Chocola seminar 13 March 2025

KLEENE-CARTESIAN BICATEGORIES FOR PROGRAM LOGICS

Elena Di Lavore
University of Oxford

j.w.w. Filippo Bonchi and Alessandro Di Giorgio
University of Pisa Tallinn University of Technology

OVERVIEW

- Introduce Kleene bicategories.
- Mix with cartesian bicategories of relations.
- Express Hoare triples and derive rules of Hoare logic.

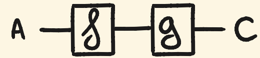
MORPHISMS AS PROGRAMS

$f : A \longrightarrow B$

$f' : A' \longrightarrow B'$

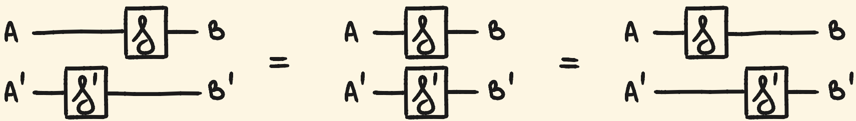
$g : B \longrightarrow C$

- Sequential composition. $f ; g : A \longrightarrow C$

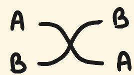


- Identity. $id_A : A \longrightarrow A$

- Parallel composition. $f \otimes f' : A \otimes A' \longrightarrow B \otimes B'$



- Symmetries. $\sigma_{A,B} : A \otimes B \longrightarrow B \otimes A$

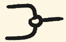


MONOIDS IN STRING DIAGRAMS

$(\mathcal{C}, \otimes, I)$ symmetric monoidal category

A commutative monoid in $\mathcal{C}$ is

$A \in \mathrm{obj}\,\mathcal{C}$ (carrier)

 : $A \otimes A \longrightarrow A$ (multiplication)

 : $I \to A$ (unit)

such that

(associativity)

(unitality)

(commutativity)

<u>ex</u> monoids in $(\mathcal{S}et, \times, \{*\})$ are monoids

COMONOIDS IN STRING DIAGRAMS

$(\mathcal{C}, \otimes, I)$ symmetric monoidal category

A commutative comonoid in $\mathcal{C}$ is

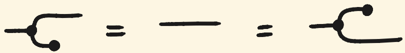
$A \in \mathrm{obj}\,\mathcal{C}$ (carrier)

 $: A \longrightarrow A \otimes A$ (comultiplication)

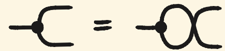
 $: A \to I$ (counit)

such that

(associativity)

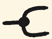


(unitality)

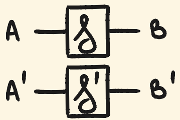


(commutativity)

<u>ex</u> $\begin{cases} \text{[comultiplication]}(x) := (x, x) \\ \text{[counit]}(x) := * \end{cases}$ is a commutative comonoid in $(\mathcal{S}et, \times, \{*\})$




TWO KINDS OF PARALLEL COMPOSITION



execute f <u>and</u> f' $\rightsquigarrow$ data flow

execute f <u>or</u> f' $\rightsquigarrow$ control flow

<u>ex</u> $(\mathbf{Set}, \times, \{*\})$ $\rightsquigarrow$ 'and'

$(\mathbf{Set}, +, \emptyset)$ $\rightsquigarrow$ 'or'

We need both kinds of parallel compositions

$\Rightarrow$ <u>rig categories</u> $(\mathcal{C}, \otimes, I, \oplus, 0)$:

$(\mathcal{C}, \otimes, I)$ and $(\mathcal{C}, \oplus, 0)$ symmetric monoidal categories

$$\delta^L_{A,B,C} : A \otimes (B \oplus C) \longrightarrow (A \otimes B) \oplus (A \otimes C) \qquad \lambda^{\bullet L}_A : 0 \otimes A \longrightarrow 0$$

$$\delta^R_{A,B,C} : (A \oplus B) \otimes C \longrightarrow (A \otimes C) \oplus (B \otimes C) \qquad \lambda^{\bullet R}_A : A \otimes 0 \longrightarrow 0$$

OUTLINE

[• The category of relations]

- Kleene bicategories
- Tape diagrams for Kleene-cartesian bicategories
- Program logics

RELATIONS FORM A RIG CATEGORY

$f : A \to B$ is $f \subseteq A \times B$

- Monoidal product $\otimes$.

  $f \otimes g := f \times g \qquad I := \{*\}$

- Monoidal product $\oplus$.

  $f \oplus g := f \sqcup g \qquad 0 := \varnothing$

- Distributivity.

  $(A \sqcup B) \times C \simeq (A \times C) \sqcup (B \times C) \qquad A \times \varnothing \simeq \varnothing$

$\rightsquigarrow$ For a monoidal monad T on a distributive category $\mathcal{C}$, Kleisli(T) is rig.

