Full abstraction à la O'Hearn–Riecke

Ohad Kammar† and Shin-ya Katsumata* and Philip Saville†

†School of Informatics
University of Edinburgh

*National Institute of Informatics
Tokyo

An apology / warning

Full abstraction	$\rightsquigarrow$	operational semantics coincides with denotational semantics

Full abstraction $\rightsquigarrow$ operational semantics coincides with denotational semantics

$$\Gamma \vdash M \simeq_{\mathrm{ctx}} M' : \sigma \iff \mathcal{C}[M] \Downarrow V \iff \mathcal{C}[M'] \Downarrow V$$

$\mathcal{C}[-]$ any closed ground context

Full abstraction $\rightsquigarrow$ operational semantics coincides with denotational semantics

$$\Gamma \vdash M \simeq_{\mathrm{ctx}} M' : \sigma \quad \Longleftrightarrow \quad \mathcal{C}[M] \Downarrow V \iff \mathcal{C}[M'] \Downarrow V$$

$\mathcal{C}[-]$ any closed ground context

Adequacy: $[\![M]\!] = [\![M']\!] \;\Rightarrow\; M \simeq_{\mathrm{ctx}} M'$

Full abstraction: $M \simeq_{\mathrm{ctx}} M' \;\Rightarrow\; [\![M]\!] = [\![M']\!]$

Full abstraction $\rightsquigarrow$ operational semantics coincides with denotational semantics

$$\Gamma \vdash M \simeq_{\mathrm{ctx}} M' : \sigma \iff \mathcal{C}[M] \Downarrow V \iff \mathcal{C}[M'] \Downarrow V$$

$\mathcal{C}[-]$ any closed ground context

Adequacy: $[\![M]\!] = [\![M']\!] \;\Rightarrow\; M \simeq_{\mathrm{ctx}} M'$

Full abstraction: $M \simeq_{\mathrm{ctx}} M' \;\Rightarrow\; [\![M]\!] = [\![M']\!]$

Loosely: adequacy is easy [logical relations] but full abstraction is hard

Full abstraction $\rightsquigarrow$ operational semantics coincides with denotational semantics

$$\Gamma \vdash M \simeq_{\mathrm{ctx}} M' : \sigma \iff \mathcal{C}[M] \Downarrow V \iff \mathcal{C}[M'] \Downarrow V$$

$\mathcal{C}[-]$ any closed ground context

Adequacy: $[\![M]\!] = [\![M']\!] \Rightarrow M \simeq_{\mathrm{ctx}} M'$

Full abstraction: $M \simeq_{\mathrm{ctx}} M' \Rightarrow [\![M]\!] = [\![M']\!]$

Loosely: adequacy is easy [logical relations] but full abstraction is hard

Problem $\rightsquigarrow$ model is too rich

Full abstraction $\rightsquigarrow$ operational semantics coincides with denotational semantics

$$\Gamma \vdash M \simeq_{\mathrm{ctx}} M' : \sigma \iff \mathcal{C}[M] \Downarrow V \iff \mathcal{C}[M'] \Downarrow V$$

$\mathcal{C}[-]$ any closed ground context

Adequacy: $[\![M]\!] = [\![M']\!] \;\Rightarrow\; M \simeq_{\mathrm{ctx}} M'$

Full abstraction: $M \simeq_{\mathrm{ctx}} M' \;\Rightarrow\; [\![M]\!] = [\![M']\!]$

Loosely: adequacy is easy [logical relations] but full abstraction is hard

Problem $\rightsquigarrow$ model is too rich, particularly at higher types:

$$f = g \text{ on definable elements does not imply } f = g$$

Full abstraction $\rightsquigarrow$ operational semantics coincides with denotational semantics

$$\Gamma \vdash M \simeq_{\mathrm{ctx}} M' : \sigma \iff \mathcal{C}[M] \Downarrow V \iff \mathcal{C}[M'] \Downarrow V$$

$\mathcal{C}[-]$ any closed ground context

Adequacy: $[\![M]\!] = [\![M']\!] \Rightarrow M \simeq_{\mathrm{ctx}} M'$

Full abstraction: $M \simeq_{\mathrm{ctx}} M' \Rightarrow [\![M]\!] = [\![M']\!]$

Loosely: adequacy is easy [logical relations] but full abstraction is hard

Problem $\rightsquigarrow$ model is too rich, particularly at higher types:

$$f = g \text{ on definable elements does not imply } f = g$$

Full abstraction for PCF $\rightsquigarrow$ games semantics, O'Hearn–Riecke

Moggi's monadic metalanguage

$$
\begin{aligned}
\text{types} ::= {} & \beta \in \mathsf{Base} \mid 1 \mid \sigma_1 * \sigma_2 \mid \sigma \mathrel{-\!\!>} \tau \mid T\sigma \\
\text{terms} ::= {} & x \\
& \mid () \\
& \mid \pi_i(M) \mid \langle M_1, M_2 \rangle \\
& \mid \lambda x : \sigma \,.\, M \mid M\,N \\
& \mid \mathrm{return}(M) \mid \mathrm{let}\ x^\sigma\ \mathrm{be}\ N\ \mathrm{in}\ M
\end{aligned}
$$

$$\text{types} ::= \beta \in \mathsf{Base} \mid 1 \mid \sigma_1 * \sigma_2 \mid \sigma \text{->} \tau \mid T\sigma$$

$$\begin{aligned}\text{terms} ::= \quad & x \\ & \mid () \\ & \mid \pi_i(M) \mid \langle M_1, M_2 \rangle \\ & \mid \lambda x : \sigma \,.\, M \mid M\,N \\ & \mid \mathrm{return}(M) \mid \mathrm{let}\ x^\sigma\ \mathrm{be}\ N\ \mathrm{in}\ M\end{aligned}$$

Model

1. CCC $(\mathbb{C}, 1, \times, \Rightarrow)$
2. strong monad (T, st)
3. interpretation of base types $s : \mathsf{Base} \to \mathbb{C}$

$$
\begin{aligned}
\text{types} &::= \beta \in \mathsf{Base} \mid 1 \mid \sigma_1 * \sigma_2 \mid \sigma \mathrel{\texttt{->}} \tau \mid T\sigma \\
\text{terms} &::= \quad x \\
&\qquad\quad \mid () \\
&\qquad\quad \mid \pi_i(M) \mid \langle M_1, M_2 \rangle \\
&\qquad\quad \mid \lambda x : \sigma \,.\, M \mid M\,N \\
&\qquad\quad \mid \mathrm{return}(M) \mid \mathrm{let}\ x^\sigma\ \mathrm{be}\ N\ \mathrm{in}\ M
\end{aligned}
$$

Model

1. CCC $(\mathbb{C}, 1, \times, \Rightarrow)$
2. strong monad (T, st)
3. interpretation of base types $s : \mathsf{Base} \to \mathbb{C}$

$\rightsquigarrow$ get an interpretation $s[\![-]\!]$ of the whole language

Model

1. CCC $(\mathbb{C}, 1, \times, \Rightarrow)$
2. strong monad (T, st)
3. interpretation of base types $s : \mathsf{Base} \to \mathbb{C}$

Contextual equivalence, semantically

$$\Gamma \vdash M \simeq_{\mathrm{ctx}} M' : \sigma \iff s[\![\mathcal{C}[M]]\!] = s[\![\mathcal{C}[M']]\!]$$

$\mathcal{C}[-]$ any closed ground context

Model

1. CCC $(\mathbb{C}, 1, \times, \Rightarrow)$
2. strong monad (T, st)
3. interpretation of base types s : Base $\to \mathbb{C}$

Contextual equivalence, semantically

$$\Gamma \vdash M \simeq_{\mathrm{ctx}} M' : \sigma \quad \iff \quad s[\![\mathcal{C}[M]]\!] = s[\![\mathcal{C}[M']]\!]$$

$\mathcal{C}[-]$ any closed ground context

$\rightsquigarrow$ for PCF, coincides with syntactic definition

Model

1. CCC $(\mathbb{C}, 1, \times, \Rightarrow)$
2. strong monad (T, st)
3. interpretation of base types $s : \mathsf{Base} \to \mathbb{C}$

Contextual equivalence, semantically

$$\Gamma \vdash M \simeq_{\mathrm{ctx}} M' : \sigma \iff s[\![\mathcal{C}[M]]\!] = s[\![\mathcal{C}[M']]\!]$$

$\mathcal{C}[-]$ any closed ground context

$(\mathbb{C}, T, s)$ is fully abstract if

$$M \simeq_{\mathrm{ctx}} M' \;\Rightarrow\; s[\![M]\!] = s[\![M']\!]$$

$$\text{Full completeness} + \text{extensionality}$$
$$\Downarrow$$
$$\text{full abstraction}$$

Model

1. CCC $(\mathbb{C}, 1, \times, \Rightarrow)$
2. strong monad (T, st)
3. interpretation of base types $s : \mathrm{Base} \to \mathbb{C}$

$(\mathbb{C}, T, s)$ is extensional if global elements distinguish maps:

$$f = g : A \to B \iff f \circ a = g \circ a \text{ for all } a : 1 \to A$$

$(\mathbb{C}, T, s)$ is fully complete if every map is definable:

$$\text{whenever } f : s[\![\Gamma]\!] \to s[\![\sigma]\!], \text{then } f = s[\![\Gamma \vdash M : \sigma]\!]$$

Model

1. CCC $(\mathbb{C}, 1, \times, \Rightarrow)$
2. strong monad (T, st)
3. interpretation of base types $s : \mathrm{Base} \to \mathbb{C}$

$(\mathbb{C}, T, s)$ is extensional if global elements distinguish maps:

$$f = g : A \to B \iff f \circ a = g \circ a \text{ for all } a : 1 \to A$$

$(\mathbb{C}, T, s)$ is fully complete if every map is definable:

$$\text{whenever } f : s[\![\Gamma]\!] \to s[\![\sigma]\!], \text{then } f = s[\![\Gamma \vdash M : \sigma]\!]$$

Every extensional, fully complete model is fully abstract

Model
1. CCC $(\mathbb{C}, 1, \times, \Rightarrow)$
2. strong monad (T, st)
3. interpretation of base types $s : \mathrm{Base} \to \mathbb{C}$

$(\mathbb{C}, T, s)$ is extensional if global elements distinguish maps:

$$f = g : A \to B \iff f \circ a = g \circ a \text{ for all } a : 1 \to A$$

$(\mathbb{C}, T, s)$ is fully complete if every map is definable:

$$\text{whenever } f : s[\![\Gamma]\!] \to s[\![\sigma]\!], \text{then } f = s[\![\Gamma \vdash M : \sigma]\!]$$

Every extensional, fully complete model is fully abstract

Suppose $(x : \sigma \vdash M \simeq_{\mathrm{ctx}} M' : \tau)$.

