

# 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>

Theoretische Informatik Seminar

Department of Computer Science

Faculty of Engineering

Friedrich-Alexander-Universität Erlangen-Nürnberg

25 June 2018



UNIVERSITY OF  
OXFORD

EPSRC  
Engineering and Physical Sciences  
Research Council



THE ROYAL  
SOCIETY



UNIVERSITY OF  
CAMBRIDGE



UNIVERSITY OF  
BIRMINGHAM

# Dynamic allocation of memory cells

## Kinds of storage

**type data** = **bool**

ground

**type linked\_list** = **1 + ref list\_cell**

full ground:  
refs to refs  
cyclic data

**type list\_cell** = **ref data \* ref linked\_list**

general:  
suspended computation  
effectful functions

**type process** = **1 + (1 → ref process)**

This talk:

full ground storage only

# Local state semantics

	full	ground	general	
operational	✓	✓	✓	syntactic, ubiquitous
relational	✓	✓	✓	dangling pointers, e.g.: Benton, Bohr, Hofmann, Birkedal et al. '06
step indexing	✓	✓	✓	syntactic or relational Ahmed '04 onwards      Birkedal et al. '10 onwards
game	✓	✓	✓	CBN-ish, partial
parametricity	✓	✓	✓	dangling pointers e.g. Reddy-Yang '04

possible-world

✓  
This work

Levy'02

Reynolds and Oles'82, Moggi'90, O'Hearn and  
Tennent'92, Stark'94, Ghica'97, Plotkin-Power'02

- ▶ Sets-with-structure and structure preserving functions
- ▶ 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

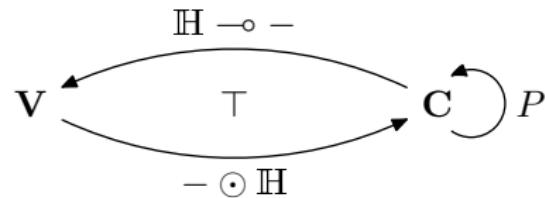
## Evaluation

- ▶ Effect masking
- ▶ Adequacy
- ▶ Local ground storage equations

# A state transformer

A monad for full ground storage

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



# Talk structure

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

# Full ground storage

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

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

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

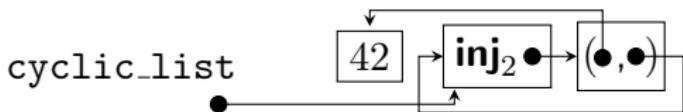
$$\mathbf{n} \stackrel{\text{def}}{=} \{\text{data}, \text{linked\_list}, \text{list\_cell}\}$$

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

# Simultaneous allocation

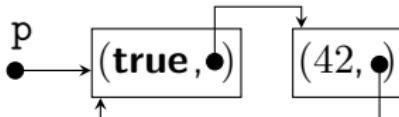
- ▶ Examples:

```
letref (payload      : ref data      ) := 42,  
       (cyclic_list: ref linked_list) := inj2head,  
       (head      : ref list_cell  ) := (payload, cyclic_list)  
in cyclic_list
```



- ▶ `letref (p: ref bit_cell) := (true, q),  
 (q: ref int_cell) := (42, p),`  
in p