CALCULUS OF RELATIONS

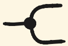
$$
\begin{aligned}
E ::= \; & R \mid id \mid E;E \\
\mid \; & E^{op} \mid \top \mid E \cap E \\
\mid \; & E^{*} \mid \bot \mid E \cup E
\end{aligned}
$$

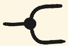
allegorical fragment
~ cartesian bicategories

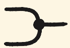
Kleene fragment
~ Kleene bicategories

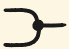
THE STRUCTURE OF $(\mathbf{Rel}, \otimes, I)$

- Commutative (co)monoid on every object. For a set A,

 $: A \to A \otimes A$

 $:= \{(x,(x,x)) \mid x \in A\}$

 $: A \otimes A \to A$

 $:= \{((x,x),x) \mid x \in A\}$

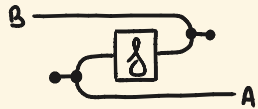
 $: A \to I$

 $:= \{(x,*) \mid x \in A\}$

 $: I \to A$

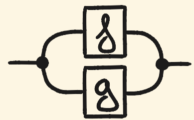
 $:= \{(*,x) \mid x \in A\}$

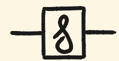
- Opposite. For $f : A \to B$,
  $f^{op} : B \to A$

$:=$


- Order. For $f, g : A \to B$,

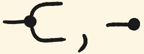
$f \leq g$ iff $f \subseteq g$ iff $f \cap g = f$ iff



$=$


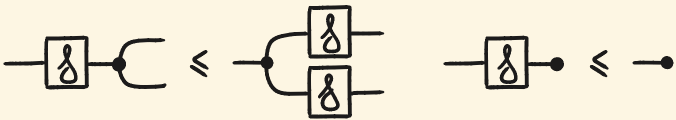
CARTESIAN BICATEGORIES OF RELATIONS

$\mathcal{C}$ monoidal posetal bicategory with

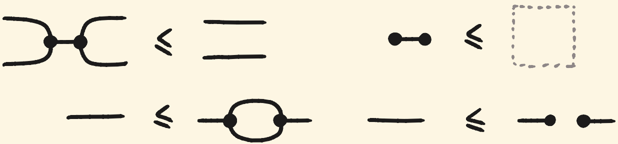
() coherent commutative comonoid

() coherent commutative monoid

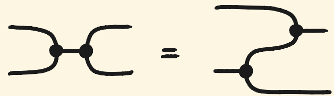
such that



(lax comonoid homomorphisms)



(comonoid ⊣ monoid)

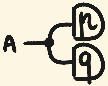


(Frobenius)

[Carboni, Walters (1987)]

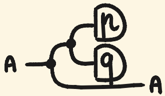
PREDICATES

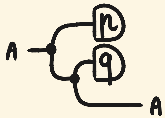
Predicates are $p : A \longrightarrow 1$.

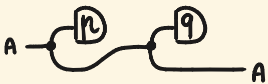
$p \wedge q =$ 

assert p = 

= 

$\Rightarrow$ assert$(p \wedge q)$ = 

= 

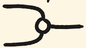
= 

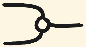
= 

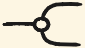
= assert p ; assert q

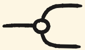
THE STRUCTURE OF $(\mathbf{Rel}, \oplus, 0)$

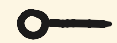
- Biproducts.

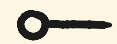
 $: A \oplus A \longrightarrow A$

 $:= \{((x,0),x) \mid x \in A\} \cup \{((x,1),x) \mid x \in A\}$

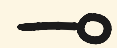
 $: A \longrightarrow A \oplus A$

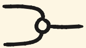
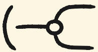
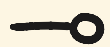
 $:= \{(x,(x,0)) \mid x \in A\} \cup \{(x,(x,1)) \mid x \in A\}$

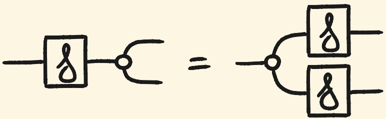
 $: 0 \longrightarrow A$

 $:= \varnothing$

 $: A \longrightarrow 0$

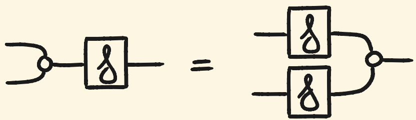
 $:= \varnothing$

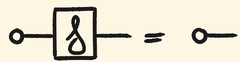
( , ), ( , ) natural coherent commutative (co)monoid



$\mapsto$ comonoid homomorphisms

monoid homomorphisms $\mapsfrom$

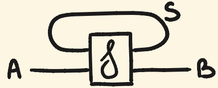




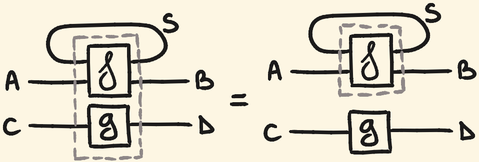
THE STRUCTURE OF $(\mathbf{Rel}, \oplus, 0)$

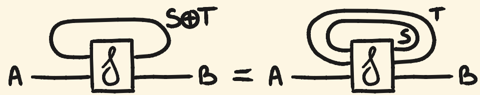
- Trace. For $f : S \oplus A \to S \oplus B$,

