

## Full abstraction à la O'Hearn–Riecke

Ohad Kammar<sup>†</sup> and Shin-ya Katsumata\* and Philip Saville<sup>†</sup>

<sup>†</sup>School of Informatics  
University of Edinburgh

\*National Institute of Informatics  
Tokyo

An apology / warning



Full abstraction



operational semantics  
coincides with  
denotational semantics

Full abstraction



operational semantics  
coincides with  
denotational semantics

$\Gamma \vdash M \simeq_{\text{ctx}} M' : \sigma$



$\mathcal{C}[M] \Downarrow V \iff \mathcal{C}[M'] \Downarrow V$   
 $\mathcal{C}[-]$  any closed ground context

Full abstraction



operational semantics  
coincides with  
denotational semantics

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

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

Adequacy:  $\llbracket M \rrbracket = \llbracket M' \rrbracket \Rightarrow M \simeq_{\text{ctx}} M'$

Full abstraction:  $M \simeq_{\text{ctx}} M' \Rightarrow \llbracket M \rrbracket = \llbracket M' \rrbracket$

Full abstraction



operational semantics  
coincides with  
denotational semantics

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

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

**Adequacy:**  $\llbracket M \rrbracket = \llbracket M' \rrbracket \Rightarrow M \simeq_{\text{ctx}} M'$

**Full abstraction:**  $M \simeq_{\text{ctx}} M' \Rightarrow \llbracket M \rrbracket = \llbracket M' \rrbracket$

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

Full abstraction



operational semantics  
coincides with  
denotational semantics

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

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

**Adequacy:**  $\llbracket M \rrbracket = \llbracket M' \rrbracket \Rightarrow M \simeq_{\text{ctx}} M'$

**Full abstraction:**  $M \simeq_{\text{ctx}} M' \Rightarrow \llbracket M \rrbracket = \llbracket M' \rrbracket$

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

Problem  model is too rich

Full abstraction



operational semantics  
coincides with  
denotational semantics


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

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

**Adequacy:**  $\llbracket M \rrbracket = \llbracket M' \rrbracket \Rightarrow M \simeq_{\text{ctx}} M'$

**Full abstraction:**  $M \simeq_{\text{ctx}} M' \Rightarrow \llbracket M \rrbracket = \llbracket M' \rrbracket$

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

Problem  model is too rich, particularly at higher types:

$f = g$  on definable elements does not imply  $f = g$



Full abstraction



operational semantics  
coincides with  
denotational semantics


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

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

Adequacy:  $\llbracket M \rrbracket = \llbracket M' \rrbracket \Rightarrow M \simeq_{\text{ctx}} M'$

Full abstraction:  $M \simeq_{\text{ctx}} M' \Rightarrow \llbracket M \rrbracket = \llbracket M' \rrbracket$

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

Problem  model is too rich, particularly at higher types:

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

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

## Moggi's monadic metalanguage

types ::=  $\beta \in \mathbf{Base} \mid 1 \mid \sigma_1 * \sigma_2 \mid \sigma \rightarrow \tau \mid T\sigma$

terms ::=  $x$

$\mid ()$

$\mid \pi_i(M) \mid \langle M_1, M_2 \rangle$

$\mid \lambda x : \sigma . M \mid M N$

$\mid \mathbf{return}(M) \mid \mathbf{let } x^\sigma \mathbf{ be } N \mathbf{ in } M$

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

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

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

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

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

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

### Contextual equivalence, semantically

---

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

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

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

### Contextual equivalence, semantically

---

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

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

 for PCF, coincides with syntactic definition

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

### Contextual equivalence, semantically

---

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

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

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

$$M \simeq_{\text{ctx}} M' \Rightarrow s[M] = s[M']$$



Full completeness + extensionality



full abstraction

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

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

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

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

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

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

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

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

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

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

---

**Every extensional, fully complete model is fully abstract**

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

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

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

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

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

**Every extensional, fully complete model is fully abstract**

---

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

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

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

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

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

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

## Every extensional, fully complete model is fully abstract

---

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

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

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

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

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

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

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

## Every extensional, fully complete model is fully abstract

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

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

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

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

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

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

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

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

## Every extensional, fully complete model is fully abstract

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

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

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  $(\mathbb{C}, \mathcal{T}, s)$ .



# 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  $\lambda_{m1}$ -definable.

## The definability problem

For any model  $(\mathbb{C}, T, s)$ ,  
characterise which maps which are  $\lambda_{\text{ml}}$ -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 $\text{Krip}(\mathbb{C}, F)$ of Kripke relations [Jung & Tiuryn, Alimohamed]



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

data:

- CCC  $\mathbb{C}$
- $F : \text{Con}^{\text{op}} \rightarrow \mathbb{C}$  cartesian

category of contexts  
and renamings

"semantic  
interpretation"



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

data:

- CCC  $\mathbb{C}$
- $F : \text{Con}^{\text{op}} \rightarrow \mathbb{C}$  cartesian

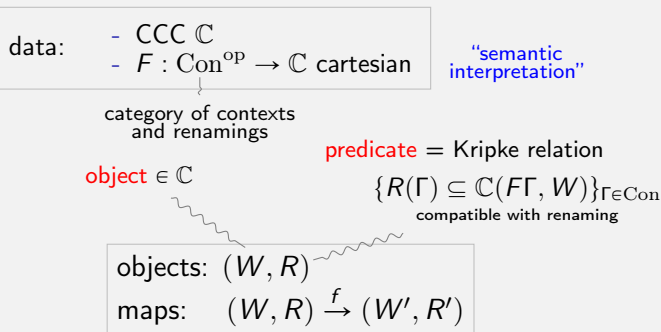
“semantic interpretation”

category of contexts  
and renamings

objects:  $(W, R)$

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

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



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

data:   
 - CCC  $\mathbb{C}$    
 -  $F : \text{Con}^{\text{op}} \rightarrow \mathbb{C}$  cartesian

"semantic interpretation"

category of contexts and renamings

object  $\in \mathbb{C}$

predicate = Kripke relation

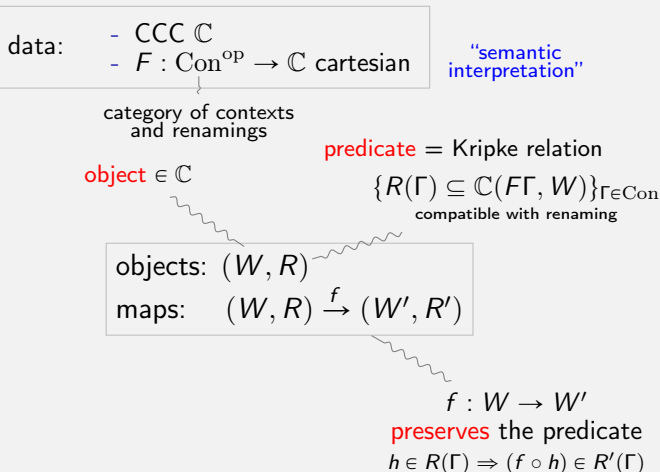
$\{R(\Gamma) \subseteq \mathbb{C}(F\Gamma, W)\}_{\Gamma \in \text{Con}}$   
compatible with renaming

objects:  $(W, R)$

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

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

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



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

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

data:    - CCC  $\mathbb{C}$   
          -  $F : \text{Con}^{\text{op}} \rightarrow \mathbb{C}$  cartesian

object  $\in \mathbb{C}$

predicate = Kripke relation

$\{R(\Gamma) \subseteq \mathbb{C}(F\Gamma, W)\}_{\Gamma \in \text{Con}}$   
compatible with renaming

objects:  $(W, R)$

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

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

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

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

data: 

