Parametricity Semantics of Symmetric Noninterference

Hongseok Yang
University of Illinois at Urbana-Champaign
hyang@cs.uiuc.edu

July 18, 2000

Contents

Symmetric Noninterference

Interference

- interference: the execution of one program affects that of the other program.
- example:
  - $(x := x + 1)$ and $(\mathbf{print}\ x)$

Noninterference in Real Life

- some program optimizations presuppose the absence of certain forms of interference. Ex. reordering codes, moving codes out of a loop.
- noninterference predicate in programming logic
  - $x \# y \Longrightarrow \{y = 1\}x := 0\{y = 1\}$
- proper understanding of noninterference becomes the basis.

Asymmetric and Symmetric Interpretations of Noninterference

For a command c and an expression e,

- asymmetric interpretation $c \# e$: the value of e remains the same while c is being executed. [Reynolds, Tennent, O'Hearn]
- symmetric interpretation $c \perp e$: c and e access disjoint memory locations. [O'Hearn, Reddy]
- asymmetric interpretation is well understood but symmetric one is not.

Topic of the Talk

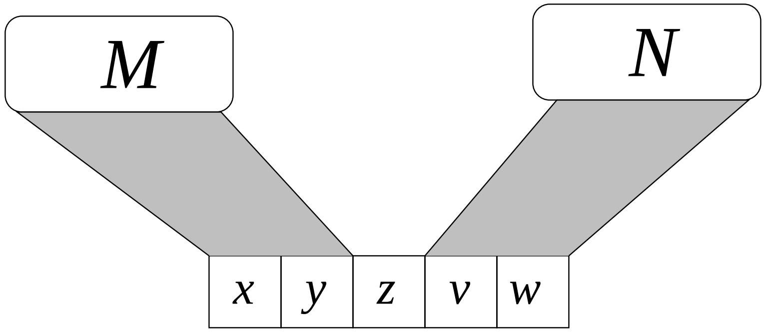
- answer the question: what does it mean that two program phrases do not interfere symmetrically?

- how?: especially using concepts in sheaf theory.

Problem: Semantics of Symmetric Noninterference

Intuition behind Symmetric Noninterference

Two program phrases M and N do not interfere symmetrically iff



The main question is how to capture this idea formally.

Principles of Symmetric Noninterference

- principles [Reynolds, O'Hearn, Reddy]:
  1. if $M_0 \perp N$ and $M_1 \perp N$, then $M_0(M_1) \perp N$.
  2. Let $FV(N) = \{x, y\}$. If $M \perp x$ and $M \perp y$, then $M \perp N$.
  3. for any nonlocal entity M and local variable x, $M \perp x$.
  4. If $M \perp N$, then $N \perp M$.

Problem

- problem: how to define symmetric noninterference on top of a "good" semantics of a programming language so that the definition satisfies the principles?

- existing work and our work:

	satisfy principles?	"good" semantics?
O'Hearn	Yes	No
Reddy	Not 2	Yes (parametricity semantics)
Our work	Yes	Yes (parametricity semantics)

Main Idea of Our Approach

- issue: how to formalize "accessed locations" for each program phrases well.

- main idea: to find a property which all program phrases have and to use this property to define "accessed locations".

- how to find such a property?
  - all program phrases including procedures should satisfy the property.
  - the property should help to define "accessed locations"
  - here comes "sheaf theory".

Main Idea of Our Approach: Sheaf Condition

- Sheaf condition captures locality:
  - Topological continuity: a function is countiuous from $\mathbf{R}$ to $\mathbf{R}$ if it is continuous on all open intervals $(i, i+2)$ for any integers i.
  - Use of resource: if a program only uses locations in $\{x, y\}$ and in $\{y, z\}$, then it only uses locations in $\{y\}$.