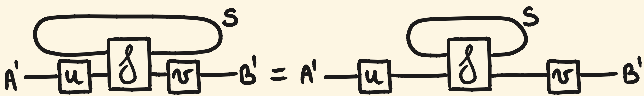
$$tr_S(f) : A \to B$$

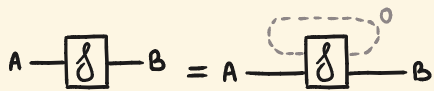
$tr_S(f) =$ 

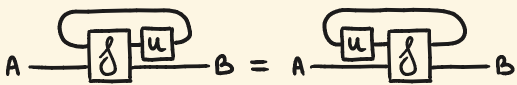
such that

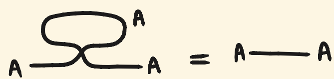












OUTLINE

- The category of relations
- Kleene bicategories
- Tape diagrams for Kleene-cartesian bicategories
- Program logics

KLEENE BICATEGORIES

How do biproducts and trace interact?

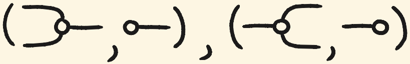
A Kleene bicategory is a category with

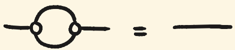
- finite biproducts s.t. $a \vee a = a$
- posetally uniform trace s.t. $id_A^* \leq id_A$

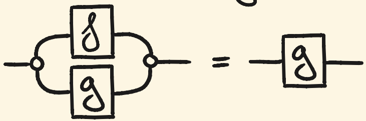
KLEENE BICATEGORIES

A Kleene bicategory is a category with

- finite biproducts s.t. $a \vee a = a$

 natural coherent commutative (co)monoid

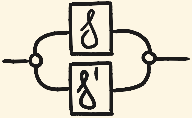
s.t.  ⇝ this gives a posetal enrichment:

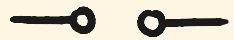
$f \leq g$ iff  iff $f \vee g = g$

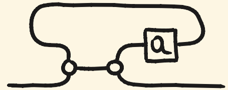
- posetally uniform trace s.t. $id_A^* \leq id_A$

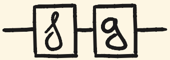
TRACES AND FIXPOINTS

For $f, f' : A \to B$, $g : B \to C$ and $a : A \to A$,

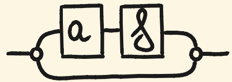
$f \vee f' :=$ 

$0_{A,B} :=$ 

$a^* :=$ 

$f \cdot g :=$ 

$1_A :=$ 

$\rightsquigarrow$ a^* is the least fixpoint of $\phi_a : f \mapsto$ 

PROPOSITION

Kleene bicategories are typed Kleene algebras.

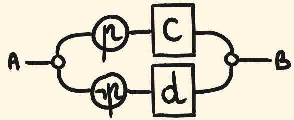
[Kozen (1994), (1998)]

COPRODUCTS FOR IF-ELSE STATEMENTS, TRACES FOR WHILE LOOPS

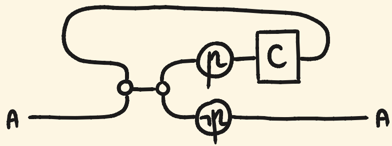
$c, d : A \longrightarrow B \qquad a : A \longrightarrow A$

$p : A \longrightarrow 1 \qquad \Rightarrow \qquad \text{assert } p : A \longrightarrow A$

