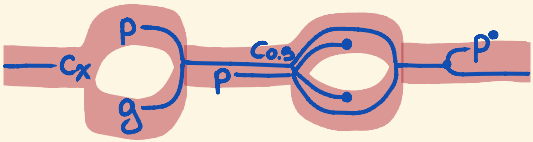
Online logic seminar 1 October 2026

DERIVING PROGRAM LOGICS FROM DISTRIBUTIVE MONOIDAL CATEGORIES

Filippo Bonchi, U. of Pisa
Elena Di Lavore, Tallinn U. of Technology
Mario Román, Tallinn U. of Technology
Sam Staton, U. of Oxford

THE PIRAÑHA PROBLEM

```
if coin(x)
| then F1 ← piranha
| else F1 ← goldfish
end
F2 ← piranha
if coin(0.5)
| then s ← F1
| else s ← F2
end
observe(s = piranha)
```



⇝ can we bound the probability of observing piranha ?

MOTIVATION AND CONTEXT

- Categorical semantics of programs
  - ↳ reason about program behaviour and properties
    abstraction and compositionality

morphisms ⇝ programs

This work:

- Categorical semantics of imperative programs with effects
- Inference rules derive from the categorical structure

IMPERATIVE PROGRAMS

A monoidal guarded command language.

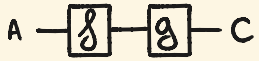
$C ::=$ skip	do nothing
$\mid$ abort	diverge
$\mid C\,C$	sequential composition
$\mid x_i \leftarrow E$	assignment
$\mid (x_{i_1}, \ldots, x_{i_k}) \leftsquigarrow S$	sampling
$\mid$ assert P	assert / observe
$\mid$ if B then C else C	conditional choice
$\mid$ while B do C	loop

MORPHISMS AS PROGRAMS

$f : A \longrightarrow B$ 

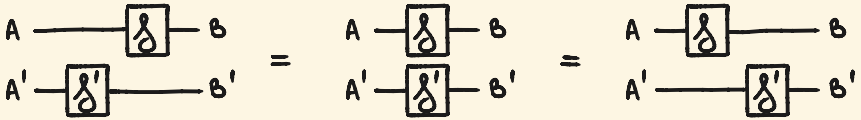
$f' : A' \longrightarrow B'$ 

$g : B \longrightarrow C$ 

- 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)

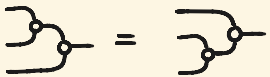


(multiplication)

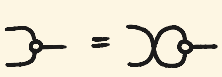


(unit)

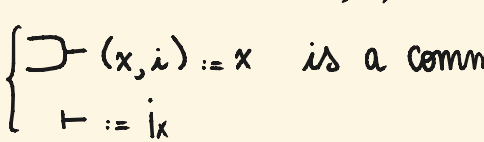
such that


(associativity)


(unitality)


(commutativity)

ex monoids in $(\mathrm{Set}, \times, \{*\})$ are monoids

ex 

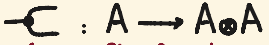
is a commutative monoid in $(\mathrm{Set}, +, \varnothing)$

COMONOIDS IN STRING DIAGRAMS

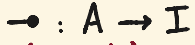
$(\mathcal{C}, \otimes, I)$ symmetric monoidal category

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

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



(comultiplication)

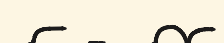


(counit)

such that


(associativity)


(unitality)

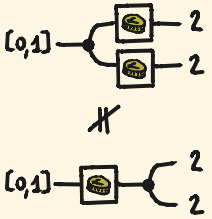

(commutativity)

ex $\begin{cases} \Delta(x) := (x,x) \\ \varepsilon(x) := * \end{cases}$ is a commutative comonoid in $(\mathrm{Set}, \times, \{*\})$

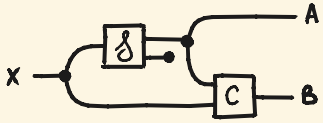
WHY STRING DIAGRAMS ?

We work with noncartesian monoidal structures.

$(\mathrm{coin}(p), \mathrm{coin}(p))$? $\not=$

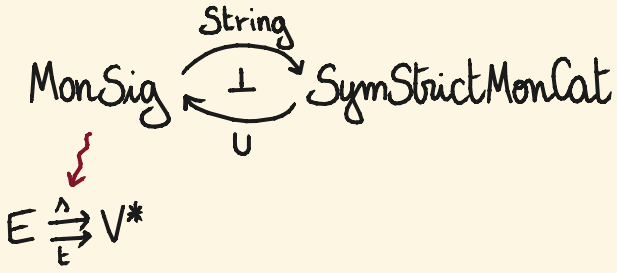


String diagrams are often easier to read.



$= \nu_X ; ((f ; \pi_1 ; \nu_A) \otimes \mathrm{id}_X) ; (\mathrm{id}_A \otimes C)$

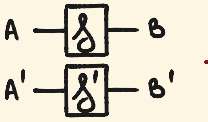
STRING DIAGRAMS ARE FREE



By strictification and coherence, string diagrams are a syntax for symmetric monoidal categories.