- CCC  $\mathbb{C}$
- $F : \text{Con}^{\text{op}} \rightarrow \mathbb{C}$  cartesian

object  $\in \mathbb{C}$

predicate = Kripke relation

$\{R(\Gamma) \subseteq \mathbb{C}(F\Gamma, W)\}_{\Gamma \in \text{Con}}$   
compatible with renaming

objects:  $(W, R)$

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

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

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

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

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

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

$$\begin{array}{ccc} \text{Krip}(\mathbb{C}, F) & \longrightarrow & \text{Sub}(\widehat{\text{Con}}) \\ \downarrow \text{proj} & & \downarrow \text{cod} \\ \text{Base} \xrightarrow{s} \mathbb{C} & \xrightarrow{\lambda X . \mathbb{C}(F(-), X)} & \widehat{\text{Con}} \end{array}$$

strictly preserves CCC

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

data:    - CCC  $\mathbb{C}$   
         -  $s[-]$  :  $\text{Con}^{\text{op}} \rightarrow \mathbb{C}$  semantic interpretation

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

definability predicate

$\{\text{DEF}_{\sigma}^{\mathbb{C}}(\Gamma) \subseteq \mathbb{C}(s[\Gamma], s[\sigma])\}_{\Gamma \in \text{Con}}$   
compatible with renaming

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

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



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

data:    - CCC  $\mathbb{C}$   
         -  $s[-]$  :  $\text{Con}^{\text{op}} \rightarrow \mathbb{C}$  semantic interpretation

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

definability predicate

$\{\text{DEF}_{\sigma}^{\mathbb{C}}(\Gamma) \subseteq \mathbb{C}(s[\Gamma], s[\sigma])\}_{\Gamma \in \text{Con}}$   
compatible with renaming

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

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

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

data:    - CCC  $\mathbb{C}$   
         -  $s[-]$  :  $\text{Con}^{\text{op}} \rightarrow \mathbb{C}$  semantic interpretation

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

definability predicate

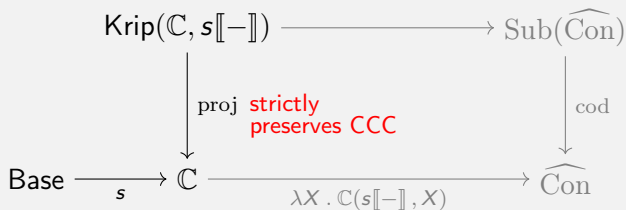
$\{\text{DEF}_\sigma^{\mathbb{C}}(\Gamma) \subseteq \mathbb{C}(s[\Gamma], s[\sigma])\}_{\Gamma \in \text{Con}}$   
compatible with renaming

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

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

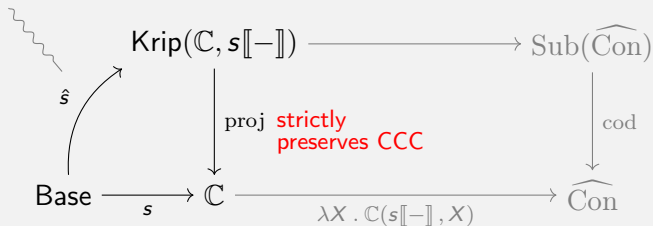
how do we characterise the maps in  $\mathbb{C}$  that are definable?

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



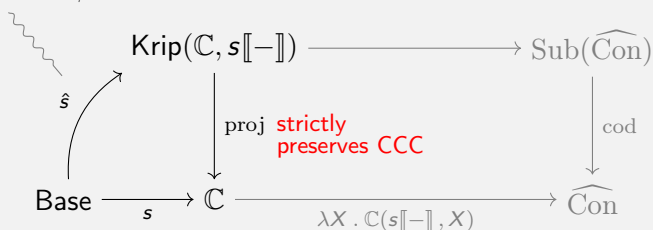
# Characterising STLC-definable maps [Jung & Tiuryn, Alimohamed]

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



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

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

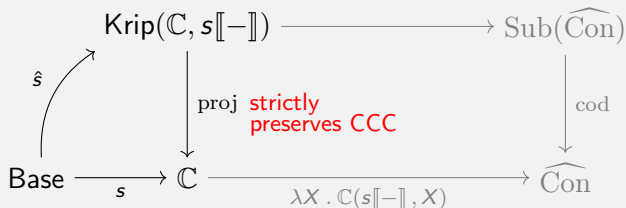


$\hat{s}[\sigma] = (s[\sigma], \text{DEF}_\sigma^{\mathbb{C}})$  for every STLC-type  $\sigma$ .



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

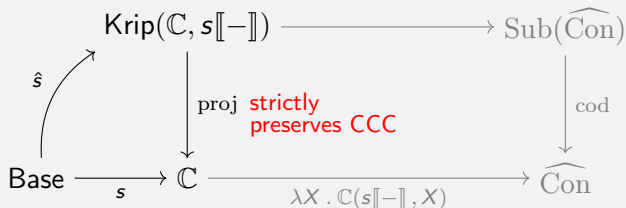
# Characterising STLC-definable maps [Jung & Tiuryn, Alimohamed]



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

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

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

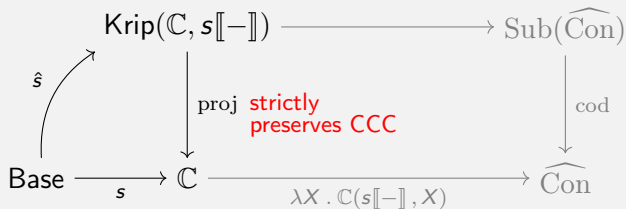


$\hat{s}[\sigma] = (s[\sigma], \text{DEF}_\sigma^{\mathbb{C}})$  for every STLC-type  $\sigma$ .

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

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

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



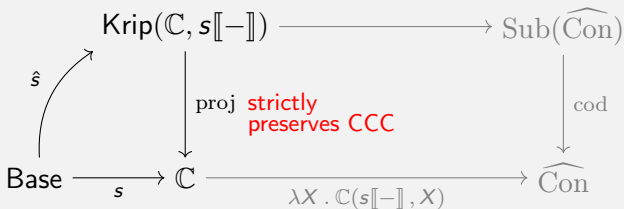
$\hat{s}[\sigma] = (s[\sigma], \text{DEF}_{\sigma}^{\mathbb{C}})$  for every STLC-type  $\sigma$ .