if p then c else d $\longmapsto$

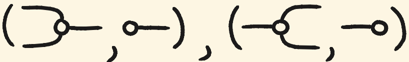


while p do c $\longmapsto$



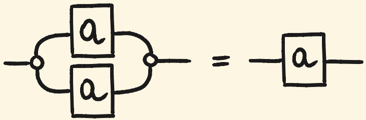
KLEENE AND CARTESIAN BICATEGORIES

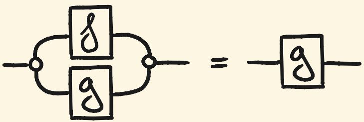
$(\mathrm{Rel}, \sqcup, \varnothing)$

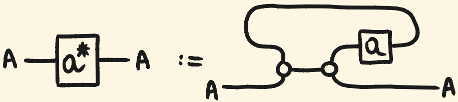


natural

monoids ⊣ comonoids



$f \leq g$ iff 



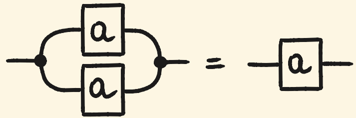
Kleene fragment

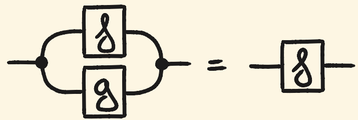
control flow

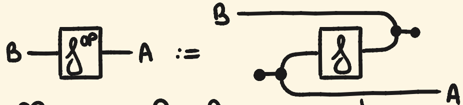
$(\mathrm{Rel}, \times, \{*\})$

lax natural

comonoids ⊣ monoids



$f \leq g$ iff 



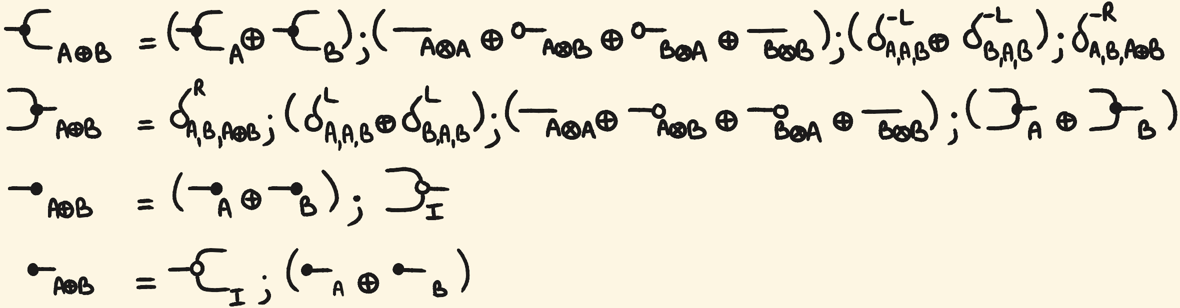
allegorical fragment

data flow

KLEENE-CARTESIAN BICATEGORIES

A Kleene-cartesian bicategory is a rig category $(\mathcal{C}, \otimes, I, \oplus, 0)$ such that

- $(\mathcal{C}, \otimes, I)$ is a cartesian bicategory
- $(\mathcal{C}, \oplus, 0)$ is a Kleene bicategory
- the black and white (co)monoids interact as



OUTLINE

- The category of relations
- Kleene bicategories
- [Tape diagrams for Kleene-cartesian bicategories]
- Program logics

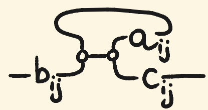
FREE KLEENE BICATEGORIES

$$
\begin{tikzcd}
KB : \mathrm{Sig} \arrow[r, bend left, "\mathrm{List}"] \arrow[r, phantom, "\perp"] & \mathrm{Cat} \arrow[l, bend left] \arrow[r, bend left, "\mathrm{Mat}"] \arrow[r, phantom, "\perp"] & \mathrm{FIBCat} \arrow[l, bend left] \arrow[r, bend left, "\mathrm{USt}"] \arrow[r, phantom, "\perp"] \arrow[d] & \mathrm{UTrFIBCat} \arrow[l, bend left] \arrow[r, bend left, "\mathrm{id}^*"] \arrow[r, phantom, "\perp"] \arrow[dr] & \mathrm{KBicat} \arrow[l, bend left] \\
 & & \mathrm{SMonCat} \arrow[r, bend left, "\mathrm{St}"] \arrow[r, phantom, "\perp"] \arrow[rr, bend left] & \mathrm{TrCat} \arrow[l, bend left] \arrow[r, bend left, "\mathrm{Unif}"] \arrow[r, phantom, "\perp"] & \mathrm{UTrCat} \arrow[l, bend left] \arrow[ll, bend left]
\end{tikzcd}
$$

Objects of KB_Σ are lists of types

Morphisms $A_1 \cdots A_n \longrightarrow B_1 \cdots B_m$ are matrices with entries typed Kleene algebra expressions over Σ :

$$\delta_{ij} = b_{ij} a_{ij}^* c =$$



FREE CARTESIAN BICATEGORIES

$$CB : \mathrm{MonSig} \underset{}{\overset{\perp}{\rightleftarrows}} \mathrm{CBicat}$$

Objects of CB_Σ are lists of types
Morphisms $A_1 \cdots A_n \longrightarrow B_1 \cdots B_m$ are discrete cospans of directed hypergraphs over Σ:

$$n \longrightarrow H \longleftarrow m$$

[Rosebrugh, Sabadini, Walters (2005), Bonchi, Seeber, Sobociński (2018)]

FREE KLEENE-CARTESIAN BICATEGORIES

$$\begin{tikzcd}
\mathrm{KCB} \colon & \mathrm{MonSig} \arrow[r, bend left] \arrow[r, phantom, "\perp"] \arrow[d] & \mathrm{KCBicat} \arrow[l, bend left] \arrow[d] \\
& \mathrm{MonSig} \arrow[r, bend left] \arrow[r, phantom, "\perp"] \arrow[d] & \mathrm{RigKBicat} \arrow[l, bend left] \arrow[d] \\
\mathrm{KB} \colon & \mathrm{Sig} \arrow[r, bend left] \arrow[r, phantom, "\perp"] & \mathrm{KBicat} \arrow[l, bend left]
\end{tikzcd}$$

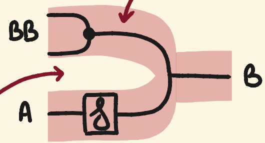
Objects of KCB_Σ are polynomials of types

$\rightsquigarrow$ what are the morphisms?

TAPE DIAGRAMS

String diagrams inside string diagrams.