[Mac Lane (1997); Joyal, Street (1993)]

TWO KINDS OF PARALLEL COMPOSITION



execute f and f' ⇝ data flow

execute f or f' ⇝ control flow

ex for $T : \mathrm{Set} \to \mathrm{Set}$ $\times$-monoidal monad,

$(\mathrm{Kl}T, \otimes, \{*\})$ ⇝ 'and'

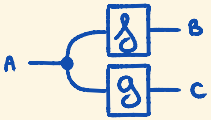
$(\mathrm{Kl}T, +, \varnothing)$ ⇝ 'or'

We need both kinds of parallel compositions

⇒ distributive monoidal categories $(\mathcal{C}, \otimes, 1, +, 0)$

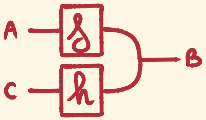
TWO MONOIDAL STRUCTURES

data flow



$A \longrightarrow B \otimes C$

control flow



$A + C \longrightarrow B$

[Arbib, Manes (1980)] [Walters (1992)]
[Jacobs (2010, 2016)]

OUTLINE

- [The structure of discrete probabilistic programs]
- The guarded command language
- Imperative categories
- Program logics

EXAMPLE : $\mathbf{Stoch}_{\leq}$

- objects are sets $A, B, \ldots$
- morphisms $f : A \to B$ are functions $f : A \to \mathcal{D}_{\leq} B$

  $\mathcal{D}_{\leq} B := \{\sigma : B \to [0,1] \mid \mathrm{supp}\,\sigma \text{ countable} \wedge \sum_B \sigma(b) \leq 1\}$

  $\rightsquigarrow f(b|a) =$ "probability that f outputs b given input a"
- composition of $f : A \to B$ and $g : B \to C$ is

  $f;g(c|a) := \sum_{b \in B} f(b|a) \cdot g(c|b)$
- identity $\mathrm{id}_A : A \to A$ is the Dirac distribution

  $\mathrm{id}_A(a'|a) = 1 \cdot [a = a']$

THE STRUCTURE OF $(\mathbf{Stoch}_{\leq}, \otimes, 1)$

$\mathbf{Stoch}_{\leq}$ has a monoidal structure.

For $f : A \to B$ and $f' : A' \to B'$,

$$f \otimes f' (b, b' | a, a') := f(b|a) \cdot f'(b'|a')$$

This monoidal structure is copy-discard.



$: A \to A \otimes A$



$(a_1, a_2 | a_3) := 1 \cdot [a_1 = a_2 = a_3]$

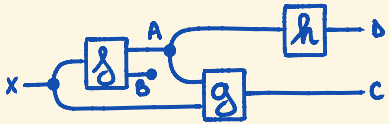


$: A \to I$



$(* | a) := 1$

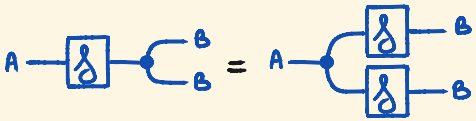
Reading diagrams in $(\mathbf{Stoch}_{\leq}, \otimes, 1)$.



$$(d, c | x) = \sum_{a \in A} \sum_{b \in B} f(a, b | x) \cdot g(c | a, x) \cdot h(d | a)$$

DETERMINISTIC AND TOTAL MORPHISMS

A morphism $f : A \to B$ is

- deterministic if 

- total if 

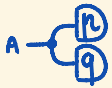
Expressions are deterministic and total morphisms



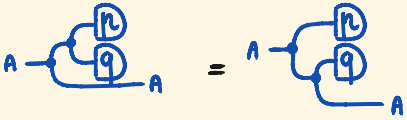
$\rightsquigarrow$ assign to the variable b the value $e(a, b)$

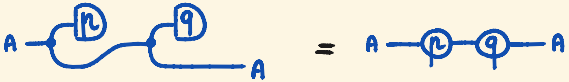
PREDICATES

Morphisms $p : A \to 1$ are fuzzy predicates.

$p \wedge q =$ 

assert $p =$ 

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

$=$ 

$=$ assert p ; assert q

THE STRUCTURE OF $(\mathbf{Stoch}_{\leq}, +, 0)$

$\mathbf{Stoch}_{\leq}$ has coproducts.

For $f_1 : A_1 \to B_1$ and $f_2 : A_2 \to B_2$,

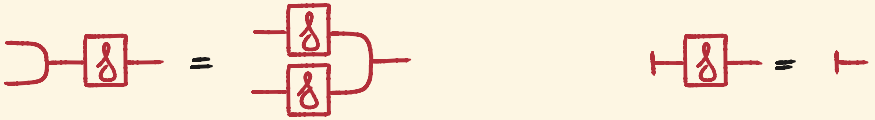
$$f_1 + f_2\,((b,i)\,|\,(a,j)) := f_i(b\,|\,a) \cdot [i = j]$$

Coproducts give natural commutative monoids.

$$\supset\!\!- : A + A \longrightarrow A \qquad \vdash : 0 \to A$$

$$\supset\!\!-\,(a\,|\,(a',i)) := 1 \cdot [a = a'] \qquad \vdash := j_A$$

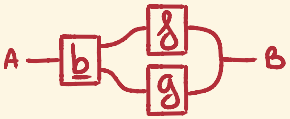
$\rightsquigarrow$ every morphism is a monoid homomorphism



GUARDS

Total morphisms $b : A \longrightarrow 1+1$ give guarded choice

$\underline{b} := A \multimap ; (b \otimes id_A) ; \delta^{-L} =$ 



$\rightsquigarrow$ b decides whether to execute f or g

THE STRUCTURE OF $(\mathbf{Stoch}_{\leq}, +, 0)$

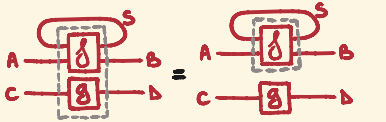
$(\mathbf{Stoch}_{\leq}, +, 0)$ is traced. For $f : S + A \to S + B$,

$$\mathrm{tr}_S(f) : A \to B \qquad \mathrm{tr}_S(f) = \quad$$

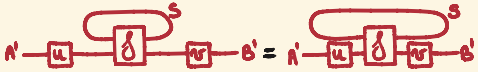


$$\mathrm{tr}_S(f)(b|a) := \sum_{n \in \mathbb{N}} \sum_{s_2, \ldots, s_n \in S} f(s_1|a) \cdot f(s_2|s_1) \cdot \ldots \cdot f(b|s_n)$$

Traces in string diagrams.

