Arrays with Garbage

J. Power[1] O. Shkaravska[2]

[1]University of Edinburgh

[2]Institute of Cybernetics at TUT

Teooriapäevad Viinistul, 2005

Outline

1. Motivation
   - Arrays: a Memory Model for Computations with Side Effects
   - Previous Works
2. Our Results
   - Arrays: a comodel for Global State
   - The Category of Arrays is equivalent to Set
   - The Category of Arrays is comonadic over Set

Outline

1. Motivation
   - Arrays: a Memory Model for Computations with Side Effects
   - Previous Works
2. Our Results
   - Arrays: a comodel for Global State
   - The Category of Arrays is equivalent to Set
   - The Category of Arrays is comonadic over Set

Pure Langauges

Pure functional languages do not subsume:

- variable assignments $x := 2$,
- field updates $x.tail := another\_list$

The example stolen from G. Plotkin's talk

```
function   Sq(x : int) : int
return     x * x
end
```

Meaning of Sq

$[\![int]\!] = \mathbb{N}$
$[\![Sq]\!] = \mathbb{N} \to \mathbb{N}$

Pure Langauges

Pure functional languages do not subsume:

- variable assignments $x := 2$,
- field updates $x.tail := another\_list$

The example stolen from G. Plotkin's talk

$$\begin{array}{ll} \texttt{function} & Sq(x : int) : int \\ \texttt{return} & x * x \\ \texttt{end} & \end{array}$$

Meaning of Sq

$$[\![int]\!] = \mathbb{N}$$
$$[\![Sq]\!] = \mathbb{N} \to \quad \mathbb{N}$$

Pure Langauges

Pure functional languages do not subsume:

- variable assignments $x := 2$,
- field updates $x.tail := another\_list$

The example stolen from G. Plotkin's talk

```
function   Sq(x : int) : int
return     x * x
end
```

Meaning of Sq

$[\![int]\!] = \mathbb{N}$
$[\![Sq]\!] = \mathbb{N} \to \mathbb{N}$

Impure Language = Pure Language + Side Effects

The next example stolen from G. Plotkin's talk

function $Sq(x : int) : int$
$y := 3$
return $x * x$
end

Meaning of Sq II

$[\![Sq]\!] = \mathbb{N} \times S \to \mathbb{N} \times S$
where $S = \mathbb{N}^{Loc}$ i.e. an ARRAY

Equivalently
$[\![Sq]\!] = \mathbb{N} \to T_{state}(\mathbb{N})$, is an arrow in Kleisli cat.,
where $T_{state}(\mathbb{N}) = (\mathbb{N} \times S)^S$

Impure Language = Pure Language + Side Effects

The next example stolen from G. Plotkin's talk

function $Sq(x : int) : int$
$y := 3$
return $x * x$
end

Meaning of Sq II

$[\![Sq]\!] = \mathbb{N} \times S \rightarrow \mathbb{N} \times S$
where $S = \mathbb{N}^{Loc}$ i.e. an ARRAY

Equivalently
$[\![Sq]\!] = \mathbb{N} \rightarrow T_{state}(\mathbb{N})$, is an arrow in Kleisli cat.,
where $T_{state}(\mathbb{N}) = (\mathbb{N} \times S)^S$

Impure Language = Pure Language + Side Effects

The next example stolen from G. Plotkin's talk

```
function   Sq(x : int) : int
y := 3
return     x * x
end
```

Meaning of Sq II

$$[\![Sq]\!] = \mathbb{N} \times S \rightarrow \mathbb{N} \times S$$
where $S = \mathbb{N}^{Loc}$ i.e. an ARRAY

Equivalently

$$[\![Sq]\!] = \mathbb{N} \rightarrow T_{state}(\mathbb{N}),$$ is an arrow in Kleisli cat.,
where $T_{state}(\mathbb{N}) = (\mathbb{N} \times S)^S$

Outline

Sketches

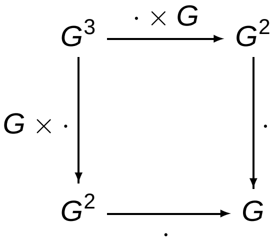
Example: associativity for groups.

For any G and $a,\ b,\ c\ \in\ G$ it holds $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

Sketches

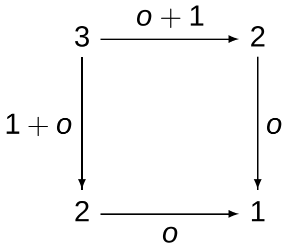
Example: associativity for groups.