$\otimes$ inside tapes



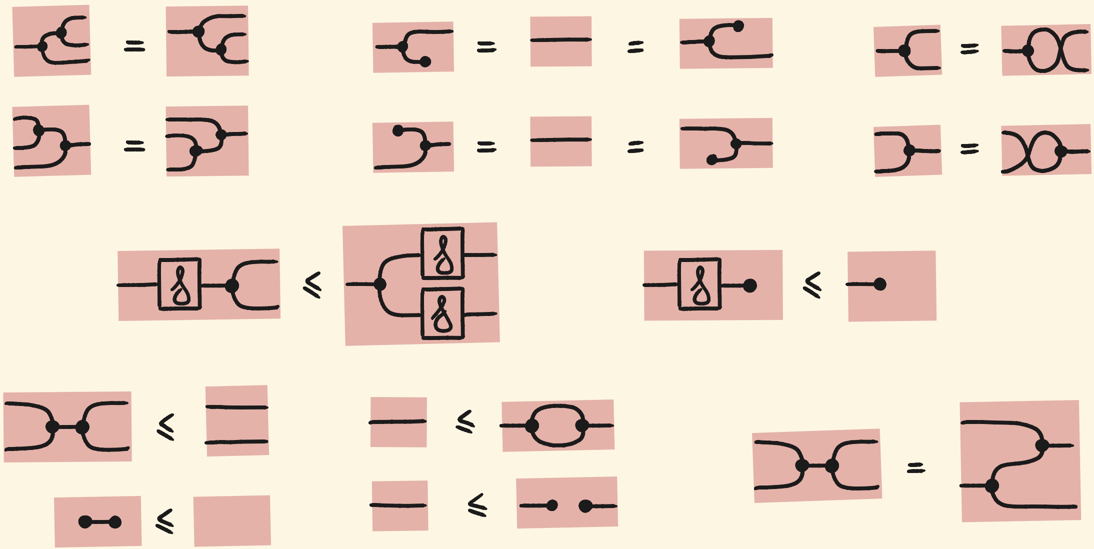
$: (B \otimes B) \oplus A \longrightarrow B$

$\oplus$ between tapes

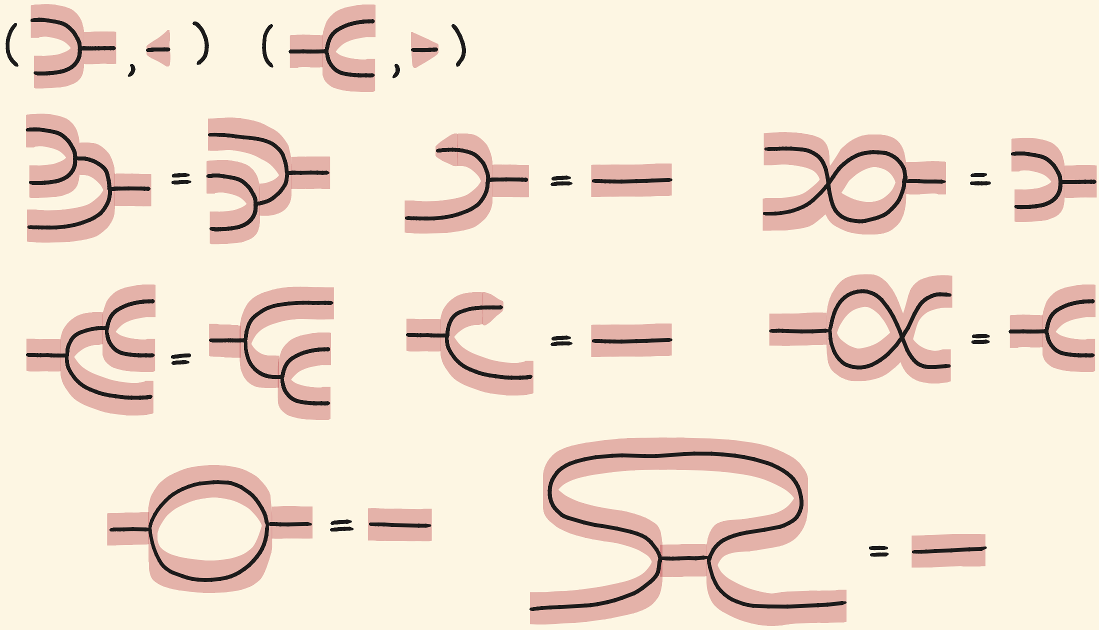
[Bonchi, Di Giorgio, Santamaria (2023)]
[Comfort, Delpeuch, Hedges (2022)]