- Technical benefit: Given a property of program expressed by a "sheaf condition", if all commands, expressions and variables satisfy the property, then all procedures satisfy it, too. (CCC)

Idealized Algol

Idealized Algol

- Idealized Algol : call-by-name typed lambda calculus with imperative base types and imperative constants
- Syntax :
  - Types : $\theta := \mathbf{exp} \mid \mathbf{com} \mid \mathbf{var} \mid \theta_1 \times \theta_2 \mid \theta_1 \to \theta_2$
  - Preterm : $M := c \mid x \mid \lambda x\colon\theta.M \mid M_1 M_2 \mid (M_1,\, M_2) \mid \mathbf{fst}\, M \mid \mathbf{snd}\, M$
  - Imperative Constants :

$$\begin{array}{rcl}
\mathbf{skip} & : & \mathbf{com} \\
\mathbf{seq} & : & \mathbf{com} \times \mathbf{com} \to \mathbf{com} \\
\mathbf{get} & : & \mathbf{var} \to \mathbf{exp} \\
:= & : & \mathbf{var} \times \mathbf{exp} \to \mathbf{com} \\
\mathbf{new} & : & (\mathbf{var} \to \mathbf{com}) \to \mathbf{com}
\end{array}$$

Examples of Idealized Algol

- $x := x + 1; y := 1$:

$$\mathbf{seq}\ \big(:= (x,\ (\mathbf{get}\, x) + 1),\ := (y,\ 1)\big)$$

- swap function :

$$\begin{array}{l} \lambda x\colon \mathbf{var}.\ \lambda y\colon \mathbf{var}. \\ \quad \mathbf{new}\ \lambda t\colon \mathbf{var}. \\ \quad\quad (t := \mathbf{get}\ x); \\ \quad\quad (x := \mathbf{get}\ y); \\ \quad\quad (y := \mathbf{get}\ t) \end{array}$$

Objects

Objects with increase and read methods can be encoded as:

$$
\begin{array}{l}
\lambda p \colon \mathbf{com} \times \mathbf{exp} \to \mathbf{com}. \\
\quad \mathbf{new}\ \lambda x \colon \mathbf{var}. \\
\qquad x := 0;\ p\left(x := (\mathbf{get}\ x) + 1,\ \mathbf{get}\ x\right)
\end{array}
$$

Parametricity Semantics

Comparison of Semantics of Idealized Algol: Examples

- Independence of Local Variable: (Functor Category Semantics)

$$(\mathbf{new}\ \lambda x{:}\,\mathbf{var}.\ C) \cong C$$

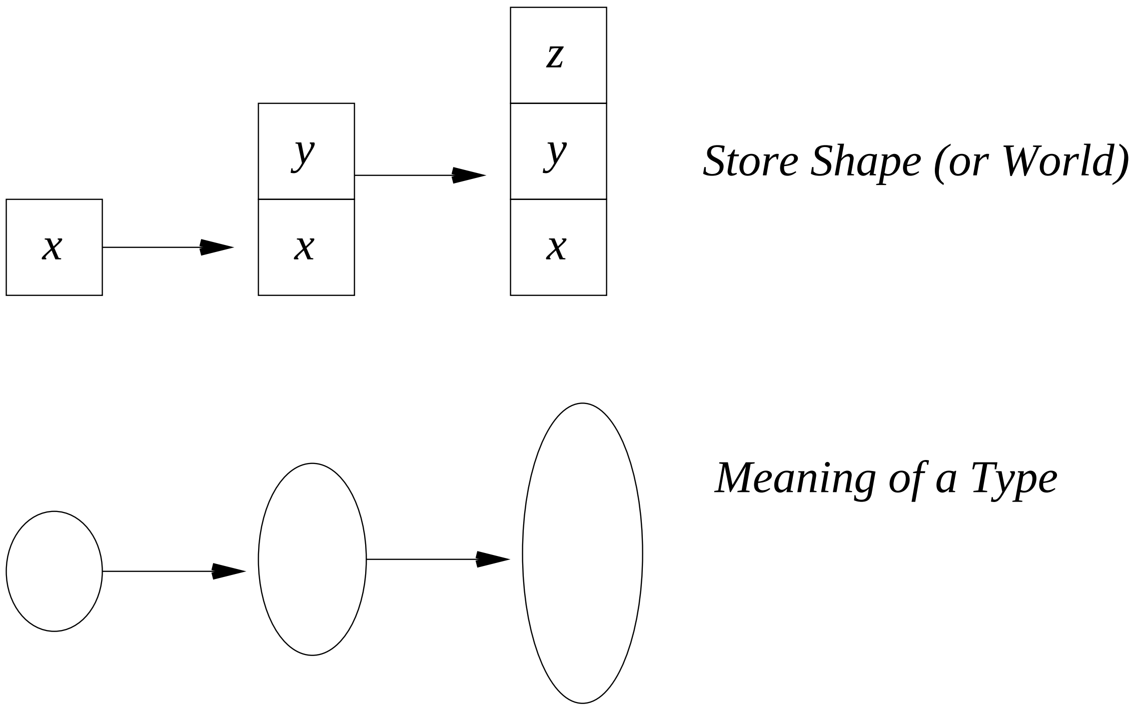
- Data Abstraction: (Parametricity Semantics)