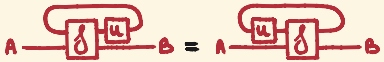






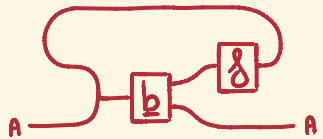






LOOPS AS TRACES

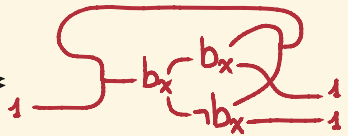
For a guard $b : A \to 1+1$ and a morphism $g : A \to A$,



$\rightsquigarrow$ while b goes left, execute g

Von Neumann's trick.

Consider a coin $b_x : 1 \to 1+1$ in $\mathbf{Stoch}_{\leq}$ of bias $x \in (0,1)$,
$b_x(0) = x$ and $b_x(1) = 1-x$.

Then, $c :=$ 

is a fair coin.

OUTLINE

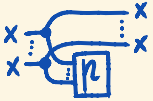
- The structure of discrete probabilistic programs
- The guarded command language
- Imperative categories
- Program logics

INTERPRETING THE GCL

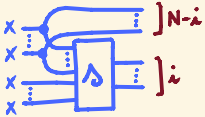
$C ::= \text{skip} \mid \text{abort} \mid C\,C \mid x_i \leftarrow E \mid (x_{i_1}, \ldots, x_{i_k}) \leftsquigarrow S$
$\mid \text{assert } P \mid \text{if } B \text{ then } C \text{ else } C \mid \text{while } B \text{ do } C$

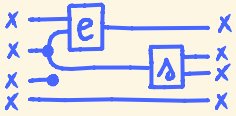
$[\![\text{skip}]\!] :=$ 

$[\![fg]\!] :=$ 

$[\![\text{assert } p]\!] :=$ 

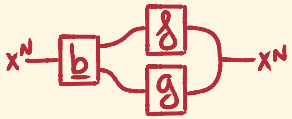
$[\![x_N \leftarrow e]\!] :=$ 

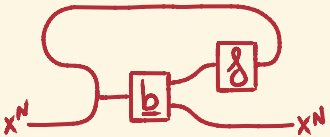
$[\![(x_{N-i+1}, \ldots, x_N) \leftsquigarrow s]\!] :=$ 

ex $[\![x_1 \leftarrow e(x_1, x_2)\ ;\ (x_2, x_3) \leftsquigarrow s(x_2)]\!] =$ 

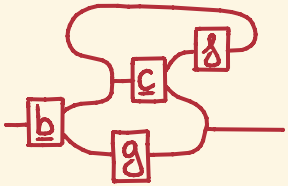
INTERPRETING THE GCL

$C ::= \text{skip} \mid \text{abort} \mid C\,C \mid x_i \leftarrow E \mid (x_{i_1}, \ldots, x_{i_k}) \leftsquigarrow S$
$\mid \text{assert}\ P \mid \text{if}\ B\ \text{then}\ C\ \text{else}\ C \mid \text{while}\ B\ \text{do}\ C$

$[\![\text{if } b \text{ then } f \text{ else } g]\!] :=$ 

$[\![\text{while } b \text{ do } f]\!] :=$ 

$[\![\text{abort}]\!] :=$ 

ex $[\![\text{if } b \text{ then } (\text{while } c \text{ do } f) \text{ else } g]\!] =$ 

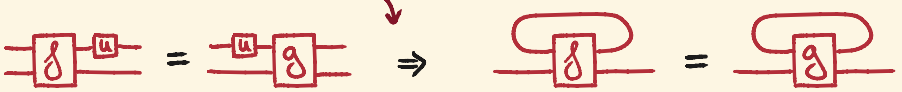
OUTLINE

- The structure of discrete probabilistic programs
- The guarded command language
- [Imperative categories]
- Program logics

IMPERATIVE CATEGORIES

category $\mathcal{C}$ with

- a copy-discard structure $(\otimes, 1)$ $\mapsto$ distributive cd (injections det & tot)
- finite coproducts $(+, 0)$
- uniform coproduct-trace $\mathrm{tr}_S : \mathcal{C}(S+A, S+B) \to \mathcal{C}(A, B)$



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

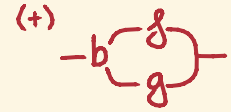
$$\delta^{L}_{A,B,C} : (A \otimes B) + (A \otimes C) \xrightarrow{\simeq} A \otimes (B + C) \qquad \zeta^{L}_{A} : 0 \xrightarrow{\simeq} 0 \otimes A$$

$$\delta^{R}_{A,B,C} : (A \otimes C) + (B \otimes C) \xrightarrow{\simeq} (A + B) \otimes C \qquad \zeta^{R}_{A} : 0 \xrightarrow{\simeq} A \otimes 0$$

IMPERATIVE CATEGORIES

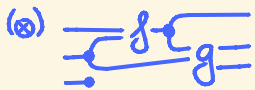
Imperative category = distributive copy-discard category with a compatible uniform coproduct-trace.