`type bit_cell = bool * ref int_cell  
type int_cell = int * ref bit_cell`



non-sequentialisable

# A call-by-value calculus: $\lambda_{\text{ref}}$ (syntax)

$\tau ::=$	types	$t, s ::=$	terms
$\mathbf{ref}\ c$	reference	$\ell$	location ( $\ell \in \mathbb{L}$ )
$\mathbf{0}$	empty	$x$	identifier
$\tau_1 + \tau_2$	binary sum	$\mathbf{inj}_i^{\tau_1 + \tau_2} t$	sum constructor
$\mathbf{1}$	unit	$()$	unit
$\tau_1 * \tau_2$	product	$(t, s)$	pair
$\tau_1 \rightarrow \tau_2$	function	$\lambda x : \tau. t$	function abstraction
$v ::=$	values	$\mathbf{match}\ t\ \mathbf{with}\ \{\}\^\tau$	pattern matching
$\ell$	location ( $\ell \in \mathbb{L}$ )	$\mathbf{match}\ t\ \mathbf{with}$	
$x$	identifier	$\{\mathbf{inj}_1 x_1 \mapsto s_1$	
$\mathbf{inj}_i^{\tau_1 + \tau_2} v$	constructor	$  \mathbf{inj}_2 x_2 \mapsto s_2\}$	
$()$	unit	$\mathbf{prj}_i t$	projections
$(v_1, v_2)$	pair	$t\ s$	function application
$\lambda x : \tau. t$	function	$t := s$	assignment
		$!t$	dereferencing
		$\mathbf{letref}$	allocation
		$(x_1 : \mathbf{ref}\ c_1) := v_1,$	
		$\vdots$	
		$(x_n : \mathbf{ref}\ c_n) := v_n$	
		$\mathbf{in}\ t$	

## Sugar

**let**  $(x : \tau) = t$  **in**  $s \equiv (\lambda x : \tau. s)$

**new** $_c t \equiv$

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

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

# A call-by-value calculus: $\lambda_{\text{ref}}$ (type system)

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

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

Heap layouts:  $w = \{\ell_1 : c_1, \dots, \ell_n : c_n\}$   
 $\underline{w} \stackrel{\text{def}}{=} \{\ell_1, \dots, \ell_n\}$

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

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

$$\frac{\Gamma \vdash_w t : \mathbf{ref} c}{\Gamma \vdash_w !t : \mathbf{type} c}$$

for  $i = 1, \dots, n$ :  $\Gamma, x_1 : \mathbf{ref} c_1, \dots, x_n : \mathbf{ref} c_n \vdash_w v_i : \mathbf{type} c_i$

$$\frac{\Gamma, x_1 : \mathbf{ref} c_1, \dots, x_n : \mathbf{ref} c_n \vdash_w t : \tau}{\Gamma \vdash_w \mathbf{letref}(x_1 : \mathbf{ref} c_1) := v_1, \dots, (x_n : \mathbf{ref} c_n) := v_n \mathbf{in} t : \tau}$$

⋮

$(x_n : \mathbf{ref} c_n) := v_n \mathbf{in} t : \tau$

# A call-by-value calculus: $\lambda_{\text{ref}}$ (operational semantics)

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

$$\overline{\langle \ell, \eta \rangle \Downarrow \langle \ell, \eta \rangle} \quad \langle t, \eta \rangle \Downarrow \langle \lambda x : \tau. t', \eta_1 \rangle \quad \langle s, \eta_1 \rangle \Downarrow \langle v', \eta_2 \rangle$$

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

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

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

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

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

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

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

$$\frac{\langle t, \eta \rangle \Downarrow \langle \mathbf{inj}_i^{\tau_1 + \tau_2} \hat{v}, \hat{\eta} \rangle \quad \langle s_i[x_i \mapsto \hat{v}], \hat{\eta} \rangle \Downarrow \langle v, \eta' \rangle}{\langle \mathbf{matcht} \text{ with } \begin{cases} \mathbf{inj}_1 x_1 \mapsto s_1 \\ | \mathbf{inj}_2 x_2 \mapsto s_2 \end{cases}, \eta \rangle \Downarrow \langle v, \eta' \rangle}$$

$$\frac{\langle t[\theta], \eta \left[ \ell_i \mapsto v_i[\theta] \right]_{i=1}^n \rangle \Downarrow \langle v, \eta' \rangle \quad \text{where } \theta \stackrel{\text{def}}{=} [x_i \mapsto \ell_i]_{i=1}^n}{\text{letref } (\#_{\underline{w}_\eta} \langle \ell_1, \dots, \ell_n \rangle)}$$

$$\frac{\langle \begin{array}{c} (x_1 : \mathbf{ref} c_1) := v_1, \\ \vdots \\ (x_n : \mathbf{ref} c_n) := v_n \end{array}, \eta \rangle \Downarrow \langle v, \eta' \rangle}{\langle \mathbf{prj}_i t, \eta \rangle \Downarrow \langle v_i, \eta' \rangle}$$

$$(x_n : \mathbf{ref} c_n) := v_n \mathbf{in} t$$

# A call-by-value calculus: $\lambda_{\text{ref}}$ (meta-theory)

## Functionality of types

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

$$\begin{aligned}\tau^v w &\subseteq \tau^v w' & \text{and} && \tau^t w &\subseteq \tau^t w' & \text{where} \\ \tau^v w &\stackrel{\text{def}}{=} \{v \mid \vdash_w v : \tau\} & \text{and} && \tau^t w &\stackrel{\text{def}}{=} \{t \mid \vdash_w t : \tau\}\end{aligned}$$

