Two types of **do** – Functional Programming with Interventions and Counterfactuals

Dario Stein – Radboud University Nijmegen

DutchCATS, Leiden – 5 June 2024

What **do** do you mean?

Haskell do-notation

A notation for effectful computation with monads

```
main :: IO ()
main = do
  name ← readLine
  return ("Hello" ++ name)
```

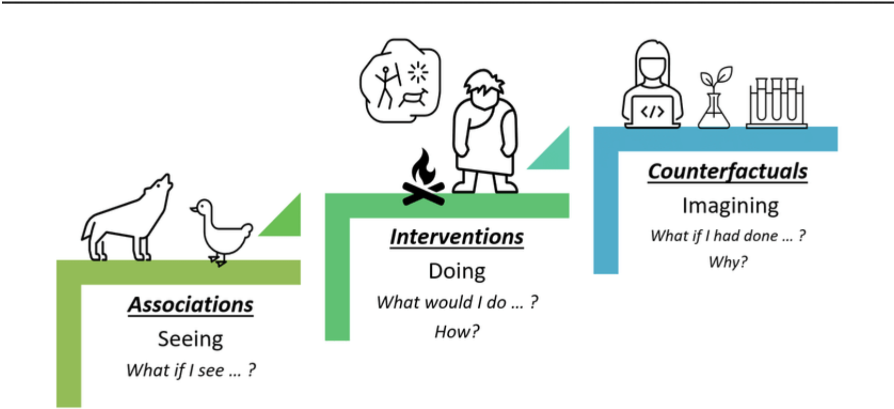
see Moggi's monadic metalanguage [Moggi'91]

Pearl's do-calculus

A calculus for **causal inference** in statistics, e.g. [Pearl'09]

$$P(Y|\mathbf{do}(X)) = \sum_z P(Y|X, Z = z)P(Z = z)$$

Pearl's Ladder of Causation



from [Carey&Wu'22]

A simple example in causal reasoning

We have two random variables X (state of a machine), and Y (indicator lamp) that are highly correlated

$$P_{XY} = \begin{pmatrix} 0.64 & 0.16 \\ 0.16 & 0.04 \end{pmatrix}$$

Question

What happens to the lamp if I turn on the machine?

Attempt 1

$$P(Y = 1|X = 1) = \frac{0.64}{0.64 + 0.16} = 0.8$$

but this is just the *correlation* (rung I). What really happens?

A simple example in causal reasoning

Interventional Question (rung II)

What happens to the lamp if I turn on the machine?

$$P(Y = 1|\mathbf{do}(X = 1)) = ??$$

We can't answer query this with the present information. Many scenarios are consistent with the rung I observations

1. machine determines lamp $X \to Y$
2. lamp determines machine $Y \to X$
3. common cause $X \leftarrow A \to Y$

A simple example in causal reasoning

What do we need to answer causal queries?

1. associations: joint distribution
2. interventions: **Bayesian networks**
3. counterfactuals: **structural causal models**

To a computer scientist:

- the **extensional meaning** of a probabilistic program is its joint distribution (rung I)
- climbing Pearl's ladder requires some **intensional aspects** (causal structure)

Goal: Capture intervenable computation with types and monads

A Programmers's View of Interventions

Good News: I need not explain you what causal structures are. Every program already has one!

```
x ← machine()
y ← display(x)
```

```
y ← button()
x ← control(x)
```

```
a ← commoncause()
x ← machine(a)
y ← display(a,x)
```

A Programmers's View of Interventions

Good News: Pearl's **do** is a simple program transformation: for $do(x = \mathbf{true})$

```
x ← true
y ← display(x)
```

```
y ← button()
x ← true
```

```
a ← commoncause()
x ← true
y ← display(a,x)
```

A Programmers's View of Interventions

Good News: Pearl's **do** is a simple program transformation
Bad News: Whole-program transformations are messy.

1. What exactly can we intervene on?
2. How to make it type safe?

Let's be explicit about the extra intensionality $\rightarrow$ Typed **intervention points**

$\mathtt{x} \leftarrow \mathit{int}(\mathtt{machine()})$
$\mathtt{y} \leftarrow \mathtt{display(x)}$

The int function returns its argument, but also creates an intervention point to which we can later return and supply an different argument.

A Programmers's View of Interventions

For a list $\boldsymbol{A} = [A_1, \ldots, A_n]$ of types, let

$$\boldsymbol{A}? = \prod_{i=1}^{n} (A_i + 1)$$

Then a computation with intervention points $\boldsymbol{A}$ is a function $X \to \boldsymbol{A}? \to Y$

- for each A_i, we can decide to do nothing (inr) or intervene ($\mathrm{inl}(a_i)$).