- coproducts
  ↳ control-flow



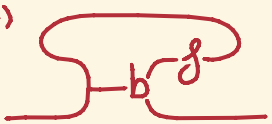
= if b then f else g

- copy-discard monoidal
  ↳ data-flow
  variable sharing



$x_1 \leftarrow f(x_1, x_2)$;
$(x_2, x_3) \leftarrow g(x_1, x_2)$

- uniform coproduct-trace
  ↳ while loops



= while b do f

[cf. Arbib, Manes (1980), Lockett (1993), Hasegawa (1997), Jacobs (2015), ...]

UNIFORMITY GIVES DIVERGENCE

PROPOSITION

$\mathcal{C}$ category with coproducts and uniform trace
$\Rightarrow$ 0 is terminal

PROOF

Consider the morphisms $A\dashv : A \to 0$ defined as $A\dashv := tr_A[id_A, id_A] =$



$\rightsquigarrow$ while TRUE do skip

By uniformity, we show that they are natural.



□

EXAMPLES

Kleisli categories of partially additive monads

- $\mathbf{Par}$, sets and partial functions
- $\mathbf{Rel}$, sets and relations
- $\mathbf{Stoch}_{\leq}$, sets and countably supported partial stochastic kernels
- $\mathbf{BorelStoch}_{\leq}$, standard Borel spaces and measurable partial stochastic kernels

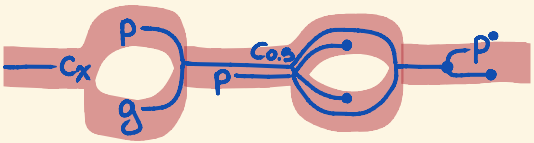
[cf. Jacobs (2010), Jlasuo (2006)]

OUTLINE

- The structure of discrete probabilistic programs
- The guarded command language
- Imperative categories
- [Program logics]

THE PIRAÑHA PROBLEM

```
if coin(x)
| then F₁ ← piranha
| else F₁ ← goldfish
end
F₂ ← piranha
if coin(0.5)
| then s ← F₁
| else s ← F₂
end
observe(s = piranha)
```

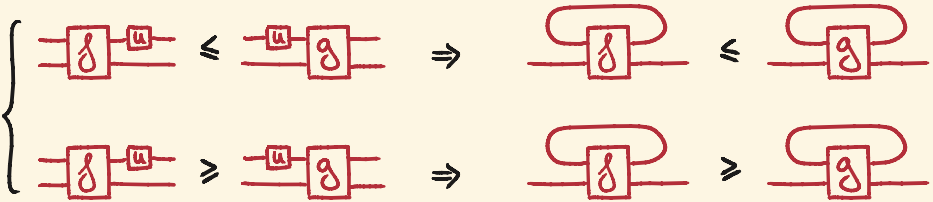


⇝ can we bound the probability of observing piranha ?
↳ we need an order on morphisms

IMPERATIVE POSETAL BICATEGORIES

A imperative category with

- a poset distributive enrichment
- posetally-uniform coproduct-trace



EXAMPLES

All our examples are poset-enriched.

- $\mathbf{Par}$, $f \leq g$ if $\forall a \in \mathrm{dom}\, f \quad f(a) = g(a)$
- $\mathbf{Rel}$, $f \leq g$ if $f \subseteq g$ as relations
- $\mathbf{Stoch}_{\leq}$, $f \leq g$ if $\forall a \in A \; \forall b \in B \quad f(b|a) \leq g(b|a)$
- $\mathbf{BorelStoch}_{\leq}$, $f \leq g$ if $\forall a \in A \; \forall E \in \sigma_B \quad f(E|a) \leq g(E|a)$

PROGRAM TRIPLES

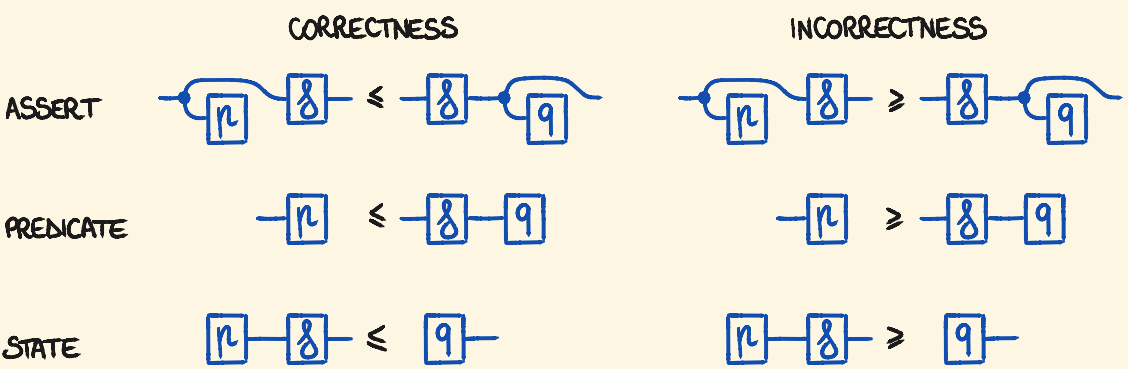
Statements of the form $\{p\}\ S\ \{q\}$.

program

precondition

postcondition

When is a triple valid?