Model

1. CCC $(\mathbb{C}, 1, \times, \Rightarrow)$
2. strong monad (T, st)
3. interpretation of base types $s : \mathrm{Base} \to \mathbb{C}$

$(\mathbb{C}, T, s)$ is extensional if global elements distinguish maps:

$$f = g : A \to B \iff f \circ a = g \circ a \text{ for all } a : 1 \to A$$

$(\mathbb{C}, T, s)$ is fully complete if every map is definable:

$$\text{whenever } f : s[\![\Gamma]\!] \to s[\![\sigma]\!], \text{then } f = s[\![\Gamma \vdash M : \sigma]\!]$$

Every extensional, fully complete model is fully abstract

Suppose $(x : \sigma \vdash M \simeq_{\mathrm{ctx}} M' : \tau)$. Suffices to show

$$s[\![M]\!] \circ \gamma = s[\![M']\!] \circ \gamma \text{ for all } \gamma : 1 \to s[\![\sigma]\!]$$

Model

1. CCC $(\mathbb{C}, 1, \times, \Rightarrow)$
2. strong monad (T, st)
3. interpretation of base types $s : \mathrm{Base} \to \mathbb{C}$

$(\mathbb{C}, T, s)$ is extensional if global elements distinguish maps:

$$f = g : A \to B \iff f \circ a = g \circ a \text{ for all } a : 1 \to A$$

$(\mathbb{C}, T, s)$ is fully complete if every map is definable:

$$\text{whenever } f : s[\![\Gamma]\!] \to s[\![\sigma]\!], \text{then } f = s[\![\Gamma \vdash M : \sigma]\!]$$

Every extensional, fully complete model is fully abstract

Suppose $(x : \sigma \vdash M \simeq_{\mathrm{ctx}} M' : \tau)$. Suffices to show

$$s[\![M]\!] \circ \gamma = s[\![M']\!] \circ \gamma \text{ for all } \gamma : 1 \to s[\![\sigma]\!]$$

By full completeness, $\gamma = s[\![\diamond \vdash G : \sigma]\!]$

Model

1. CCC $(\mathbb{C}, 1, \times, \Rightarrow)$
2. strong monad (T, st)
3. interpretation of base types $s : \mathrm{Base} \to \mathbb{C}$

$(\mathbb{C}, T, s)$ is extensional if global elements distinguish maps:

$$f = g : A \to B \iff f \circ a = g \circ a \text{ for all } a : 1 \to A$$

$(\mathbb{C}, T, s)$ is fully complete if every map is definable:

$$\text{whenever } f : s[\![\Gamma]\!] \to s[\![\sigma]\!], \text{then } f = s[\![\Gamma \vdash M : \sigma]\!]$$

Every extensional, fully complete model is fully abstract

Suppose $(x : \sigma \vdash M \simeq_{\mathrm{ctx}} M' : \tau)$. Suffices to show

$$s[\![M]\!] \circ \gamma = s[\![M']\!] \circ \gamma \text{ for all } \gamma : 1 \to s[\![\sigma]\!]$$

By full completeness, $\gamma = s[\![\diamond \vdash G : \sigma]\!]$

$$s[\![M]\!] \circ \gamma = s[\![M[x \mapsto G]]\!] = s[\![M'[x \mapsto G]]\!] = s[\![M']\!] \circ \gamma$$

How do you build a fully complete model?

Starting data: a model $(\mathbb{C}, T, s)$.

How do you build a fully complete model?

Starting data: a model $(\mathbb{C}, T, s)$.

Two goals:

1. Make every morphism definable,
2. $f = g \in (X \Rightarrow Y)$ iff they agree on definable elements.

How do you build a fully complete model?

Starting data: a model $(\mathbb{C}, T, s)$.

Two goals:

1. Make every morphism definable,
2. $f = g \in (X \Rightarrow Y)$ iff they agree on definable elements.

Two solutions [*c.f.* O'Hearn–Riecke]:

1. Characterise definability as a logical relation,
   and ensure every map preserves such relations,

How do you build a fully complete model?

Starting data: a model $(\mathbb{C}, T, s)$.

Two goals:

1. Make every morphism definable,
2. $f = g \in (X \Rightarrow Y)$ iff they agree on definable elements.

Two solutions [*c.f.* O'Hearn–Riecke]:

1. Characterise definability as a logical relation,
   and ensure every map preserves such relations,
2. Ensure every object is concrete.

How do you build a fully complete model?

Starting data: a model $(\mathbb{C}, T, s)$.

Two goals:

1. Make every morphism definable,
2. $f = g \in (X \Rightarrow Y)$ iff they agree on definable elements.

Two solutions [*c.f.* O'Hearn–Riecke]:

1. Characterise definability as a logical relation,
   and ensure every map preserves such relations,
2. Ensure every object is concrete.

The definability problem

For any model $(\mathbb{C}, T, s)$,
characterise which maps which are λ_{ml}-definable.

The definability problem

For any model $(\mathbb{C}, T, s)$,
characterise which maps which are λ_{ml}-definable.

1. Define a category of relations over objects in $\mathbb{C}$,
2. Show this defines a model,
3. Definable maps = those which preserve every relation.

The category Krip($\mathbb{C}, F$) of Kripke relations [Jung & Tiuryn, Alimohamed]

The category $\mathrm{Krip}(\mathbb{C}, F)$ of Kripke relations [Jung & Tiuryn, Alimohamed]

data:
- CCC $\mathbb{C}$
- $F : \underbrace{\mathrm{Con}}_{\text{category of contexts and renamings}}^{\mathrm{op}} \to \mathbb{C}$ cartesian

"semantic interpretation"

The category Krip($\mathbb{C}, F$) of Kripke relations [Jung & Tiuryn, Alimohamed]

data:
- CCC $\mathbb{C}$
- $F : \underbrace{\mathrm{Con}^{\mathrm{op}}}_{\text{category of contexts and renamings}} \to \mathbb{C}$ cartesian

"semantic interpretation"

objects: (W, R)

maps: $(W, R) \xrightarrow{f} (W', R')$

The category $\mathrm{Krip}(\mathbb{C}, F)$ of Kripke relations [Jung & Tiuryn, Alimohamed]

data:
- CCC $\mathbb{C}$
- $F : \mathrm{Con}^{\mathrm{op}} \to \mathbb{C}$ cartesian

"semantic interpretation"

category of contexts and renamings

predicate = Kripke relation

$\{R(\Gamma) \subseteq \mathbb{C}(F\Gamma, W)\}_{\Gamma \in \mathrm{Con}}$

compatible with renaming

object $\in \mathbb{C}$

objects: (W, R)

maps: $(W, R) \xrightarrow{f} (W', R')$

The category Krip($\mathbb{C}, F$) of Kripke relations [Jung & Tiuryn, Alimohamed]

data:
- CCC $\mathbb{C}$
- $F : \mathrm{Con}^{\mathrm{op}} \to \mathbb{C}$ cartesian

"semantic interpretation"

category of contexts and renamings

predicate = Kripke relation

$\{R(\Gamma) \subseteq \mathbb{C}(F\Gamma, W)\}_{\Gamma \in \mathrm{Con}}$

compatible with renaming

object $\in \mathbb{C}$

objects: (W, R)

maps: $(W, R) \xrightarrow{f} (W', R')$

over Set with $|F\Gamma| = n \rightsquigarrow R$ an n-ary relation

The category $\mathrm{Krip}(\mathbb{C}, F)$ of Kripke relations [Jung & Tiuryn, Alimohamed]

data:
- CCC $\mathbb{C}$
- $F : \mathrm{Con}^{\mathrm{op}} \to \mathbb{C}$ cartesian

"semantic interpretation"

category of contexts and renamings

predicate = Kripke relation

object $\in \mathbb{C}$

$\{R(\Gamma) \subseteq \mathbb{C}(F\Gamma, W)\}_{\Gamma \in \mathrm{Con}}$

compatible with renaming

objects: (W, R)

maps: $(W, R) \xrightarrow{f} (W', R')$

$f : W \to W'$

preserves the predicate

$h \in R(\Gamma) \Rightarrow (f \circ h) \in R'(\Gamma)$

over Set with $|F\Gamma| = n \rightsquigarrow R$ an n-ary relation

The category Krip($\mathbb{C}$, F) of Kripke relations [Jung & Tiuryn, Alimohamed]

data:
- CCC $\mathbb{C}$
- $F : \mathrm{Con}^{\mathrm{op}} \to \mathbb{C}$ cartesian

object $\in \mathbb{C}$

predicate = Kripke relation

$\{R(\Gamma) \subseteq \mathbb{C}(F\Gamma, W)\}_{\Gamma \in \mathrm{Con}}$

compatible with renaming

objects: (W, R)

maps: $(W, R) \xrightarrow{f} (W', R')$

Fact: Krip($\mathbb{C}$, F) is a CCC

Notation: $(W, R) \Rightarrow (X, S) := (W \Rightarrow X, R \supset S)$

The category $\mathrm{Krip}(\mathbb{C}, F)$ of Kripke relations [Jung & Tiuryn, Alimohamed]

data:
- CCC $\mathbb{C}$
- $F : \mathrm{Con}^{\mathrm{op}} \to \mathbb{C}$ cartesian

predicate = Kripke relation

$\{R(\Gamma) \subseteq \mathbb{C}(F\Gamma, W)\}_{\Gamma \in \mathrm{Con}}$

compatible with renaming

object $\in \mathbb{C}$

objects: (W, R)

maps: $(W, R) \xrightarrow{f} (W', R')$

Fact: $\mathrm{Krip}(\mathbb{C}, F)$ is a CCC

Notation: $(W, R) \Rightarrow (X, S) := (W \Rightarrow X, R \supset S)$

$(f_1, \ldots, f_n) \in (R \supset S)(\Gamma)$ iff, for any $\rho : \Gamma \to \Delta$,

$(x_1, \ldots, x_m) \in R(\Delta) \subseteq W^{F\Gamma} \Rightarrow (f_{\rho(1)} x_1, \ldots, f_{\rho(m)} x_m) \in S(\Delta) \subseteq X^{F\Delta}$

$\mathsf{Krip}(\mathbb{C}, F)$, abstractly

$$\begin{tikzcd}
 & \mathsf{Krip}(\mathbb{C}, F) \arrow[r] \arrow[d, "\mathrm{proj}", "\text{strictly preserves CCC}"'] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathbb{C} \arrow[r, "\lambda X \,.\, \mathbb{C}(F(-)\,, X)"'] & \widehat{\mathrm{Con}}
\end{tikzcd}$$

Kripke relations for definability [Jung & Tiuryn, Alimohamed, . . .]

data:
- CCC $\mathbb{C}$
- $s[\![-]\!] : \mathrm{Con}^{\mathrm{op}} \to \mathbb{C}$ semantic interpretation

$s[\![\sigma]\!] \in \mathbb{C}$

definability predicate

$\{\mathrm{DEF}^{\mathbb{C}}_{\sigma}(\Gamma) \subseteq \mathbb{C}(s[\![\Gamma]\!], s[\![\sigma]\!])\}_{\Gamma \in \mathrm{Con}}$

compatible with renaming

objects: (W, R)

maps: $(W, R) \xrightarrow{f} (W', R')$

$$\mathrm{DEF}^{\mathbb{C}}_{\sigma}(\Gamma) := \{s[\![\Gamma \vdash M : \sigma]\!] \mid M \text{ is derivable}\}$$

Kripke relations for definability [Jung & Tiuryn, Alimohamed, ...]

data:
- CCC $\mathbb{C}$
- $s[\![-]\!] : \mathrm{Con}^{\mathrm{op}} \to \mathbb{C}$ semantic interpretation

$s[\![\sigma]\!] \in \mathbb{C}$

definability predicate

$\{\mathrm{DEF}^{\mathbb{C}}_{\sigma}(\Gamma) \subseteq \mathbb{C}(s[\![\Gamma]\!], s[\![\sigma]\!])\}_{\Gamma \in \mathrm{Con}}$

compatible with renaming

objects: (W, R)

maps: $(W, R) \xrightarrow{f} (W', R')$

$$(s[\![\sigma]\!], \mathrm{DEF}_{\sigma}) \in \mathsf{Krip}(\mathbb{C}, s[\![-]\!]) \text{ for every } \sigma \in \mathrm{Type}$$

Kripke relations for definability [Jung & Tiuryn, Alimohamed, ...]

data:
- CCC $\mathbb{C}$
- $s[\![-]\!] : \mathrm{Con}^{\mathrm{op}} \to \mathbb{C}$ semantic interpretation

$s[\![\sigma]\!] \in \mathbb{C}$

definability predicate

$\{\mathrm{DEF}^{\mathbb{C}}_{\sigma}(\Gamma) \subseteq \mathbb{C}(s[\![\Gamma]\!], s[\![\sigma]\!])\}_{\Gamma \in \mathrm{Con}}$

compatible with renaming

objects: (W, R)

maps: $(W, R) \xrightarrow{f} (W', R')$

$$(s[\![\sigma]\!], \mathrm{DEF}_{\sigma}) \in \mathsf{Krip}(\mathbb{C}, s[\![-]\!]) \text{ for every } \sigma \in \mathrm{Type}$$

how do we characterise the maps in $\mathbb{C}$ that are definable?

