Decomposing Probabilistic Lambda-Calculi

Willem Heijltjes
University of Bath

Joint work with Ugo Dal Lago and Giulio Guerrieri

Mathematical Foundations Seminar, 31 March 2020

Probabilistic λ-calculus

$$M, N \quad ::= \quad x \mid \lambda x.N \mid NM \mid N \oplus M$$

$$C[\,(\lambda x.N)M\,] \quad \to_\beta \quad C[\,N[M/x]\,]$$

$$C[\,N \oplus M\,] \quad \to_\oplus \quad C[\,N\,] + C[\,M\,]$$

$$C \quad ::= \quad [\,] \mid \lambda x.C \mid CM \mid NC \mid C \oplus M \mid N \oplus C$$

. . . is non-confluent

cbn :

$$\begin{array}{lll} (\lambda x.\, x = x)(\top \oplus \bot) & \to_\beta & (\top \oplus \bot) = (\top \oplus \bot) \\ & \twoheadrightarrow & (\top = \top) + (\top = \bot) + (\bot = \top) + (\bot = \bot) \\ & = & \top + \bot \end{array}$$

cbv :

$$\begin{array}{lll} (\lambda x.\, x = x)(\top \oplus \bot) & \to & (\lambda x.\, x = x)\,\top + (\lambda x.\, x = x)\,\bot \\ & \twoheadrightarrow_\beta & (\top = \top) + (\bot = \bot) \\ & = & \top \end{array}$$

The problem: duplicating probabilistic events

$$M, N \quad ::= \quad x \mid \lambda x.N \mid NM \mid N \oplus M$$

$$C[\,(\lambda x.N)M\,] \quad \rightarrow_\beta \quad C[\,N[M/x]\,]$$

$$C[\,N \oplus M\,] \quad \rightarrow_\oplus \quad C[\,N\,] + C[\,M\,]$$

$$C \quad ::= \quad [\,] \mid \lambda x.C \mid CM \mid NC \mid C \oplus M \mid N \oplus C$$

Analysing the problem

A probabilistic sum $N \oplus M$

1) flips a coin, and then
2) evaluates to N for heads and M for tails.

Does duplicating $N \oplus M$ mean

a) flipping different coins for each copy, or (call-by-name)
b) flipping the same coin for all copies? (call-by-value)

Decomposing the probabilistic sum

$$N \oplus M \quad \stackrel{\Delta}{=} \quad \boxed{a}.\, N \overset{a}{\oplus} M$$

$\boxed{a}$ (*generator*): a probabilistic quantifier that binds a
— it flips a coin and stores the result as a

$N \overset{a}{\oplus} M$ (*choice*): a conditional on a
— it evaluates to N if $a = 0$ and M if $a = 1$.

encoding cbn :

$$
\begin{array}{lll}
(\lambda x.\, x = x)(\boxed{a}.\, \top \overset{a}{\oplus} \bot) & \to_\beta & (\boxed{a}.\, \top \overset{a}{\oplus} \bot) = (\boxed{b}.\, \top \overset{b}{\oplus} \bot) \\
& \twoheadrightarrow & (\top = \top) + (\top = \bot) + (\bot = \top) + (\bot = \bot) \\
& = & \top + \bot
\end{array}
$$

encoding cbv :

$$
\begin{array}{lll}
\boxed{a}.\, (\lambda x.\, x = x)(\top \overset{a}{\oplus} \bot) & \to_\beta & \boxed{a}.\, (\top \overset{a}{\oplus} \bot) = (\top \overset{a}{\oplus} \bot) \\
& \twoheadrightarrow & (\top = \top) + (\bot = \bot) \\
& = & \top
\end{array}
$$

The Probabilistic Event λ-Calculus (PEL)

$$M, N \quad ::= \quad x \mid \lambda x.N \mid NM \mid N \overset{a}{\oplus} M \mid \boxed{a}.N$$

Overview:

- Confluent
- Two forms of probabilistic reduction:
  - *permutative* reduction ($\twoheadrightarrow_{\mathsf{p}}$)
    — small-step; gives natural probabilistic normal forms
  - *projective* reduction ($\twoheadrightarrow_{\pi}$)
    — big-step; intuitive; efficient
- Simple types are simple

Principal ideas:

- Confluence is non-negotiable
- Call-by-name is correct for duplicating probabilistic events
- Call-by-value has a valid intuition that should be expressible

Permutative reduction

Conditionals satisfy the equivalence

$$C[N \overset{a}{\oplus} M] \;\sim\; C[N] \overset{a}{\oplus} C[M] \qquad \text{(if } a \text{ is free)}$$

which is readily cast as rewriting . . .