HOARE LOGIC (ASSERT-CORRECTNESS) [Floyd (1969)]

$$\frac{}{\{p\}\ \mathrm{skip}\ \{p\}} \qquad \frac{\{p\}\,f\,\{q\} \quad \{q\}\,g\,\{r\}}{\{p\}\,f;g\,\{r\}} \qquad \frac{p \leqslant p' \quad \{p'\}\,f\,\{q'\} \quad q' \leqslant q}{\{p\}\,f\,\{q\}} \qquad \frac{\{p_1\}\,f\,\{q_1\} \quad \{p_2\}\,f\,\{q_2\}}{\{p_1 \wedge p_2\}\,f\,\{q_1 \wedge q_2\}}$$

$$\frac{}{\{p\}\ \mathrm{abort}\ \{q\}} \qquad \frac{}{\{p(x_1,\ldots,x_i,\ldots,x_n)\}\ x_i \leftarrow e\ \{p\}} \qquad \frac{}{\{q\}\ \mathrm{assert}\ p\ \{p \wedge q\}} \qquad \frac{q \wedge r \leqslant \bot}{\{r \Rightarrow q\}\ \mathrm{assert}\ r\ \{p \wedge \neg b\}}$$

$$\frac{}{\{\bot\}\,f\,\{q\}} \qquad \frac{}{\{p\}\,f\,\{\top\}} \qquad \frac{\{p\}\,f\,\{q\} \quad \{p\}\,g\,\{q\}}{\{p\}\ \mathrm{if}\ b\ \mathrm{then}\ f\ \mathrm{else}\ g\ \{q\}} \qquad \frac{\{p \wedge b\}\,f\,\{q\} \quad \{p \wedge \neg b\}\,g\,\{q\} \quad b\ \mathrm{det}}{\{p\}\ \mathrm{if}\ b\ \mathrm{then}\ f\ \mathrm{else}\ g\ \{q\}}$$

$$\frac{\{p\}\ \mathrm{if}\ b\ \mathrm{then}\ (f;\ \mathrm{while}\ b\ \mathrm{do}\ f)\ \mathrm{else}\ \mathrm{skip}\ \{q\}}{\{p\}\ \mathrm{while}\ b\ \mathrm{do}\ f\ \{q\}} \qquad \frac{\{p\}\,f\,\{p\}}{\{p\}\ \mathrm{while}\ b\ \mathrm{do}\ f\ \{p\}} \qquad \frac{\{p \wedge b\}\,f\,\{p\} \quad b\ \mathrm{det}}{\{p\}\ \mathrm{while}\ b\ \mathrm{do}\ f\ \{p \wedge \neg b\}}$$

THEOREM

The rules of assert-correctness program logics are valid in any poset al imperative category where, for all morphisms f and all predicates p, $\mathrm{abort} \leqslant f$ and $p \leqslant \top$.

INCORRECTNESS LOGIC (STATE - INCORRECTNESS)

[de Vries, Koutavas (2011), O'Hearn (2019)]

$$\frac{}{\{s\}\ \mathsf{skip}\ \{s\}} \qquad \frac{\{s\}\ f\ \{t\} \quad \{t\}\ g\ \{u\}}{\{s\}\ f;g\ \{u\}} \qquad \frac{\{s\}\ f\ \{\bot\}}{\{s\}\ f;g\ \{\bot\}} \qquad \frac{s \geq s' \quad \{s'\}\ f\ \{t'\} \quad t' \geq t}{\{s\}\ f\ \{t\}} \qquad \frac{\{s_1\}\ f\ \{t_1\} \quad \{s_2\}\ f\ \{t_2\} \quad b\ \text{const}}{\{s_1 +_{\mathsf{S}} s_2\}\ f\ \{t_1 +_{\mathsf{S}} t_2\}}$$

$$\frac{}{\{s\}\ x_i \leftarrow e\ \{s[x_i \backslash e]\}} \qquad \frac{}{\{s\}\ (x_{i_1}, \ldots, x_{i_m}) \xleftarrow{\$} s_0\ \{\coprod_{\vec{x}} s \cdot s_0\}} \qquad \frac{}{\{s\}\ \mathsf{abort}\ \{\bot\}} \qquad \frac{}{\{s\}\ f\ \{\bot\}} \qquad \frac{}{\{s\}\ \mathsf{assert}\ p\ \{s \downarrow p\}}$$

$$\frac{\{s \downarrow b\}\ f\ \{t\}}{\{s\}\ \mathsf{if}\ b\ \mathsf{then}\ f\ \mathsf{else}\ g\ \{t\}} \qquad \frac{\{s \downarrow \neg b\}\ g\ \{t\}}{\{s\}\ \mathsf{if}\ b\ \mathsf{then}\ f\ \mathsf{else}\ g\ \{t\}} \qquad \frac{\{s \downarrow b\}\ f;\mathsf{while}\ b\ \mathsf{do}\ f\ \{t\}}{\{s\}\ \mathsf{while}\ b\ \mathsf{do}\ f\ \{t\}} \qquad \frac{}{\{s\}\ \mathsf{while}\ b\ \mathsf{do}\ f\ \{s \downarrow \neg b\}}$$

THEOREM

The rules of state-incorrectness program logics are valid in any poset-al imperative category where, for all morphisms f, $\mathsf{abort} \leqslant f$.