## Non-functionality of heaps

For  $\mathbf{H}w := \{\eta \mid \underline{w}_\eta = \underline{w}, \forall (\ell : c) \in w. \vdash_w \eta(\ell) : \mathbf{type} c\}$  and  $w \leq w'$  we **lack** functions

$$\mathbf{H}w \rightarrow \mathbf{H}w' \quad \text{and} \quad \mathbf{H}w \leftarrow \mathbf{H}w'$$

# A call-by-value calculus: $\lambda_{\text{ref}}$ (meta-theory)

## Non-functoriality of heaps

For  $\mathbf{H}w := \{\eta \mid \underline{w}_\eta = \underline{w}, \forall (\ell : c) \in w. \vdash_w \eta(\ell) : \mathbf{type} c\}$  and  $w \leq w'$  we **lack** functions

$$\mathbf{H}w \rightarrow \mathbf{H}w' \quad \text{and} \quad \mathbf{H}w \leftarrow \mathbf{H}w'$$

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

$$\langle \rangle \in \mathbf{H}\emptyset \mapsto ? \in \mathbf{H} \{ \ell : \text{ptr} \}$$
$$? \in \mathbf{H} \{ \ell_1 : \text{ptr} \} \leftarrow \begin{cases} \ell_1 \mapsto \ell_2, \\ \ell_2 \mapsto \ell_1 \end{cases} \in \mathbf{H} \{ \ell_1, \ell_2 : \text{ptr} \}$$

for **type**  $\text{ptr} = \text{ref}$   $\text{ptr}$ .

# A call-by-value calculus: $\lambda_{\text{ref}}$ (meta-theory)

## Theorem (totality)

*All well-typed closed programs fully evaluate:*

$$\forall \vdash_w t : \tau, w_1 \geq w, \eta_1 \in \mathbf{H}w_1.$$

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

## Proof highlights

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

**Allocation case:** initialisation data  $\vdash_{w \oplus \{\ell_1:c_1, \dots, \ell_n:c_n\}} v_1, \dots, v_n$  needs to extend  $\eta \in \mathbf{H}w_1$ ,  $w_1 \geq w$ :

$$\begin{array}{c} w \leq w \oplus \{\ell_1 : c_1, \dots, \ell_n : c_n\} \\ | \wedge \quad \quad \quad | \wedge \\ w_1 \leq w_1 \oplus \{\ell_1 : c_1, \dots, \ell_n : c_n\} \end{array}$$

# Talk structure

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

# Worlds

The category  $\mathbb{W}$

Objects: layouts  $w$

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

Interpreting full ground types

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

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

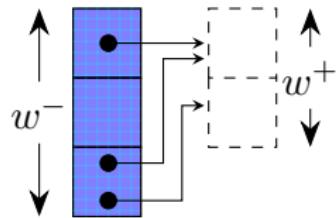
$$[\![\mathbf{1}]\!] \stackrel{\text{def}}{=} \mathbb{1} \quad [\![\gamma_1 + \gamma_2]\!] \stackrel{\text{def}}{=} [\![\gamma_1]\!] \times [\![\gamma_2]\!]$$

$$[\![\mathbf{ref}\; c]\!] w \stackrel{\text{def}}{=} \{\ell \in w \mid w(\ell) = c\} \quad [\![\mathbf{ref}\; c]\!] \rho(\ell) \stackrel{\text{def}}{=} \rho(\ell)$$

# Semantic heaps

## Heaplets

Define  $\underline{\mathbb{H}} : \mathbb{W}^{\text{op}} \times \mathbb{W} \rightarrow \mathbf{Set}$ :