Characterising STLC-definable maps [Jung & Tiuryn, Alimohamed]

$$\begin{tikzcd}
& \mathsf{Krip}(\mathbb{C}, s[\![-]\!]) \arrow[r] \arrow[d, "\mathrm{proj}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathbb{C} \arrow[r, "\lambda X\,.\,\mathbb{C}(s[\![-]\!]\,,X)"'] & \widehat{\mathrm{Con}}
\end{tikzcd}$$

strictly
preserves CCC

Characterising STLC-definable maps [Jung & Tiuryn, Alimohamed]

$\hat{s}(\beta) := (s[\![\beta]\!], \mathrm{DEF}^{\mathbb{C}}_{\beta})$

$$\begin{tikzcd}
 & \mathsf{Krip}(\mathbb{C}, s[\![-]\!]) \arrow[r] \arrow[d, "\mathrm{proj}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[ur, bend left, "\hat{s}"] \arrow[r, "s"'] & \mathbb{C} \arrow[r, "\lambda X\,.\,\mathbb{C}(s[\![-]\!]\,,X)"'] & \widehat{\mathrm{Con}}
\end{tikzcd}$$

strictly
preserves CCC

Characterising STLC-definable maps [Jung & Tiuryn, Alimohamed]

$\hat{s}(\beta) := (s[\![\beta]\!], \mathrm{DEF}^{\mathbb{C}}_{\beta})$

$$
\begin{tikzcd}
 & \mathsf{Krip}(\mathbb{C}, s[\![-]\!]) \arrow[r] \arrow[d, "\mathrm{proj}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[ur, bend left, "\hat{s}"] \arrow[r, "s"'] & \mathbb{C} \arrow[r, "\lambda X . \mathbb{C}(s[\![-]\!], X)"'] & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

strictly preserves CCC

$\hat{s}[\![\sigma]\!] = (s[\![\sigma]\!], \mathrm{DEF}^{\mathbb{C}}_{\sigma})$ for every STLC-type σ.

i.e. $(\mathrm{DEF}^{\mathbb{C}}_{\sigma} \supset \mathrm{DEF}^{\mathbb{C}}_{\tau}) = \mathrm{DEF}^{\mathbb{C}}_{\sigma \Rightarrow \tau}$ *etc.*

Characterising STLC-definable maps [Jung & Tiuryn, Alimohamed]

$$
\begin{tikzcd}
 & \mathsf{Krip}(\mathbb{C}, s[\![-]\!]) \arrow[r] \arrow[d, "\mathrm{proj}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[ur, bend left, "\hat{s}"] \arrow[r, "s"'] & \mathbb{C} \arrow[r, "\lambda X \,.\, \mathbb{C}(s[\![-]\!]\,, X)"'] & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

strictly preserves CCC

$\hat{s}[\![\sigma]\!] = (s[\![\sigma]\!], \mathrm{DEF}^{\mathbb{C}}_{\sigma})$ for every STLC-type σ.

$f : s[\![\Gamma]\!] \to s[\![\sigma]\!]$ is definable $\iff f : \hat{s}[\![\Gamma]\!] \to \hat{s}[\![\sigma]\!]$

Characterising STLC-definable maps [Jung & Tiuryn, Alimohamed]

$$
\begin{tikzcd}
 & \mathrm{Krip}(\mathbb{C}, s[\![-]\!]) \arrow[r] \arrow[d, "\mathrm{proj}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathrm{Base} \arrow[ur, bend left, "\hat{s}"] \arrow[r, "s"'] & \mathbb{C} \arrow[r, "\lambda X \,.\, \mathbb{C}(s[\![-]\!]\,, X)"'] & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

strictly preserves CCC

$$\hat{s}[\![\sigma]\!] = (s[\![\sigma]\!], \mathrm{DEF}^{\mathbb{C}}_{\sigma}) \text{ for every STLC-type } \sigma.$$

$$f : s[\![\Gamma]\!] \to s[\![\sigma]\!] \text{ is definable } \iff f : \hat{s}[\![\Gamma]\!] \to \hat{s}[\![\sigma]\!]$$

$$\hat{s}[\![\Gamma]\!] \xrightarrow{\quad f \quad} \hat{s}[\![\sigma]\!]$$

Characterising STLC-definable maps [Jung & Tiuryn, Alimohamed]

$$
\begin{tikzcd}
 & \mathrm{Krip}(\mathbb{C}, s[\![-]\!]) \arrow[r] \arrow[d, "\mathrm{proj}\ \text{strictly preserves CCC}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathrm{Base} \arrow[r, "s"'] \arrow[ur, bend left, "\hat{s}"] & \mathbb{C} \arrow[r, "\lambda X\,.\,\mathbb{C}(s[\![-]\!]\,,X)"'] & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

$\hat{s}[\![\sigma]\!] = (s[\![\sigma]\!], \mathrm{DEF}^{\mathbb{C}}_{\sigma})$ for every STLC-type σ.

$f : s[\![\Gamma]\!] \to s[\![\sigma]\!]$ is definable $\iff f : \hat{s}[\![\Gamma]\!] \to \hat{s}[\![\sigma]\!]$

$$(s[\![\Gamma]\!], \mathrm{DEF}^{\mathbb{C}}_{\Gamma}) \xrightarrow{f} (s[\![\sigma]\!], \mathrm{DEF}^{\mathbb{C}}_{\sigma})$$

Characterising STLC-definable maps [Jung & Tiuryn, Alimohamed]

$$\begin{tikzcd}
 & \mathsf{Krip}(\mathbb{C}, s\llbracket - \rrbracket) \arrow[r] \arrow[d, "\mathrm{proj}\ \text{strictly preserves CCC}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[ur, bend left, "\hat{s}"] \arrow[r, "s"'] & \mathbb{C} \arrow[r, "\lambda X \,.\, \mathbb{C}(s\llbracket - \rrbracket\,, X)"'] & \widehat{\mathrm{Con}}
\end{tikzcd}$$

$\hat{s}\llbracket \sigma \rrbracket = (s\llbracket \sigma \rrbracket, \mathrm{DEF}^{\mathbb{C}}_{\sigma})$ for every STLC-type σ.

$f : s\llbracket \Gamma \rrbracket \to s\llbracket \sigma \rrbracket$ is definable $\iff f : \hat{s}\llbracket \Gamma \rrbracket \to \hat{s}\llbracket \sigma \rrbracket$

$$(s\llbracket \Gamma \rrbracket, \mathrm{DEF}^{\mathbb{C}}_{\Gamma}) \xrightarrow{f} (s\llbracket \sigma \rrbracket, \mathrm{DEF}^{\mathbb{C}}_{\sigma})$$

$\mathrm{id}_{\llbracket \Gamma \rrbracket}$ is definable $\longmapsto f \circ \mathrm{id}_{\llbracket \Gamma \rrbracket} \in \mathrm{DEF}^{\mathbb{C}}_{\sigma}(\Gamma)$

Characterising STLC-definable maps [Jung & Tiuryn, Alimohamed]

$$
\begin{tikzcd}
& \mathrm{Krip}(\mathbb{C}, s[\![-]\!]) \arrow[d, "\mathrm{proj}"] \arrow[rr] & & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathrm{Base} \arrow[ur, bend left, "\hat{s}"] \arrow[r, "s"'] & \mathbb{C} \arrow[rr, "\lambda X \,.\, \mathbb{C}(s[\![-]\!]\,, X)"'] & & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

strictly preserves CCC

$\hat{s}[\![\sigma]\!] = (s[\![\sigma]\!], \mathrm{DEF}^{\mathbb{C}}_{\sigma})$ for every STLC-type σ.

the definable maps are exactly those that lift to $(\mathrm{Krip}(\mathbb{C}, s[\![-]\!]), \hat{s})$

What about monadic types?

What about monadic types?

Diversion: lifting monads up fibrations

$$
\begin{tikzcd}
\mathbb{E} \arrow[d, "\text{fibration } p"] \\
\mathbb{B} \arrow[loop below, "\text{strong monad } T"]
\end{tikzcd}
$$

A lifting of T is a monad $\hat{T}$ on $\mathbb{E}$ such that

$$p(\hat{T}W) = T(pW)$$

$$p(\hat{\mathrm{st}}) = \mathrm{st}, p(\hat{\eta}) = \eta_p, p(\hat{\mu}) = \mu_p$$

$$\begin{tikzcd} \mathbb{E} \arrow[d, "\text{fibration } p"] \\ \mathbb{B} \arrow[loop below, "\text{strong monad } T"] \end{tikzcd}$$

A lifting of T is a monad $\hat{T}$ on $\mathbb{E}$ such that

$$p(\hat{T}W) = T(pW)$$

$$p(\hat{\mathrm{st}}) = \mathrm{st}, p(\hat{\eta}) = \eta_p, p(\hat{\mu}) = \mu_p$$

$\top\top$-**lifting** [*c.f.* Katsumata] $\rightsquigarrow$ defines $\hat{S}$ from $\hat{T}$ and $\tau : S \Rightarrow T$

$$\begin{tikzcd} \mathbb{E} \arrow[loop above, "\text{lifting } \hat{T}"] \arrow[d, "p"'] \\ \mathbb{B} \arrow[loop below, "\tau : S \Rightarrow T"] \end{tikzcd}$$

$$\begin{array}{c}\mathbb{E}\\ \Big\downarrow \text{fibration } p\\ \mathbb{B}\\ \circlearrowleft\\ \text{strong monad } T\end{array}$$

A lifting of T is a monad $\hat{T}$ on $\mathbb{E}$ such that

$$p(\hat{T}W) = T(pW)$$

$$p(\hat{\mathrm{st}}) = \mathrm{st}, p(\hat{\eta}) = \eta_p, p(\hat{\mu}) = \mu_p$$

$\top\top$-**lifting** [c.f. Katsumata] $\rightsquigarrow$ defines $\hat{S}$ from $\hat{T}$ and $\tau : S \Rightarrow T$

$$\begin{array}{c}\text{lifting } \hat{T}\\ \circlearrowright\\ \mathbb{E}\\ p\Big\downarrow\\ \mathbb{B}\\ \circlearrowleft\\ \tau : S \Rightarrow T\end{array}$$

$$\hat{S}X \overset{\tau_X^*}{\underset{\text{def.}}{\dashrightarrow}} \hat{T}X$$

$$S(pX) \xrightarrow[\tau_{pX}]{} T(pX)$$

monad morphism
$\tau : S \Rightarrow T$

$\top\top$-**lifting** [Katsumata] $\rightsquigarrow$ defines $\hat{T}$ from lifting parameter (TW, R)

$$
\begin{tikzcd}
\hat{\mathrm{K}}_{(TW,R)} \arrow[d, phantom, "\circlearrowleft"] & & \\
\mathrm{Krip}(\mathbb{C}, F) \arrow[dd, "\mathrm{proj}"'] & \hat{T}(A,S) \arrow[r, dashed, "\text{def.}"] & ((A,S) \mathrel{\hat{\Rightarrow}} (TW,R)) \mathrel{\hat{\Rightarrow}} (TW,R) \\
& & \\
\mathbb{C} \arrow[d, phantom, "\circlearrowright"] & T(A) \arrow[r, "{\substack{\text{canonical} \\ \text{monad morphism} \\ t \mapsto \lambda h\,.\,h^{\#}(t)}}"'] & (A \Rightarrow TW) \Rightarrow TW \\
T\,, \mathrm{K}_{TW} & &
\end{tikzcd}
$$

$\top\top$-**lifting** [Katsumata] $\rightsquigarrow$ defines $\hat{T}$ from lifting parameter (TW, R)

$$
\begin{tikzcd}
\hat{\mathrm{K}}_{(TW,R)} \arrow[d, phantom, "\circlearrowright"] & \\
\mathrm{Krip}(\mathbb{C}, F) \arrow[d, "\mathrm{proj}"'] & \hat{T}(A,S) \arrow[r, dashed, "\text{def.}"] & ((A,S) \hat{\Rightarrow} (TW, R)) \hat{\Rightarrow} (TW, R) \\
\mathbb{C} \arrow[d, phantom, "\circlearrowleft"] & T(A) \arrow[r, "\substack{\text{canonical}\\ \text{monad morphism}\\ t \mapsto \lambda h\,.\,h^{\#}(t)}"'] & (A \Rightarrow TW) \Rightarrow TW \\
T, \mathrm{K}_{TW} & &
\end{tikzcd}
$$

Omitting varying arity, for binary relations:

$$
t \in \hat{T}S \iff \text{for any } f \in (S \supset R),\ f^{\#}(t) \in R
$$

where $S \subseteq (TA)^2$, $R \subseteq (TW)^2$.