For any G and $a,\ b,\ c\ \in\ G$ it holds $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.



Sketches

Associativity holds for any group G.
Remove concrete G from the diagram – to get its *sketch*.



Sketches

Lawvere Theory

- A category L with f.p.
- id. on objects strict f.p.p. $J : Nat^{op} \longrightarrow L$

New arrows in L

$$2 \longrightarrow^{o} 1$$
$$1 \longrightarrow^{-1} 1$$
$$0 \longrightarrow^{e} 1$$

$$\begin{array}{ccc} 3 & \xrightarrow{o+1} & 2 \\ {\scriptstyle 1+o}\downarrow & & \downarrow{\scriptstyle o} \\ 2 & \xrightarrow[o]{} & 1 \end{array}$$

Sketches

Lawvere Theory

- A category L with f.p.
- id. on objects strict f.p.p. $J : Nat^{op} \to L$

New arrows in L

$$2 \longrightarrow^{o} 1$$
$$1 \longrightarrow^{-1} 1$$
$$0 \longrightarrow^{e} 1$$

$$\begin{array}{ccc} 3 & \xrightarrow{o+1} & 2 \\ {\scriptstyle 1+o}\downarrow & & \downarrow{\scriptstyle o} \\ 2 & \xrightarrow[o]{} & 1 \end{array}$$

Model of L

- a f.p.p. functor $M : L \to C$,
- such functors form a category $Mod(L, C)$, n.t. as arrows.

$C :=$ **Set**

$M(1) := G$

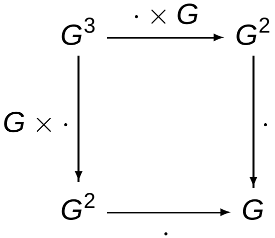
$n = 1 + \ldots + 1$, $M(n) := G^n$

$$\begin{array}{ccc} G^3 & \xrightarrow{\cdot \times G} & G^2 \\ {\scriptstyle G \times \cdot}\downarrow & & \downarrow{\scriptstyle \cdot} \\ G^2 & \xrightarrow[\cdot]{} & G \end{array}$$

Model of L

- a f.p.p. functor $M : L \to C$,
- such functors form a category $Mod(L, C)$, n.t. as arrows.

$C :=$ **Set**
$M(1) := G$
$n = 1 + \ldots + 1$, $M(n) := G^n$



Comodel of L

- a f.cp.p. functor $L^{op} \to C$,
- they form a category $Comod(L, C)$, n.t. as arrows

$$Comod(L, C) \cong Mod(L, C^{op})^{op}$$

Example: additional arrow $n \to 1$.
$M(1) := X$,
the map $X \to X + \ldots + X$ (n times),
i.e. $X \to n \times X$

Comodel of L

- a f.cp.p. functor $L^{op} \to C$,
- they form a category $Comod(L, C)$, n.t. as arrows

$$Comod(L, C) \cong Mod(L, C^{op})^{op}$$

Example: additional arrow $n \to 1$.
$M(1) := X$,
the map $X \to X + \ldots + X$ (n times),
i.e. $X \to n \times X$

Comodel of L

- a f.cp.p. functor $L^{op} \to C$,
- they form a category $Comod(L,\ C)$, n.t. as arrows

$$Comod(L,\ C) \cong Mod(L,\ C^{op})^{op}$$

Example: additional arrow $n \to 1$.
$M(1) := X$,
the map $X \to X + \ldots + X$ (n times),
i.e. $X \to n \times X$

Comodel of L

- a f.cp.p. functor $L^{op} \to C$,
- they form a category $Comod(L, C)$, n.t. as arrows

$$Comod(L, C) \cong Mod(L, C^{op})^{op}$$

Example: additional arrow $n \to 1$.
$M(1) := X$,
the map $X \to X + \ldots + X$ (n times),
i.e. $X \to n \times X$

Theory of Global State of P&P

The side-effect monad $(S \times (-))^S$, $S = V^{Loc}$, where
Loc is a finite set of locations,
V is a countable set of values.

$$l : V \longrightarrow Loc$$
$$u : 1 \longrightarrow Loc \times V$$

$$M(1) := A$$
$$lookup : A^V \longrightarrow A^{Loc}$$
$$update : A \longrightarrow A^{Loc \times V}$$

