
Complements

Given $\rho : w_1 \to w_2$, define:

$$\underline{w_2 \ominus \rho} \stackrel{\text{def}}{=} \underline{w_2} \setminus \mathrm{Im}\,(\rho)$$

$$w_2 \ominus \rho(\ell) \stackrel{\text{def}}{=} w_2(\ell)$$

$$\rho^{\complement} : w_2 \ominus \rho \to w_2$$
$$\ell \mapsto \ell$$

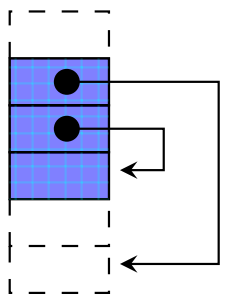


$$\underline{\mathbb{S}}(w_2, w) \xrightarrow{\underline{\mathbb{S}}(\cong, w)} \underline{\mathbb{S}}(w_1 \oplus (w_2 \ominus \rho), w)$$

The category $\mathbb{E}$ of initialisations

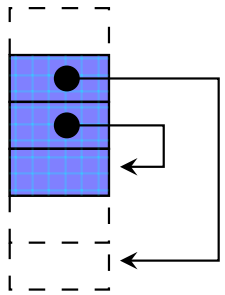
Objects: layouts w

Morphisms $\varepsilon : w \to w'$: injections $u\varepsilon : w \to w'$ and **initialisation data** $\sigma_\varepsilon \in \underline{\mathbb{S}}(w' \ominus u\varepsilon, w')$



Initialisations

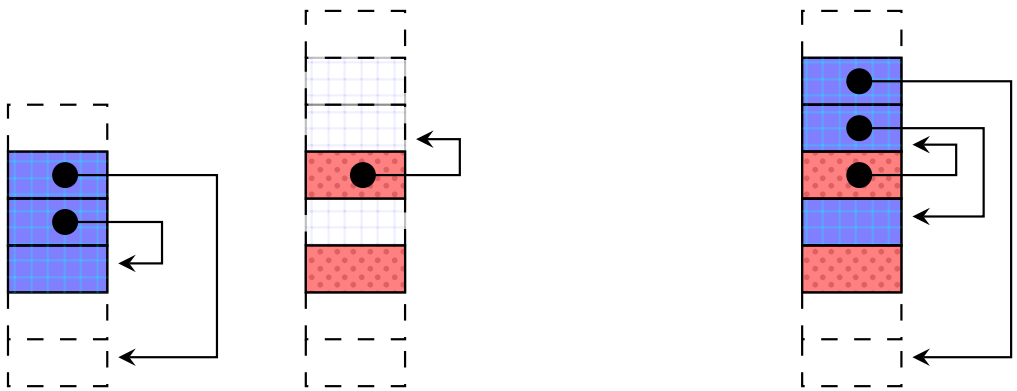
The category $\mathbb{E}$ of initialisations



Objects: layouts w

Morphisms $\varepsilon : w \to w'$: injections $u\varepsilon : w \to w'$ and **initialisation data** $\sigma_\varepsilon \in \underline{\mathbb{S}}(w' \ominus u\varepsilon, w')$

Composition



σ_{ε_1} σ_{ε_2} $\sigma_{\varepsilon_2 \circ \varepsilon_1}$

Functoriality of stores

$$\mathbb{S}w = \underline{\mathbb{S}}(w, w) \cong \mathbb{E}(\emptyset, w)$$

so $\mathbb{S}$ is a representable functor in $\mathbf{C} \stackrel{\text{def}}{=} [\mathbb{E}, \mathbf{Set}]$

Talk structure

- Full ground storage
- Worlds and initialisations
- A monad for hiding/encapsulation
- The monad, explicitly
- Evaluation
- Conclusion

Aspiration

Computational motivation alongside technical development

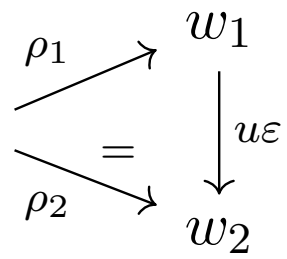
The hiding monad

Comma category

For the forgetful $u : \mathbb{E} \to \mathbb{W}$, the comma $w \downarrow u$ is:

Objects: $\mathbb{W}$-morphisms $\rho : w \to w'$

Morphisms $\left(\begin{smallmatrix} w \\ \rho_1 \downarrow \\ w_1 \end{smallmatrix} \right) \xrightarrow{\varepsilon} \left(\begin{smallmatrix} w \\ \rho_2 \downarrow \\ w_2 \end{smallmatrix} \right)$: Initialisations $w_1' \xrightarrow{\varepsilon} w_2'$ s.t.