Characterising λ_{ml}-definable maps

$\hat{s}(\beta) := (s[\![\beta]\!], \mathrm{DEF}^{\mathbb{C}}_{\beta})$

$$
\begin{tikzcd}
 & \mathsf{Krip}(\mathbb{C}, s[\![-]\!]) \arrow[r] \arrow[d, "\mathrm{proj}"] \arrow[d, phantom, shift left=12ex, "\text{strictly preserves CCC}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[ur, bend left, "\hat{s}"] \arrow[r, "s"'] & \mathbb{C} \arrow[r, "\lambda X\,.\,\mathbb{C}(s[\![-]\!]\,,X)"'] & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

$\hat{s}[\![\sigma]\!] = (s[\![\sigma]\!], \mathrm{DEF}^{\mathbb{C}}_{\sigma})$ for every STLC-type σ

Characterising λ_{ml}-definable maps

$\hat{s}(\beta) := (s[\![\beta]\!], \mathrm{DEF}^{\mathbb{C}}_{\beta})$

$$
\begin{tikzcd}
 & \mathsf{Krip}(\mathbb{C}, s[\![-]\!]) \arrow[loop above, "\hat{T}"] \arrow[d, "\mathrm{proj}"] \arrow[rr] & & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[ur, bend left, "\hat{s}"] \arrow[r, "s"'] & \mathbb{C} \arrow[loop below, "T"] \arrow[rr, "\lambda X\,.\,\mathbb{C}(s[\![-]\!]\,,X)"'] & & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

strictly
preserves CCC
and monad

$\hat{s}[\![\sigma]\!] = (s[\![\sigma]\!], \mathrm{DEF}^{\mathbb{C}}_{\sigma})$ for every STLC-type σ

Characterising λ_{ml}-definable maps

$\hat{s}(\beta) := (s[\![\beta]\!], \mathrm{DEF}^{\mathbb{C}}_{\beta})$

$$\begin{tikzcd} & \mathsf{Krip}(\mathbb{C}, s[\![-]\!]) \arrow[d, "\mathrm{proj}"] \arrow[loop above, "\hat{T}"] \arrow[r] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\ \mathsf{Base} \arrow[ur, bend left, "\hat{s}"] \arrow[r, "s"'] & \mathbb{C} \arrow[loop below, "T"] \arrow[r, "\lambda X \,.\, \mathbb{C}(s[\![-]\!]\,, X)"'] & \widehat{\mathrm{Con}} \end{tikzcd}$$

strictly preserves CCC and monad

$\hat{T}(s[\![\sigma]\!], \mathrm{DEF}^{\mathbb{C}}_{\sigma}) = (s[\![T\sigma]\!], \mathrm{DEF}^{\mathbb{C}}_{T\sigma})$ with the right lifting parameters

$$\Downarrow$$

$\hat{s}[\![\sigma]\!] = (s[\![\sigma]\!], \mathrm{DEF}^{\mathbb{C}}_{\sigma})$ for every $\sigma \in \mathrm{Type}$

$\hat{s}(\beta) := (s[\![\beta]\!], \mathrm{DEF}^{\mathbb{C}}_{\beta})$

$$
\begin{tikzcd}
 & \mathsf{Krip}(\mathbb{C}, s[\![-]\!]) \arrow[loop above, "\hat{T}"] \arrow[rr] \arrow[d, "\mathrm{proj}"] & & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[ur, bend left, "\hat{s}"] \arrow[r, "s"'] & \mathbb{C} \arrow[loop below, "T"] \arrow[rr, "\lambda X\,.\,\mathbb{C}(s[\![-]\!]\,,X)"'] & & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

strictly
preserves CCC
and monad

$$\hat{s}[\![\sigma]\!] = (s[\![\sigma]\!], \mathrm{DEF}^{\mathbb{C}}_{\sigma}) \text{ for every } \sigma \in \mathrm{Type}$$

$$f : s[\![\Gamma]\!] \to s[\![\sigma]\!] \text{ is definable } \iff f : \hat{s}[\![\Gamma]\!] \to \hat{s}[\![\sigma]\!]$$

Characterising λ_{ml}-definable maps

$\hat{s}(\beta) := (s[\![\beta]\!], \mathrm{DEF}^{\mathbb{C}}_{\beta})$

$$
\begin{tikzcd}
 & \mathsf{Krip}(\mathbb{C}, s[\![-]\!]) \arrow[loop above, "\hat{T}"] \arrow[d, "\mathrm{proj}"] \arrow[r] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[ur, bend left, "\hat{s}"] \arrow[r, "s"'] & \mathbb{C} \arrow[loop below, "T"] \arrow[r, "\lambda X \,.\, \mathbb{C}(s[\![-]\!]\,, X)"'] & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

strictly preserves CCC and monad

$\hat{s}[\![\sigma]\!] = (s[\![\sigma]\!], \mathrm{DEF}^{\mathbb{C}}_{\sigma})$ for every $\sigma \in \mathrm{Type}$

the definable maps are exactly those that lift to $(\mathsf{Krip}(\mathbb{C}, s[\![-]\!]), \hat{T}, \hat{s})$

How do you build a fully complete model?

Starting data: a model $(\mathbb{C}, T, s)$.

Two goals:

1. Make every morphism definable,
2. $f = g \in (X \Rightarrow Y)$ iff they agree on definable elements.

Two solutions [*c.f.* O'Hearn–Riecke]:

1. Characterise definability as a logical relation,
   and ensure every map preserves such relations
2. Ensure every object is concrete.

How do you build a fully complete model?

Starting data: a model $(\mathbb{C}, T, s)$.

Two goals:

1. Make every morphism definable,
2. $f = g \in (X \Rightarrow Y)$ iff they agree on definable elements.

Two solutions [*c.f.* O'Hearn–Riecke]:

1. Characterise definability as a logical relation, ✓
   and ensure every map preserves such relations
2. Ensure every object is concrete.

How do you build a fully complete model?

Starting data: a model $(\mathbb{C}, T, s)$.

Two goals:

1. Make every morphism definable,
2. $f = g \in (X \Rightarrow Y)$ iff they agree on definable elements.

Two solutions [*c.f.* O'Hearn–Riecke]:

1. Characterise definability as a logical relation, ✓
   and ensure every map preserves such relations
2. Ensure every object is concrete.

How do you build a fully complete model?

Starting data: a model $(\mathbb{C}, T, s)$.

Two goals:

1. Make every morphism definable,
2. $f = g \in (X \Rightarrow Y)$ iff they agree on definable elements.

Two solutions [*c.f.* O'Hearn–Riecke]:

1. Characterise definability as a logical relation, ✓
   and ensure every map preserves such relations
2. Ensure every object is concrete.

$\rightsquigarrow$ For simplicity, over $\mathrm{Set}_\kappa \hookrightarrow \mathrm{Set}$ [*e.g.* hereditarily-κ sets]

Kripke relations over Set_κ

data: $F : \mathrm{Con}^{\mathrm{op}} \to \mathrm{Set}_\kappa$ cartesian

predicate

set

$\{R(\Gamma) \subseteq W^{F\Gamma}\}_{\Gamma \in \mathrm{Con}}$

compatible with renaming

objects: (W, R)

maps: $(W, R) \xrightarrow{f} (W', R')$

$f : W \to W'$

preserves the predicate

$h \in R(\Gamma) \Rightarrow (f \circ h) \in R'(\Gamma)$

Concreteness [*c.f.* O'Hearn–Riecke]

For $(W, R) \in \mathsf{Krip}(\mathrm{Set}_\kappa, F)$,

1. $x \in W$ is concrete if, for any $\Gamma \in \mathrm{Con}$,

$$\lambda\gamma^{F\Gamma} \,.\, x \in R(\Gamma)$$

2. (W, R) is concrete if every $x \in W$ is concrete.

Concreteness [*c.f.* O'Hearn–Riecke]

For $(W, R) \in \mathsf{Krip}(\mathrm{Set}_\kappa, F)$,

1. $x \in W$ is concrete if, for any $\Gamma \in \mathrm{Con}$,

$$\lambda\gamma^{F\Gamma} \,.\, x \in R(\Gamma)$$

2. (W, R) is concrete if every $x \in W$ is concrete.

Intuitively, for $(W, R), (X, S) \in \mathsf{Krip}(\mathrm{Set}_\kappa, F)$:

$f \in (W \Rightarrow X)$ is concrete if

$$(x_1, \ldots, x_n) \in R(\Gamma) \subseteq W^{F\Gamma} \Rightarrow (fx_1, \ldots, fx_n) \in S(\Gamma) \subseteq X^{F\Gamma}$$

Concreteness [*c.f.* O'Hearn–Riecke]

For $(W, R) \in \mathsf{Krip}(\mathrm{Set}_\kappa, F)$,

1. $x \in W$ is concrete if, for any $\Gamma \in \mathrm{Con}$,
$$\lambda\gamma^{F\Gamma} . x \in R(\Gamma)$$
2. (W, R) is concrete if every $x \in W$ is concrete.

Intuitively, for $(W, R), (X, S) \in \mathsf{Krip}(\mathrm{Set}_\kappa, F)$:

$f \in (W \Rightarrow X)$ is concrete if
$$(x_1, \ldots, x_n) \in R(\Gamma) \subseteq W^{F\Gamma} \Rightarrow (fx_1, \ldots, fx_n) \in S(\Gamma) \subseteq X^{F\Gamma}$$

$\rightsquigarrow$ can detect if $f = g$ within R, S [*c.f.* sequentiality]

Concreteness [*c.f.* O'Hearn–Riecke]

For $(W, R) \in \mathsf{Krip}(\mathrm{Set}_\kappa, F)$,

1. $x \in W$ is concrete if, for any $\Gamma \in \mathrm{Con}$,
$$\lambda\gamma^{F\Gamma} \,.\, x \in R(\Gamma)$$
2. (W, R) is concrete if every $x \in W$ is concrete.

