CAUSAL PROBABILISTIC PROGRAMMING VIA NON-ASSOCIATIVE DO-NOTATION

MARIO ROMÁN

ABSTRACT. We introduce a foundational framework for causal probabilistic programming based on magmadic do-notation and the consideration of identifiability problems as derived constructs in probabilistic programming, rather than primitives.

1. INTRODUCTION

1.1. **Example — confounding in observational data.** In 1986, an observational study by Charig et al. [CWPW86] reported the relative success of different modalities of kidney stone surgery. Attending to the study abstract, *open surgery* seemed slightly less successful than *closed surgery*: "success was achieved in 273 (78%) patients after open surgery, [and] 289 (83%) after percutaneous nephrolithotomy [closed surgery]".

However, the observational data happened to also be segregated by stone diameter. After separating patients by the diameter of the stone, one could observe that open surgery was separately more successful for small stones ($< 2\,\text{cm}$), with a success rate of 81/87(93%) versus 234/270(87%); and for large stones ($\geq 2\,\text{cm}$), with a success rate of 192/263(73%) versus 55/80(69%). The explanation for this apparent paradox is that the choice of treatment was being influenced by the severity of the case — while smaller stones were treated by closed surgery, difficult larger stones were systematically sent for open surgery.

Surgery	Open		Closed	
Diameter	Small	Large	Small	Large
Success	81	192	234	55
Failure	6	71	36	25

FIGURE 1. Data on kidney stone surgery [CWPW86].

This was a real-world instance of *Simpson's paradox*, as noted in 1994 by Julious and Mullee [JM94]. Their comment concluded that *"randomised trials are therefore necessary to demonstrate any treatment effect."* Certainly, randomised trials would have been sufficient: e.g., if we had truly randomized the application of open or closed surgery, we would have been able to compare both under equal conditions.

But do we really *need* randomised trials to estimate the treatment effect?

Causal inference is the problem of estimating effects from observational data and a causal model [Pea09], and the solvability of causal inference problems is decidable [SP08]. Let us introduce first probabilistic inference and then causal inference.

Date: August 18, 2026.
Key words and phrases. Category theory, categorical semantics.

1.2. **Probabilistic inference.** Probabilistic inference is the problem of estimating a conditional probability from observational data and a statistical model. It is a mathematically-solved problem, but it is conceptually difficult to apply correctly. Probabilistic programming is the paradigm where statistical models are first-class citizens of the language; it facilitates probabilistic inference.

```
(do (stone surgery outcome) <- observationalData
    () <- (observe surgery 'open)
    return (outcome))
> '(((success) 39/50) ((failure) 11/50))

(do (stone surgery outcome) <- observationalData
    () <- (observe surgery 'closed)
    return (outcome))
> '(((success) 289/350) ((failure) 61/350))
```

FIGURE 2. Naive surgery outcome estimation.

However, a naive application of probabilistic inference would fall prey to the confounding variable problem (see the program in Figure 2). Conditioning on the observation does not help, for we really need a causal model to identify the problem. Endowed with a causal model, causal inference is solvable [SP08].

1.3. **Example — causal reasoning with observational data.** In particular, the problem of estimating a randomised control trial from these data is solvable.

(1) Note that — assuming that observational data is representative, as the original study did — we can estimate the prevalence of small and large stones from the observational data. Small stones (51%) are slightly more frequent than large stones (49%)[1].
(2) We can estimate the success of open surgery by weighting conditional probabilities by stone diameter, $51\% \cdot 93\% + 49\% \cdot 73\% \approx 83\%$.
(3) Analogously, we can estimate the success of closed surgery by weighting conditional probabilities, $51\% \cdot 87\% + 49\% \cdot 69\% \approx 78\%$.

We thus have a mathematically valid estimator for a randomized control trial. Of course, the validity of this estimator relies on how the original data was collected; but let us note that no randomized controlled trial is necessary for this estimate.

1.4. **Causal probabilistic programming.** Causal inference is a mathematically-solved problem [SP08], but it is still tedious to apply correctly. In the same sense, that probabilistic inference was rendered much easier by *probabilistic programming*. Could we render *causal inference* easier with *causal probabilistic programming*?