. . . if we have an order on choice variables:

$$(N \overset{a}{\oplus} M) \overset{b}{\oplus} P \;\sim\; (N \overset{b}{\oplus} P) \overset{a}{\oplus} (M \overset{b}{\oplus} P)$$

Definition $a < b$ if $\boxed{b}$ occurs in the scope of $\boxed{a}$:

$$\boxed{a}.\, C\,[\,\boxed{b}.\, N]$$

(assuming Barendregts convention: binders have unique names).

Reduction rules

Beta-reduction:

$$(\lambda x.N)M \twoheadrightarrow_\beta N[M/x] \qquad (\beta)$$

Permutative reduction: (continued on next slide)

$$N \overset{a}{\oplus} N \to_{\mathsf{p}} N \qquad \text{(i)}$$

$$(N \overset{a}{\oplus} M) \overset{a}{\oplus} P \to_{\mathsf{p}} N \overset{a}{\oplus} P \qquad (\mathsf{c}_1)$$

$$N \overset{a}{\oplus} (M \overset{a}{\oplus} P) \to_{\mathsf{p}} N \overset{a}{\oplus} P \qquad (\mathsf{c}_2)$$

$$\lambda x.\, (N \overset{a}{\oplus} M) \to_{\mathsf{p}} (\lambda x.\, N) \overset{a}{\oplus} (\lambda x.\, M) \qquad (\oplus\lambda)$$

$$(N \overset{a}{\oplus} M) P \to_{\mathsf{p}} (NP) \overset{a}{\oplus} (MP) \qquad (\oplus\mathsf{f})$$

$$N (M \overset{a}{\oplus} P) \to_{\mathsf{p}} (NM) \overset{a}{\oplus} (NP) \qquad (\oplus\mathsf{a})$$

$$(N \overset{a}{\oplus} M) \overset{b}{\oplus} P \to_{\mathsf{p}} (N \overset{b}{\oplus} P) \overset{a}{\oplus} (M \overset{b}{\oplus} P) \quad (\text{if } a < b) \qquad (\oplus\oplus_1)$$

$$N \overset{b}{\oplus} (M \overset{a}{\oplus} P) \to_{\mathsf{p}} (N \overset{b}{\oplus} M) \overset{a}{\oplus} (N \overset{b}{\oplus} P) \quad (\text{if } a < b) \qquad (\oplus\oplus_2)$$

$$\boxed{b}.\, (N \overset{a}{\oplus} M) \to_{\mathsf{p}} (\boxed{b}.\, N) \overset{a}{\oplus} (\boxed{b}.\, M) \quad (\text{if } a < b) \qquad (\oplus\Box)$$

$$\boxed{a}.\, N \to_{\mathsf{p}} N \quad (\text{if } a \notin \mathsf{fv}(N)) \qquad (\not\Box)$$

$$\lambda x.\, \boxed{a}.\, N \to_{\mathsf{p}} \boxed{a}.\, \lambda x.\, N \qquad (\Box\lambda)$$

$$(\boxed{a}.\, N) M \to_{\mathsf{p}} \boxed{a}.\, (NM) \quad (\text{if } a \notin \mathsf{fv}(M)) \qquad (\Box\mathsf{f})$$

Example

Another reduction path for the cbv-translation of $(\lambda x.x = x)(\top + \bot)$

$$
\begin{array}{lll}
\boxed{a}.\,(\lambda x.\,x{=}x)(\top \overset{a}{\oplus} \bot) & \to_{\mathsf{p}} & \boxed{a}.\,(\lambda x.\,x{=}x)\top \overset{a}{\oplus} (\lambda x.\,x{=}x)\bot \qquad (\oplus\mathsf{a}) \\
 & \twoheadrightarrow_{\beta} & \boxed{a}.\,(\top{=}\top) \overset{a}{\oplus} (\bot{=}\bot) \\
 & = & \boxed{a}.\,\top \overset{a}{\oplus} \top \\
 & \to_{\mathsf{p}} & \boxed{a}.\,\top \qquad (\mathsf{i}) \\
 & \to_{\mathsf{p}} & \top \qquad (\not\boxed{\ }) \\
\end{array}
$$

Observations

- The "missing" rule which would otherwise generate non-confluence is

$$N\,(\boxed{a}.\,M) \not\to_{\mathsf{p}} \boxed{a}.\,N\,M$$

- We don't allow *generators* to permute out of argument position, but we do *choice*:

$$N\,(M \overset{a}{\oplus} P) \;\to_{\mathsf{p}}\; (NM) \overset{a}{\oplus} (NP)$$

- The rules for $\overset{a}{\oplus}$ alone are the rewriting rules for *ordered binary decision trees*.[1]

[1] [Zantema & Van de Pol, 2001]

Characterization

Theorem Permutative reduction is SN

(A direct application of *recursive path orders*[2])

Normal forms:

$$\begin{array}{lcl} P_0 & ::= & P_1 \mid P_0 \oplus P_0 \\ P_1 & ::= & x \mid \lambda x.P_1 \mid P_1\, P_0 \end{array}$$

Theorem Permutative reduction is confluent

[2] [Dershowitz, 2001]

Reduction

Definition Reduction $\rightarrow$ is $\rightarrow_\beta \cup \rightarrow_{\mathsf{P}}$.