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

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



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



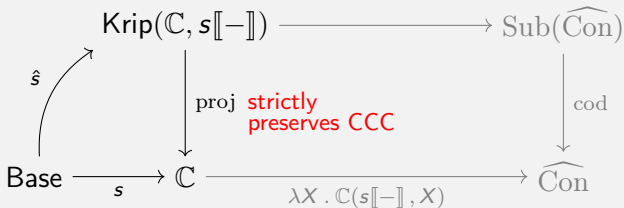
$\widehat{s}[\sigma] = (s[\sigma], \text{DEF}_{\sigma}^{\mathbb{C}})$  for every STLC-type  $\sigma$ .

$f : s[\Gamma] \rightarrow s[\sigma]$  is definable  $\iff f : \widehat{s}[\Gamma] \rightarrow \widehat{s}[\sigma]$

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

$\text{id}_{[\Gamma]}$  is definable  $\longmapsto f \circ \text{id}_{[\Gamma]} \in \text{DEF}_{\sigma}^{\mathbb{C}}(\Gamma)$

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



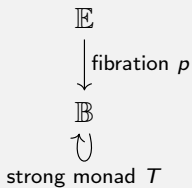
$\hat{s}[\sigma] = (s[\sigma], \text{DEF}_\sigma^{\mathbb{C}})$  for every STLC-type  $\sigma$ .

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

What about monadic types?

What about monadic types?

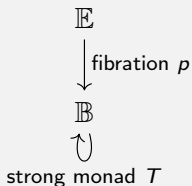
**Diversion:** lifting monads up fibrations



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

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

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

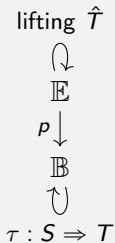


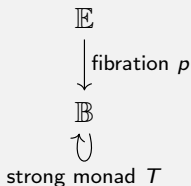
A **lifting** of  $T$  is a monad  $\hat{T}$  on  $\mathbb{E}$  such that

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

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

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



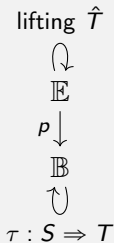


A **lifting** of  $T$  is a monad  $\hat{T}$  on  $\mathbb{E}$  such that

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

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

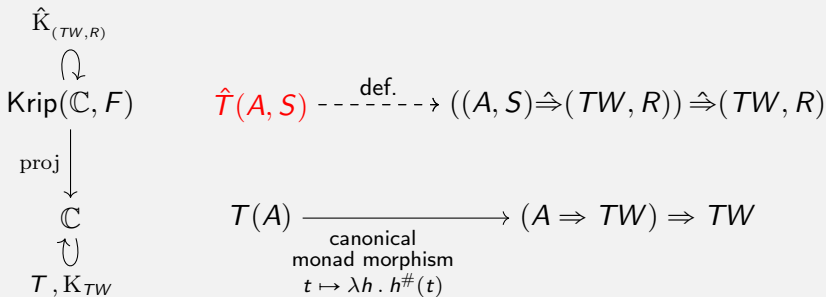
**TT-lifting** [c.f. Katsumata]  $\rightsquigarrow$  defines  $\hat{S}$  from  $\hat{T}$  and  $\tau : S \Rightarrow T$



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

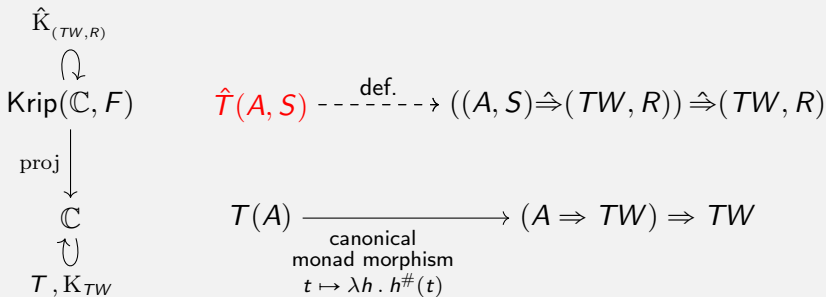
$$S(pX) \xrightarrow{\tau_{pX}} T(pX) \quad \text{monad morphism } \tau : S \Rightarrow T$$

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





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



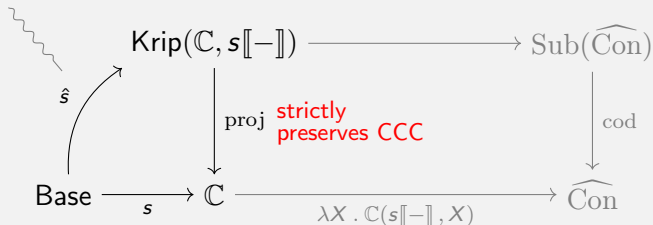
Omitting varying arity, for binary relations:

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

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

# Characterising $\lambda_{ml}$ -definable maps

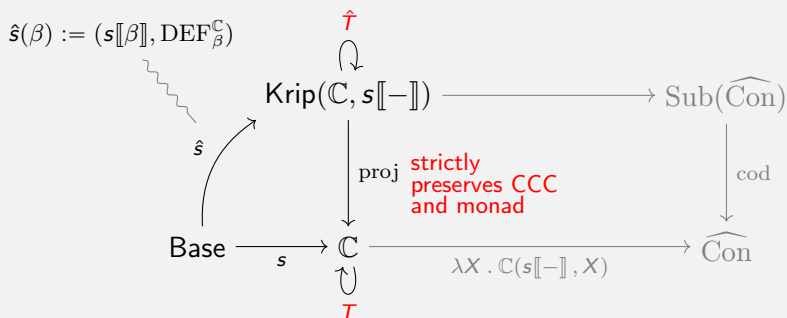
$$\hat{s}(\beta) := (s[\beta], \text{DEF}_{\beta}^{\mathbb{C}})$$



---

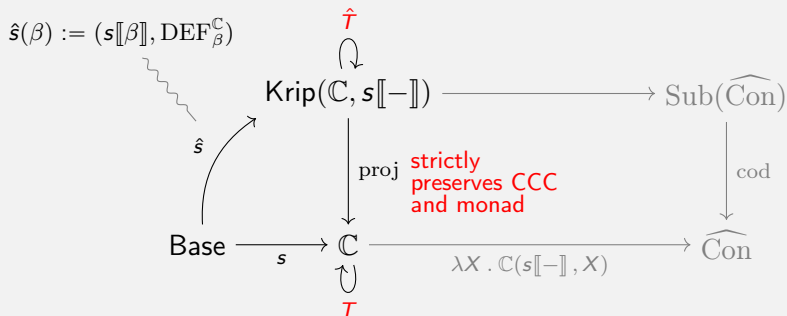
$$\hat{s}[\sigma] = (s[\sigma], \text{DEF}_{\sigma}^{\mathbb{C}}) \text{ for every STLC-type } \sigma$$

# Characterising $\lambda_{ml}$ -definable maps



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

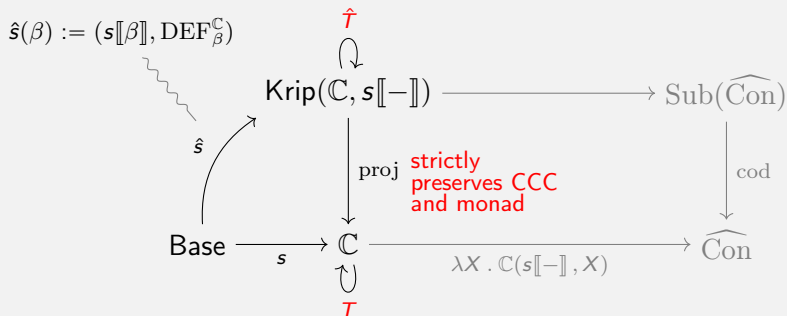
# Characterising $\lambda_{ml}$ -definable maps