Causal programming has been the subject of recent theoretical work [Cc23, WSB+]. We introduce a simple syntax and algebraic denotational semantics for causal programming. Our technique is not to consider *causal programming* as a new paradigm requiring new primitives, but to introduce derived constructs on top of *probabilistic programming*.

[1]This assumption seems unrealistic and an artifact of how data was collected in the original study. Small stones seem to be exceedingly more common in reality. Let us assume the original data was unbiased, or that we would take another source to estimate prevalence.

As an example, consider the problem that we presented above, and consider that we want to estimate the effect of a 1:1 randomized control trial: we uniformly assign patients to an arm and interpret an intervention on the observational data. The following would be our query.

```
(do (arm) <- (uniform '(open) '(closed))
    (outcome) <- (Intervene observationalData
                  WithModel (do
                    stone <- ()
                    surgery <- (stone)
                    outcome <- (stone surgery)
                    return (stone surgery outcome))
                  Setting (surgery) To (arm)
                  In (outcome))
    return (arm outcome))
>>> '(((open success) 634983/1525400)
>>>  ((open failure) 127717/1525400)
>>>  ((closed success) 6231/16000)
>>>  ((closed failure) 1769/16000))
```

Most of the weightlifting seems to be on the "Intervene" operator, which takes a distribution and a causal model and yields the intervened distribution — or an exception when impossible. This "Intervene" operator is not a primitive itself: any program using it can be rewritten into a program not doing so; this is a result that relies on completeness for causal inference [SP08]. For instance, before evaluating it, the previous causal program got automatically rewritten as follows.

```
(do (arm) <- (uniform '(open) '(closed))
    (outcome) <- (do
       (stone) <- (do
          (stone20 surgery21 outcome22) <- observationalData
          return (stone20))
       (outcome) <- (do
          (stone24 surgery25 outcome26) <- observationalData
          () <- (observe arm surgery25)
          () <- (observe stone stone24)
          return (outcome26))
       return (outcome))
    return (arm outcome))
```

From this point of view, the most important primitive of our syntax is its only one: do-notation sequentiation of programs. We will use a non-associative variant of do-notation that we recall first [DRS26] (Section 2). The last section rewrites Shpitser and Pearl's algorithm for identifiability in terms of do-notation (Section 3).

1.5. **Related work.** Literature towards a denotational semantics of causal probabilistic programming has not yet explicitly addressed the problem of identifiability [WSB+]. We follow the identifiability algorithm of Shpitser and Pearl [SP08].

2. Non-associative Do-notation

Do-notation is right-associating by default [Mog91, Wad92]. We instead employ left-associating do-notation. We take semantics in the magmad — the non-associative monad — of normalized distributions [DRS26].

2.1. Magmad of normalized distributions.

We recall the magmad, or *non-associative monad*, of normalized distributions [DRS26]. A normalized distribution is either an empty formal sum — meaning it "fails", or that it is partially defined — or a full distribution. Normalized distributions provide normalized stochastic semantics, but they do not form a monad.

Definition 2.1.1 (Magmad of normalized distributions)**.** A *normalized distribution* over a set is a finite formal sum over its elements whose coefficients are positive and add up to exactly 1 or 0. The *finitary normalized distribution magmad*, $\mathbf{N}\colon \mathsf{Set} \to \mathsf{Set}$, assigns, to each set, the set of normalized distributions over it,

$$\mathbf{N}(X) = \left\{ \sum_{i=0}^{n} \lambda_i \left|x_i\right\rangle \;\middle|\; x_i \in X,\ \lambda_i \in \mathbb{R}^+,\ \sum_{i=0}^{n} \lambda_i = 1 \text{ or } 0 \right\}.$$