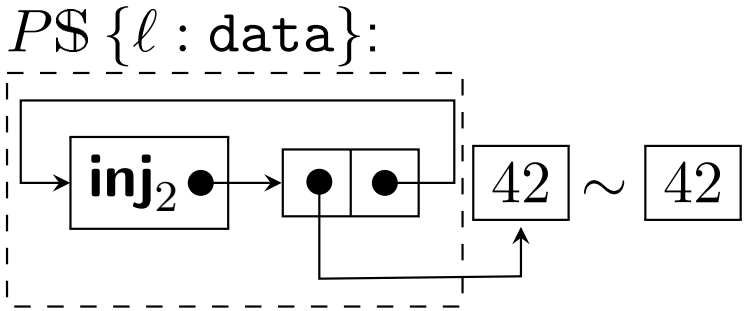
Object map

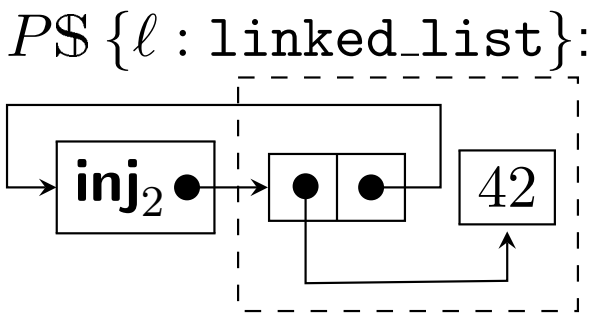
Define for $A \in \mathbf{C} = [\mathbb{E}, \mathbf{Set}]$:

$$PAw \stackrel{\text{def}}{=} \int^{w \to w' \in w \downarrow u} A$$

$$\stackrel{\text{def}}{=} \left(\sum_{w \to w' \in w \downarrow u} Aw' \right) / \sim$$

In $[\rho : w \to w', x] \in PAw$ the locations in $w' \ominus \rho$ are private to x





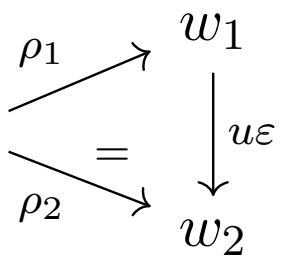
The hiding monad

Comma category

For the forgetful $u : \mathbb{E} \to \mathbb{W}$, the comma $w \downarrow u$ is:

Objects: $\mathbb{W}$-morphisms $\rho : w \to w'$

Morphisms $\left(\begin{smallmatrix} w \\ \rho_1 \downarrow \\ w_1 \end{smallmatrix} \right) \xrightarrow{\varepsilon} \left(\begin{smallmatrix} w \\ \rho_2 \downarrow \\ w_2 \end{smallmatrix} \right)$: Initialisations $w_1' \xrightarrow{\varepsilon} w_2'$ s.t.



Object map

Define for $A \in \mathbf{C} = [\mathbb{E}, \mathbf{Set}]$:

$$PAw \stackrel{\text{def}}{=} \int^{w \to w' \in w \downarrow u} A$$

$$\stackrel{\text{def}}{=} \Big(\sum_{w \to w' \in w \downarrow u} Aw' \Big) / \sim$$

In $[\rho : w \to w', x] \in PAw$ the locations in $w' \ominus \rho$ are private to x

Subtleties [standard]

- $PAw = \mathrm{Colim}_{w \to w' \in w \downarrow u} Aw'$ convenient coend notation
- The injections $\rho : w \to w'$ define the quotient