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

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

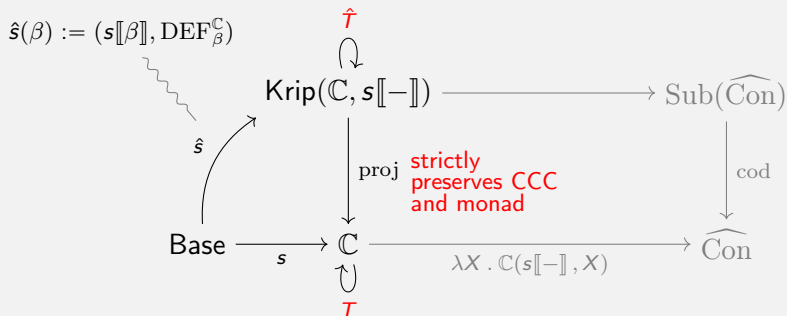
# Characterising $\lambda_{ml}$ -definable maps



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

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

# Characterising $\lambda_{ml}$ -definable maps



---


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

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

# 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,  $\checkmark$   
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.

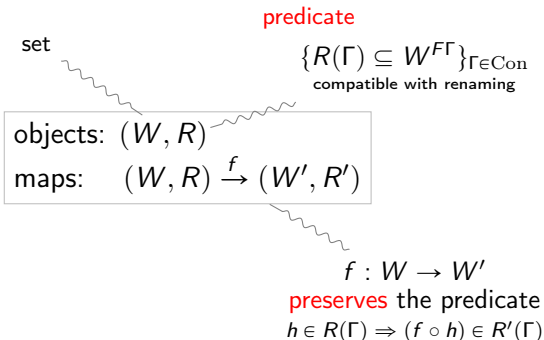
Two solutions [c.f. O'Hearn–Riecke]:

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

$\rightsquigarrow$  For simplicity, over  $\text{Set}_\kappa \leftrightarrow \text{Set}$  [e.g. hereditarily- $\kappa$  sets]

# Kripke relations over $\text{Set}_\kappa$

data:  $F : \text{Con}^{\text{op}} \rightarrow \text{Set}_\kappa$  cartesian



## Concreteness [c.f. O'Hearn–Riecke]

---

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

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

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

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

## Concreteness [c.f. O'Hearn–Riecke]

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

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

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

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

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

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

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

## Concreteness [c.f. O'Hearn–Riecke]

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

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

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

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

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

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

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

↪ can detect if  $f = g$  within  $R, S$  [c.f. [sequentiality](#)]

## Concreteness [c.f. O'Hearn–Riecke]

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

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

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

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

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

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

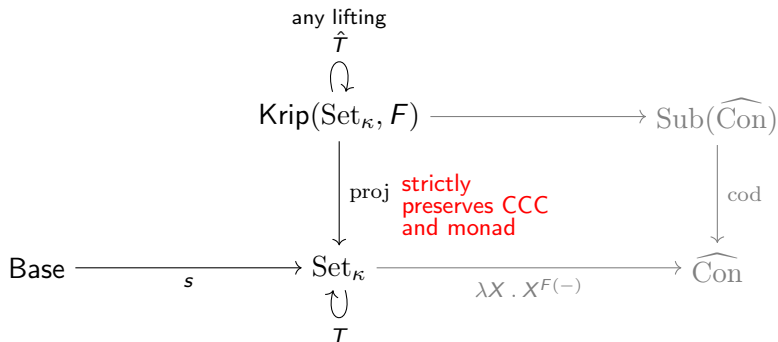
$$(x_1, \dots, x_n) \in R(\Gamma) \subseteq W^{F\Gamma} \Rightarrow (fx_1, \dots, fx_n) \in S(\Gamma) \subseteq X^{F\Gamma}$$

↪ can detect if  $f = g$  within  $R, S$  [c.f. [sequentiality](#)]

↪ full subcategory of concrete objects

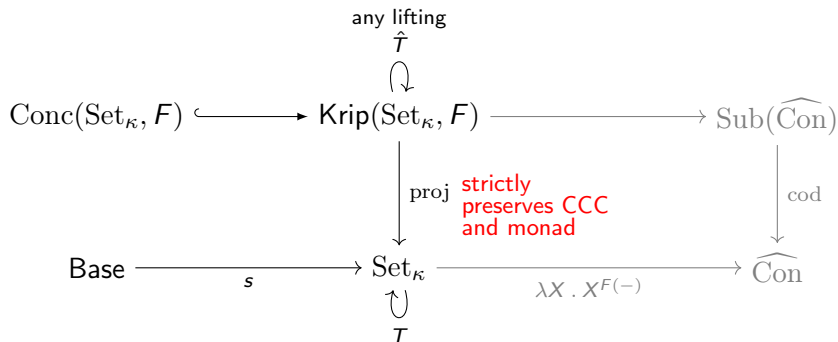
$$j : \text{Conc}(\text{Set}_\kappa, F) \hookrightarrow \text{Krip}(\text{Set}_\kappa, F)$$

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

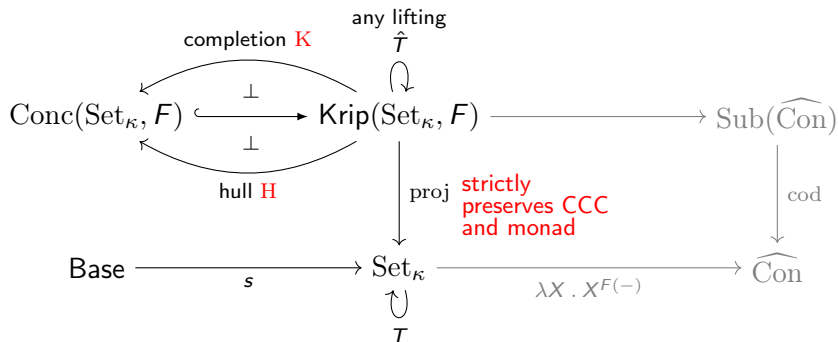




# Making $\text{Conc}(\text{Set}_\kappa, F)$ a model



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

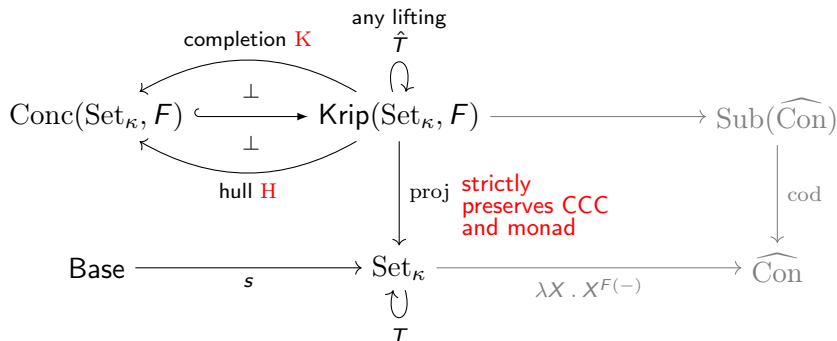


For  $(W, R) \in \text{Krip}(\text{Set}_\kappa, F)$ :