$$\begin{array}{l} \lambda p{:}\,\mathbf{com} \times \mathbf{exp} \to \mathbf{com}. \\ \quad \mathbf{new}\ \lambda x{:}\,\mathbf{var}. \\ \quad\quad x := 0; \\ \quad\quad p\left(x := (\mathbf{get}\ x) + 1,\ \mathbf{get}\ x\right) \end{array} \cong \begin{array}{l} \lambda p{:}\,\mathbf{com} \times \mathbf{exp} \to \mathbf{com}. \\ \quad \mathbf{new}\ \lambda x{:}\,\mathbf{var}. \\ \quad\quad x := 0; \\ \quad\quad p\left(x := (\mathbf{get}\ x) - 1,\ -(\mathbf{get}\ x)\right) \end{array}$$

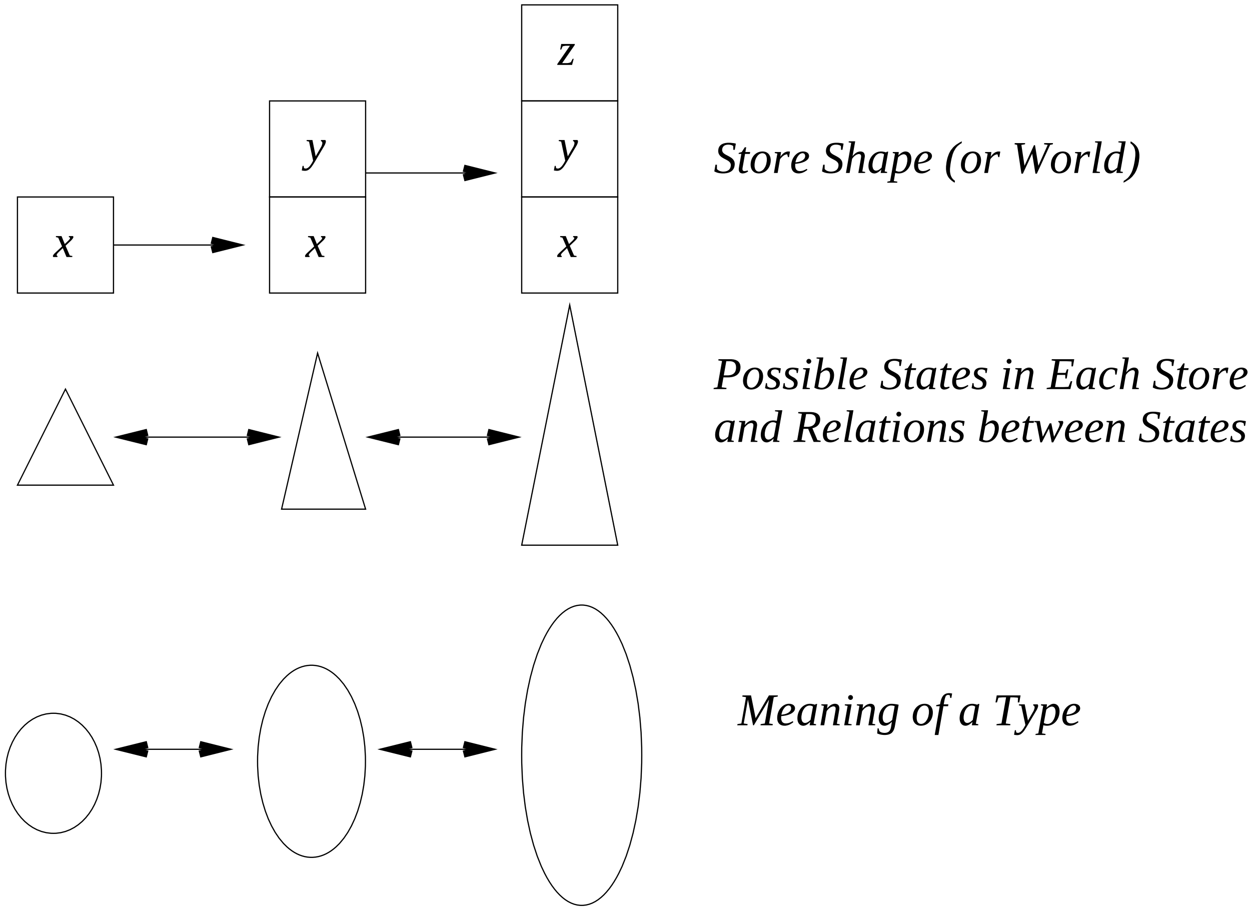
Comparisons of Semantics of Idealized Algol