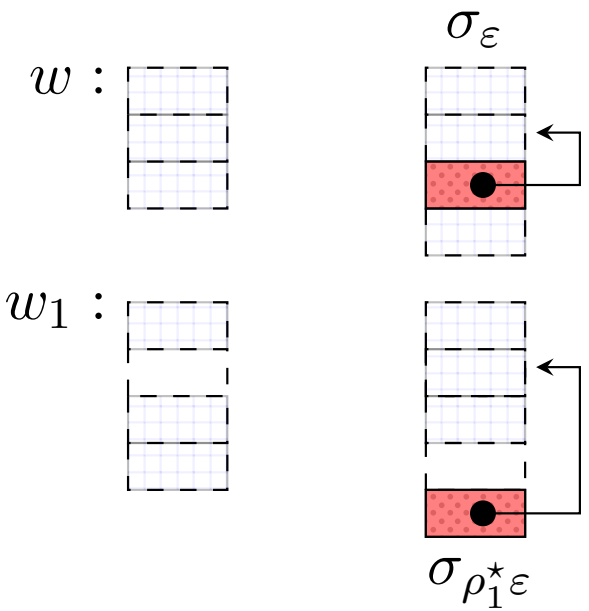
The initialisation $\rho^\star\varepsilon$

Local independent coproducts

Given $w_1 \xleftarrow{\rho_1} w \xrightarrow{\rho_2} w_2$, define:

$$\rho_1 \oplus_w \rho_2 := w \oplus (w_1 \ominus \rho_1) \oplus (w_2 \ominus \rho_2)$$

$$\begin{array}{ccc} w & \xrightarrow{\rho_2} & w_2 \\ {\scriptstyle \rho_1}\downarrow & = & \downarrow{\scriptstyle \rho_2^\star\rho_1} \\ w_1 & \xrightarrow[\rho_1^\star\rho_2]{} & \rho_1 \oplus_w \rho_2 \end{array}$$



when $\rho_2 = u\varepsilon$ there is an initialisation $\rho_1^\star\varepsilon : w_1 \to \rho_1 \oplus_w \rho_2$ with $u\rho_1^\star\varepsilon = \rho_1^\star u\varepsilon$.

Semantic counterpart of

$$\begin{array}{ccc} w & \leq & w \oplus \{\ell_1 : c_1, \ldots, \ell_n : c_n\} \\ |\wedge & & |\wedge \\ w' & \leq & w' \oplus \{\ell_1 : c_1, \ldots, \ell_n : c_n\} \end{array}$$

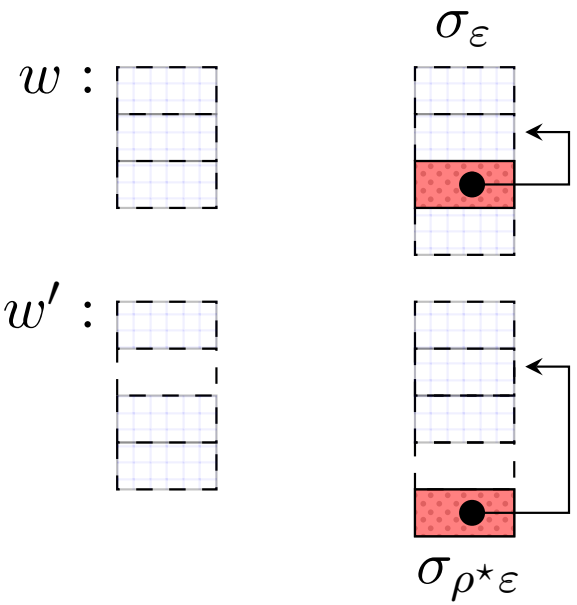
The hiding monad

Object map

Define for $A \in \mathbf{C} = [\mathbb{E}, \mathbf{Set}]$:

$$PAw \stackrel{\text{def}}{=} \int^{w \to w' \in w\downarrow u} A$$

$$\stackrel{\text{def}}{=} \Big(\sum_{w \to w' \in w\downarrow u} Aw' \Big) / \sim$$