$K \rightsquigarrow$  union each  $R\Gamma$  with the diagonal

$H \rightsquigarrow$  restrict to subset of concrete elements

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



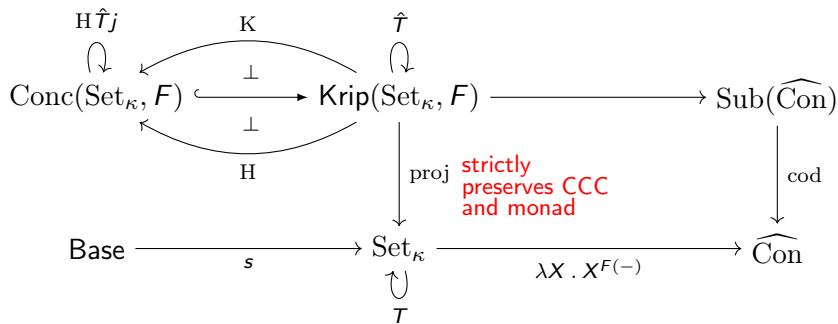
For  $(W, R) \in \text{Krip}(\text{Set}_\kappa, F)$ :

$K \rightsquigarrow$  union each  $R\Gamma$  with the diagonal

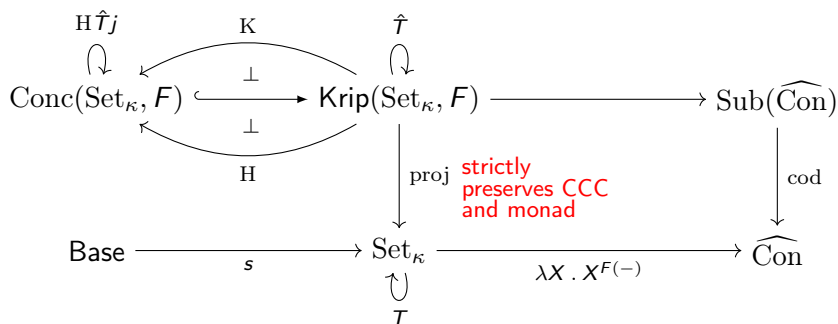
$H \rightsquigarrow$  restrict to subset of concrete elements

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

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



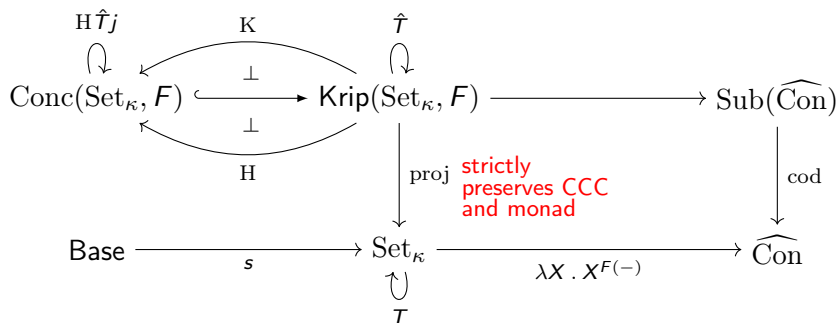
## Making $\text{Conc}(\text{Set}_\kappa, F)$ a model



By abstract nonsense:

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

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



By abstract nonsense:

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

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

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

Starting from:

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

Starting from:

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

**Then:**

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



Starting from:

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

**Then:**

1.  $\text{Conc}(\text{Set}_\kappa, F)$  becomes a CCC with a strong monad  $H\hat{T}_j$ ,
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  $(\mathbb{C}, T, s)$ .

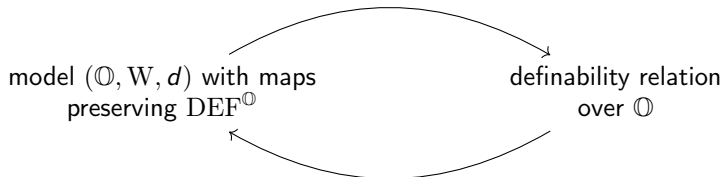
Two goals:

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]:

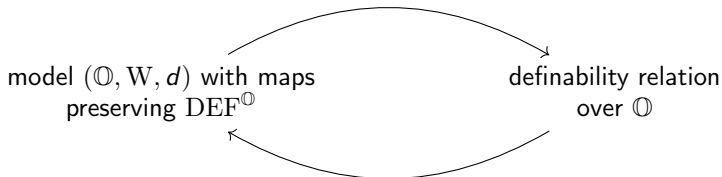
1. Characterise definability as a logical relation, ✓  
and ensure every map preserves such relations
2. Ensure every object is concrete. ✓

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



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

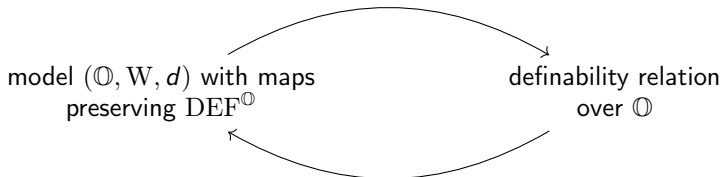
Trick: cut the loop!



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

Trick: cut the loop!

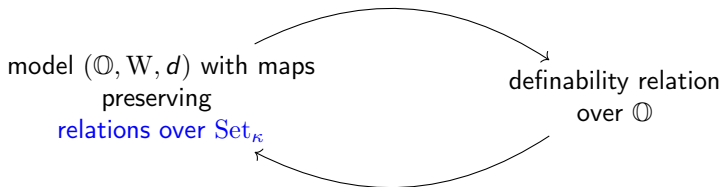
If  $\text{DEF}^\mathbb{O}$  is *also* a relation over our starting model...



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

Trick: cut the loop!

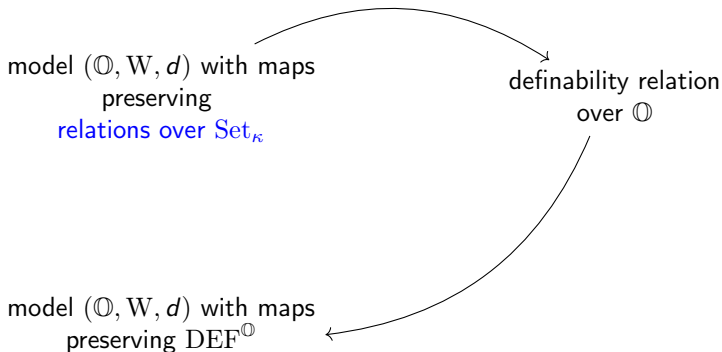
If  $\text{DEF}^\mathbb{O}$  is *also* a relation over our starting model...



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

Trick: cut the loop!

If  $\text{DEF}^\circledast$  is *also* a relation over our starting model...