Intuitively, for $(W, R), (X, S) \in \mathsf{Krip}(\mathrm{Set}_\kappa, F)$:

$f \in (W \Rightarrow X)$ is concrete if
$$(x_1, \ldots, x_n) \in R(\Gamma) \subseteq W^{F\Gamma} \Rightarrow (fx_1, \ldots, fx_n) \in S(\Gamma) \subseteq X^{F\Gamma}$$

$\rightsquigarrow$ can detect if $f = g$ within R, S [*c.f.* sequentiality]

$\rightsquigarrow$ full subcategory of concrete objects
$$j : \mathrm{Conc}(\mathrm{Set}_\kappa, F) \hookrightarrow \mathsf{Krip}(\mathrm{Set}_\kappa, F)$$

Making $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$ a model

any lifting

$$
\begin{tikzcd}
 & \mathsf{Krip}(\mathrm{Set}_\kappa, F) \arrow[loop above, "\hat{T}"] \arrow[r] \arrow[d, "\mathrm{proj}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[r, "\lambda X \,.\, X^{F(-)}"'] & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

strictly
preserves CCC
and monad

Making $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$ a model

$$
\begin{tikzcd}
 & \text{any lifting} & \\
\mathrm{Conc}(\mathrm{Set}_\kappa, F) \arrow[r, hook] & \mathsf{Krip}(\mathrm{Set}_\kappa, F) \arrow[loop above, "\hat{T}"] \arrow[r] \arrow[d, "\mathrm{proj}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[r, "\lambda X\,.\,X^{F(-)}"'] & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

strictly
preserves CCC
and monad

Making $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$ a model

$$
\begin{tikzcd}
\mathrm{Conc}(\mathrm{Set}_\kappa, F) \arrow[r, hook] \arrow[r, phantom, shift left=3, "\perp"] \arrow[r, phantom, shift right=3, "\perp"'] & \mathsf{Krip}(\mathrm{Set}_\kappa, F) \arrow[l, bend right=40, "\text{completion } \mathrm{K}"'] \arrow[l, bend left=40, "\text{hull } \mathrm{H}"] \arrow[loop above, "\substack{\text{any lifting} \\ \hat{T}}"] \arrow[r] \arrow[d, "\mathrm{proj}\ \text{strictly preserves CCC and monad}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[r, "\lambda X\,.\,X^{F(-)}"'] & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

For $(W, R) \in \mathsf{Krip}(\mathrm{Set}_\kappa, F)$:

$\mathrm{K} \rightsquigarrow$ union each $R\Gamma$ with the diagonal

$\mathrm{H} \rightsquigarrow$ restrict to subset of concrete elements

Making $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$ a model

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
\mathrm{Conc}(\mathrm{Set}_\kappa, F) \arrow[r, hook] \arrow[r, phantom, shift left=4, "\perp"] \arrow[r, phantom, shift right=4, "\perp"'] & \mathsf{Krip}(\mathrm{Set}_\kappa, F) \arrow[l, bend right=40, "\text{completion } \mathrm{K}"'] \arrow[l, bend left=40, "\text{hull } \mathrm{H}"] \arrow[loop above, "\overset{\text{any lifting}}{\hat{T}}"] \arrow[r] \arrow[d, "\mathrm{proj}\ \ \text{strictly preserves CCC and monad}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[r, "\lambda X\,.\,X^{F(-)}"'] & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

For $(W, R) \in \mathsf{Krip}(\mathrm{Set}_\kappa, F)$:

$\mathrm{K} \rightsquigarrow$ union each $R\Gamma$ with the diagonal

$\mathrm{H} \rightsquigarrow$ restrict to subset of concrete elements

N.B.: K only changes the *relation*; H also changes the *carrier*

Making $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$ a model

$$
\begin{tikzcd}
\mathrm{Conc}(\mathrm{Set}_\kappa, F) \arrow[loop above, "\mathrm{H}\hat{T}j"] \arrow[r, hook] & \mathsf{Krip}(\mathrm{Set}_\kappa, F) \arrow[loop above, "\hat{T}"] \arrow[l, bend right=40, "\mathrm{K}"'] \arrow[l, bend left=40, "\mathrm{H}"] \arrow[d, "\mathrm{proj}"] \arrow[r] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[r, "\lambda X \,.\, X^{F(-)}"'] & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

strictly
preserves CCC
and monad

Making $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$ a model

$$
\begin{tikzcd}
\mathrm{Conc}(\mathrm{Set}_\kappa, F) \arrow[loop above, "\mathrm{H}\hat{T}j"] \arrow[r, hook] \arrow[r, phantom, shift left=4, "\perp"] \arrow[r, phantom, shift right=4, "\perp"] & \mathsf{Krip}(\mathrm{Set}_\kappa, F) \arrow[loop above, "\hat{T}"] \arrow[l, bend right=40, "\mathrm{K}"'] \arrow[l, bend left=40, "\mathrm{H}"] \arrow[r] \arrow[d, "\mathrm{proj}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[r, "\lambda X \,.\, X^{F(-)}"'] & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

strictly
preserves CCC
and monad

By abstract nonsense:

1. $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$ inherits products from $\mathsf{Krip}(\mathrm{Set}_\kappa, F)$,
2. $\mathrm{H}\hat{T}j$ becomes a strong monad on $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$,
3. $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$ is cartesian closed

Making $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$ a model

$$
\begin{tikzcd}
\mathrm{Conc}(\mathrm{Set}_\kappa, F) \arrow[loop above, "\mathrm{H}\hat{T}j"] \arrow[r, hook] \arrow[r, phantom, shift left=4, "\perp"] \arrow[r, phantom, shift right=4, "\perp"] & \mathrm{Krip}(\mathrm{Set}_\kappa, F) \arrow[l, bend right=40, "\mathrm{K}"'] \arrow[l, bend left=40, "\mathrm{H}"] \arrow[loop above, "\hat{T}"] \arrow[r] \arrow[d, "\mathrm{proj}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathrm{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[r, "\lambda X \,.\, X^{F(-)}"'] & \widehat{\mathrm{Con}}
\end{tikzcd}
$$

strictly
preserves CCC
and monad

By abstract nonsense:

1. $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$ inherits products from $\mathrm{Krip}(\mathrm{Set}_\kappa, F)$,
2. $\mathrm{H}\hat{T}j$ becomes a strong monad on $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$,
3. $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$ is cartesian closed:

$$(W, R) \blacktriangleright (X, S) := \mathrm{H}(j(W, R) \Rightarrow j(X, S))$$

$$\mathrm{eval}^{\blacktriangleright} := \text{restrict } \mathrm{eval}^{\Rightarrow} \text{ to concrete subset}$$

Starting from:

- Model $(\mathrm{Set}_\kappa, T, s)$,
- Cartesian functor $F : \mathrm{Con}^{\mathrm{op}} \to \mathrm{Set}_\kappa$,
- Lifting $\hat{T}$ of T to Krip(Set_κ, F).

Starting from:

- Model $(\mathrm{Set}_\kappa, T, s)$,
- Cartesian functor $F : \mathrm{Con}^{\mathrm{op}} \to \mathrm{Set}_\kappa$,
- Lifting $\hat{T}$ of T to $\mathsf{Krip}(\mathrm{Set}_\kappa, F)$.

Then:

1. $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$ becomes a CCC with a strong monad $\mathrm{H}\hat{T}j$,

Starting from:

- Model $(\mathrm{Set}_\kappa, T, s)$,
- Cartesian functor $F : \mathrm{Con}^{\mathrm{op}} \to \mathrm{Set}_\kappa$,
- Lifting $\hat{T}$ of T to Krip(Set_κ, F).

Then:

1. $\mathrm{Conc}(\mathrm{Set}_\kappa, F)$ becomes a CCC with a strong monad $\mathrm{H}\hat{T}j$,
2. ... with exponentials cut down by H

How do you build a fully complete model?

Starting data: a model $(\mathbb{C}, T, s)$.

Two goals:

1. Make every morphism definable,
2. $f = g \in (X \Rightarrow Y)$ iff they agree on definable elements.

Two solutions [*c.f.* O'Hearn–Riecke]:

1. Characterise definability as a logical relation, ✓
   and ensure every map preserves such relations
2. Ensure every object is concrete. ✓

How do you build a fully complete model?

Starting data: a model $(\mathbb{C}, T, s)$.

Two goals:

1. Make every morphism definable,
2. $f = g \in (X \Rightarrow Y)$ iff they agree on definable elements.

Two solutions [*c.f.* O'Hearn–Riecke]:

1. Characterise definability as a logical relation, ✓
   and ensure every map preserves such relations
2. Ensure every object is concrete. ✓

The chicken and the egg starting model $(\mathrm{Set}_\kappa, T, s)$

$$
\begin{tikzcd}
\begin{array}{c}\text{model } (\mathbb{O}, \mathrm{W}, d) \text{ with maps} \\ \text{preserving } \mathrm{DEF}^{\mathbb{O}}\end{array} \arrow[r, bend left] & \begin{array}{c}\text{definability relation} \\ \text{over } \mathbb{O}\end{array} \arrow[l, bend left]
\end{tikzcd}
$$

Trick: cut the loop!

$$\begin{tikzcd}
\text{model } (\mathbb{O}, \mathrm{W}, d) \text{ with maps} \\ \text{preserving } \mathrm{DEF}^{\mathbb{O}} \arrow[r, bend left] & \text{definability relation} \\ \text{over } \mathbb{O} \arrow[l, bend left]
\end{tikzcd}$$

The chicken and the egg starting model $(\mathrm{Set}_\kappa, T, s)$

Trick: cut the loop!

If $\mathrm{DEF}^{\mathbb{O}}$ is *also* a relation over our starting model...

$$\begin{tikzcd}
\begin{array}{c}\text{model } (\mathbb{O}, \mathrm{W}, d) \text{ with maps} \\ \text{preserving } \mathrm{DEF}^{\mathbb{O}}\end{array} \arrow[r, bend left] & \begin{array}{c}\text{definability relation} \\ \text{over } \mathbb{O}\end{array} \arrow[l, bend left]
\end{tikzcd}$$

The chicken and the egg starting model $(\mathrm{Set}_\kappa, T, s)$

Trick: cut the loop!

If $\mathrm{DEF}^{\mathbb{O}}$ is *also* a relation over our starting model...

$$
\begin{tikzcd}
\text{model } (\mathbb{O}, \mathrm{W}, d) \text{ with maps preserving relations over } \mathrm{Set}_\kappa \arrow[r, bend left] & \text{definability relation over } \mathbb{O} \arrow[l, bend left]
\end{tikzcd}
$$

The chicken and the egg starting model $(\mathrm{Set}_\kappa, T, s)$

Trick: cut the loop!