Theorem Reduction is confluent.

Simple types

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash N : B}{\Gamma \vdash \lambda x.\, N : A \to B} \qquad \frac{\Gamma \vdash N : A \to B \qquad \Gamma \vdash M : A}{\Gamma \vdash NM : B}$$

$$\frac{\Gamma \vdash N : A \qquad \Gamma \vdash M : A}{\Gamma \vdash N \overset{a}{\oplus} M : A} \qquad \frac{\Gamma \vdash N : A}{\Gamma \vdash \boxed{a}.\, N : A}$$

Theorem Typed reduction is SN

(By abstract reducibility)

Encoding cbn

$$[\![x]\!]_{\mathsf{n}} = x \qquad [\![\lambda x.N]\!]_{\mathsf{n}} = \lambda x.[\![N]\!]_{\mathsf{n}} \qquad [\![NM]\!]_{\mathsf{n}} = [\![N]\!]_{\mathsf{n}}[\![M]\!]_{\mathsf{n}}$$

$$[\![N \oplus M]\!]_{\mathsf{n}} = \boxed{a}.\, [\![N]\!]_{\mathsf{n}} \overset{a}{\oplus} [\![M]\!]_{\mathsf{n}}$$

Encoding cbv

$$[\![N]\!]_{\mathsf{v}} = \lfloor [\![N]\!] \rfloor$$

$$[\![x]\!] = x \qquad [\![\lambda x.N]\!] = \lambda x.[\![N]\!] \qquad [\![NM]\!] = [\![N]\!][\![M]\!]$$

$$[\![N \oplus M]\!] = [\![N]\!] \overset{a}{\oplus} [\![M]\!]$$

where each choice variable a is fresh;

$$\lfloor N \rfloor = \boxed{a_1} \ldots \boxed{a_n} . N$$

where $a_1 \ldots a_n$ are the free choice variables of N.

Next: to show that the encodings work. . .

Definition a-Projections $\pi_0^a(-)$ and $\pi_1^a(-)$:

$$\pi_0^a(N \overset{a}{\oplus} M) = \pi_0^a(N) \qquad \pi_i^a(\lambda x.\, N) = \lambda x.\, \pi_i^a(N)$$

$$\pi_1^a(N \overset{a}{\oplus} M) = \pi_1^a(M) \qquad \pi_i^a(NM) = \pi_i^a(N)\, \pi_i^a(M)$$

$$\pi_i^a(\boxed{a}.\, N) = \boxed{a}.\, N \qquad \pi_i^a(N \overset{b}{\oplus} M) = \pi_i^a(N) \overset{b}{\oplus} \pi_i^a(M) \qquad \text{if } a \neq b$$

$$\pi_i^a(x) = x \qquad \pi_i^a(\boxed{b}.\, N) = \boxed{b}.\, \pi_i^a(N) \qquad \text{if } a \neq b.$$

Definition Head contexts: $H ::= [\,] \mid \lambda x.\, H \mid H\, N$

Proposition Permutative normal forms are given by $\downarrow_{\mathsf{p}}$:

$$H[\boxed{a}.\, N]\downarrow_{\mathsf{p}} = \begin{cases} H\downarrow_{\mathsf{p}} [N\downarrow_{\mathsf{p}}] & \text{if } a \notin \mathrm{fv}(N) \\ H\downarrow_{\mathsf{p}} [\pi_0^a(N)\downarrow_{\mathsf{p}}] \;\oplus\; H\downarrow_{\mathsf{p}} [\pi_1^a(N)\downarrow_{\mathsf{p}}] & \text{otherwise} \end{cases}$$

$$[\,]\downarrow_{\mathsf{p}} = [] \qquad \lambda x.H\downarrow_{\mathsf{p}} = \lambda x.(H\downarrow_{\mathsf{p}}) \qquad HN\downarrow_{\mathsf{p}} = (H\downarrow_{\mathsf{p}})\,(N\downarrow_{\mathsf{p}})$$