semantics	type	term
global state semantics	set	function
functor category	family of sets	family of functions
parametricity	family of sets and family of relations	family of functions

Parametricity Semantics — Family of Sets



Parametricity Semantics — Family of Relations



Parametricity Semantics — Family Indexed by What?

- worlds (or store shape): finite sets which denote a set of variables currently allocated on the memory.
- family of sets: indexed by worlds
- family of relations: indexed by a relation between possible states at two store shapes.
  - $State(W) = W \to \mathbf{Int}$
  - $R\colon State(W_0) \leftrightarrow State(W_1)$

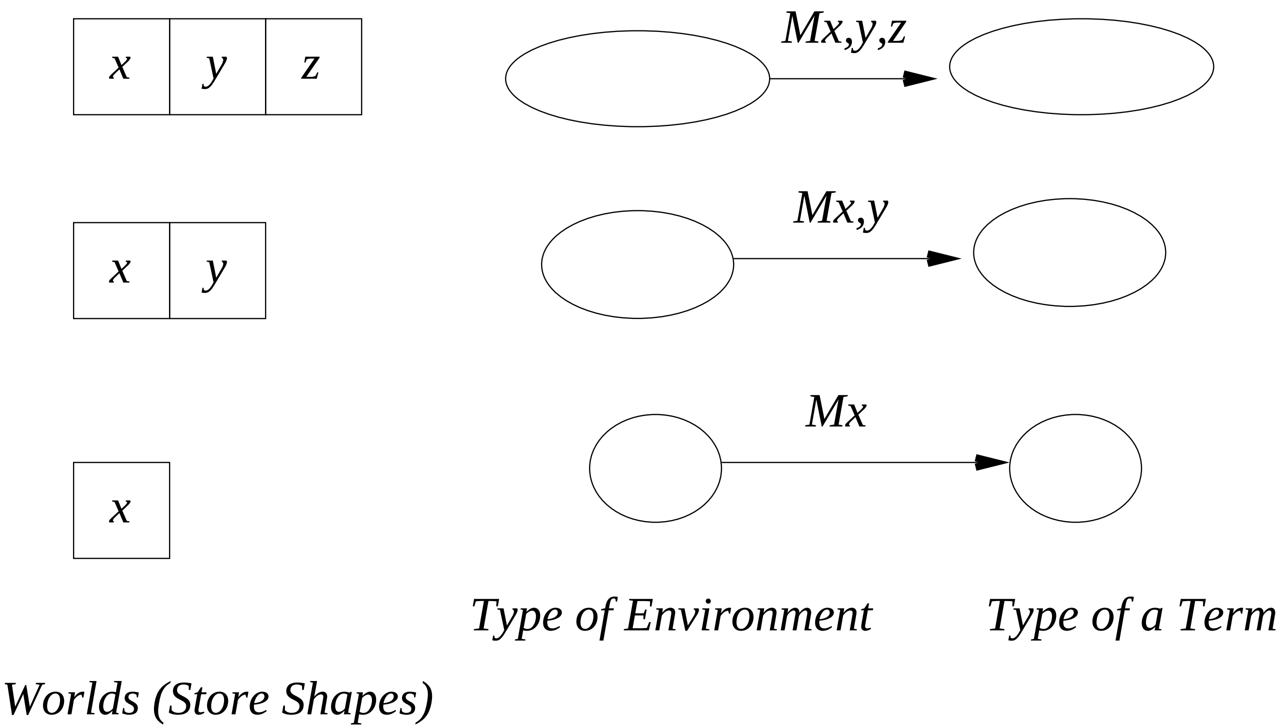
Interpretation of Types in Parametricity Semantics