A Programmers's View of Interventions

Graded monad of interventions

The type constructor $\mathrm{Int}^{\boldsymbol{A}}(X) = \boldsymbol{A}? \to X$ is a graded monad over lists of types

1. $\eta : X \to \mathrm{Int}^{[]}(X)$
2. $\gg\!= : \mathrm{Int}^{\boldsymbol{A}}(X) \to (X \to \mathrm{Int}^{\boldsymbol{B}}(Y)) \to \mathrm{Int}^{\boldsymbol{A} + \!\!+ \boldsymbol{B}}(Y)$

To define

$$\begin{aligned} \gg\!= &: (\boldsymbol{A}? \to X) \to (X \to \boldsymbol{B}? \to Y) \to (\boldsymbol{A} + \!\!+ \boldsymbol{B})? \to Y \\ F \gg\!= G &= \lambda(\boldsymbol{a}, \boldsymbol{b}).G(F(\boldsymbol{a}))(\boldsymbol{b}) \end{aligned}$$

we use the isomorphism $(\boldsymbol{A} + \!\!+ \boldsymbol{B})? \cong \boldsymbol{A}? \times \boldsymbol{B}?$

Intervention points are created using *int* : $X \to \mathrm{Int}^{[X]}(X)$

$$int(x_1)(inr) = x_1, \qquad int(x_1)(inl(x_2)) = x_2$$

A Programmers's View of Interventions

do-notation for graded monads works directly in Agda

```
display : Bool → Int [ Bool ] (Bool x Bool)
display machine = do
  x ← int(machine)
  y ← display(x)
  return (x , y)
```

Computations now have two ‘directions‘ of effects: data (probability) and grading (intervention points)

A Categorists's View of Interventions

How to manipulate the grading? Grading in monoid $\to$ grading in monoidal category.

General semantics

Let $(\mathbb{C}, \otimes, 1)$ be a semicartesian closed category with coproducts, and let $\mathcal{I}$ be the free smc over $\mathrm{ob}(\mathbb{C})$.

1. objects are lists $\boldsymbol{A} = [A_1, \ldots, A_n]$ of $\mathbb{C}$-objects
2. morphisms are welltyped permutations

Then $\mathrm{Int}^{\boldsymbol{A}}(X) = \boldsymbol{A}? \multimap X$ defines an $\mathcal{I}$-graded monad, i.e. a lax monoidal functor

$$\mathrm{Int} : \mathcal{I} \to [\mathbb{C}, \mathbb{C}]$$

A Categorists's View of Interventions

There are many different views on the same construction, e.g.

- $\mathbb{C}$ has the structure of an *actegory* (i.e. monoidal action)

  $$\bullet : \mathcal{I} \times \mathbb{C} \to \mathbb{C}, (\boldsymbol{A}, X) \mapsto \boldsymbol{A}? \otimes X$$

  This works even without closed structure on $\mathbb{C}$; we just need $\mathbb{C}$ to be semicartesian distributive (relevant examples: **FinStoch**, **sfKer**)
- we can form the monoidal bicategory $\mathrm{Para}_\bullet(\mathbb{C})$ where morphisms $X \to Y$ are pairs $(\boldsymbol{A}, f)$ with $f : \boldsymbol{A} \bullet X \to Y$. But this loses access to the grading.

A Double Category of Interventions

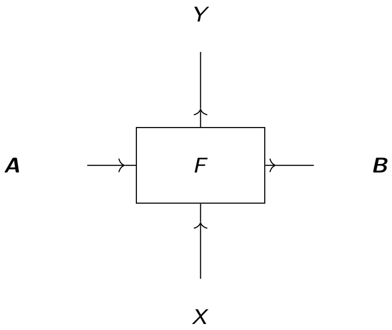
Define a one-object double category as follows

- vertical morphisms are gradings $\boldsymbol{A}$ $\leftarrow$ strict composition with $+\!\!+$
- horizontal morphisms are types X $\leftarrow$ weak composition with $\otimes$
- squares are bi-graded computations $F : \boldsymbol{A} \bullet X \to \boldsymbol{B} \bullet Y$

$$\begin{tikzcd}
* \arrow[r, "Y"] & * \\
* \arrow[u, "\boldsymbol{A}"] \arrow[r, "X"'] \arrow[ur, phantom, "F"] & * \arrow[u, "\boldsymbol{B}"']
\end{tikzcd}$$