# Preserving many relations at once [I some index set]

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

$$\begin{array}{ccc}
 \text{Krip}(\text{Set}_{\kappa}, F_{\varphi i}) & \longrightarrow & \text{Sub}(\widehat{\text{Con}}) \\
 \downarrow \text{proj} & & \downarrow \text{cod} \\
 \text{Base} \xrightarrow{s} \text{Set}_{\kappa} & \xrightarrow{\lambda X . \text{Set}_{\kappa}(F_{\varphi i}(-), X)} & \widehat{\text{Con}} \\
 \uparrow T & & 
 \end{array}$$

# Preserving many relations at once [I some index set]

$$\begin{array}{ccc}
 \mathbb{L} & \longrightarrow & \prod_{i \in \mathbb{I}} \widehat{\text{Sub}}(\widehat{\text{Con}}) \\
 \downarrow \text{proj} & & \downarrow \text{cod} \\
 \text{Base} \xrightarrow{s} \text{Set}_\kappa & \xrightarrow{\langle \lambda X . \text{Set}_\kappa(F_{\varphi_i}(-), X) \rangle_{i \in \mathbb{I}}} & \prod_{i \in \mathbb{I}} \widehat{\text{Con}} \\
 \uparrow \tau & & 
 \end{array}$$

# Preserving many relations at once [I some index set]

$$\begin{array}{ccc}
 \mathbb{L} & \xrightarrow{\quad} & \prod_{i \in \mathbb{I}} \widehat{\text{Sub}}(\widehat{\text{Con}}) \\
 \downarrow \text{proj} & & \downarrow \text{cod} \\
 \text{Base} \xrightarrow{s} \text{Set}_{\kappa} & \xrightarrow{\langle \lambda X . \text{Set}_{\kappa}(F_{\varphi_i}(-), X) \rangle_{i \in \mathbb{I}}} & \prod_{i \in \mathbb{I}} \widehat{\text{Con}} \\
 \uparrow T & & 
 \end{array}$$

Object of  $\mathbb{L}$ :

$(\underline{A}, \overline{A})$  where  $(\underline{A}, \overline{A}(i)) \in \text{Krip}(\text{Set}_{\kappa}, F_{\varphi_i})$

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

## Preserving many relations at once [I some index set]

$$\begin{array}{ccc}
 \mathbb{L} & \xrightarrow{\quad} & \prod_{i \in I} \widehat{\text{Sub}}(\widehat{\text{Con}}) \\
 \downarrow \text{proj} & & \downarrow \text{cod} \\
 \text{Base} \xrightarrow{s} \text{Set}_{\kappa} & \xrightarrow{\langle \lambda X . \text{Set}_{\kappa}(F_{\varphi_i}(-), X) \rangle_{i \in I}} & \prod_{i \in I} \widehat{\text{Con}} \\
 \uparrow T & & 
 \end{array}$$

Object of  $\mathbb{L}$ :

$$(\underline{A}, \overline{A}) \text{ where } (\underline{A}, \overline{A}(i)) \in \text{Krip}(\text{Set}_{\kappa}, F_{\varphi_i})$$

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

↪ Maps in  $\mathbb{L}$  are maps in  $\text{Set}_{\kappa}$  that **preserve every relation**

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

## Preserving many relations at once [I some index set]

$$\begin{array}{ccc}
 \mathbb{L} & \xrightarrow{\quad} & \prod_{i \in \mathbb{I}} \widehat{\text{Sub}}(\widehat{\text{Con}}) \\
 \downarrow \text{proj} & & \downarrow \text{cod} \\
 \text{Base} \xrightarrow{s} \text{Set}_{\kappa} & \xrightarrow{\langle \lambda X . \text{Set}_{\kappa}(F_{\varphi_i}(-), X) \rangle_{i \in \mathbb{I}}} & \prod_{i \in \mathbb{I}} \widehat{\text{Con}} \\
 \uparrow T & & 
 \end{array}$$

Object of  $\mathbb{L}$ :

$(\underline{A}, \overline{A})$  where  $(\underline{A}, \overline{A}(i)) \in \text{Krip}(\text{Set}_{\kappa}, F_{\varphi_i})$

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

↪ Maps in  $\mathbb{L}$  are maps in  $\text{Set}_{\kappa}$  that preserve every relation

↪ If we choose  $\mathbb{I}$  big enough, maps in  $\mathbb{L}$  will preserve  $\text{DEF}^{\circledast}$