each type θ, worlds W, L and $R\colon State(W_0) \leftrightarrow State(W_1)$

- $[\![\theta]\!]_{\mathbf{Set}}(W)$: possible values when only W are allocated on the memory
- expand$(W, W \cup L)_\theta : [\![\theta]\!]_{\mathbf{Set}}(W) \to [\![\theta]\!]_{\mathbf{Set}}(W \cup L)$
- $[\![\theta]\!]_{\mathbf{Rel}}(R)\colon [\![\theta]\!]_{\mathbf{Set}}(W_0) \leftrightarrow [\![\theta]\!]_{\mathbf{Set}}(W_1)$

$$\begin{tikzcd} State(W_0) \arrow[d, leftrightarrow, "R"'] & & {[\![\theta]\!]_{\mathbf{Set}}(W_0)} \arrow[d, leftrightarrow, "{[\![\theta]\!]_{\mathbf{Rel}}(R)}"] \\ State(W_1) & \longmapsto & {[\![\theta]\!]_{\mathbf{Set}}(W_1)} \end{tikzcd}$$

Interpretation of Terms in Parametricity Semantics

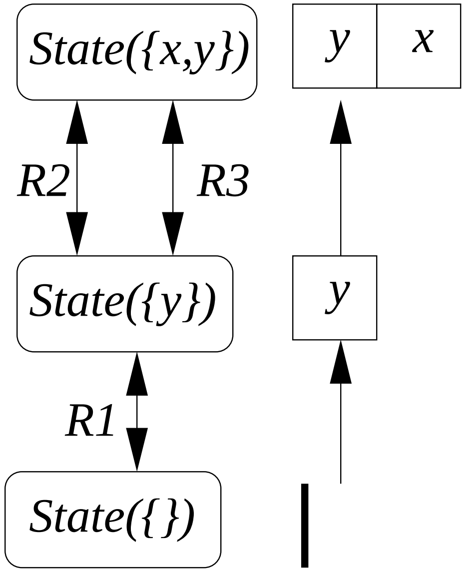


Each term $x_0\colon\theta_0 \ldots x_n\colon\theta_n \vdash M\colon\theta$ denotes a "uniform" family of functions from $[\![\Gamma]\!](W)$ to $[\![\theta]\!](W)$ for each world W