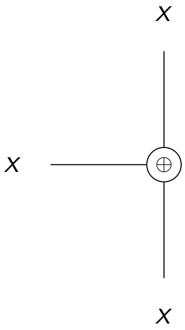
String Diagrams for Double Categories

Using the graphical calculus of double categories [Myers'16], we draw squares as



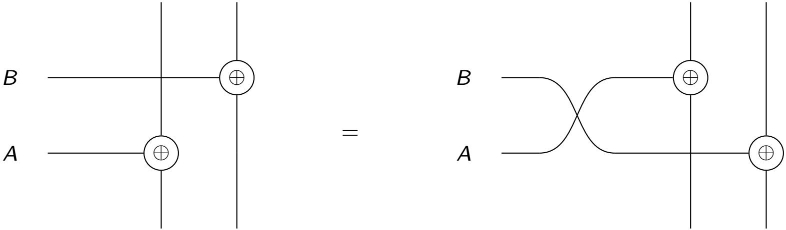
String Diagrams for Interventions

Creating an intervention point is represented as follows



String Diagrams for Interventions

The graded interchange law

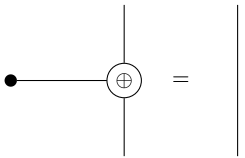


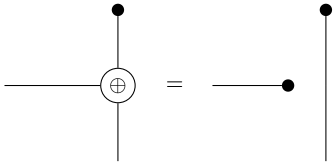
String Diagrams for Interventions

Additional structure: Intervention points can be

1. created out of nowhere (we don't use their result)
2. terminated (we supply inr)

$\Rightarrow$ the canonical choice of grading is partial type-preserving injections ($\mathcal{I} = \mathbf{pInj}$).



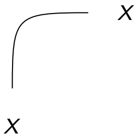


String Diagrams for Interventions

Lastly, we define Pearl's do-operator as the left inclusion

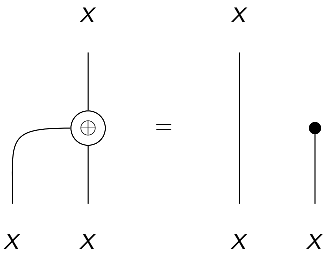
$$\mathrm{do} : X \to [X] \bullet 1 = (X + 1) \otimes 1$$

This can be represented as bend from the data direction to the grading direction.



and satisfies the expected equations of an intervention

String Diagrams for Interventions



Note the other 3 bends don't have a meaningful interpretation. There is no calculus of conjoints/mates in this case.

More Context

A very general notion of grading (originally [Wood'76,78], recently [Levy'19])

Definition (Locally graded category)

Let $\mathcal{I}$ be semicartesian. A local $\mathcal{I}$-grading of a monoidal category $\mathbb{C}$ has

1. for each grade $a \in \mathrm{ob}(\mathcal{I})$ graded homsets $\mathbb{C}_a(X, Y)$
2. for each $\rho : a \to b$ a pullback action $\rho^* : \mathbb{C}_b(X, Y) \to \mathbb{C}_a(X, Y)$
3. graded composition $\circ : \mathbb{C}_a(X, Y) \times \mathbb{C}_b(Y, Z) \to \mathbb{C}_{a \otimes b}(X, Z)$
4. graded tensor $\otimes : \mathbb{C}_a(X_1, Y_1) \times \mathbb{C}_b(X_2, Y_2) \to \mathbb{C}_{a \otimes b}(X_1 \otimes X_2, Y_1 \otimes Y_2)$

which suitably cohere and coincide with $\mathbb{C}$ for $a = 1$.

More Context

Proposition [Levy'19]

A local $\mathcal{I}$-grading is the same as an $([\mathcal{I}^{\mathrm{op}}, \mathbf{Set}], \widehat{\otimes})$-enrichment, where $\widehat{\otimes}$ is Day convolution.

Interesting: The topos $[\mathbf{pInj}^{\mathrm{op}}, \mathbf{Set}]$ is the 'Staton topos' [Pitts], the category of nominal restriction sets (and monadic over **Nom**).

To summarize

tl;dr We studied intervenable computation in a graded monadic framework: The two types of **do** finally meet.

There are two orthogonal ‘directions’ of dataflow (probability & grading), and a double-categorical graphical calculus to relate them.

Future Work

1. Zoo of different presentations: actegories, bicategories, double categories, graded monads, locally graded categories ... Which one is the most convenient?
2. What are convenient type theories for such things?
3. Study connections of interventions to name generation via enrichment
4. Liell-Cock & Staton recently used local grading to study probability+nondeterminism. Grading gets around the Eckmann-Hilton argument (where two symmetric idempotent operations $+_{0.5}$, $\vee$ must be identified)
5. Extend to Rung III (Counterfactuals): might involve grading in trace types [Lew&al'19]
6. Apply to existing probabilistic programming languages for causal inference (OmegaC, ChiRho).

References

1. Judea Pearl, "Causality" (2009)
2. Jack Liell-Cock, Sam Staton. "Compositional imprecise probability." arXiv preprint arXiv:2405.09391 (2024).
3. Paul Blain Levy, "Locally graded categories" (2019), `https://www.cs.bham.ac.uk/~pbl/papers/locgrade.pdf`
4. Bruno Gavranović. "Fundamental Components of Deep Learning: A category-theoretic approach." (PhD thesis)
5. Zenna Tavares, et al. "A language for counterfactual generative models." International conference on machine learning. PMLR (2021)
6. `https://github.com/BasisResearch/chirho`