# 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. ✓

~~~~> Now to put it all together!

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

$$\begin{array}{ccc}
 \mathbb{L} & \longrightarrow & \prod_{i \in \mathbb{I}} \widehat{\text{Con}} \\
 \downarrow \text{proj} & & \downarrow \text{cod} \\
 \text{Base} \xrightarrow{s} \text{Set}_\kappa & \xrightarrow{\langle \lambda X . \text{Set}_\kappa(F_{\varphi_i}(-), X) \rangle_{i \in \mathbb{I}}} & \prod_{i \in \mathbb{I}} \widehat{\text{Con}} \\
 \uparrow T & & 
 \end{array}$$

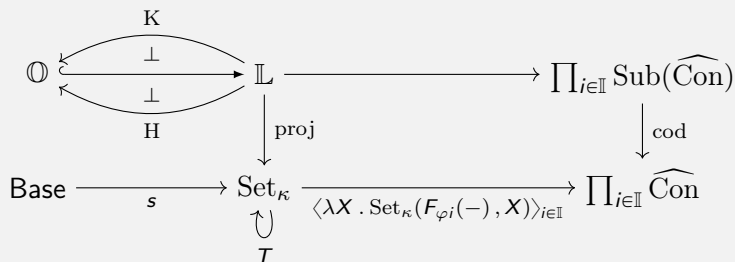
Object of  $\mathbb{L}$ :

$(\underline{A}, \overline{A})$  where  $(\underline{A}, \overline{A}(i)) \in \text{Krip}(\text{Set}_\kappa, F_{\varphi_i})$

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



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



Object of  $\mathbb{O}$ :

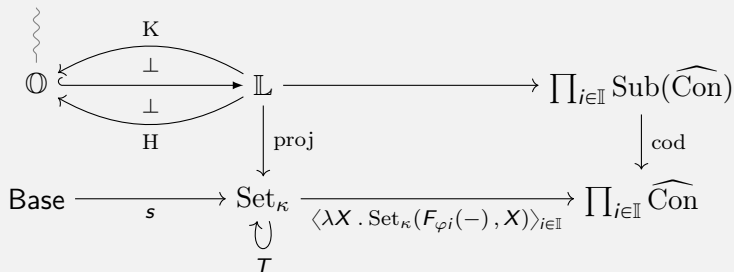
$(\underline{A}, \overline{A})$  where  $(\underline{A}, \overline{A}(i)) \in \text{Krip}(\text{Set}_\kappa, F_{\varphi i})$

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

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

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

O'Hearn–Riecke  
model



Object of  $\mathbb{O}$ :

$(\underline{A}, \overline{A})$  where  $(\underline{A}, \overline{A}(i)) \in \text{Conc}(\text{Set}_\kappa, F_{\varphi_i})$

$\overline{A}(i)(\Gamma) \subseteq \text{Set}_\kappa(F_{\varphi_i}\Gamma, \underline{A})$  + 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  $\text{DEF}^{\mathbb{O}}$  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

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

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

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

+ compatibility with renaming

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

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

## Choosing $\mathbb{I}$ , part 1

$$\begin{aligned}(\underline{A}, \overline{A}) &\in \mathbb{O}: \\(\underline{A}, \overline{A}(i)) &\in \text{Conc}(\text{Set}_\kappa, F_{\varphi i}) \\ \overline{A}(i)(\Gamma) &\subseteq \text{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A}) \\ &+ \text{compatibility with renaming}\end{aligned}$$

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

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

## Choosing $\mathbb{I}$ , part 1

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

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

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

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



## Choosing $\mathbb{I}$ , part 1

$$\begin{aligned}(\underline{A}, \overline{A}) &\in \mathbb{O}: \\ (\underline{A}, \overline{A}(i)) &\in \text{Conc}(\text{Set}_\kappa, F_{\varphi i}) \\ \overline{A}(i)(\Gamma) &\subseteq \text{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A}) \\ &+ \text{compatibility with renaming}\end{aligned}$$

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

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

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

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

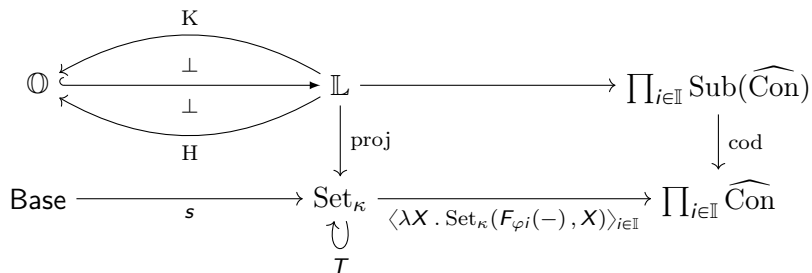
## Choosing $\mathbb{I}$ , part 1

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

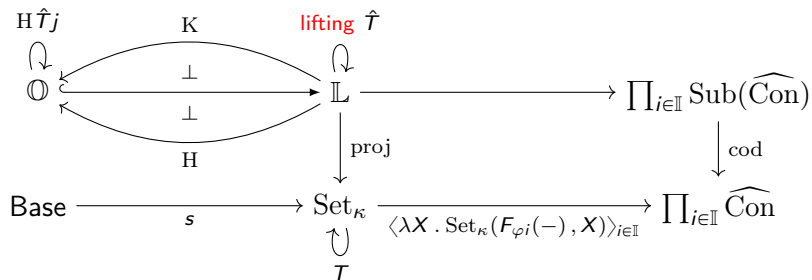
**Defining  $\mathbb{I}$ :**

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

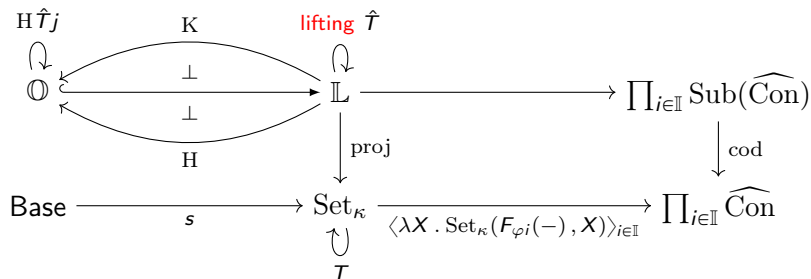
## Choosing $\mathbb{I}$ , part 2



## Choosing II, part 2

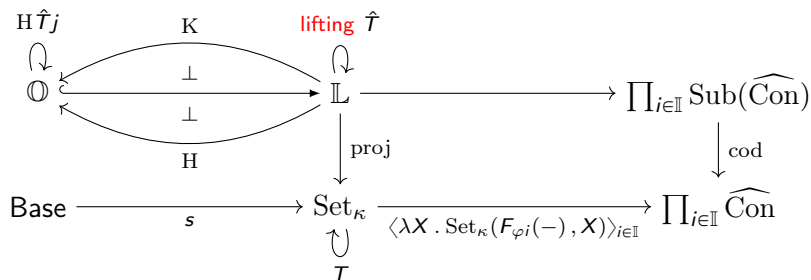


## Choosing $\mathbb{I}$ , part 2



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

## Choosing $\mathbb{I}$ , part 2



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



need a lifting parameter for each  $\varphi_i$

## Choosing II, part 2

**Defining II:**

---

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

---

$\mathcal{K}r(F)$

parameters for  $\top\top$ -lifting  
to  $\text{Krip}(\text{Set}_{\kappa}, F)$

---

## Choosing II, part 3

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

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

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

+ compatibility with renaming

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

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



## Choosing II, part 3

$$\begin{aligned}(\underline{A}, \overline{A}) &\in \mathbb{O}: \\ (\underline{A}, \overline{A}(i)) &\in \text{Conc}(\text{Set}_\kappa, F_{\varphi i}) \\ \overline{A}(i)(\Gamma) &\subseteq \text{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A}) \\ &+ \text{compatibility with renaming}\end{aligned}$$

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

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

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

## Choosing II, part 3

$$\begin{aligned}(\underline{A}, \overline{A}) &\in \mathbb{O}: \\(\underline{A}, \overline{A}(i)) &\in \text{Conc}(\text{Set}_\kappa, F_{\varphi i}) \\ \overline{A}(i)(\Gamma) &\subseteq \text{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A}) \\ &+ \text{compatibility with renaming}\end{aligned}$$

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

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

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

Trick:

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

## Choosing II, part 3

$$\begin{aligned}(\underline{A}, \overline{A}) &\in \mathbb{O}: \\(\underline{A}, \overline{A}(i)) &\in \text{Conc}(\text{Set}_\kappa, F_{\varphi i}) \\ \overline{A}(i)(\Gamma) &\subseteq \text{Set}_\kappa(F_{\varphi i}\Gamma, \underline{A}) \\ &+ \text{compatibility with renaming}\end{aligned}$$

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

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

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

Trick:

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

Then:

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

## Defining II

---

|                                                                          |                                                                                                                                                                   |
|--------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\mathbb{F} := \text{Cart}(\text{Con}^{\text{op}}, \text{Set}_{\kappa})$ | cartesian functors into $\text{Set}_{\kappa}$                                                                                                                     |
| $\mathcal{K}r(F)$                                                        | parameters for TT-lifting<br>to $\text{Krip}(\text{Set}_{\kappa}, F)$                                                                                             |
| $\mathcal{I}nt(F)$                                                       | $r : \text{Base} \rightarrow \widehat{\text{Con}}$ such that<br>$(s[\beta], r(\beta)) \in \text{Conc}(\text{Set}_{\kappa}, F)$<br>for all $\beta \in \text{Base}$ |

---

## Defining $\mathbb{I}$

|                                                                          |                                                                                                                                                                   |
|--------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\mathbb{F} := \text{Cart}(\text{Con}^{\text{op}}, \text{Set}_{\kappa})$ | cartesian functors into $\text{Set}_{\kappa}$                                                                                                                     |
| $\mathcal{K}r(F)$                                                        | parameters for $\text{TT}$ -lifting<br>to $\text{Krip}(\text{Set}_{\kappa}, F)$                                                                                   |