If $\mathrm{DEF}^{\mathbb{O}}$ is *also* a relation over our starting model...

$$
\begin{tikzcd}[column sep=large, row sep=huge]
\begin{array}{c}\text{model } (\mathbb{O}, \mathrm{W}, d) \text{ with maps}\\ \text{preserving}\\ \text{relations over } \mathrm{Set}_\kappa\end{array} \arrow[r, bend left] & \begin{array}{c}\text{definability relation}\\ \text{over } \mathbb{O}\end{array} \arrow[dl, bend left] \\
\begin{array}{c}\text{model } (\mathbb{O}, \mathrm{W}, d) \text{ with maps}\\ \text{preserving } \mathrm{DEF}^{\mathbb{O}}\end{array} &
\end{tikzcd}
$$

For every $i \in \mathbb{I}$ and fixed $\varphi : \mathbb{I} \to \mathrm{Cart}(\mathrm{Con}^{\mathrm{op}}, \mathrm{Set}_\kappa)$:

$$\begin{tikzcd}
 & \mathsf{Krip}(\mathrm{Set}_\kappa, F_{\varphi i}) \arrow[r] \arrow[d, "\mathrm{proj}"] & \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[r, "\lambda X\,.\,\mathrm{Set}_\kappa(F_{\varphi i}(-)\,,X)"'] \arrow[loop below, "T"] & \widehat{\mathrm{Con}}
\end{tikzcd}$$

Preserving many relations at once [$\mathbb{I}$ some index set]

$$
\begin{tikzcd}
 & \mathbb{L} \arrow[r] \arrow[d, "\mathrm{proj}"] & \prod_{i\in\mathbb{I}} \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[r, "\langle \lambda X \,.\, \mathrm{Set}_\kappa(F_{\varphi i}(-)\,,X)\rangle_{i\in\mathbb{I}}"'] & \prod_{i\in\mathbb{I}} \widehat{\mathrm{Con}}
\end{tikzcd}
$$

Preserving many relations at once [$\mathbb{I}$ some index set]

$$\begin{tikzcd}
 & \mathbb{L} \arrow[r] \arrow[d, "\mathrm{proj}"] & \prod_{i\in\mathbb{I}} \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[r, "\langle \lambda X \,.\, \mathrm{Set}_\kappa(F_{\varphi i}(-)\,, X)\rangle_{i\in\mathbb{I}}"'] \arrow[loop below, "T"] & \prod_{i\in\mathbb{I}} \widehat{\mathrm{Con}}
\end{tikzcd}$$

Object of $\mathbb{L}$:

$(\underline{A}, \overline{A})$ where $\left(\underline{A}, \overline{A}(i)\right) \in \mathsf{Krip}(\mathrm{Set}_\kappa, F_{\varphi i})$

$\overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$ + compatibility with renaming

Preserving many relations at once [$\mathbb{I}$ some index set]

$$
\begin{tikzcd}
 & \mathbb{L} \arrow[r] \arrow[d, "\mathrm{proj}"] & \prod_{i\in\mathbb{I}} \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[r, "\langle \lambda X \,.\, \mathrm{Set}_\kappa(F_{\varphi i}(-)\,,X)\rangle_{i\in\mathbb{I}}"'] \arrow[loop below, "T"] & \prod_{i\in\mathbb{I}} \widehat{\mathrm{Con}}
\end{tikzcd}
$$

Object of $\mathbb{L}$:

$(\underline{A}, \overline{A})$ where $\left(\underline{A}, \overline{A}(i)\right) \in \mathsf{Krip}(\mathrm{Set}_\kappa, F_{\varphi i})$

$\overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$ + compatibility with renaming

$\rightsquigarrow$ Maps in $\mathbb{L}$ are maps in Set_κ that preserve every relation

$$h \in \overline{A}(i) \Rightarrow (f \circ h) \in \overline{B}(i)$$

Preserving many relations at once [$\mathbb{I}$ some index set]

$$\begin{tikzcd}
 & \mathbb{L} \arrow[r] \arrow[d, "\mathrm{proj}"] & \prod_{i\in\mathbb{I}} \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[r, "\langle \lambda X\,.\,\mathrm{Set}_\kappa(F_{\varphi i}(-)\,,X)\rangle_{i\in\mathbb{I}}"'] & \prod_{i\in\mathbb{I}} \widehat{\mathrm{Con}}
\end{tikzcd}$$

Object of $\mathbb{L}$:

$(\underline{A}, \overline{A})$ where $\left(\underline{A}, \overline{A}(i)\right) \in \mathsf{Krip}(\mathrm{Set}_\kappa, F_{\varphi i})$

$\overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$ + compatibility with renaming

$\rightsquigarrow$ Maps in $\mathbb{L}$ are maps in Set_κ that preserve every relation

$\rightsquigarrow$ If we choose $\mathbb{I}$ big enough, maps in $\mathbb{L}$ will preserve $\mathrm{DEF}^{\mathbb{O}}$

How do you build a fully complete model?

Starting data: a model $(\mathbb{C}, T, s)$.

Two goals:

1. Make every morphism definable,
2. $f = g \in (X \Rightarrow Y)$ iff they agree on definable elements.

Two solutions [*c.f.* O'Hearn–Riecke]:

1. Characterise definability as a logical relation, ✓
   and ensure every map preserves such relations ✓
2. Ensure every object is concrete. ✓

How do you build a fully complete model?

Starting data: a model $(\mathbb{C}, T, s)$.

Two goals:

1. Make every morphism definable,
2. $f = g \in (X \Rightarrow Y)$ iff they agree on definable elements.

Two solutions [*c.f.* O'Hearn–Riecke]:

1. Characterise definability as a logical relation, ✓
   and ensure every map preserves such relations ✓
2. Ensure every object is concrete. ✓

$\rightsquigarrow$ Now to put it all together!

O'Hearn–Riecke construction ($\mathbb{I}$ t.b.d.) any model $(\mathrm{Set}_\kappa, T, s)$

$$\begin{tikzcd}
 & \mathbb{L} \arrow[r] \arrow[d, "\mathrm{proj}"] & \prod_{i\in\mathbb{I}} \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathrm{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[r, "\langle \lambda X\,.\,\mathrm{Set}_\kappa(F_{\varphi i}(-)\,,X)\rangle_{i\in\mathbb{I}}"'] & \prod_{i\in\mathbb{I}} \widehat{\mathrm{Con}}
\end{tikzcd}$$

Object of $\mathbb{L}$:

$(\underline{A}, \overline{A})$ where $\left(\underline{A}, \overline{A}(i)\right) \in \mathsf{Krip}(\mathrm{Set}_\kappa, F_{\varphi i})$

$\overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$ + compatibility with renaming

O'Hearn–Riecke construction ($\mathbb{I}$ t.b.d.) any model $(\mathrm{Set}_\kappa, T, s)$

$$
\begin{tikzcd}
\mathbb{O} \arrow[r, hook, "\perp", "\perp"'] & \mathbb{L} \arrow[l, bend right, "\mathrm{K}"'] \arrow[l, bend left, "\mathrm{H}"] \arrow[r] \arrow[d, "\mathrm{proj}"] & \prod_{i\in\mathbb{I}} \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[r, "\langle \lambda X \,.\, \mathrm{Set}_\kappa(F_{\varphi i}(-)\,,X)\rangle_{i\in\mathbb{I}}"'] & \prod_{i\in\mathbb{I}} \widehat{\mathrm{Con}}
\end{tikzcd}
$$

Object of $\mathbb{O}$:

$(\underline{A}, \overline{A})$ where $\left(\underline{A}, \overline{A}(i)\right) \in \mathsf{Krip}(\mathrm{Set}_\kappa, F_{\varphi i})$

$\overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$ + compatibility with renaming

+ concreteness $(\mathrm{Diag}_{\underline{A}} \subseteq \overline{A}(i)(\Gamma)$ for all $\Gamma)$

O'Hearn–Riecke construction ($\mathbb{I}$ t.b.d.) any model $(\mathrm{Set}_\kappa, T, s)$

O'Hearn–Riecke model

$$
\begin{tikzcd}[column sep=large]
 & \mathbb{O} \arrow[rr] \arrow[rr, phantom, shift left=3, "\perp"] \arrow[rr, phantom, shift right=3, "\perp"] & & \mathbb{L} \arrow[ll, bend right=40, "\mathrm{K}"'] \arrow[ll, bend left=40, "\mathrm{H}"] \arrow[r] \arrow[d, "\mathrm{proj}"] & \prod_{i \in \mathbb{I}} \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathrm{Base} \arrow[rrr, "s"'] & & & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[r, "\langle \lambda X\,.\,\mathrm{Set}_\kappa(F_{\varphi i}(-)\,,X)\rangle_{i \in \mathbb{I}}"'] & \prod_{i \in \mathbb{I}} \widehat{\mathrm{Con}}
\end{tikzcd}
$$

Object of $\mathbb{O}$:

$$(\underline{A}, \overline{A}) \text{ where } \left(\underline{A}, \overline{A}(i)\right) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$$

$$\overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A}) + \text{compatibility with renaming}$$

How do we choose $\mathbb{I}$?

How do we choose $\mathbb{I}$?

See what we need as we go along!

How do we choose $\mathbb{I}$?

See what we need as we go along!

ensuring $\mathrm{DEF}^{\mathbb{O}}$ is one of the relations preserved
defining a lifting of T to $\mathbb{L}$
defining an interpretation d in $\mathbb{O}$

Choosing $\mathbb{I}$, part 1

$(\underline{A}, \overline{A}) \in \mathbb{O}$:
$\quad\big(\underline{A}, \overline{A}(i)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$
$\quad\overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$
$\quad$+ compatibility with renaming

We want an interpretation d and $i \in \mathbb{I}$ so that

$$\overline{d[\![\sigma]\!]}(i)(\Gamma) = \mathrm{DEF}^{\mathbb{O}}_{\sigma}(\Gamma)$$

Choosing $\mathbb{I}$, part 1

$(\underline{A}, \overline{A}) \in \mathbb{O}$:

$\quad \big(\underline{A}, \overline{A}(i)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$

$\quad \overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$

$\quad$ + compatibility with renaming

We want an interpretation d and $i \in \mathbb{I}$ so that

$$\begin{aligned}\overline{d[\![\sigma]\!]}(i)(\Gamma) &= \mathrm{DEF}^{\mathbb{O}}_\sigma(\Gamma) \\ &\subseteq \mathbb{O}(d[\![\Gamma]\!], d[\![\sigma]\!])\end{aligned}$$

Choosing $\mathbb{I}$, part 1

$(\underline{A}, \overline{A}) \in \mathbb{O}$:
$\big(\underline{A}, \overline{A}(i)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$
$\overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$
+ compatibility with renaming

We want an interpretation d and $i \in \mathbb{I}$ so that

$$\begin{aligned}\overline{d[\![\sigma]\!]}(i)(\Gamma) &= \mathrm{DEF}^{\mathbb{O}}_{\sigma}(\Gamma) \\ &\subseteq \mathbb{O}(d[\![\Gamma]\!], d[\![\sigma]\!]) \\ &\subseteq \mathrm{Set}_\kappa(\mathrm{U}d[\![\Gamma]\!], \mathrm{U}d[\![\sigma]\!])\end{aligned}$$

where $\mathrm{U} := (\mathbb{O} \hookrightarrow \mathbb{L} \xrightarrow{\mathrm{proj}} \mathrm{Set}_\kappa)$

Choosing $\mathbb{I}$, part 1

$(\underline{A}, \overline{A}) \in \mathbb{O}$:
$\quad (\underline{A}, \overline{A}(i)) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$
$\quad \overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$
$\quad$ + compatibility with renaming

We want an interpretation d and $i \in \mathbb{I}$ so that

$$\begin{aligned} \overline{d[\![\sigma]\!]}(i)(\Gamma) &= \mathrm{DEF}^{\mathbb{O}}_{\sigma}(\Gamma) \\ &\subseteq \mathbb{O}(d[\![\Gamma]\!], d[\![\sigma]\!]) \\ &\subseteq \mathrm{Set}_\kappa(\mathrm{U}d[\![\Gamma]\!], \mathrm{U}d[\![\sigma]\!]) \end{aligned}$$

where $\mathrm{U} := (\mathbb{O} \hookrightarrow \mathbb{L} \xrightarrow{\mathrm{proj}} \mathrm{Set}_\kappa)$

Want: $F_{\varphi i} := (\mathrm{Con}^{\mathrm{op}} \xrightarrow{d[\![-]\!]} \mathbb{O} \xrightarrow{\mathrm{U}} \mathrm{Set}_\kappa)$

Choosing $\mathbb{I}$, part 1

Want: $F_{\varphi i} := (\mathrm{Con}^{\mathrm{op}} \xrightarrow{d[\![-]\!]} \mathbb{O} \xrightarrow{\mathrm{U}} \mathrm{Set}_\kappa)$