Its unit, $\eta^{\mathbf{N}}\colon X \to \mathbf{N}X$, is defined by $\eta^{\mathbf{N}}(x) = 1\left|x\right\rangle$. Its binding, $\beta^{\mathbf{N}}\colon \mathbf{N}X \times (X \to \mathbf{N}Y) \to \mathbf{N}Y$, is defined by

$$\beta^{\mathbf{N}}\left(\sum_{i=0}^{n} \lambda_i \left|x_i\right\rangle ; \left[x_i \mapsto \sum_{j=0}^{m_i} \rho_{ij} \left|y_j\right\rangle\right]\right) = \frac{1}{\sum_{i=0}^{n}\sum_{j=0}^{m_i} \lambda_i\rho_{ij}} \sum_{i=0}^{n}\sum_{j=0}^{m_i} \lambda_i\rho_{ij} \left|y_j\right\rangle.$$

Note that, whenever $\sum_{i=0}^{n}\sum_{j=0}^{m_i} \lambda_i\rho_{ij} = 0$, then there must be no summands and the value of the binding is the empty formal sum, 0.

Magmadic do notation. Because we do not have a monad but only a magmad, the usual monadic programming language [Mog91] can take two meanings depending on where we associate by default. Moreover, nested do-statements do not reduce to a single do-statement, as they would do if we were to assume associativity.

Definition 2.1.2 (Magmadic do notation)**.** Magmadic do notation is defined recursively by the following rules. The last rule *(iv)* groups two lines, using $m = \beta(m_1; \lambda x.\beta(m_2; \lambda y.\eta(x,y)))$; the rest of the rules *(i,ii,iii)* deal with the three base cases.

```
(do return (y ...))
```
$\overset{(i)}{=}\ \eta(y_1, ..., y_m)$

```
(do (x ...) <- m
    return (y ...))
```
$\overset{(ii)}{=}\ \beta(m; \lambda(x_1, ..., x_n).\ \eta(y_1, ..., y_m)).$

```
(do (x ...) <- m1
    (y ...) <- m2
    return (z ...))
```
$\overset{(iii)}{=}\ \beta(m_1; \lambda(x_1, ..., x_n).\ \beta(m_2; \lambda(y_1, ..., y_m).\ \eta(z_1, ..., z_k)))$

```
(do (x ...) <- m1
    (y ...) <- m2
    rest ...)
```
$\overset{(iv)}{=}$
```
(do (x ... y ...) <-
       (do (x ...) <- m1
           (y ...) <- m2
           return (x ... y ...))
   rest ...)
```

3. Identifiability

3.1. **Identifiability algorithms.** The **Identify** algorithm checks if a set of variables C can be identified inside of a single confounded component T containing them, $C \subseteq T$. It is based on Shpitser and Pearl's identify algorithm [SP08], but this algorithm is different in multiple ways: *(i)* it does not use multiplications or divisions, but only do-notation; *(ii)* it computes conditionals separately, using the observation structure; and *(iii)* it outputs a selected variable as a single separated output, $x \in C$, which will be needed later in the algorithm.

Definition 3.1.1 (Causal model)**.** A *(semimarkovian) causal model* consists of a directed acyclic graph with a subset of *visible* variables, such that each non-visible variable does not have any parents and is the parent of exactly two visible variables.

Algorithm 3.1.2 (Identify algorithm)**.** Let G be a causal model inducing a topological partial order $(<)$. Let T be a confounded component; let $C \subseteq T$ be a subset of it; and let $x \in C$ be a variable. Let q be a distribution over T.

The following algorithm computes $\text{Identify}(q, G, T, C, x)$.

(1) Compute $A = \text{Ancestors}(C)_{G(T)}$, the ancestors of C in the subgraph given by the component T.
(2) If $A = C$, then return the conditional distribution $q(x \mid \{c < x \mid c \in C\})$.
(3) If $A \neq C$ and $A = T$, then fail.
(4) If $C \subset A \subset T$, then let T' be the confounded component of C in $G(A)$.
   (a) Compute a conditional distribution over the subset using the fact that it is a separated confounded component.

```
q' = (do t1' <- q(t1' | {a | a < t1'})
         ...
         tn' <- q(tn' | {a | a < tn'})
         return (t1' ... tn'))
```

   (b) Recursively apply $\text{Identify}(q', G(T'), T', C, x)$.