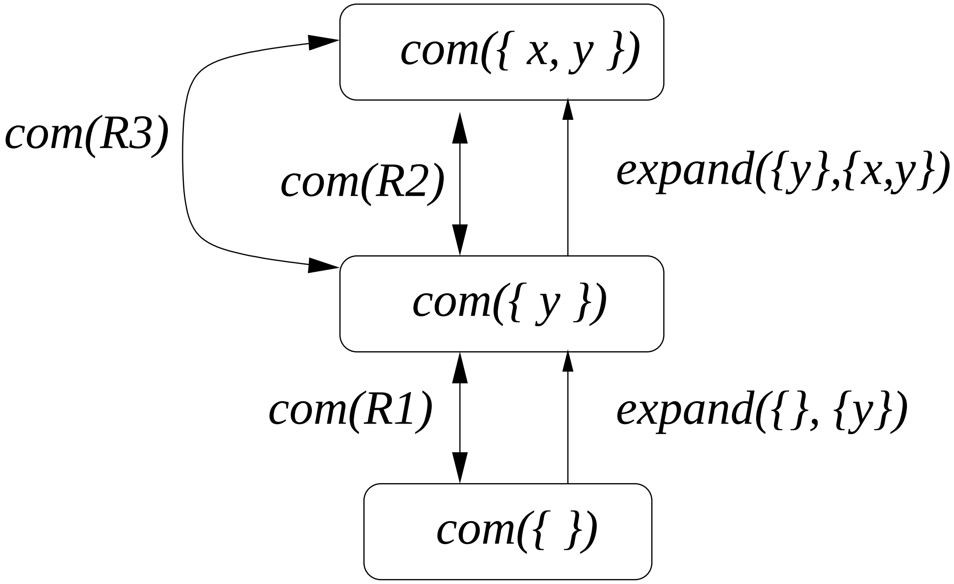
Interpretation of Command Type

- commands:
  - $[\![\mathbf{com}]\!](W) = State(W) \to State(W)$
  - $[\![\mathbf{com}]\!](R : State(W_0) \leftrightarrow State(W_1)) = R \to R$
  - informally, $\mathsf{expand}(W, W \cup L)_{\mathbf{com}}(c) = c \times \mathsf{id}_{State(W \cup L - W)}$

Interpretation of Command Type



Relations and Worlds



Interpretation of Command Type

Semantics of Symmetric Noninterference

Semantics of Symmetric Noninterference — Support

- support captures "accessed locations by a program phrase"
- support: for any $a \in [\![\theta]\!](W)$, $supp(a)$ is a subset of W such that
  - there exists $a_0 \in supp(W)$ such that $\mathrm{expand}(supp(a), W)_\theta(a_0) = a$

$$\begin{tikzcd} a \in [\![\theta]\!](W) \\ a_0 \in [\![\theta]\!](supp(a)) \arrow[u, mapsto, "\mathrm{expand}(supp(a), W)"] \end{tikzcd}$$

  - $supp(a)$ is the least such set

- Example: for $(x := 3), (y := 3; x := 2) \in [\![\mathbf{com}]\!](\{x, y\})$,
  - $supp(x := 3) = \{x\}$
  - $supp(y := 3; x := 2) = \{x, y\}$
- meaning of symmetric noninterference, guess what?

Semantics of Symmetric Noninterference

- noninterference : for any $a \in [\![\theta_0]\!](W), b \in [\![\theta_1]\!](W)$, $a \perp b$ iff they access disjoint memory locations, i.e., $supp(a) \cap supp(b) = \emptyset$

- noninterference well defined because

  Theorem 1. [minimal resource] *For any IA types θ and $a \in [\![\theta]\!](W)$, $supp(a)$ is defined.*

- the second principle holds because

  Lemma 1. *If $a \perp b$ and $a \perp c$, then $a \perp (b, c)$.*

Expected Properties of IA Types

- why Minimal Resource? because IA types are not just families of sets and families of relations!!

- expected properties of IA types θ:

  1. if a program phrase M doesn't use a variable x and M doesn't use a variable y, then M doesn't use both x and y.
     - if expand$(W - \{x\}, W)m_x = m$ and expand$(W - \{y\}, W)m_y = m$, then there is a unique $m_{x,y} \in [\![\theta]\!](W - \{x, y\})$ such that expand$(W - \{x, y\}, W)m_{x,y} = m$.
  2. expansions preserve and reflect "relatedness". The expansions of two program phrases M and N are related $[\![\theta]\!](R \times S)$ iff M and N are related by $[\![\theta]\!](R)$.

- the first expected property imply minimal resource.