Defining $\mathbb{I}$:

$$\mathbb{F} := \mathrm{Cart}(\mathrm{Con}^{\mathrm{op}}, \mathrm{Set}_\kappa) \;\Big|\; \text{cartesian functors into } \mathrm{Set}_\kappa$$

Choosing $\mathbb{I}$, part 2

$$
\begin{tikzcd}[column sep=huge, row sep=large]
\mathbb{O} \arrow[r, hook] \arrow[r, phantom, shift left=2.5ex, "\perp"] \arrow[r, phantom, shift right=2.5ex, "\perp"] & \mathbb{L} \arrow[l, bend right=40, "\mathrm{K}"'] \arrow[l, bend left=40, "\mathrm{H}"] \arrow[r] \arrow[d, "\mathrm{proj}"] & \prod_{i\in\mathbb{I}} \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathrm{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[r, "\langle \lambda X \,.\, \mathrm{Set}_\kappa(F_{\varphi i}(-)\,,X)\rangle_{i\in\mathbb{I}}"'] & \prod_{i\in\mathbb{I}} \widehat{\mathrm{Con}}
\end{tikzcd}
$$

Choosing $\mathbb{I}$, part 2

$$
\begin{tikzcd}
\mathbb{O} \arrow[loop above, "\mathrm{H}\,\hat{T}j"] \arrow[rr, hook] & & \mathbb{L} \arrow[ll, bend right, "\mathrm{K}"'] \arrow[ll, bend left, "\mathrm{H}"] \arrow[ll, phantom, shift right=3, "\perp"] \arrow[ll, phantom, shift left=3, "\perp"] \arrow[loop above, "\text{lifting}\ \ \hat{T}"] \arrow[rr] \arrow[d, "\mathrm{proj}"] & & \prod_{i \in \mathbb{I}} \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[rr, "s"'] & & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[rr, "\langle \lambda X \,.\, \mathrm{Set}_\kappa(F_{\varphi i}(-)\,,X)\rangle_{i \in \mathbb{I}}"'] & & \prod_{i \in \mathbb{I}} \widehat{\mathrm{Con}}
\end{tikzcd}
$$

Choosing $\mathbb{I}$, part 2

$$\begin{tikzcd}
\mathbb{O} \arrow[loop above, "\mathrm{H}\hat{T}j"] \arrow[rr, hook] \arrow[rr, phantom, shift left=3, "\perp"] \arrow[rr, phantom, shift right=3, "\perp"] & & \mathbb{L} \arrow[ll, bend right, "\mathrm{K}"'] \arrow[ll, bend left, "\mathrm{H}"] \arrow[loop above, "\textcolor{red}{\text{lifting}}\ \hat{T}"] \arrow[rr] \arrow[d, "\mathrm{proj}"] & & \prod_{i \in \mathbb{I}} \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[rr, "s"'] & & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[rr, "\langle \lambda X \,.\, \mathrm{Set}_\kappa(F_{\varphi i}(-)\,,X)\rangle_{i \in \mathbb{I}}"'] & & \prod_{i \in \mathbb{I}} \widehat{\mathrm{Con}}
\end{tikzcd}$$

$\hat{T}$ defined by giving a lifting $\hat{T}^{(i)}$ to $\mathsf{Krip}(\mathrm{Set}_\kappa, F_{\varphi i})$ for each $i \in \mathbb{I}$.

Choosing $\mathbb{I}$, part 2

$$
\begin{tikzcd}[column sep=huge, row sep=large]
\mathbb{O} \arrow[loop above, "\mathrm{H}\hat{T}j"] \arrow[r, hook] \arrow[r, phantom, shift left=2.5ex, "\perp"] \arrow[r, phantom, shift right=2.5ex, "\perp"] & \mathbb{L} \arrow[l, bend right=40, "\mathrm{K}"'] \arrow[l, bend left=40, "\mathrm{H}"] \arrow[loop above, "\text{lifting } \hat{T}"] \arrow[r] \arrow[d, "\mathrm{proj}"] & \prod_{i\in\mathbb{I}} \mathrm{Sub}(\widehat{\mathrm{Con}}) \arrow[d, "\mathrm{cod}"] \\
\mathsf{Base} \arrow[r, "s"'] & \mathrm{Set}_\kappa \arrow[loop below, "T"] \arrow[r, "\langle \lambda X \,.\, \mathrm{Set}_\kappa(F_{\varphi i}(-)\,,X)\rangle_{i\in\mathbb{I}}"'] & \prod_{i\in\mathbb{I}} \widehat{\mathrm{Con}}
\end{tikzcd}
$$

$\hat{T}$ defined by giving a lifting $\hat{T}^{(i)}$ to $\mathsf{Krip}(\mathrm{Set}_\kappa, F_{\varphi i})$ for each $i \in \mathbb{I}$.

$$\rightsquigarrow$$

need a lifting parameter for each φi

Defining $\mathbb{I}$:

$\mathbb{F} := \mathrm{Cart}(\mathrm{Con}^{\mathrm{op}}, \mathrm{Set}_\kappa)$	cartesian functors into Set_κ
$\mathcal{K}r(F)$	parameters for $\top\top$-lifting to $\mathsf{Krip}(\mathrm{Set}_\kappa, F)$

Choosing $\mathbb{I}$, part 3

$(\underline{A}, \overline{A}) \in \mathbb{O}$:

$\big(\underline{A}, \overline{A}(i)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$

$\overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$

$+$ compatibility with renaming

Want d : Base $\to \mathbb{O}$ lying over s:

$$d(\beta) = \big(s[\![\beta]\!], \overline{d[\![\beta]\!]}\big)$$

Choosing $\mathbb{I}$, part 3

$(\underline{A}, \overline{A}) \in \mathbb{O}$:

$\quad \big(\underline{A}, \overline{A}(i)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$

$\quad \overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$

$\quad$ + compatibility with renaming

Want $d : \mathsf{Base} \to \mathbb{O}$ lying over s:

$$d(\beta) = \big(s[\![\beta]\!], \overline{d[\![\beta]\!]}\big)$$

So need: $\quad \big(s[\![\beta]\!], \overline{d[\![\beta]\!]}(i)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i}) \quad$ (all i)

Choosing $\mathbb{I}$, part 3

$(\underline{A}, \overline{A}) \in \mathbb{O}$:

$\big(\underline{A}, \overline{A}(i)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$

$\overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$

$+$ compatibility with renaming

Want $d : \mathrm{Base} \to \mathbb{O}$ lying over s:

$$d(\beta) = \big(s[\![\beta]\!], \overline{d[\![\beta]\!]}\big)$$

So need: $\big(s[\![\beta]\!], \overline{d[\![\beta]\!]}(i)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$ (all i)

Trick:

quantify over all $r : \mathrm{Base} \to \widehat{\mathrm{Con}}$ such that
$(s[\![\beta]\!], r(\beta)) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$

Choosing $\mathbb{I}$, part 3

$(\underline{A}, \overline{A}) \in \mathbb{O}$:

$\big(\underline{A}, \overline{A}(i)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$

$\overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$

$+$ compatibility with renaming

Want $d : \mathrm{Base} \to \mathbb{O}$ lying over s:

$$d(\beta) = \big(s[\![\beta]\!], \overline{d[\![\beta]\!]}\big)$$

So need: $\big(s[\![\beta]\!], \overline{d[\![\beta]\!]}(i)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$ (all i)

Trick:

quantify over all $r : \mathrm{Base} \to \widehat{\mathrm{Con}}$ such that

$(s[\![\beta]\!], r(\beta)) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$

Then:

$$\overline{d[\![\beta]\!]}(\ldots, F_{\varphi i}, r) = r(\beta)$$

Defining $\mathbb{I}$

$\mathbb{F} := \mathrm{Cart}(\mathrm{Con}^{\mathrm{op}}, \mathrm{Set}_\kappa)$	cartesian functors into Set_κ
$\mathcal{K}r(F)$	parameters for $\top\top$-lifting to $\mathsf{Krip}(\mathrm{Set}_\kappa, F)$
$\mathcal{I}nt(F)$	$r : \mathsf{Base} \to \widehat{\mathrm{Con}}$ such that $(s[\![\beta]\!], r(\beta)) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F)$ for all $\beta \in \mathsf{Base}$

Defining $\mathbb{I}$

$\mathbb{F} := \mathrm{Cart}(\mathrm{Con}^{\mathrm{op}}, \mathrm{Set}_\kappa)$	cartesian functors into Set_κ
$\mathcal{K}r(F)$	parameters for $\top\top$-lifting to $\mathsf{Krip}(\mathrm{Set}_\kappa, F)$
$\mathcal{I}nt(F)$	$r : \mathsf{Base} \to \widehat{\mathrm{Con}}$ such that $(s[\![\beta]\!], r(\beta)) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F)$ for all $\beta \in \mathsf{Base}$

$$\mathbb{I} := \left\{(F, K, r) \,\middle|\, F \in \mathbb{F}, K \in \mathcal{K}r(F), r \in \mathcal{I}nt(F)\right\}$$
$$\varphi : (F, K, r) \mapsto F$$

Defining $\mathbb{I}$

$\mathbb{F} := \mathrm{Cart}(\mathrm{Con}^{\mathrm{op}}, \mathrm{Set}_\kappa)$	cartesian functors into Set_κ
$\mathcal{K}r(F)$	parameters for $\top\top$-lifting to $\mathsf{Krip}(\mathrm{Set}_\kappa, F)$
$\mathcal{I}nt(F)$	$r : \mathsf{Base} \to \widehat{\mathrm{Con}}$ such that $(s[\![\beta]\!], r(\beta)) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F)$ for all $\beta \in \mathsf{Base}$

$$\mathbb{I} := \left\{(F, K, r) \,\middle|\, F \in \mathbb{F}, K \in \mathcal{K}r(F), r \in \mathcal{I}nt(F)\right\}$$

$$\varphi : (F, K, r) \mapsto F$$

$$\overline{d[\![\beta]\!]}(F, K, r) = r(\beta)$$

$(\underline{A}, \overline{A}) \in \mathbb{O}$:

$\big(\underline{A}, \overline{A}(i)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$

$\overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$

+ compatibility with renaming

$$\overline{d[\![\sigma]\!]}(F, K, r)(\Gamma) \subseteq \mathrm{Set}_\kappa(F\Gamma, \mathrm{U}d[\![\sigma]\!])$$

$(\underline{A}, \overline{A}) \in \mathbb{O}$:

$\big(\underline{A}, \overline{A}(i)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$

$\overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$

$+$ compatibility with renaming

$$\overline{d[\![\sigma]\!]}(F, K, r)(\Gamma) \subseteq \mathrm{Set}_\kappa(F\Gamma, \mathrm{U}d[\![\sigma]\!])$$

$$\rightsquigarrow$$

$$\overline{d[\![\sigma]\!]}(\mathrm{U}d[\![-]\!], K, r)(\Gamma) \subseteq \mathrm{Set}_\kappa(\mathrm{U}d[\![\Gamma]\!], \mathrm{U}d[\![\sigma]\!])$$

$(\underline{A}, \overline{A}) \in \mathbb{O}$:

$\quad (\underline{A}, \overline{A}(i)) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F_{\varphi i})$

$\quad \overline{A}(i)(\Gamma) \subseteq \mathrm{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A})$