TAPE DIAGRAMS FOR KLEENE-CARTESIAN BICATEGORIES

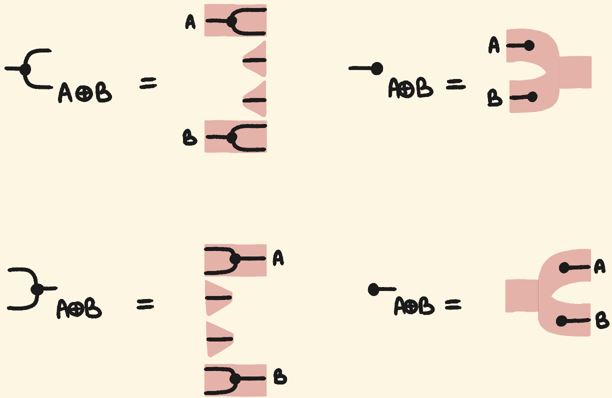
 such that



TAPE DIAGRAMS FOR KLEENE-CARTESIAN BICATEGORIES



TAPE DIAGRAMS FOR KLEENE-CARTESIAN BICATEGORIES



OUTLINE

- The category of relations
- Kleene bicategories
- Tape diagrams for Kleene-cartesian bicategories
- [Program logics]

INTERPRETING A PROGRAMMING LANGUAGE

$Var, \Sigma_E, \Sigma_P, \Sigma_C$

$E ::= x \in Var \mid e(E, \dots, E),\ e \in \Sigma_E(n)$

$P ::= p \in \Sigma_P \mid \neg p,\ p \in \Sigma_P \mid P \wedge P \mid P \vee P \mid E = E$

$C ::= c \in \Sigma_C \mid \text{skip} \mid CC \mid x \leftarrow E \mid \text{assert } P \mid \text{if } P \text{ then } C \text{ else } C \mid \text{while } P \text{ do } C$

choose a type X for variables

$N := \# Var$

$\Rightarrow$ expressions are $X^N \longrightarrow X$

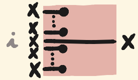
predicates are $X^N \longrightarrow I$

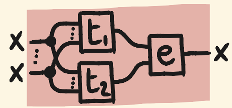
programs are $X^N \longrightarrow X^N$

INTERPRETING A PROGRAMMING LANGUAGE

$E ::= x_i \in Var \mid e(E, \dots, E),\ e \in \Sigma_E(n)$

expressions are $X^N \to X$ deterministic

$x_i \mapsto \pi_i =$ 

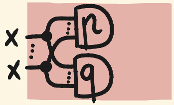
$e(t_1, \dots, t_n) \mapsto -\!\!<^{n}_{X^N} ; ([\![t_1]\!] \otimes \dots \otimes [\![t_n]\!]) ; [\![e]\!] \overset{n=2}{=}$ 

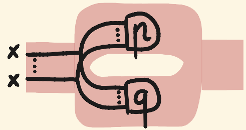
INTERPRETING A PROGRAMMING LANGUAGE

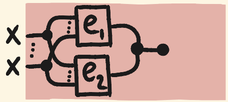
$P ::= p \in \Sigma_P \mid \neg p,\ p \in \Sigma_P \mid P \wedge P \mid P \vee P \mid E = E$

predicates are $X^N \to I$

$p \mapsto [\![p]\!] =$  for $p \in \Sigma_P \cup \neg\Sigma_P$

$p \wedge q \mapsto -\!\!\bullet\!\!\!\prec_{X^N} ; ([\![p]\!] \otimes [\![q]\!]) =$ 

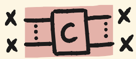
$p \vee q \mapsto -\!\!\circ\!\!\!\prec_{X^N} ; ([\![p]\!] \oplus [\![q]\!]) ; \succ\!\!\!\circ\!\!-_{I} =$ 

$e_1 = e_2 \mapsto -\!\!\bullet\!\!\!\prec_{X^N} ; ([\![e_1]\!] \otimes [\![e_2]\!]) ; \succ\!\!\!\bullet\!\!-\!\bullet_{X} =$ 

INTERPRETING A PROGRAMMING LANGUAGE (1)

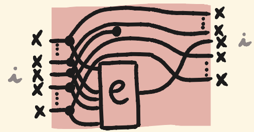
$C ::= c \in \Sigma_c \mid \mathrm{skip} \mid CC \mid x \leftarrow E \mid \mathrm{assert}\ P \mid \mathrm{if}\ P\ \mathrm{then}\ C\ \mathrm{else}\ C \mid \mathrm{while}\ P\ \mathrm{do}\ C$

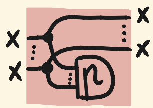
programs are $X^N \longrightarrow X^N$

$c \longmapsto [\![c]\!] =$  for $c \in \Sigma_c$

$\mathrm{skip} \longmapsto id_{X^N} =$ 

$fg \longmapsto [\![f]\!];[\![g]\!] =$ 

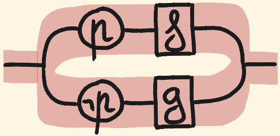
$x_i \leftarrow e \longmapsto \prec_{X^N} ; (\pi_{-i} \otimes [\![e]\!]) ; \bar{x}_{X^{N-i},X} =$ 

$\mathrm{assert}\ p \longmapsto \prec_{X^N} ; (id_{X^N} \otimes [\![p]\!]) =$ 

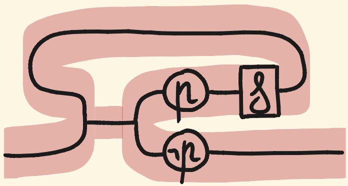
INTERPRETING A PROGRAMMING LANGUAGE (2)