| $\mathcal{I}nt(F)$                                                       | $r : \text{Base} \rightarrow \widehat{\text{Con}}$ such that<br>$(s[\beta], r(\beta)) \in \text{Conc}(\text{Set}_{\kappa}, F)$<br>for all $\beta \in \text{Base}$ |

$$\mathbb{I} := \{(F, K, r) \mid F \in \mathbb{F}, K \in \mathcal{K}r(F), r \in \mathcal{I}nt(F)\}$$
$$\varphi : (F, K, r) \mapsto F$$

## Defining II

|                                                                          |                                                                                                                                                                   |
|--------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\mathbb{F} := \text{Cart}(\text{Con}^{\text{op}}, \text{Set}_{\kappa})$ | cartesian functors into $\text{Set}_{\kappa}$                                                                                                                     |
| $\mathcal{K}r(F)$                                                        | parameters for $\text{TT}$ -lifting<br>to $\text{Krip}(\text{Set}_{\kappa}, F)$                                                                                   |
| $\mathcal{I}nt(F)$                                                       | $r : \text{Base} \rightarrow \widehat{\text{Con}}$ such that<br>$(s[\beta], r(\beta)) \in \text{Conc}(\text{Set}_{\kappa}, F)$<br>for all $\beta \in \text{Base}$ |

$$\mathbb{I} := \{(F, K, r) \mid F \in \mathbb{F}, K \in \mathcal{K}r(F), r \in \mathcal{I}nt(F)\}$$
$$\varphi : (F, K, r) \mapsto F$$



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

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

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

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

+ compatibility with renaming

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

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

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

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

+ compatibility with renaming

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



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



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

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

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

+ compatibility with renaming

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



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

$$\text{DEF}_\sigma^\mathbb{O}(\Gamma) \subseteq \mathbb{O}(d[\Gamma], d[\sigma])$$

$$\subseteq \text{Set}_\kappa(\text{Ud}[\Gamma], \text{Ud}[\sigma])$$

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

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

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

### Key property of $(\mathbb{O}, \text{H}\hat{T}_j, d)$

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

$$\overline{d[\sigma]}(F, K, r) = \text{DEF}_{\sigma}^{\circ}$$

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

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

$(\underline{A}, \overline{A}(F, K, r)) \in \text{Conc}(\text{Set}_\kappa, F)$

$\overline{A}(F, K, r)(\Gamma) \subseteq \text{Set}_\kappa(F\Gamma, \underline{A})$

+ compatibility with renaming

$d : \text{Base} \rightarrow \mathbb{O}$

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

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

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

$(\underline{A}, \overline{A}(F, K, r)) \in \text{Conc}(\text{Set}_\kappa, F)$

$\overline{A}(F, K, r)(\Gamma) \subseteq \text{Set}_\kappa(F\Gamma, \underline{A})$

+ compatibility with renaming

$d : \text{Base} \rightarrow \mathbb{O}$

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

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

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

$$\begin{array}{l}
 (\underline{A}, \overline{A}) \in \mathbb{O}: \\
 (\underline{A}, \overline{A})(F, K, r) \in \text{Conc}(\text{Set}_\kappa, F) \\
 \overline{A}(F, K, r)(\Gamma) \subseteq \text{Set}_\kappa(F\Gamma, \underline{A}) \\
 + \text{compatibility with renaming}
 \end{array}$$

---


$$\begin{array}{l}
 d : \text{Base} \rightarrow \mathbb{O} \\
 d(\beta) = (s[\beta], \overline{d[\beta]}) \\
 \overline{d[\beta]}(F, K, r) = r(\beta)
 \end{array}$$

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

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

### Key property

---

If  $(s[\beta], \text{DEF}_\beta^\circ) \in \text{Conc}(\text{Set}_\kappa, \text{Ud}[-])$  for all  $\beta \in \text{Base}$ , then

$$\overline{d[\sigma]}(\text{Ud}[-], K, \lambda\beta. \text{DEF}_\beta^\circ) = \text{DEF}_\sigma^\circ$$

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

### Key property

---

If  $(s[\beta], \text{DEF}_\beta^\circ) \in \text{Conc}(\text{Set}_\kappa, \text{Ud}[-])$  for all  $\beta \in \text{Base}$ , then

$$\overline{d[\sigma]}(\text{Ud}[-], K, \lambda\beta. \text{DEF}_\beta^\circ) = \text{DEF}_\sigma^\circ$$

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

→ concreteness needed for exponentials and monadic types

## Key property

---

If  $(s[\beta], \text{DEF}_\beta^\circ) \in \text{Conc}(\text{Set}_\kappa, \text{Ud}[-])$  for all  $\beta \in \text{Base}$ , then

$$\overline{d[\sigma]}(\text{Ud}[-], K, \lambda\beta. \text{DEF}_\beta^\circ) = \text{DEF}_\sigma^\circ$$

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

From the Key Property to full completeness:

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



## Key property

If  $(s[\beta], \text{DEF}_\beta^\circ) \in \text{Conc}(\text{Set}_\kappa, \text{Ud}[-])$  for all  $\beta \in \text{Base}$ , then

$$\overline{d[\sigma]}(\text{Ud}[-], K, \lambda\beta. \text{DEF}_\beta^\circ) = \text{DEF}_\sigma^\circ$$

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

From the Key Property to full completeness:

if  $f : d[\Gamma] \rightarrow d[\sigma]$ , 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  $(s[\beta], \text{DEF}_\beta^\circ) \in \text{Conc}(\text{Set}_\kappa, \text{Ud}[-])$  for all  $\beta \in \text{Base}$ , then

$$\overline{d[\sigma]}(\text{Ud}[-], K, \lambda\beta. \text{DEF}_\beta^\circ) = \text{DEF}_\sigma^\circ$$

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

From the Key Property to full completeness:

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

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

## Key property

If  $(s[\beta], \text{DEF}_\beta^\circ) \in \text{Conc}(\text{Set}_\kappa, \text{Ud}[-])$  for all  $\beta \in \text{Base}$ , then

$$\overline{d[\sigma]}(\text{Ud}[-], K, \lambda\beta. \text{DEF}_\beta^\circ) = \text{DEF}_\sigma^\circ$$

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

From the Key Property to full completeness:

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

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

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

## Key property

---

If  $(s[\beta], \text{DEF}_\beta^\circ) \in \text{Conc}(\text{Set}_\kappa, \text{Ud}[-])$  for all  $\beta \in \text{Base}$ , then

$$\overline{d[\sigma]}(\text{Ud}[-], \mathcal{K}, \lambda\beta. \text{DEF}_\beta^\circ) = \text{DEF}_\sigma^\circ$$

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

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

---

If  $(s[\beta], \text{DEF}_\beta^\circ) \in \text{Conc}(\text{Set}_\kappa, \text{Ud}[-])$  for all  $\beta \in \text{Base}$ , then

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

is fully complete and extensional, hence fully abstract.

## 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  $\lambda_c$  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  $\lambda_c$  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, ...