Functorial action

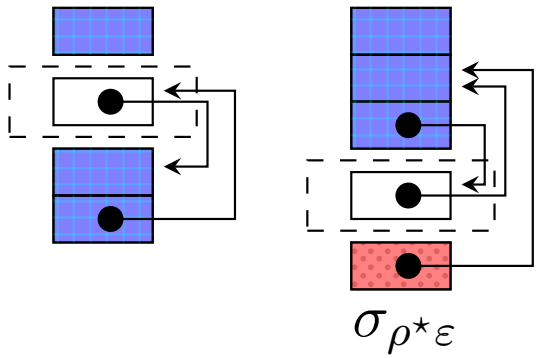
For $\varepsilon : w_1 \to w_2$ in $\mathbb{E}$:

$$PA\varepsilon : \quad PAw_1 \to PAw_2$$

$$[\rho : w_1 \to w', a] \mapsto [\varepsilon^\star\rho, A(\rho^\star\varepsilon)(a)]$$

where

$$\begin{array}{ccc} w_1 & \xrightarrow{\quad\varepsilon\quad} & w_2 \\ {\scriptstyle\rho}\downarrow & = & \downarrow{\scriptstyle\varepsilon^\star\rho} \\ w' & \xrightarrow[\rho^\star\varepsilon]{} & \rho \oplus_w u\varepsilon \end{array}$$



$$P\$w_1 \xrightarrow{P\$\varepsilon} P\$w_2$$

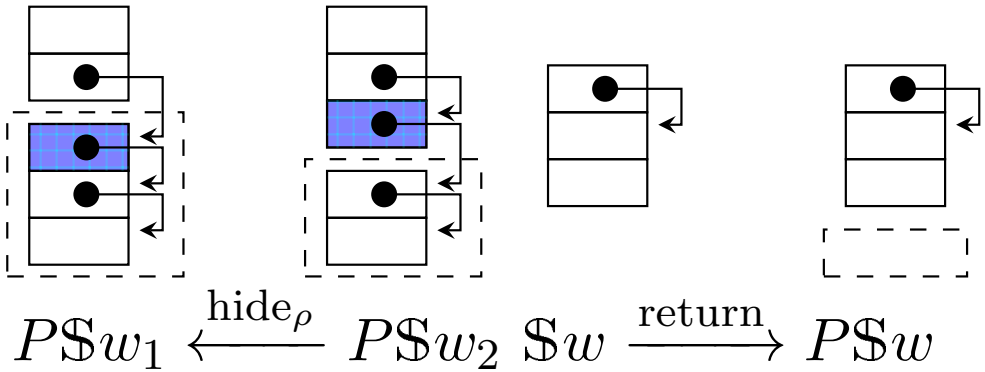
The hiding monad

Object map

Define for $A \in \mathbf{C} = [\mathbb{E}, \mathbf{Set}]$:

$$PAw \stackrel{\text{def}}{=} \int^{w \to w' \in w\downarrow u} A$$

$$\stackrel{\text{def}}{=} \left(\sum_{w \to w' \in w\downarrow u} Aw' \right) / \sim$$



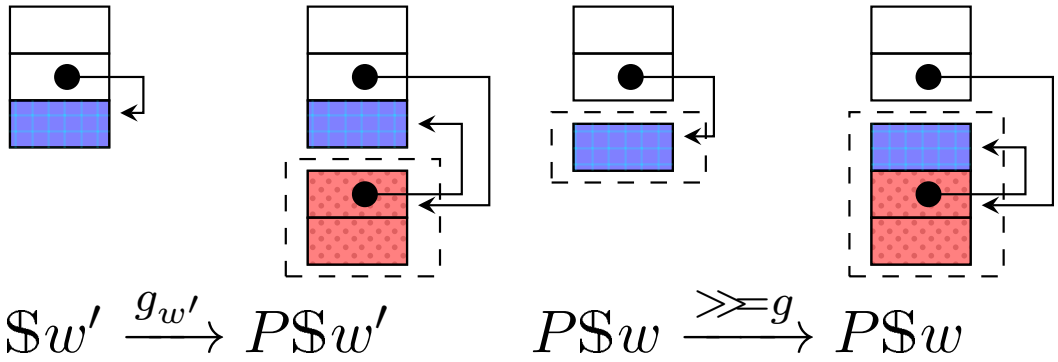
Hiding

For $\rho : w_1 \to w_2$ we have

$\text{hide}_\rho : PAw_1 \leftarrow PAw_2$

Return

$\text{return}_w a \stackrel{\text{def}}{=} [\text{id}_w, a]$



Bind

For $g : A \to PB$, define:

$([\rho : w \to w', a] \gg\!= g) \stackrel{\text{def}}{=} \text{hide}_\rho(g_{w'}(a))$

A state transformer

Enrichment

$\mathbf{C} = [\mathbb{E}, \mathbf{Set}]$ is enriched over $\mathbf{V} = [\mathbb{W}, \mathbf{Set}]$ with tensors:

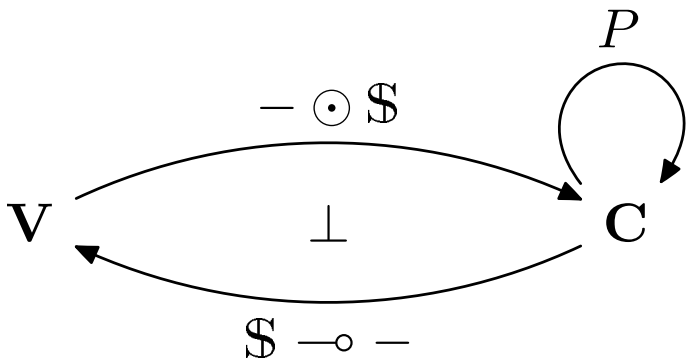
$$X \odot A \stackrel{\text{def}}{=} (X \circ u) \times A$$

$$(A \multimap B)w \stackrel{\text{def}}{=} \int_{w \to w' \in w\downarrow u} Aw' \Rightarrow (Bw')$$

(directly/as a $\mathbf{V}$-actegory [Gordon-Power'97, Janelidze-Kelly'01])

A monad for full ground storage

Take $T \stackrel{\text{def}}{=} \mathbb{S} \multimap P(- \odot \mathbb{S})$



Talk structure

- Full ground storage
- Worlds and initialisations
- A monad for hiding/encapsulation
- The monad, explicitly
- Evaluation
- Conclusion

Aspiration

Computational motivation alongside technical development

The full ground storage monad

Outline

For every $X \in \mathbf{V} = [\mathbb{W}, \mathbf{Set}]$:

$TXw \subseteq \prod_{w \to w' \in \mathbb{W}} \mathbb{S}w' \Rightarrow \left(\sum_{w' \to w'' \in \mathbb{W}} Xw'' \times \mathbb{S}w''\right) / \sim$

The monad

$$(TX)w = \int_{w \to w' \in w \downarrow u} \mathbb{S}w' \Rightarrow \left(\int^{w' \to w'' \in w \downarrow u} X \circ uw'' \times \mathbb{S}w'' \right)$$

Subtleties

- $\int^{w' \to w'' \in w \downarrow u} X \circ uw'' \times \mathbb{S}w''$ is contravariant in w', but the end is taken w.r.t. the covariant action of P
- The functorial action $TX\rho$ is given via the end

Analogous to $T_{\mathbf{ground}} X \rho$ in the ground case [cf. Plotkin-Power'02]:

$$T_{\mathbf{ground}} Xw = \mathbb{S}_{\mathbf{ground}} w \Rightarrow \int^{w \to w' \in w/\mathbb{W}} X \times \mathbb{S}_{\mathbf{ground}}$$

The full ground storage monad

Outline

For every $X \in \mathbf{V} = [\mathbb{W}, \mathbf{Set}]$:

$$TXw \subseteq \prod_{w \to w' \in \mathbb{W}} \$w' \Rightarrow \left(\sum_{w' \to w'' \in \mathbb{W}} Xw'' \times \$w''\right) / \sim$$

The monad

$$(TX)w = \int_{w \to w' \in w\downarrow u} \$w' \Rightarrow \left(\int^{w' \to w'' \in w\downarrow u} \quad X \circ uw'' \times \$w''\right)$$