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

## Heaps

$$\mathbb{H}w \stackrel{\text{def}}{=} \underline{\mathbb{H}}(w, w)$$

↗  
mixed variance

# Combining store fragments

Independent coproducts [Simpson'17]

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

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

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

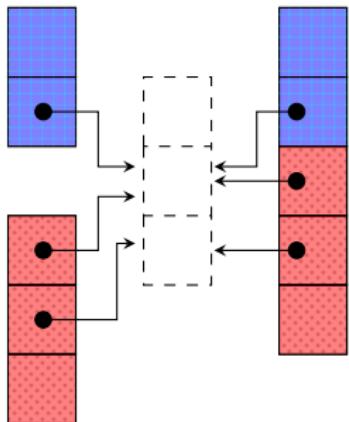
$$\iota_i^\oplus : \underline{w}_i \rightarrow \underline{w}_1 \oplus \underline{w}_2$$

$$\iota_1^\oplus : \ell \mapsto \ell$$

$$\iota_2^\oplus : \#n \mapsto \#(\#w_1 + n)$$

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

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



$\mathbb{H}^\oplus$

Canonical isomorphisms

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

# Initialisations

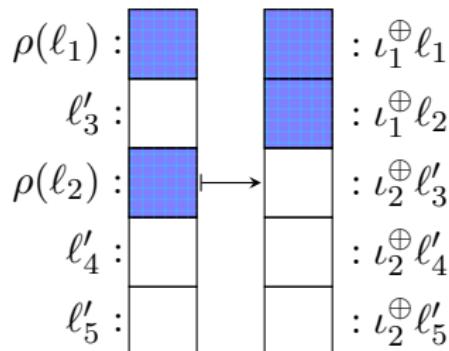
## Complements

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

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

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

$$\begin{aligned}\rho^C : w_2 \ominus \rho &\rightarrow w_2 \\ \ell &\mapsto \ell\end{aligned}$$

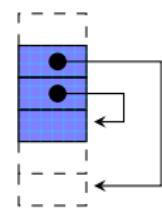


$$\underline{\mathbb{H}}(w_2, w) \xrightarrow{\underline{\mathbb{H}}(\cong, w)} \underline{\mathbb{H}}(w_1 \oplus (w_2 \ominus \rho), w)$$

## The category $\mathbb{E}$ of initialisations

Objects: layouts  $w$

Morphisms  $\varepsilon : w \rightarrow w'$ : injections  $u\varepsilon : w \rightarrow w'$  and  
**initialisation data**  $\eta_\varepsilon \in \underline{\mathbb{H}}(w' \ominus u\varepsilon, w')$

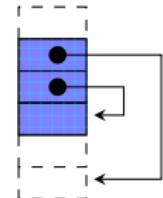


# Initialisations

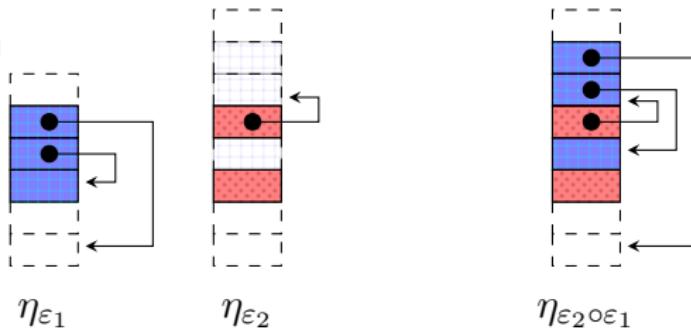
## The category $\mathbb{E}$ of initialisations

Objects: layouts  $w$

Morphisms  $\varepsilon : w \rightarrow w'$ : injections  $u\varepsilon : w \rightarrow w'$  and  
**initialisation data**  $\eta_\varepsilon \in \underline{\mathbb{H}}(w' \ominus u\varepsilon, w')$



## Composition



## Functionality of heaps

representable functor  $\mathbb{H}w = \underline{\mathbb{H}}(w, w) \cong \mathbb{E}(\emptyset, w)$   
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