WEAKEST PRECONDITIONS (PREDICATE-CORRECTNESS)

[cf. Kaminski (2018); Zilberstein, Dreyer, Silva (2023)]

$$\frac{}{\{p\}\ \mathsf{skip}\ \{p\}} \qquad \frac{\{p\}\ f\ \{q\} \quad \{q\}\ g\ \{r\}}{\{p\}\ f;g\ \{r\}} \qquad \frac{p \leqslant p' \quad \{p'\}\ f\ \{q'\} \quad q' \leqslant q}{\{p\}\ f\ \{q\}} \qquad \frac{\{p_1\}\ f\ \{q_1\} \quad \{p_2\}\ f\ \{q_2\} \quad b\ \text{const}}{\{p_1 +_b p_2\}\ f\ \{q_1 +_b q_2\}} \qquad \frac{\neg b \wedge q \leqslant \bot \quad b\ \text{det}}{\{p +_b q\}\ \mathsf{assert}\ b\ \{p\}}$$

$$\frac{}{\{p(x_1,\dots,x_{i-1},e,x_{i+1},\dots,x_N)\}\ x_i \leftarrow e\ \{p\}} \qquad \frac{}{\{p(x_{i_1},\dots,x_{i_m} \setminus s)\}\ x_{i_1},\dots,x_{i_m} \overset{\$}{\leftarrow} s\ \{p\}}$$

$$\frac{\{p\}\ \mathsf{if}\ b\ \mathsf{then}\ (f;\mathsf{while}\ b\ \mathsf{do}\ f)\ \mathsf{else}\ \mathsf{skip}\ \{q\}}{\{p\}\ \mathsf{while}\ b\ \mathsf{do}\ f\ \{q\}}$$

$$\frac{\{p \wedge b\}\ f\ \{q\} \quad \{p \wedge \neg b\}\ g\ \{q\} \quad b\ \text{det}}{\{p\}\ \mathsf{if}\ b\ \mathsf{then}\ f\ \mathsf{else}\ g\ \{q\}} \qquad \frac{\{p\}\ f\ \{q\} \quad \{p\}\ g\ \{q\}}{\{p\}\ \mathsf{if}\ b\ \mathsf{then}\ f\ \mathsf{else}\ g\ \{q\}} \qquad \frac{}{\{\bot\}\ f\ \{q\}} \qquad \frac{}{\{p \wedge q\}\ \mathsf{assert}\ p\ \{q\}}$$

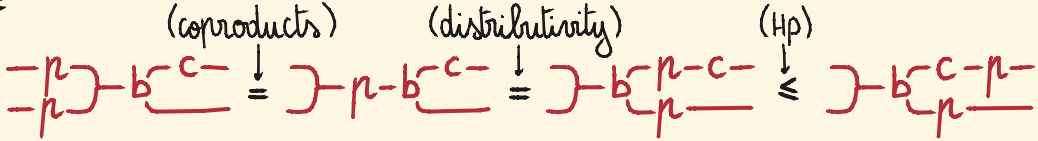
THEOREM

The rules of predicate-correctness program logics are valid in any posetal imperative category where, for all morphisms f, $\mathsf{abort} \leqslant f$.

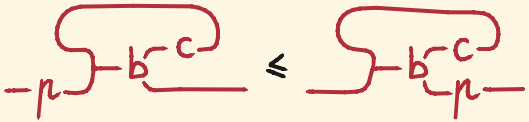
EXAMPLE DERIVATION: LOOP INVARIANTS

$$\frac{[\![\text{assert } p]\!] ; [\![c]\!] \leqslant [\![c]\!] ; [\![\text{assert } p]\!]}{[\![\text{assert } p]\!] ; [\![\text{while } b \text{ do } c]\!] \leqslant [\![\text{while } b \text{ do } c]\!] ; [\![\text{assert } p]\!]}$$

PROOF



(uniformity) ⇒

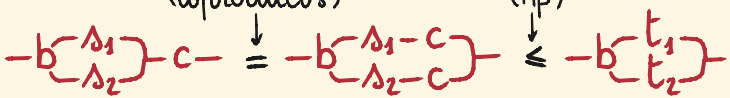


□

EXAMPLE DERIVATION: GUARDED DISJUNCTION

$$\frac{[\![s_1]\!] \,;\, [\![c]\!] \leqslant [\![t_1]\!] \qquad [\![s_2]\!] \,;\, [\![c]\!] \leqslant [\![t_2]\!]}{[\![s_1 +_b s_2]\!] \,;\, [\![c]\!] \leqslant [\![t_1 +_b t_2]\!]}$$

PROOF

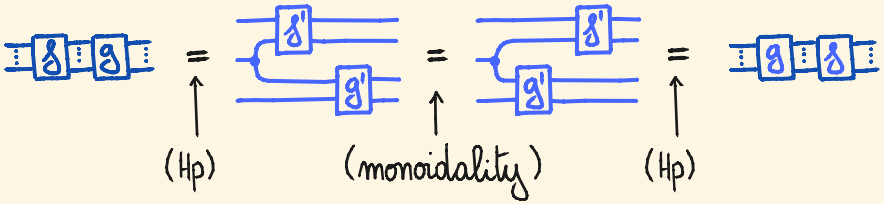


□

EXAMPLE DERIVATION: INTERCHANGE

$$\frac{freeVar(f) \cap modVar(g) = \emptyset \qquad freeVar(g) \cap modVar(f) = \emptyset}{[\![f;g]\!] = [\![g;f]\!]}$$

PROOF



□

CONCLUSIONS

- We propose imperative categories as denotational semantics of imperative programs with effects
- We derive the rules of multiple program logics from the categorical structure

MORE IN THE PAPER

- A monoidal guarded command language and its algebra (cf. GKAT [Smolka, Foster, Hsu, Kappé, Kozen, Silva (2019)])
- Relational program logics

FUTURE WORK

- More expressive assertions: (nonconvex) sets of distributions
- Relative completeness