- now, how can we show the others?

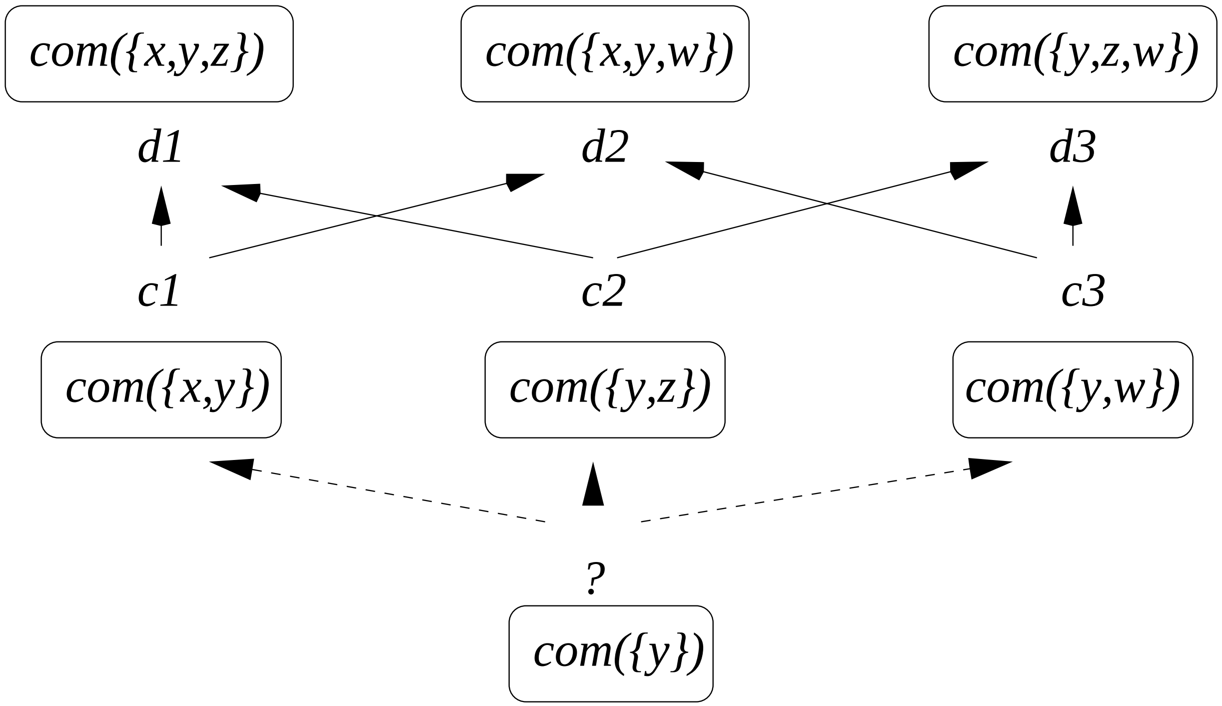
Expected Properties of IA Types Expressed as “Sheaf Condition”

Find a “sheaf condition” such that

- “sheaf condition” on a set implies the first expected property
- “reflect” part in the second expected property = “sheaf condition” on a relation
  - “preserve” part is a well-known fact in parametricity semantics

Then to show all commands, expressions and variables satisfy it.