Theory of Global State of P&P

The side-effect monad $(S \times (-))^S$, $S = V^{Loc}$, where
Loc is a finite set of locations,
V is a countable set of values.

$$l : V \longrightarrow Loc$$
$$u : 1 \longrightarrow Loc \times V$$

$$M(1) := A$$
$$lookup : A^V \longrightarrow A^{Loc}$$
$$update : A \longrightarrow A^{Loc \times V}$$

Theory of Global State of P&P

The side-effect monad $(S \times (-))^S$, $S = V^{Loc}$, where
Loc is a finite set of locations,
V is a countable set of values.

$$l : V \longrightarrow Loc$$
$$u : 1 \longrightarrow Loc \times V$$

$$M(1) := A$$
$$lookup : A^V \longrightarrow A^{Loc}$$
$$update : A \longrightarrow A^{Loc \times V}$$

Theory of Global State of P&P

$$M(1) := A = (S \times X)^S$$
$$lookup : A^V \longrightarrow A^{Loc}$$
$$update : A \longrightarrow A^{Loc \times V}$$

$$\Big(lookup(t)\Big)(loc)(s) = \texttt{let } v := s(loc) \texttt{ in } t(v)(s)$$
$$\Big(update(t)\Big)(loc,\ v)(s) = t(s[loc := v])$$

Theory of Global State of P&P

$$M(1) := A = (S \times X)^S$$
$$lookup : A^V \longrightarrow A^{Loc}$$
$$update : A \longrightarrow A^{Loc \times V}$$

$$\Big(lookup(t)\Big)(loc)(s) = \texttt{let } v := s(loc) \texttt{ in } t(v)(s)$$
$$\Big(update(t)\Big)(loc,\ v)(s) = t(s[loc := v])$$

Outline

Comodel for Global State

$$l : V \longrightarrow Loc$$
$$u : 1 \longrightarrow Loc \times V$$

(Loc, V)-array is a set $M(1) = A$ together with functions

$$sel : A \times Loc \longrightarrow A \times V$$
$$upd : A \times Loc \times V \longrightarrow A$$

Intuitively, looking up a cell does not change the state:
$sel(a, loc) = (a, v)$.
For the sake of simplicity:

$$sel : A \times Loc \longrightarrow V$$
$$upd : A \times Loc \times V \longrightarrow A$$

Comodel for Global State

$$l : V \longrightarrow Loc$$
$$u : 1 \longrightarrow Loc \times V$$

(Loc, V)-array is a set $M(1) = A$ together with functions

$$sel : A \times Loc \longrightarrow A \times V$$
$$upd : A \times Loc \times V \longrightarrow A$$

Intuitively, looking up a cell does not change the state: $sel(a, loc) = (a, v)$.
For the sake of simplicity:

$$sel : A \times Loc \longrightarrow V$$
$$upd : A \times Loc \times V \longrightarrow A$$

Comodel for Global State

$$l : V \longrightarrow Loc$$
$$u : 1 \longrightarrow Loc \times V$$

(Loc, V)-array is a set $M(1) = A$ together with functions

$$sel : A \times Loc \longrightarrow A \times V$$
$$upd : A \times Loc \times V \longrightarrow A$$

Intuitively, looking up a cell does not change the state:
$sel(a,\ loc) = (a,\ v)$.
For the sake of simplicity:

$$sel : A \times Loc \longrightarrow V$$
$$upd : A \times Loc \times V \longrightarrow A$$

Comodel for Global State

$$l : V \longrightarrow Loc$$
$$u : 1 \longrightarrow Loc \times V$$

(Loc, V)-array is a set $M(1) = A$ together with functions

$$sel : A \times Loc \longrightarrow A \times V$$
$$upd : A \times Loc \times V \longrightarrow A$$

Intuitively, looking up a cell does not change the state:
$sel(a, loc) = (a, v)$.
For the sake of simplicity:

$$sel : A \times Loc \longrightarrow V$$
$$upd : A \times Loc \times V \longrightarrow A$$