# The hiding monad

## Comma category

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

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

Morphisms  $(\begin{smallmatrix} w \\ \rho_1 \downarrow \\ w_1 \end{smallmatrix}) \xrightarrow{\varepsilon} (\begin{smallmatrix} w \\ \rho_2 \downarrow \\ w_2 \end{smallmatrix})$ : Initialisations  $w'_1 \xrightarrow{\varepsilon} w'_2$  s.t.  $w \begin{array}{c} \nearrow \rho_1 \\ = \\ \searrow \rho_2 \end{array} \begin{array}{c} w_1 \\ \downarrow u\varepsilon \\ w_2 \end{array}$

## Object map

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

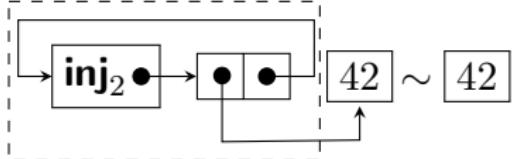
$$PAw \stackrel{\text{def}}{=} \int^{w \rightarrow w' \in w \downarrow u} A$$

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

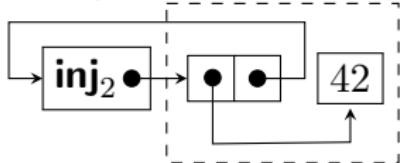
In  $[\rho : w \rightarrow w', x] \in PAw$  the locations in  $w' \ominus \rho$  are private to  $x$

$$w \begin{array}{c} \nearrow \rho_1 \\ = \\ \searrow \rho_2 \end{array} \begin{array}{c} w_1 \\ \downarrow u\varepsilon \\ w_2 \end{array}$$

$P\mathbb{H}\{\ell : \text{data}\}$ :



$P\mathbb{H}\{\ell : \text{linked\_list}\}$ :



# The hiding monad

## Comma category

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

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

Morphisms  $(\begin{smallmatrix} w \\ \rho_1 \downarrow \\ w_1 \end{smallmatrix}) \xrightarrow{\varepsilon} (\begin{smallmatrix} w \\ \rho_2 \downarrow \\ w_2 \end{smallmatrix})$ : Initialisations  $w'_1 \xrightarrow{\varepsilon} w'_2$  s.t.

Object map

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

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

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

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

$$\begin{array}{ccc} & & w_1 \\ & \nearrow \rho_1 & \downarrow u\varepsilon \\ w & = & \searrow \rho_2 \\ & & w_2 \end{array}$$

## Subtleties [standard]

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

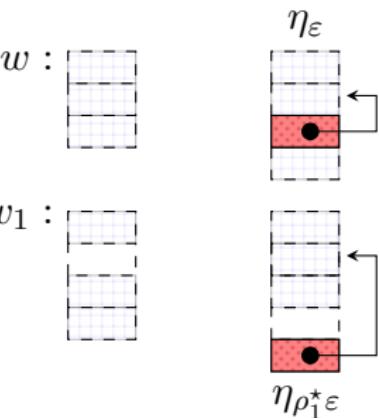
# The initialisation $\rho^\star\varepsilon$

## Local independent coproducts

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

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

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



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

Semantic counterpart of

$$w \leq w \oplus \{\ell_1 : c_1, \dots, \ell_n : c_n\}$$

$$|\wedge \hspace{10em} |\wedge$$

$$w' \leq w' \oplus \{\ell_1 : c_1, \dots, \ell_n : c_n\}$$

# The hiding monad

## Object map

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

$$\begin{aligned} PAw &\stackrel{\text{def}}{=} \int^{w \rightarrow w' \in w \downarrow u} A \\ &\stackrel{\text{def}}{=} \left( \sum_{w \rightarrow w' \in w \downarrow u} Aw' \right) / \sim \end{aligned}$$