$C ::= c \in \Sigma_C \mid \text{skip} \mid C\,C \mid x \leftarrow E \mid \text{assert}\ P \mid \text{if}\ P\ \text{then}\ C\ \text{else}\ C \mid \text{while}\ P\ \text{do}\ C$

$$\begin{array}{l}\text{if}\ p\\ \text{then}\ f\\ \text{else}\ g\end{array} \mapsto -\!\circ\!\!\subset ; ((\llbracket \text{assert}\ p \rrbracket ; \llbracket f \rrbracket) \oplus (\llbracket \text{assert}\ \neg p \rrbracket ; \llbracket g \rrbracket)) ; \supset\!\!\circ\!-$$

$=$ 

$$\begin{array}{l}\text{while}\ p\\ \text{do}\ f\end{array} \mapsto tr(\supset\!\!\circ\!- ; -\!\circ\!\!\subset ; ((\llbracket \text{assert}\ p \rrbracket ; \llbracket f \rrbracket) \oplus \llbracket \text{assert}\ \neg p \rrbracket))$$

$=$ 

HOARE LOGIC

$\{p\}\, c\, \{q\} \rightsquigarrow$ if the input satisfies p and c succeeds, then its output satisfies q

$$\frac{}{\{p\}\ \mathrm{skip}\ \{p\}} \qquad \frac{}{\{p[e/x]\}\ x \leftarrow e\ \{p\}} \qquad \frac{}{\{q\}\ \mathrm{assert}\ p\ \{p \wedge q\}}$$

$$\frac{p \subseteq p' \quad \{p'\}\, c\, \{q'\} \quad q' \subseteq q}{\{p\}\, c\, \{q\}} \qquad \frac{\{p\}\, c\, \{q\} \quad \{q\}\, d\, \{r\}}{\{p\}\, cd\, \{r\}}$$

$$\frac{\{p \wedge b\}\, c\, \{q\} \quad \{p \wedge \neg b\}\, d\, \{q\}}{\{p\}\ \mathrm{if}\ b\ \mathrm{then}\ c\ \mathrm{else}\ d\ \{q\}} \qquad \frac{\{p \wedge b\}\, c\, \{p\}}{\{p\}\ \mathrm{while}\ b\ \mathrm{do}\ c\ \{p \wedge \neg b\}}$$

INTERPRETING HOARE TRIPLES

THEOREM

$$\begin{cases} \mathcal{C} \text{ Kleene-cartesian bicategory} \\ [\![-]\!] : KCB_\Sigma \longrightarrow \mathcal{C} \end{cases}$$

$\{p\}\, c\, \{q\}$ is derivable with the rules of Hoare logic

$\Rightarrow [\![p]\!]^{op} ; [\![c]\!] \leq [\![q]\!]^{op}$

DERIVATION RULES (1)

$$\frac{}{\{p\}\,\mathrm{skip}\,\{p\}}$$

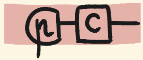
$[\![p]\!]^{op} ; [\![\mathrm{skip}]\!]$

$=$ 

$= [\![p]\!]^{op}$

$$\frac{p \subseteq p' \quad \{p'\}\,c\,\{q'\} \quad q' \subseteq q}{\{p\}\,c\,\{q\}}$$

$[\![p]\!]^{op} ; [\![c]\!]$

$=$ 

$\leqslant$ 

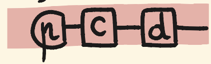
$\leqslant$ 

$\leqslant$ 

$= [\![q]\!]^{op}$

$$\frac{\{p\}\,c\,\{q\} \quad \{q\}\,d\,\{r\}}{\{p\}\,cd\,\{r\}}$$

$[\![p]\!]^{op} ; [\![cd]\!]$

$=$ 

$\leqslant$ 

$\leqslant$ 

$= [\![r]\!]^{op}$

DERIVATION RULES (2)

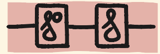
$$\dfrac{}{\{p[e/x]\}\ x \leftarrow e\ \{p\}}$$

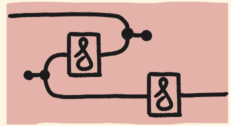
$[\![p[e/x]]\!]^{op} ; [\![x \leftarrow e]\!]$