Axiomatics of Arrays

$$sel(upd(a,\ loc,\ v),\ loc) = v$$

$$\begin{array}{ccc}
A \times Loc \times V & \longrightarrow & A \times Loc \times V \times Loc \\
\Big\downarrow \pi_V & & \Big\downarrow upd \times Loc \\
V & \underset{sel}{\longleftarrow} & A \times Loc
\end{array}$$

with $\delta_{Loc} : Loc \longrightarrow Loc \times Loc$

Axiomatics of Arrays

$$upd(a,\ loc,\ sel(a,\ loc)) = a$$

$$\begin{array}{ccc}
A \times Loc & \longrightarrow & A \times Loc \times A \times Loc \\
\Big\downarrow \pi_A & & \Big\downarrow A \times Loc \times sel \\
A & \underset{upd}{\longleftarrow} & A \times Loc \times V
\end{array}$$

with diagonal $\delta_{A \times Loc} : A \times Loc \longrightarrow A \times Loc \times A \times Loc$

Axiomatics of Arrays

$$upd(upd(a,\ loc,\ v),\ loc,\ v') = upd(a,\ loc,\ v')$$

$upd(upd(a,\ loc,\ v),\ loc',\ v') = upd(upd(a,\ loc',\ v'),\ loc,\ v)$, where $loc \neq loc'$

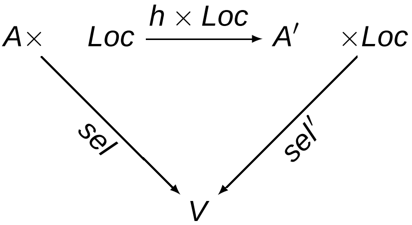
Axiomatics of Arrays

$$upd(upd(a, loc, v), loc, v') = upd(a, loc, v')$$

$upd(upd(a, loc, v), loc', v') = upd(upd(a, loc', v'), loc, v)$, where $loc \neq loc'$

Array Morphisms

A *map* of arrays from (A, sel, upd) to (A', sel', upd'):
$h : A \rightarrow A'$.



Array Morphisms

$$
\begin{array}{ccc}
A \times Loc \times V & \xrightarrow{h \times Loc \times V} & A' \times Loc \times V \\
\downarrow{\scriptstyle upd} & & \downarrow{\scriptstyle upd'} \\
A & \xrightarrow[h]{} & A'
\end{array}
$$

We have a category (Loc, V)-*Array*

Array Morphisms

$$\begin{array}{ccc} A \times Loc \times V & \xrightarrow{h \times Loc \times V} & A' \times Loc \times V \\ {\scriptstyle upd}\downarrow & & \downarrow{\scriptstyle upd'} \\ A & \xrightarrow[h]{} & A' \end{array}$$

We have a category (Loc, V)-*Array*

Outline

From **Set** to $(Loc, V) - Array$

The Functor

$$\mathbf{Set} \longrightarrow (Loc, V) - Array$$
$$R \mapsto V^{Loc} \times R$$

with the structure maps

$$sel\Big((v_1, \ldots, v_n, \ r), \ loc\Big) := v_{loc}$$
$$upd\big((v_1, \ldots, v_n, \ r), \ loc, \ v\big) := (v_1, \ldots, v_{loc-1}, v, v_{loc+1} \ldots, \ r)$$

is an equivalence of categories.

From **Set** to $(Loc, V) - Array$

The Functor

$$\begin{array}{l} \mathbf{Set} \longrightarrow (Loc, V) - Array \\ R \mapsto V^{Loc} \times R \end{array}$$

with the structure maps

$$sel\Big((v_1, \ldots, v_n, \ r), \ loc\Big) := v_{loc}$$
$$upd((v_1, \ldots, v_n, \ r), \ loc, \ v\Big) := (v_1, \ldots, v_{loc-1}, v, v_{loc+1} \ldots, \ r)$$

is an equivalence of categories.

From $(Loc, V) - Array$ to **Set**

The pseudoinverse:

$$\begin{array}{l}(Loc, V) - Array \longrightarrow \mathbf{Set} \\ (A,\ sel,\ upd) \mapsto R_A := A/\theta\end{array}$$

θ is the relation of the final reachability:
$a \approx_\theta b$ iff $b = upd_{l_k}\left(upd_{l_{k-1}}(\ldots a \ldots),\ v_{l_k}\right)$
Isomorphism $\varphi : (A,\ sel,\ upd) \rightarrow (V^{Loc} \times R_A,\ sel',\ upd')$?

$$\varphi :\ a\ \mapsto\ \left(sel_1(a),\ \ldots,\ sel_n(a),\ \ [a]_\theta\right)$$

From $(Loc, V) - Array$ to **Set**

The pseudoinverse:

$$\begin{array}{l} (Loc, V) - Array \longrightarrow \mathbf{Set} \\ (A,\ sel,\ upd) \mapsto R_A := A/\theta \end{array}$$