Return

$$(\pi_{\rho : w \to w'} \circ \operatorname{return} x)(\sigma) = [\mathrm{id}_{w'}, (X\rho)x, \sigma]$$

The full ground storage monad

Outline

For every $X \in \mathbf{V} = [\mathbb{W}, \mathbf{Set}]$:

$$TXw \subseteq \prod_{w \to w' \in \mathbb{W}} \$w' \Rightarrow \left(\sum_{w' \to w'' \in \mathbb{W}} Xw'' \times \$w''\right) / \sim$$

The monad

$$(TX)w = \int_{w \to w' \in w\downarrow u} \$w' \Rightarrow \left(\int^{w' \to w'' \in w\downarrow u} \quad X \circ uw'' \times \$w''\right)$$

Bind

Given $f : X \to TY$ in $\mathbf{V}$ and $\alpha \in TX$,

$$(\pi_{\rho:w \to w'} \alpha \gg\!= f)(\sigma') = [\rho'' \circ \rho', y, \sigma'''']$$

where

$$(\pi_\rho \alpha)\sigma' = [\rho' : w' \to w'', x, \sigma'']$$
$$(\pi_{\mathrm{id}_{w''}} \circ f_{w''}(x))(\sigma'') = [\rho'' : w'' \to w''', y, \sigma''']$$

The full ground storage monad

Outline

For every $X \in \mathbf{V} = [\mathbb{W}, \mathbf{Set}]$:

$$TXw \subseteq \prod_{w \to w' \in \mathbb{W}} \$w' \Rightarrow \left(\sum_{w' \to w'' \in \mathbb{W}} Xw'' \times \$w''\right) / \sim$$

The monad

$$(TX)w = \int_{w \to w' \in w\downarrow u} \$w' \Rightarrow \left(\int^{w' \to w'' \in w\downarrow u} X \circ uw'' \times \$w''\right)$$

State manipulation

$$\begin{aligned}
\mathrm{get}_c &: [\![\mathbf{ref}\ c]\!] \to T\,[\![\mathbf{type}\ c]\!] \\
&\quad (\pi_\rho \circ \mathrm{get}_c(\ell))(\sigma) \quad = [\mathrm{id}_{w'}, \sigma(\rho(\ell)), \sigma] \\
\mathrm{set}_c &: [\![\mathbf{ref}\ c]\!] \times [\![\mathbf{type}\ c]\!] \to T\mathbb{1} \\
&\quad (\pi_\rho \circ \mathrm{set}_c(\ell, a))(\sigma) = [\mathrm{id}_{w'}, \star, \sigma[\rho(\ell) \mapsto [\![\mathbf{type}\ c]\!]\,\rho(a)]
\end{aligned}$$

where $\sigma(\ell) \stackrel{\text{def}}{=} \pi_\ell \sigma$, and

$$\sigma[\ell \mapsto x](\ell') = \begin{cases} x & \ell' = \ell \\ \sigma(\ell') & \text{otherwise} \end{cases}$$

The full ground storage monad

Outline

For every $X \in \mathbf{V} = [\mathbb{W}, \mathbf{Set}]$:

$$TXw \subseteq \prod_{w \to w' \in \mathbb{W}} \mathbb{S}w' \Rightarrow \left(\sum_{w' \to w'' \in \mathbb{W}} Xw'' \times \mathbb{S}w''\right) / \sim$$

The monad

$$(TX)w = \int_{w \to w' \in w \downarrow u} \mathbb{S}w' \Rightarrow \left(\int^{w' \to w'' \in w \downarrow u} \quad X \circ uw'' \times \mathbb{S}w''\right)$$

Allocation

Using

$$\partial_{w_0} : \mathbf{V} \to \mathbf{V} \qquad \mathrm{init}_{w_0} : \prod_{(\ell:c) \in w_0} \partial_{w_0} [\![\mathbf{type}\ c]\!] \to \mathbb{E}(w, w \oplus w_0)$$
$$X \mapsto X(- \oplus w_0)$$

define

$$\mathrm{new}_{w_0} : \prod_{(\ell:c) \in w_0} \partial_{w_0} [\![\mathbf{type}\ c]\!] \to T \prod_{(\ell:c) \in w_0} [\![\mathbf{ref}\ c]\!]$$
$$(\pi_\rho \circ \mathrm{new}_{w_0}(\langle a_\ell \rangle)\sigma = [\varepsilon^\star \rho, (\langle \varepsilon^\star \rho(\ell) \rangle_{\ell \in w_0}, \mathbb{S}(\rho^\star \varepsilon)(\sigma)]$$

where $\varepsilon \stackrel{\mathsf{def}}{=} \mathrm{init}\, \langle a_\ell \rangle$.