"Sheaf Condition" on Sets

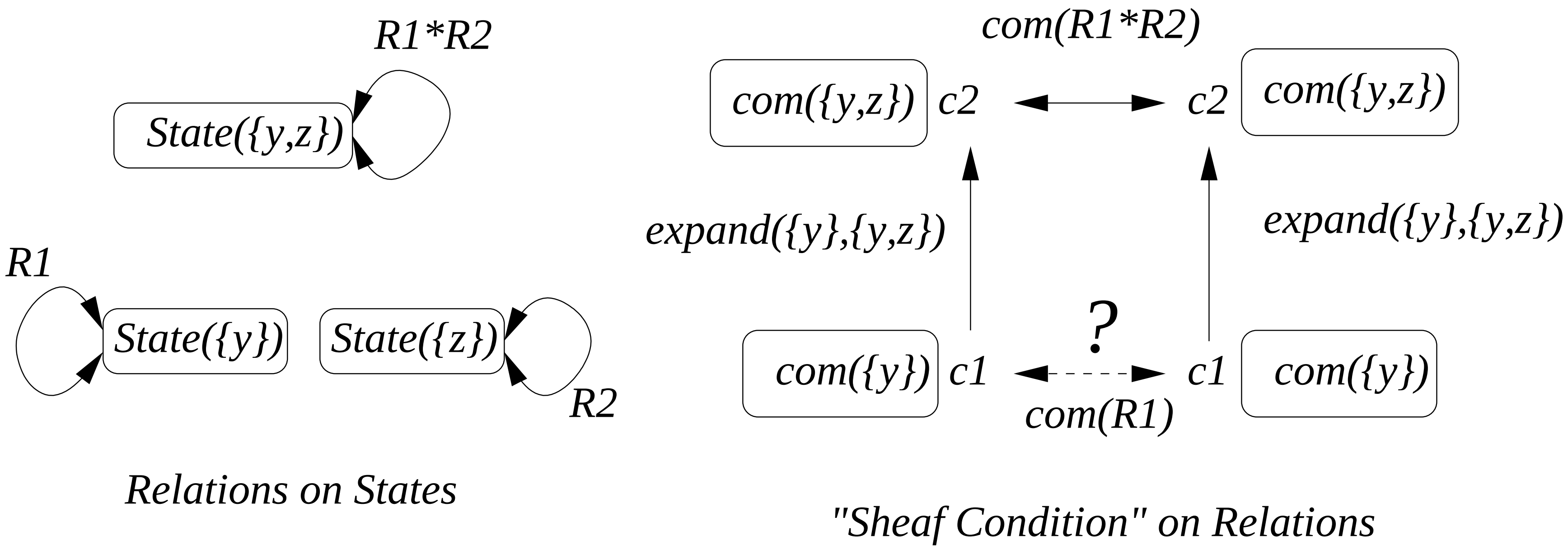


"Sheaf Condition" on Sets

"Sheaf Condition" on Sets

- for any finite family of worlds $\{W_i\}_{i \in I}$ and $\{a_i \in [\![\theta]\!](W_i)\}_{i \in I}$,
  - if expand$(W_i, W_i \cup W_j)a_i =$ expand$(W_j, W_i \cup W_j)a_j$, then there exists a unique $a \in [\![\theta]\!](\bigcap_{i \in I} W)$ such that expand$(W, W_i)(a) = a_i$

"Sheaf Condition" on Relations



“Sheaf Condition” on Relations

- for any “nonidentity” $R : State(W_0) \leftrightarrow State(W_1)$ and any nonempty relation $S : State(L_0) \leftrightarrow State(L_1)$,
  - if expand$(W_0, W_0 \cup L_0)(a_0)\, [\![\theta]\!](R \times S)$ expand$(W_1, W_1 \cup L_1)(a_1)$, then $a_0\, [\![\theta]\!](R)\, a_1$

Minimal Resource and Sheaf Condition

Theorem 2. [sheaf condition] *For any IA type θ, $[\![\theta]\!]$ satisfies the sheaf condition.*

Theorem 3. *For any type θ, if $[\![\theta]\!]$ satisfies the sheaf conditions and its expansions are injective, any $a \in [\![\theta]\!]_{\mathbf{Set}}(W)$ has the support.*

Conclusion

Conclusion

- Research on states has dealt with local variable (functor category), data abstraction (parametricity), state irreversibility (linearity), noninterference (pullback preserving functors).
  - Parametric Sheaf Model: local variable + data abstraction + noninterference
  - Linear Parametricity Model [O'Hearn and Reynolds]: local variable + data abstraction + state irreversibility
- Papers: `http://www.students.uiuc.edu/~hyang/`
  Prelim Proposal (buggy, but readable though), Parametric Sheaves for modelling Store Locality (oriented more on mathematics)