Algorithm 3.1.3 (ID algorithm)**.** The following algorithm computes $\text{ID}(p, G, T, S)$.

(1) Compute the ancestors of S in the graph where all the variables in T are removed, $D = \text{Ancestors}(S)_{G(V-T)}$.
(2) For each variable $d_i \in D$, let S_i be its confounded component.
(3) For each variable $d_i \in D$, let D_i be its confounded component in the graph $G(D)$, restricted to variables in D.
(4) For each variable $d_i \in D$, let QS_i be computed as a conditional distribution on a separated confounded component.

```
QSi = (do si1 <- p(si1 | {v | v < si1})
          ...
          sim <- p(sim | {v | v < sim})
          return (si1 ... sim))
```

(5) For each variable in D, identify it using a call to the main Identify algorithm, $\text{Identify}(QS_i, G, S_i, D_i, d_i)$.

```
q = (do d1 <- Identify(QS1, G, S1, D1, d1)
        ...
        dn <- Identify(QSn, G, Sn, Dn, dn)
        return (s1 ... sn))
```

Remark 3.1.4. Shpitser and Pearl have shown that this algorithm is both sound — i.e., when it succeeds, it computes a correct estimator of the intervened distribution — and complete — i.e., it only fails when it is impossible to provide a correct estimator [SP08]. However, they have done so assuming a particular semantics: that of discrete probability distributions. This version of the algorithm opens the way to prove soundness and completeness under a more general theory; we leave this for further work.

3.2. **Identifiability as a derived construct.** We introduce a single construct to magmadic do notation. It represents the result of modeling a distribution p with a causal model G and, after performing an intervention that sets some variables $(v_1, ..., v_n)$ to some values $(x_1, ..., x_n)$, identifying the value of some output variables $(y_1, ..., y_m)$. We use the previous identifiability algorithm.

```
(Intervene p
 WithModel G
 Setting (v ...) To (x ...)
 In (y ...))
```

$$= \text{ID}(p, G, \{v_1, ..., v_n\}, \{y_1, ..., y_m\})(x_1, ..., x_n)$$

Remark 3.2.1. The causal model can be itself declared using a form of anonymous do-notation, as we do in the examples.

Acknowledgements

The author thanks helpful discussion with Márk Széles and Elena Di Lavore.

Funding. Mario Román was supported by the Estonian Research Council grant PRG 3215 ("Post Cartesian Programming - Logic, Probability and Computation with String Diagrams"), and by the Advanced Research + Invention Agency (ARIA) Safeguarded AI Programme.

References

[Cc23] The ChiRho contributors. ChiRho: An experimental language for causal reasoning, 2023.

[CWPW86] Clive R Charig, David R Webb, Stephen Richard Payne, and John E Wickham. Comparison of treatment of renal calculi by open surgery, percutaneous nephrolithotomy, and extracorporeal shockwave lithotripsy. *British medical journal (Clinical research ed.)*, 292(6524):879–882, 1986.

[DRS26] Elena Di Lavore, Mario Román, and Márk Széles. Normalized probabilistic semantics is not associative, 2026.

[JM94] Steven A Julious and Mark A Mullee. Confounding and simpson's paradox. *British Medical Journal*, 309(6967):1480–1481, 1994.

[Mog91] Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, 93(1):55–92, 1991.

[Pea09] Judea Pearl. *Causality*. Cambridge University Press, 2009.

[SP08] Ilya Shpitser and Judea Pearl. Complete identification methods for the causal hierarchy. *Journal of Machine Learning Research*, 9(64):1941–1979, 2008.

[Wad92] Philip Wadler. The essence of functional programming. In Ravi Sethi, editor, *Conference Record of the Nineteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Albuquerque, New Mexico, USA, January 19-22, 1992*, pages 1–14. ACM Press, 1992.

[WSB+] Theo Wang, Dario Stein, Eli Bingham, John Feser, Ohad Kammar, Michael Lee, and Jeremy Yallop. Typed abstractions for causal probabilistic programming.