BONUS SLIDES

THE GUARDED COMMAND LANGUAGE (GCL)

- Signature

  $\mathsf{Var}, \Sigma_E, \Sigma_G, \Sigma_P, \Sigma_C, \Sigma_D$

- Expressions

  $E ::= x \in \mathsf{Var} \mid e(E, \ldots, E),\ e \in \Sigma_E$

- Guards

  $G ::= b(E, \ldots, E),\ b \in \Sigma_G \mid \mathsf{L} \mid \mathsf{R} \mid G \wedge G \mid \neg G$

- Predicates

  $P ::= p(E, \ldots, E),\ p \in \Sigma_P \mid \top \mid \bot \mid P \wedge P \mid P +_G P \mid \breve{G}$

- Commands

  $C ::= c \in \Sigma_C \mid \mathsf{skip} \mid \mathsf{abort} \mid x \leftarrow E \mid (x_1, \ldots, x_m) \xleftarrow{\$} \sigma,\ \sigma \in \Sigma_D \mid C\,C$
  $\mid \mathsf{assert}\ P \mid \mathsf{if}\ B\ \mathsf{then}\ C\ \mathsf{else}\ C \mid \mathsf{while}\ B\ \mathsf{do}\ C$

INTERPRETING THE GUARDED COMMAND LANGUAGE

- Choose a type X for variables and fix $N := \#\mathrm{Var}$.
- Commands are endomorphisms of $X^N = \underbrace{X \otimes \cdots \otimes X}_{N \text{ times}}$
  ⇝ state-passing translation
- Expressions are deterministic and total $X^N \longrightarrow X$
- Guards are total $X^N \longrightarrow 1+1$
- Predicates are $X^N \longrightarrow 1$

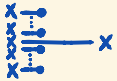
→ we also constructed an internal language (in the paper)
  ↳ succint with few primitives
  ↳ programs with arbitrary types

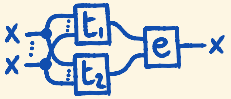
INTERPRETING THE GCL : EXPRESSIONS

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

Choose $\bar{e} : X^n \longrightarrow X$ for each $e \in \Sigma_E(n)$.

Define

$[\![x_i]\!] := \pi_i =$ 

$[\![e(t_1, \dots, t_m)]\!] := \Delta^n_m ; ([\![t_1]\!] \otimes \dots \otimes [\![t_m]\!]) ; \bar{e} \overset{m=2}{=}$ 

⇒ copy-discard structure for variable handling
↳ compatible cocommutative comonoid on every object

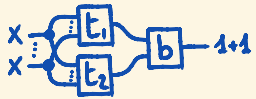
INTERPRETING THE GCL : GUARDS

$G ::= b(E,\dots,E),\ b \in \Sigma_G(m) \mid L \mid R \mid G \wedge G \mid \neg G$

Choose $\mathfrak{b} : X^m \longrightarrow 1+1$ total for each $b \in \Sigma_G(m)$

Define

$[\![b(t_1,\dots,t_m)]\!] := X^N \to m \,;\, ([\![t_1]\!] \otimes \cdots \otimes [\![t_m]\!]) \,;\, \mathfrak{b}$ $\overset{m=2}{=}$



$[\![L]\!] := X^N \to 0 \,;\, 1 \vdash 1 = X^N \to 0 \vdash 1$



$[\![b_1 \wedge b_2]\!] := X^N \to [\![b_1]\!] \,;\, [\![b_2]\!] \,;\, 1 \vdash 1$

$[\![R]\!] := X^N \to 0 \,;\, 1 \vdash 1 = X^N \vdash 0 \to 1$

$[\![\neg b]\!] := X^N \to \mathfrak{b} \,;\, 1 \vdash 1$

⇒ coproducts for choices

INTERPRETING THE GCL : PREDICATES

$P ::= p(E,\dots,E),\ p \in \Sigma_P(n) \mid \top \mid \bot \mid P \wedge P \mid P +_6 P \mid \breve{G}$

Choose $\bar{p} : X^n \longrightarrow 1$ for each $p \in \Sigma_P(n)$

Define

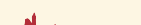
$[\![p(t_1,\dots,t_m)]\!] := X^N \to m \,;\, ([\![t_1]\!] \otimes \cdots \otimes [\![t_m]\!]) \,;\, \bar{p}$ $\overset{m=2}{=}$



$[\![\top]\!] :=$



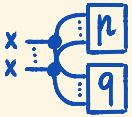
$[\![\bot]\!] :=$



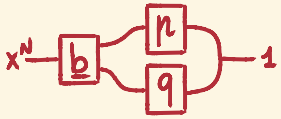
$[\![\breve{b}]\!] :=$



$[\![p \wedge q]\!] :=$



$[\![p +_b q]\!] :=$



RELATIONAL PROGRAM TRIPLES

Statements of the form $\{p\}\ f \sim g\ \{q\}$.