$\quad$ + compatibility with renaming

$$\overline{d[\![\sigma]\!]}(F, K, r)(\Gamma) \subseteq \mathrm{Set}_\kappa(F\Gamma, \mathrm{U}d[\![\sigma]\!])$$

$$\rightsquigarrow$$

$$\overline{d[\![\sigma]\!]}(\mathrm{U}d[\![-]\!], K, r)(\Gamma) \subseteq \mathrm{Set}_\kappa(\mathrm{U}d[\![\Gamma]\!], \mathrm{U}d[\![\sigma]\!])$$

$$\begin{aligned} \mathrm{DEF}^{\mathbb{O}}_\sigma(\Gamma) &\subseteq \mathbb{O}(d[\![\Gamma]\!], d[\![\sigma]\!]) \\ &\subseteq \mathrm{Set}_\kappa(\mathrm{U}d[\![\Gamma]\!], \mathrm{U}d[\![\sigma]\!]) \end{aligned}$$

$$\overline{d\llbracket\sigma\rrbracket}(F, K, r)(\Gamma) \subseteq \mathrm{Set}_\kappa(F\Gamma, \mathrm{U}d\llbracket\sigma\rrbracket)$$

$$\overline{d\llbracket\sigma\rrbracket}({\color{red}\mathrm{U}d\llbracket - \rrbracket}, K, r)(\Gamma) \subseteq \mathrm{Set}_\kappa({\color{red}\mathrm{U}d\llbracket\Gamma\rrbracket}, \mathrm{U}d\llbracket\sigma\rrbracket)$$

$$\begin{aligned} \mathrm{DEF}^{\mathbb{O}}_\sigma(\Gamma) &\subseteq \mathbb{O}(d\llbracket\Gamma\rrbracket, d\llbracket\sigma\rrbracket) \\ &\subseteq \mathrm{Set}_\kappa(\mathrm{U}d\llbracket\Gamma\rrbracket, \mathrm{U}d\llbracket\sigma\rrbracket) \end{aligned}$$

Key property of $(\mathbb{O}, \mathrm{H}\hat{T}j, d)$

There exists $(F, K, r) \in \mathbb{I}$ such that

$$\overline{d\llbracket\sigma\rrbracket}(F, K, r) = \mathrm{DEF}^{\mathbb{O}}_\sigma$$

for all $\sigma \in \mathrm{Type}$.

$(\underline{A}, \overline{A}) \in \mathbb{O}$:	$\big(\underline{A}, \overline{A}(F, K, r)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F)$ $\overline{A}(F, K, r)(\Gamma) \subseteq \mathrm{Set}_\kappa(F\Gamma, \underline{A})$ + compatibility with renaming
$d : \mathrm{Base} \to \mathbb{O}$	$d(\beta) = (s[\![\beta]\!], \overline{d[\![\beta]\!]})$ $\overline{d[\![\beta]\!]}(F, K, r) = r(\beta)$

$(\underline{A}, \overline{A}) \in \mathbb{O}$:	$\big(\underline{A}, \overline{A}(F, K, r)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F)$ $\overline{A}(F, K, r)(\Gamma) \subseteq \mathrm{Set}_\kappa(F\Gamma, \underline{A})$ + compatibility with renaming
$d : \mathsf{Base} \to \mathbb{O}$	$d(\beta) = (s[\![\beta]\!], \overline{d[\![\beta]\!]})$ $\overline{d[\![\beta]\!]}(F, K, r) = r(\beta)$

Since $\mathrm{DEF}^{\mathbb{O}}_{\beta}(\Gamma) \subseteq \mathbb{O}(d[\![\Gamma]\!], d[\![\sigma]\!]) \subseteq \mathrm{Set}_\kappa(\mathrm{U}d[\![\Gamma]\!], \mathrm{U}d[\![\sigma]\!])$:

$(\underline{A}, \overline{A}) \in \mathbb{O}$:	$\big(\underline{A}, \overline{A}(F, K, r)\big) \in \mathrm{Conc}(\mathrm{Set}_\kappa, F)$ $\overline{A}(F, K, r)(\Gamma) \subseteq \mathrm{Set}_\kappa(F\Gamma, \underline{A})$ + compatibility with renaming
$d : \mathsf{Base} \to \mathbb{O}$	$d(\beta) = (s[\![\beta]\!], \overline{d[\![\beta]\!]})$ $\overline{d[\![\beta]\!]}(F, K, r) = r(\beta)$

Since $\mathrm{DEF}^{\mathbb{O}}_\beta(\Gamma) \subseteq \mathbb{O}(d[\![\Gamma]\!], d[\![\sigma]\!]) \subseteq \mathrm{Set}_\kappa(\mathrm{U}d[\![\Gamma]\!], \mathrm{U}d[\![\sigma]\!])$:

$$\overline{d[\![\beta]\!]}(\mathrm{U}d[\![-]\!], K, \lambda\beta\,.\,\mathrm{DEF}^{\mathbb{O}}_\beta) = \mathrm{DEF}^{\mathbb{O}}_\beta$$

Key property

If $(s[\![\beta]\!], \mathrm{DEF}^{\mathbb{O}}_{\beta}) \in \mathrm{Conc}(\mathrm{Set}_{\kappa}, \mathrm{U}d[\![-]\!])$ for all $\beta \in$ Base, then

$$\overline{d[\![\sigma]\!]}(\mathrm{U}d[\![-]\!], K, \lambda\beta \,.\, \mathrm{DEF}^{\mathbb{O}}_{\beta}) = \mathrm{DEF}^{\mathbb{O}}_{\sigma}$$

for all $\sigma \in \mathrm{Type}$.

Key property

If $(s[\![\beta]\!], \mathrm{DEF}^{\mathbb{O}}_{\beta}) \in \mathrm{Conc}(\mathrm{Set}_{\kappa}, \mathrm{U}d[\![-]\!])$ for all $\beta \in$ Base, then

$$\overline{d[\![\sigma]\!]}(\mathrm{U}d[\![-]\!], K, \lambda\beta \,.\, \mathrm{DEF}^{\mathbb{O}}_{\beta}) = \mathrm{DEF}^{\mathbb{O}}_{\sigma}$$

for all $\sigma \in \mathrm{Type}$.

$\rightsquigarrow$ concreteness needed for exponentials and monadic types

Key property

If $(s[\![\beta]\!], \mathrm{DEF}^{\mathbb{O}}_{\beta}) \in \mathrm{Conc}(\mathrm{Set}_{\kappa}, \mathrm{U}d[\![-]\!])$ for all $\beta \in$ Base, then

$$\overline{d[\![\sigma]\!]}(\mathrm{U}d[\![-]\!], K, \lambda\beta \,.\, \mathrm{DEF}^{\mathbb{O}}_{\beta}) = \mathrm{DEF}^{\mathbb{O}}_{\sigma}$$

for all $\sigma \in \mathrm{Type}$.

From the Key Property to full completeness:

if $f : d[\![\Gamma]\!] \to d[\![\sigma]\!]$, then

Key property

If $(s[\![\beta]\!], \mathrm{DEF}^{\mathbb{O}}_{\beta}) \in \mathrm{Conc}(\mathrm{Set}_{\kappa}, \mathrm{U}d[\![-]\!])$ for all $\beta \in$ Base, then

$$\overline{d[\![\sigma]\!]}(\mathrm{U}d[\![-]\!], K, \lambda\beta\,.\,\mathrm{DEF}^{\mathbb{O}}_{\beta}) = \mathrm{DEF}^{\mathbb{O}}_{\sigma}$$

for all $\sigma \in \mathrm{Type}$.

From the Key Property to full completeness:

if $f : d[\![\Gamma]\!] \to d[\![\sigma]\!]$, then

$h \in \overline{d[\![\Gamma]\!]}(F, K, r)(\Gamma) \Rightarrow f \circ h \in \overline{d[\![\sigma]\!]}(F, K, r)(\Gamma)$

Key property

If $(s[\![\beta]\!], \mathrm{DEF}^{\mathbb{O}}_{\beta}) \in \mathrm{Conc}(\mathrm{Set}_{\kappa}, \mathrm{U}d[\![-]\!])$ for all $\beta \in$ Base, then

$$\overline{d[\![\sigma]\!]}(\mathrm{U}d[\![-]\!], K, \lambda\beta \,.\, \mathrm{DEF}^{\mathbb{O}}_{\beta}) = \mathrm{DEF}^{\mathbb{O}}_{\sigma}$$

for all $\sigma \in \mathrm{Type}$.

From the Key Property to full completeness:

if $f : d[\![\Gamma]\!] \to d[\![\sigma]\!]$, then

$h \in \mathrm{DEF}_{\Gamma}(\Gamma) \Rightarrow f \circ h \in \mathrm{DEF}_{\sigma}(\Gamma)$

Key property

If $(s[\![\beta]\!], \mathrm{DEF}^{\mathbb{O}}_{\beta}) \in \mathrm{Conc}(\mathrm{Set}_{\kappa}, \mathrm{U}d[\![-]\!])$ for all $\beta \in$ Base, then

$$\overline{d[\![\sigma]\!]}(\mathrm{U}d[\![-]\!], K, \lambda\beta \,.\, \mathrm{DEF}^{\mathbb{O}}_{\beta}) = \mathrm{DEF}^{\mathbb{O}}_{\sigma}$$

for all $\sigma \in \mathrm{Type}$.

From the Key Property to full completeness:

if $f : d[\![\Gamma]\!] \to d[\![\sigma]\!]$, then

$h \in \mathrm{DEF}_{\Gamma}(\Gamma) \Rightarrow f \circ h \in \mathrm{DEF}_{\sigma}(\Gamma)$

Hence $f \in \mathrm{DEF}_{\sigma}(\Gamma)$

Key property

If $(s[\![\beta]\!], \mathrm{DEF}^{\mathbb{O}}_{\beta}) \in \mathrm{Conc}(\mathrm{Set}_{\kappa}, \mathrm{U}d[\![-]\!])$ for all $\beta \in$ Base, then

$$\overline{d[\![\sigma]\!]}(\mathrm{U}d[\![-]\!], K, \lambda\beta\,.\,\mathrm{DEF}^{\mathbb{O}}_{\beta}) = \mathrm{DEF}^{\mathbb{O}}_{\sigma}$$

for all $\sigma \in \mathrm{Type}$.

Theorem $(\mathrm{Set}_{\kappa}, T, s)$ any model

If $(s[\![\beta]\!], \mathrm{DEF}^{\mathbb{O}}_{\beta}) \in \mathrm{Conc}(\mathrm{Set}_{\kappa}, \mathrm{U}d[\![-]\!])$ for all $\beta \in$ Base, then

$$(\mathbb{O}, \mathrm{H}\,\hat{T}j, d)$$

is fully complete and extensional, hence fully abstract.

Summary and future work

Summary and future work

1. Full completeness + extensionality $\Rightarrow$ full abstraction

Summary and future work

1. Full completeness + extensionality $\Rightarrow$ full abstraction
2. Definability is a logical relation;
   if f preserves DEF, then f is definable

Summary and future work

1. Full completeness + extensionality $\Rightarrow$ full abstraction
2. Definability is a logical relation;
   if f preserves DEF, then f is definable
3. Maps in O'Hearn–Riecke model $\mathbb{O}$ preserve
   enough relations

Summary and future work

1. Full completeness + extensionality $\Rightarrow$ full abstraction
2. Definability is a logical relation;
   if f preserves DEF, then f is definable
3. Maps in O'Hearn–Riecke model $\mathbb{O}$ preserve
   enough relations

Extends to λ_{c} over a general model $(\mathbb{C}, T, s)$

[modulo assumptions]

Summary and future work

1. Full completeness + extensionality $\Rightarrow$ full abstraction
2. Definability is a logical relation;
   if f preserves DEF, then f is definable
3. Maps in O'Hearn–Riecke model $\mathbb{O}$ preserve
   enough relations

Extends to λ_{c} over a general model $(\mathbb{C}, T, s)$

[modulo assumptions]

Still to do

1. Weakening assumptions: extensionality, hull functor H, ...
2. Checking examples: esp. presheaf models (names, QBS, ...)
3. Richer language: sum types, effect operations, primitives, ...