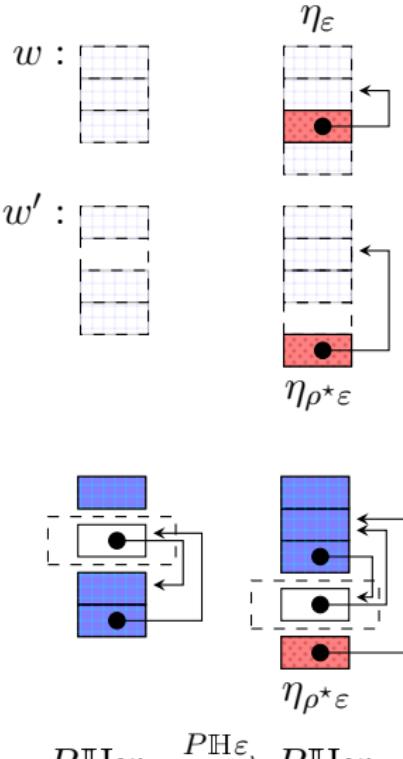
## Functorial action

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

$$PA\varepsilon : \quad PAw_1 \rightarrow \quad PAw_2$$
$$[\rho : w_1 \rightarrow w', a] \mapsto [\varepsilon^*\rho, A(\rho^*\varepsilon)(a)]$$

where

$$\begin{array}{ccc} w_1 & \xrightarrow{\varepsilon} & w_2 \\ \rho \downarrow & = & \downarrow \varepsilon^*\rho \\ w' & \xrightarrow{\rho^*\varepsilon} & \rho \oplus_w u\varepsilon \end{array}$$



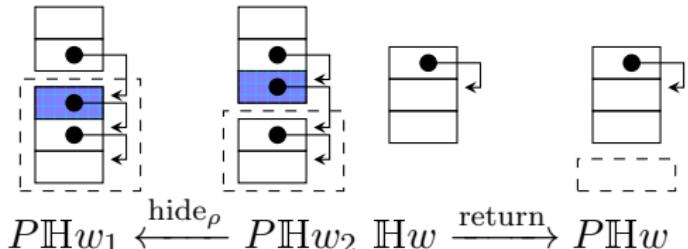
$$P\mathbb{H}w_1 \xrightarrow{P\mathbb{H}\varepsilon} P\mathbb{H}w_2$$

# The hiding monad

## Object map

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

$$\begin{aligned} PAw &\stackrel{\text{def}}{=} \int^{w \rightarrow w' \in w \downarrow u} A \\ &\stackrel{\text{def}}{=} \left( \sum_{w \rightarrow w' \in w \downarrow u} Aw' \right) / \sim \end{aligned}$$



## Hiding

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

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

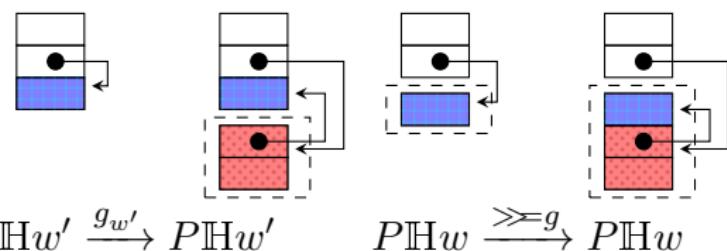
## Return

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

## Bind

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

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



# A state transformer

## Enrichment

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

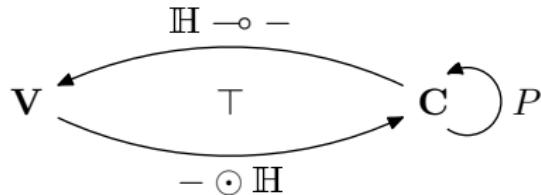
$$X \odot A \stackrel{\text{def}}{=} (X \circ u) \times A$$

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

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

## A monad for full ground storage

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



# Talk structure

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

# The full ground storage monad

## Outline

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

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

## The monad

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

## Subtleties

- ▶  $\int^{w' \rightarrow w'' \in w' \downarrow u} X \circ uw'' \times \mathbb{H}w''$  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  $T_{\text{ground}}X\rho$  in the ground case [cf. Plotkin-Power'02]:

$$T_{\text{ground}}Xw = \mathbb{H}_{\text{ground}}w \Rightarrow \int^{w \rightarrow w' \in w/\mathbb{W}} X \times \mathbb{H}_{\text{ground}}$$