Projective head reduction

Definition Projective head reduction $\rightarrow\!\!\!\triangleright_{\pi\mathsf{h}}$ is given by

$$H[\boxed{a}.\, N] \;\rightarrow\!\!\!\triangleright_{\pi\mathsf{h}}\; H[\pi_0^a(N)] + H[\pi_1^a(N)]$$

Permutative reduction simulates projective head reduction:

$$H[\boxed{a}.\, N] \;\twoheadrightarrow_{\mathsf{P}}\; \begin{cases} H[N] & \text{if } a \notin \mathsf{fv}(N) \\ H[\pi_0^a(N)] \oplus H[\pi_1^a(N)] & \text{otherwise.} \end{cases}$$

This is sufficient to capture call-by-name.

But: we *should* be able to reduce in any *linear* context, i.e. also under generators and choices. However, we cannot simulate that with permutative reduction, since it would require exchanging generators.

$$\boxed{a}.\,\boxed{b}.\,N \overset{?}{\sim} \boxed{b}.\,\boxed{a}.\,N$$

Projective reduction

Definition N^s labels N with a binary *stream* $s \in \mathbb{S} = \{0, 1\}^{\mathbb{N}}$

$$(\lambda x.\, N)^s = \lambda x.\, N^s \qquad (\boxed{a}.\, N)^{i \cdot s} = \boxed{a}^i.\, N^s$$

$$(N\, M)^s = N^s\, M \qquad (N \overset{a}{\oplus} M)^s = N^s \overset{a}{\oplus} M^s$$

Definition Projective reduction $\twoheadrightarrow_\pi$ is

$$\boxed{a}^i.\, N \;\twoheadrightarrow_\pi\; \pi_i^a(N)\ .$$

Extend β-reduction to stream-labelled terms with a substitution case for labelled variables: $x^s[M/x] = M^s$

$$
\begin{array}{ccccc}
(\boxed{a}.\boxed{b}.N)^{i\cdot j\cdot s} & = & \boxed{a}^{i}.\boxed{b}^{j}.N^{s} & \xrightarrow{\pi} & \pi_i^a(\pi_j^b(N^s)) \\
\sim & & & & = \\
(\boxed{b}.\boxed{a}.N)^{j\cdot i\cdot s} & = & \boxed{b}^{j}.\boxed{a}^{i}.N^{s} & \xrightarrow{\pi} & \pi_j^b(\pi_i^a(N^s))
\end{array}
$$

Philosophy:

$$N = N^0 + N^1$$

Proposition $\rightarrow_\pi$ includes $\rightarrow_{\pi h}$:

$$H[\boxed{a}.N] \;=\; H[\boxed{a}^0.N] + H[\boxed{a}^1.N] \;\rightarrow_\pi\; H[\pi_0^a(N)] + H[\pi_1^a(N)]\,.$$

Proposition If $M \rightarrow N$ by a step other than $\not\Box$ then $M^s \rightarrow N^s$.

(Note that $\not\Box$ is included in $\rightarrow_{\pi h}$)

Call-by-value

$$M, N ::= x \mid \lambda x.N \mid MN \mid M \oplus N \qquad C[(\lambda x.N)V] \to_{\beta v} C[N[V/x]]$$

$$V, W ::= x \mid \lambda x.V \mid VW \qquad C[M \oplus N] \to_{v} C[M] + C[N]$$

If $N \to_{v} M + P$ then there is an $n \in \mathbb{N}$ such that for any finite stream s of length n and any stream t:

$$[\![N]\!]_{v}^{s \cdot 0 \cdot t} \to_{\pi} [\![M]\!]_{v}^{s \cdot t} \qquad [\![N]\!]_{v}^{s \cdot 1 \cdot t} \to_{\pi} [\![P]\!]_{v}^{s \cdot t}$$

Why the fuss

- Composing computational effects is not satisfactorily solved
- *Linearity* is enforced through monadic types
- But the *spine* of a λ-term is also linear

Compare

$(\lambda x.\, x = x)(\boxed{a}.\, \top \overset{a}{\oplus} \bot)$

```
f :: Random Bool
f = g random
  where
    g x = do
      a <- x
      b <- x
      return (a == b)
```

$\boxed{a}.\, (\lambda x.\, x = x)(\top \overset{a}{\oplus} \bot)$

```
f :: Random Bool
f = do
    a <- random
    return (h a)
  where
    h x = x == x
```

- Our approach generalizes to any *read* operation (think STDin, ROM, etc.)
- We know how to do *write* operations analogously, so we can do IO and State (with individual mutable variables)
- Important questions to make this work:
  - Types
  - Pointer calculations
  - Parallellism/concurrency