Talk structure

- Full ground storage
- Worlds and initialisations
- A monad for hiding/encapsulation
- The monad, explicitly
- Evaluation
- Conclusion

Aspiration

Computational motivation alongside technical development

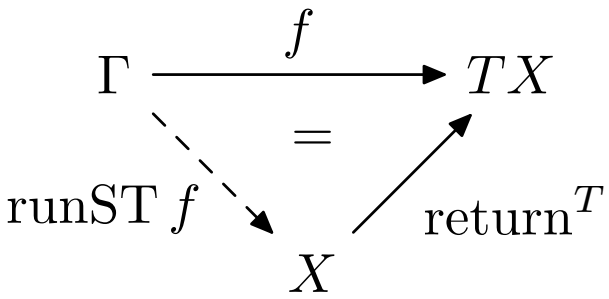
Garbage collection

Constant functors

Functors $X \in \mathbf{V}$ whose action $X\rho$ is a bijection.

Theorem (effect masking)

For every pair of constant functors $\Gamma, X \in \mathbf{V}$, every morphism $f : \Gamma \to TX$ factors uniquely through the monadic unit:



Interprets a multi-monadic-metalanguage with a **runST** construct [Launchbury-Peyton Jones'94]

Garbage collection

Proof.

$$\begin{array}{rl}
\mathbb{1} \longrightarrow \mathbb{\$} \multimap P(X \odot \mathbb{\$}) & \text{in } \mathbf{V} \\
\hline\hline
\mathbb{\$} \longrightarrow P(X \odot \mathbb{\$}) & \text{in } \mathbf{C} \\
\hline\hline
\mathbb{\$} \longrightarrow X \odot P\mathbb{\$} & \text{by lemma} \\
\hline\hline
\mathbb{E}(\underline{0}, -) \longrightarrow X \odot P\mathbb{\$} & \text{in } \mathbf{C} \\
\hline\hline
\mathbb{1} \longrightarrow (X \odot P\mathbb{\$})\underline{0} & \text{in } \mathbf{Set}\text{, by Yoneda} \\
\hline\hline
\mathbb{1} \longrightarrow X\underline{0} & \text{by lemma} \\
\hline\hline
\mathbb{1} \longrightarrow X & \text{in } \mathbf{V}
\end{array}$$

and chase a generic morphism upwards. □

A call-by-value calculus: λ_{ref} (denotational semantics)

With a standard denotational semantics we have:

Theorem (adequacy)

For all terms $\Gamma \vdash_w t_1, t_2 : \tau$, *if* $[\![t_1]\!] = [\![t_2]\!]$ *then* $\Gamma \vdash_w t_1 \simeq_{\mathrm{ctx}} t_2 : \tau$.

Program transformations

The semantics validates the Hilbert-Post complete program equations for local ground state [Staton'10], which can be divided into three groups [*loc.cit.*, Melliés'14]:

- Global state equations [Plotkin-Power'02]
- Block algebra equations (allocation only)
- Local state equations (state-allocation interaction)

Summary

Conclusions

- A monad for full ground storage
- Heaps/stores functoriality over initialisations
- Decomposition into state-transformed encapsulation monad
- Evaluation: effect masking theorem and local ground state equations

Further work

- Haskell's **runST** and effect masking
- A denotational account of the value restriction [Wright'95]
- General storage
- Algebraic presentations, Hilbert-Post completeness, handlers
- Independence structures
- Combining effects
- Further analysis
- Garbage collectors and mallocators
- Comparison with ground case
- Nominal accounts