θ is the relation of the final reachability:

$a \approx_\theta b$ iff $b = upd_{l_k}\left(upd_{l_{k-1}}(\ldots a \ldots),\ v_{l_k}\right)$

Isomorphism $\varphi : (A,\ sel,\ upd) \rightarrow (V^{Loc} \times R_A,\ sel',\ upd')$?

$$\varphi :\ a\ \mapsto\ \left(sel_1(a),\ \ldots,\ sel_n(a),\ \ [a]_\theta\right)$$

From $(Loc, V) - Array$ to **Set**

The pseudoinverse:

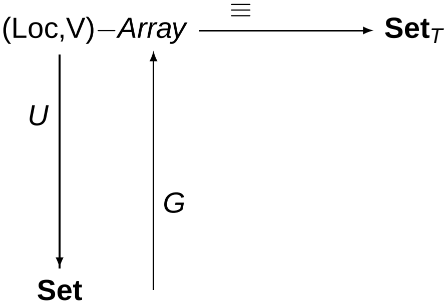
$$\begin{array}{l} (Loc, V) - Array \longrightarrow \mathbf{Set} \\ (A,\ sel,\ upd) \mapsto R_A := A/\theta \end{array}$$

θ is the relation of the final reachability:
$a \approx_\theta b$ iff $b = upd_{l_k}\left(upd_{l_{k-1}}(\ldots a \ldots),\ v_{l_k}\right)$
Isomorphism $\varphi : (A,\ sel,\ upd) \rightarrow (V^{Loc} \times R_A,\ sel',\ upd')$?

$$\varphi :\ a\ \mapsto\ \left(sel_1(a),\ \ldots,\ sel_n(a),\ \ [a]_\theta\right)$$

Outline

Comonadicity



$$G(X) = V^{Loc} \times X^{V^{Loc}}$$
$$T(-) := V^{Loc} \times (-)^{V^{Loc}}$$

Comonadicity

The Eilenberg-Moore comparison Functor

$\Phi : \ (A,\ sel,\ upd) \mapsto \ (A,\ U\eta_{(A,\, sel,\, upd)}),$

$\eta_{(A,\, sel,\, upd)} : (A,\ sel,\ upd) \longrightarrow (V^{Loc} \times A^{V^{Loc}},\ sel',\ upd')$

$U\eta_{(A,\, sel,\, upd)} : A \longrightarrow V^{Loc} \times A^{V^{Loc}}$

$\eta_{(A,\, sel,\, upd)} : a \mapsto \left(\overline{sel}(a),\ \overline{upd}(a,\ -)\right)$, where

$\overline{sel}(a) = \left(sel_1(a),\ \ldots,\ sel_n(a)\right)$

$\overline{upd}\left(a,\ (v_1,\ \ldots,\ v_n)\right) = upd_n\left(upd_{n-1}(\ldots a \ldots),\ v_n\right)$

Comonadicity

The Eilenberg-Moore comparison Functor

$\Phi : \ (A,\ sel,\ upd) \mapsto \ (A,\ U\eta_{(A,\ sel,\ upd)}),$

$\eta_{(A,\ sel,\ upd)} : (A,\ sel,\ upd) \to (V^{Loc} \times A^{V^{Loc}},\ sel',\ upd')$

$U\eta_{(A,\ sel,\ upd)} : A \to V^{Loc} \times A^{V^{Loc}}$

$\eta_{(A,\ sel,\ upd)} : a \mapsto \left(\overline{sel}(a),\ \overline{upd}(a,\ -)\right)$, where

$\overline{sel}(a) = \left(sel_1(a),\ \ldots,\ sel_n(a)\right)$

$\overline{upd}\left(a,\ (v_1,\ \ldots,\ v_n)\right) = upd_n\left(upd_{n-1}(\ldots a \ldots),\ v_n\right)$

Infinite Arrays

Aim: to model **new** operation

P&P approach – a **block** for **new** – might look a bit artificial.

To model C-like allocation/deallocation use

- arrays with countable *Loc*,
- $\perp$ to present a fresh cell.

Alternatives

- Use $(\perp + V)^{\omega}$ as a "canonic model",
  note: $(0\ 1\ \ldots \perp\ \ldots\ 1\ 1\ 1\ \ldots)$ is possible,
- Use $(\perp + V)^{*}\perp^{\omega}$ as a "canonic model".