# The full ground storage monad

## Outline

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

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

## The monad

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

## Return

$$(\pi_{\rho:w \rightarrow w'} \circ \text{return } x)(\eta) = [\text{id}_{w'}, (X\rho)x, \eta]$$

# The full ground storage monad

## Outline

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

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

## The monad

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

## Bind

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

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

where

$$\text{cf. } \frac{\langle t, \eta' \rangle \Downarrow \langle a, \eta'' \rangle \quad \langle s[x \mapsto a], \eta'' \rangle \Downarrow \langle b, \eta''' \rangle}{\langle \text{let } x = t \text{ in } s, \eta' \rangle \Downarrow \langle b, \eta''' \rangle}$$

$$(\pi_\rho \alpha)\eta' = [\rho' : w' \rightarrow w'', x, \eta'']$$

$$(\pi_{\text{id}_{w''}} \circ f_{w''}(x))(\eta'') = [\rho'' : w'' \rightarrow w''', y, \eta''']$$

# The full ground storage monad

## Outline

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

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

## The monad

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

## State manipulation

$$\text{get}_c : [\text{ref } c] \rightarrow T[\text{type } c]$$

$$(\pi_\rho \circ \text{get}_c(\ell))(\eta) = [\text{id}_{w'}, \eta(\rho(\ell)), \eta]$$

$$\text{set}_c : [\text{ref } c] \times [\text{type } c] \rightarrow T\mathbb{1}$$

$$(\pi_\rho \circ \text{set}_c(\ell, a))(\eta) = [\text{id}_{w'}, \star, \eta[\rho(\ell) \mapsto [\text{type } c] \rho(a)]]$$

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

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

# The full ground storage monad

## Outline

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

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

## The monad

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

## Allocation

### Using

$$\begin{aligned} \partial_{w_0} : \mathbf{V} &\rightarrow \mathbf{V} & \text{init}_{w_0} : \prod_{(\ell:c) \in w_0} \partial_{w_0} [\mathbf{type } c] &\rightarrow \mathbb{E}(w, w \oplus w_0) \\ X &\mapsto X(- \oplus w_0) \end{aligned}$$

### define

$$\begin{aligned} \text{new}_{w_0} : \prod_{(\ell:c) \in w_0} \partial_{w_0} [\mathbf{type } c] &\rightarrow T \prod_{(\ell:c) \in w_0} [\mathbf{ref } c] \\ (\pi_\rho \circ \text{new}_{w_0}(\langle a_\ell \rangle))\eta &= [\varepsilon^\star \rho, (\langle \varepsilon^\star \rho(\ell) \rangle_{\ell \in w_0}, \mathbb{H}(\rho^\star \varepsilon)(\eta))] \end{aligned}$$

where  $\varepsilon \stackrel{\text{def}}{=} \text{init } \langle a_\ell \rangle$ .

# Talk structure

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

# 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  $f : \Gamma \rightarrow TX$  factors uniquely through the monadic unit:

$$\begin{array}{ccc} \Gamma & \xrightarrow{f} & TX \\ & \searrow \text{runST } f & \swarrow = \\ & X & \end{array}$$

$\nearrow \text{return}^T$

A commutative diagram showing the factorization of a morphism  $f$  from  $\Gamma$  to  $TX$ . The top arrow is  $f$ . The bottom arrow is  $X$ . A dashed diagonal arrow labeled "runST  $f$ " points from  $\Gamma$  down to  $X$ . A solid diagonal arrow labeled "=" points from  $X$  up to  $TX$ . A solid arrow labeled "return $^T$ " points from  $X$  up to  $TX$ .

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

# Garbage collection

Proof.

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

and chase a generic morphism upwards. □

# A call-by-value calculus: $\lambda_{\text{ref}}$ (denotational semantics)

With a standard denotational semantics we have:

## Theorem (adequacy)

For all terms  $\Gamma \vdash_w t_1, t_2 : \tau$ , if  $\llbracket t_1 \rrbracket = \llbracket t_2 \rrbracket$  then  $\Gamma \vdash_w t_1 \simeq_{\text{ctx}} t_2 : \tau$ .

## 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