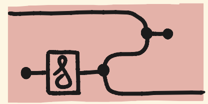
$=$ 

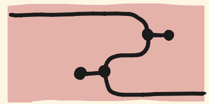
$\leqslant$ 

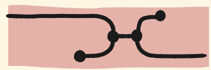
$= [\![p]\!]^{op}$



$:=$ 

(DET)
$=$ 

$\leqslant$ 

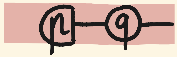
$=$ 

$=$ 

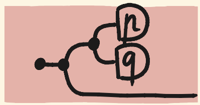
DERIVATION RULES (3)

$$\frac{}{\{p\}\ \text{assert}\ q\ \{p \wedge q\}}$$

$$[\![p]\!]^{op} ; [\![\text{assert}\ q]\!]$$

$=$ 

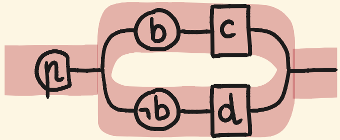
$=$ 

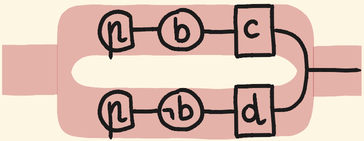
$=$ 

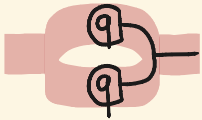
$$= [\![p \wedge q]\!]^{op}$$

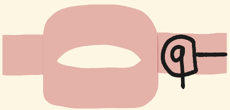
DERIVATION RULES (4)

$$\frac{\{p \wedge b\}\ c\ \{q\} \quad \{p \wedge \neg b\}\ d\ \{q\}}{\{p\}\ \text{if}\, b\ \text{then}\ c\ \text{else}\ d\ \{q\}}$$

$[\![p]\!]^{op} ; [\![\text{if}\, b\ \text{then}\ c\ \text{else}\ d]\!] =$ 

$\overset{(NAT)}{=}$ 

$\leq$ 

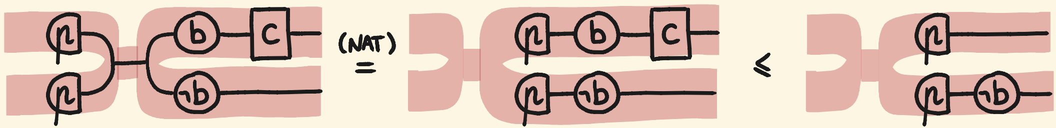
$\overset{(NAT)}{=}$ 

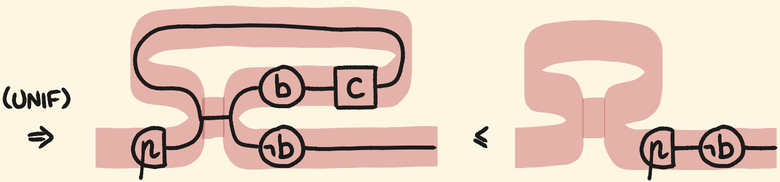
$\overset{(IDEM)}{=}$ 

$= [\![q]\!]^{op}$

DERIVATION RULES (5)

$$\frac{\{p \wedge b\}\ c\ \{p\}}{\{p\}\ \text{while } b \text{ do } c\ \{p \wedge \neg b\}}$$





$\|$ $\qquad\qquad$ $\wedge$ (id*)

$[\![p]\!]^{op} ; [\![\text{while } b \text{ do } c]\!]$

$= [\![p \wedge \neg b]\!]^{op}$

OTHER PROGRAM LOGICS

HOARE

$\{p\}\, c\, \{q\}$ $\qquad$ $[\![p]\!]^{op} ; [\![c]\!] \leqslant [\![q]\!]^{op}$ $\qquad$ if c terminates from p, its output is in q

INCORRECTNESS

$[p]\, c\, [q]$ $\qquad$ $[\![p]\!]^{op} ; [\![c]\!] \geqslant [\![q]\!]^{op}$ $\qquad$ all outputs in q can be reached by c starting in q

SUFFICIENT INCORRECTNESS

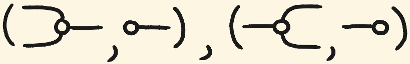
$\langle\!\langle p \rangle\!\rangle\, c\, \langle\!\langle q \rangle\!\rangle$ $\qquad$ $[\![p]\!] \leqslant [\![c]\!] ; [\![q]\!]$ $\qquad$ starting in p, c can terminate in q

NECESSARY

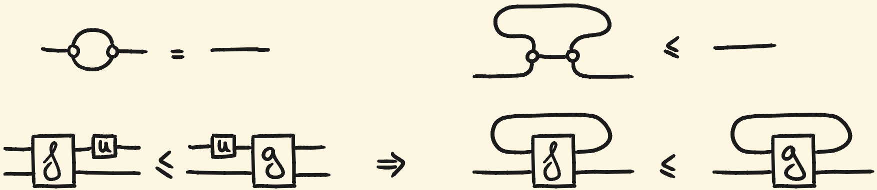
$(p)\, c\, (q)$ $\qquad$ $[\![p]\!] \geqslant [\![c]\!] ; [\![q]\!]$ $\qquad$ if c terminates in q, its input was in p

SUMMARY

- Kleene bicategories ≈ structure of $(\mathrm{Rel}, \sqcup, \varnothing)$.

 natural coherent commutative (co)monoid

s.t.



- Kleene-cartesian bicategories express both control flow and data flow of classical programs.
- Kleene-cartesian bicategories are sound for Hoare logic.

FUTURE WORK

- Effectful programs and their logics.
- Normal form for morphisms in $KB_{\mathcal{C}}$ and KCB_{Σ}.