Infinite Arrays

Aim: to model **new** operation

P&P approach – a **block** for **new** – might look a bit artificial.
To model C-like allocation/deallocation use

- arrays with countable *Loc*,
- $\perp$ to present a fresh cell.

Alternatives

- Use $(\perp + V)^\omega$ as a "canonic model",
  note: $(0\ 1\ \ldots \perp\ \ldots\ 1\ 1\ 1\ \ldots)$ is possible,
- Use $(\perp + V)^* \perp^\omega$ as a "canonic model".

Infinite Arrays

Aim: to model **new** operation

P&P approach – a **block** for **new** – might look a bit artificial.
To model C-like allocation/deallocation use

- arrays with countable *Loc*,
- $\perp$ to present a fresh cell.

Alternatives

- Use $(\perp + V)^\omega$ as a "canonic model",
  note: $(0\ 1\ \ldots \perp\ \ldots\ 1\ 1\ 1\ \ldots)$ is possible,
- Use $(\perp + V)^* \perp^\omega$ as a "canonic model".

Infinite Arrays

Aim: to model **new** operation

P&P approach – a **block** for **new** – might look a bit artificial.
To model C-like allocation/deallocation use

- arrays with countable *Loc*,
- $\perp$ to present a fresh cell.

Alternatives

- Use $(\perp + V)^\omega$ as a "canonic model",
  note: $(0\ 1\ \ldots \perp\ \ldots\ 1\ 1\ 1\ \ldots)$ is possible,
- Use $(\perp + V)^* \perp^\omega$ as a "canonic model".

Infinite Arrays

Aim: to model **new** operation

P&P approach – a **block** for **new** – might look a bit artificial.
To model C-like allocation/deallocation use

- arrays with countable *Loc*,
- $\perp$ to present a fresh cell.

Alternatives

- Use $(\perp + V)^\omega$ as a "canonic model",
  note: $(0\ 1\ \ldots \perp\ \ldots\ 1\ 1\ 1\ \ldots)$ is possible,
- Use $(\perp + V)^* \perp^\omega$ as a "canonic model".

Infinite Arrays

Status of the research

- *Proven:* the (Loc, V) – *Arrays* with countable *Loc* is equivalent to $\mathbf{Set}/\mathcal{F}$,
  where $\mathcal{F}$ is the Frechet-filtered product,
- *Conjecture:* the (Loc, V) – *Arrays* is comonadic over $\mathbf{Set}/\mathcal{F}$, the comonad is boring.

Infinite Arrays

Status of the research

- *Proven:* the (Loc, V) – *Arrays* with countable *Loc* is equivalent to **Set**$/\mathcal{F}$,
  where $\mathcal{F}$ is the Frechet-filtered product,
- *Conjecture:* the (Loc, V) – *Arrays* is comonadic over **Set**$/\mathcal{F}$, the comonad is boring.

Infinite Arrays

Status of the research

- *Proven:* the $(Loc, V) - Arrays$ with countable *Loc* is equivalent to $\mathbf{Set}/\mathcal{F}$, where $\mathcal{F}$ is the Frechet-filtered product,
- *Conjecture:* the $(Loc, V) - Arrays$ is comonadic over $\mathbf{Set}/\mathcal{F}$, the comonad is boring.

Conclusions

- (Loc, V) – *Arrays* is a category of comodels for the theory of Global State.
- (Loc, V) – *Arrays* is equivalent to **Set** and comonadic over it.

- Future Work
  - Choose a good alternative for countable arrays to model the effect of **new**.
  - Find equational axiomatics for **new**.

Conclusions

- $(Loc, V) - Arrays$ is a category of comodels for the theory of Global State.
- $(Loc, V) - Arrays$ is equivalent to **Set** and comonadic over it.

- Future Work
  - Choose a good alternative for countable arrays to model the effect of **new**.
  - Find equational axiomatics for **new**.

Conclusions

- $(Loc, V) - Arrays$ is a category of comodels for the theory of Global State.
- $(Loc, V) - Arrays$ is equivalent to **Set** and comonadic over it.

- Future Work
  - Choose a good alternative for countable arrays to model the effect of **new**.
  - Find equational axiomatics for **new**.

Conclusions

- $(Loc, V) - Arrays$ is a category of comodels for the theory of Global State.
- $(Loc, V) - Arrays$ is equivalent to **Set** and comonadic over it.

- Future Work
  - Choose a good alternative for countable arrays to model the effect of **new**.
  - Find equational axiomatics for **new**.