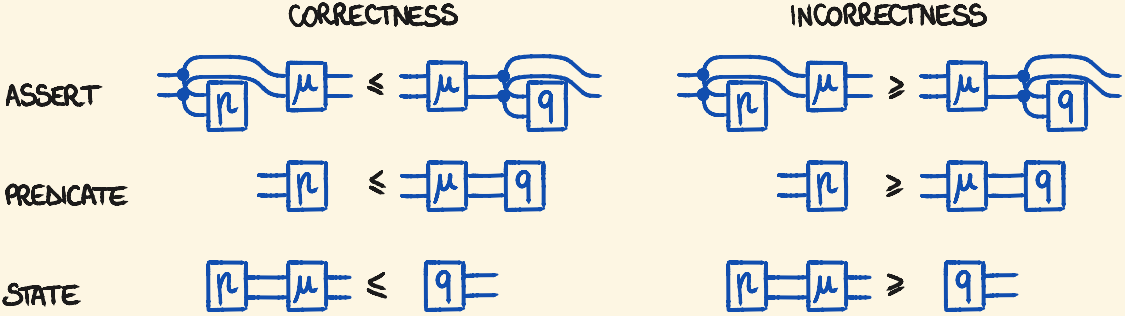
programs

precondition

postcondition

When is a triple valid?

There is a coupling $\mu \triangleright f \& g$ such that



[Benton (2004)] [Avanzini, Barthe, Davoli, Grégoire (2025)]

COUPLINGS

A strict coupling μ of $f : A \to A$ and $g : B \to B$ is $\mu : A \otimes B \to A \otimes B$ st



$\rightsquigarrow$ this forces f and g to have the same termination behaviour

A coupling μ of $f : A \to A$ and $g : B \to B$ is $\mu : A \otimes B \to A \otimes B + A + B$ st



RELATIONAL HOARE LOGIC (ASSERT-CORRECTNESS)

$$\frac{}{\{p\}\ \mathsf{skip} \sim \mathsf{skip}\ \{p\}} \qquad \frac{\{p\}\ c_1 \sim d_1\ \{q\} \quad \{q\}\ c_2 \sim d_2\ \{r\}}{\{p\}\ c_1 c_2 \sim d_1 d_2\ \{r\}} \qquad \frac{p \leq p' \quad \{p'\}\ c \sim d\ \{q'\} \quad q' \leq q}{\{p\}\ c \sim d\ \{q\}} \qquad \frac{\{p\}\ c \sim d\ \{q\}}{\{\sigma p\}\ c \sim d\ \{\sigma q\}}$$

$$\frac{}{\{p[(u_1,u_2)/(e_1,e_2)]\}\ u_1 \leftarrow e_1 \sim u_2 \leftarrow e_2\ \{p\}} \qquad \frac{\{p\}\ c_1 \sim c_2\ \{q\} \quad \{p\}\ c_1 \sim d_2\ \{q\} \quad \{p\}\ d_1 \sim c_2\ \{q\} \quad \{p\}\ d_1 \sim d_2\ \{q\}}{\{p\}\ \mathsf{if}\ b_1\ \mathsf{then}\ c_1\ \mathsf{else}\ d_1 \sim \mathsf{if}\ b_2\ \mathsf{then}\ c_2\ \mathsf{else}\ d_2\ \{q\}}$$

$$\frac{\{(\vec{b}_1 \circledast \vec{b}_2) \wedge p\}\ c_1 \sim c_2\ \{q\} \quad \{(\neg\vec{b}_1 \circledast \neg\vec{b}_2) \wedge p\}\ d_1 \sim d_2\ \{q\} \quad b_1, b_2\ \mathsf{det}}{\{(b_1 = b_2) \wedge p\}\ \mathsf{if}\ b_1\ \mathsf{then}\ c_1\ \mathsf{else}\ d_1 \sim \mathsf{if}\ b_2\ \mathsf{then}\ c_2\ \mathsf{else}\ d_2\ \{q\}}$$

$$\frac{\{(\vec{b}_1 \circledast \vec{b}_2) \wedge p\}\ c_1 \sim c_2\ \{(b_1 = b_2) \wedge p\} \quad b_1, b_2\ \mathsf{det}}{\{(b_1 = b_2) \wedge p\}\ \mathsf{while}\ b_1\ \mathsf{do}\ c_1 \sim \mathsf{while}\ b_2\ \mathsf{do}\ c_2\ \{(\neg\vec{b}_1 \circledast \neg\vec{b}_2) \wedge p\}} \qquad \frac{\{p\}\ c_1 \sim c_2\ \{p\} \quad \{p\}\ c_1 \sim \mathsf{skip}\ \{p\} \quad \{p\}\ \mathsf{skip} \sim c_2\ \{p\}}{\{p\}\ \mathsf{while}\ b_1\ \mathsf{do}\ c_1 \sim \mathsf{while}\ b_2\ \mathsf{do}\ c_2\ \{p\}}$$

+ one-sided rules

THEOREM

The rules of relational assert-correctness program logics are valid in any poretal imperative category where, for all morphisms f, $\mathsf{abort} \leq f$.

GUARDED MONOIDS

- $(C, 1, \cdot, 0)$ monoid with absorbing element
- $(B, (-+_{(-)}-) : B^3 \to B, \neg(-) : B \to B, L \in B)$ such that

$$a +_L b = a \qquad \neg(a +_b c) = \neg a +_b \neg c$$
$$L +_a \neg L = a \qquad (a +_b c) +_d (e +_b g) = (a +_d e) +_b (c +_d g)$$
$$a +_b c = c +_{\neg b} a \qquad (a +_b c) +_d (a +_e c) = a +_{(b +_d e)} c$$

- an operation $(-\oplus_{(-)}-) : C \times B \times C \to C$ such that

$$g +_L g = g \qquad (g +_b g) \cdot h = (g \cdot h) +_b (g \cdot h)$$
$$g +_b g = g +_{\neg b} g \qquad (g +_b g) +_a (h +_b l) = (g +_a h) +_b (g +_a l)$$
$$(g +_a g) +_b (g +_c g) = g +_{(a +_b c)} g$$