# Trace Types for Sound Programmable Inference in Probabilistic Languages – Supplementary Material

ALEXANDER K. LEW, Massachusetts Institute of Technology
MARCO F. CUSUMANO-TOWNER, Massachusetts Institute of Technology
BENJAMIN SHERMAN, Massachusetts Institute of Technology
MICHAEL CARBIN, Massachusetts Institute of Technology
VIKASH K. MANSINGHKA, Massachusetts Institute of Technology



## A IMPLEMENTATIONS OF CONTROL FLOW CONSTRUCTS.

Because our control flow constructs introduce only a single label into a program's trace, their disintegrations are trivial (either their entire traces are observed, or nothing is observed). But here, we describe the constructions of their samplers and densities.

### A.1 Branching.

We first present the implementation of our stochastic branching construct, $\textbf{withProbability}_l$.

$$\textbf{withProbability}_l\ p\ \mu\ \textbf{else}\ \nu$$

```
1  traced.sampler = M.do {
2    u ← sample
3    if u < p then M.do {
4      t ← μ.traced.sampler
5      return {l = inl t}
6    } else M.do {
7      t ← ν.traced.sampler
8      return {l = inr t}
9    }
10 }

12 traced.density(t) = match t.l with
13   inl s → p * μ.traced.density(s)
14   inr s → (1 − p) * ν.traced.density(s)

16 returnValue(t) = match t.l with
17   inl s → μ.returnValue(s)
18   inr s → ν.returnValue(s)
```

### A.2 Looping.

Next, we implement each of our three looping constructs.

$$\textbf{for}_l\ i\ \textbf{inRandomRange}\ d\ p(i)$$

2020. 2475-1421/2020/1-ART19
https://doi.org/10.1145/3371087

```
1  traced.sampler = M.do {
2   n ← d
3   snd (fold_N n (n, return {l = Nil}) (λ (i, μ_i). (i − 1, M.do {
4    t_i ← μ_i
5    subtrace ← p(i).traced.sampler
6    return {l = Cons subtrace t_i.l}
7   }))
8  }

10 traced.density(t) = d.density(length t.l) * ∏_{i∈{1,...,length t.l}} p(i).traced.density(nth t.l i)

12 returnValue(t) = indexedMap (λ (i, t_i). p(i).returnValue(t_i)) t.l
```

$$\textbf{while}_l\ (s := s_0;\ \min(p(s), p_{max}))\ \mu(s)$$

```
1  traced.sampler = M.do {
2   n ← geometric p_max
3   (_, res) ← fold_N n (return ((false, s_0), Nil)) (λ soFar. M.do {
4    ((stopped, s), t_i) ← soFar
5    u ← sample
6    let stopped' = stopped or u < p(s)/p_max
7    if stopped' then return ((stopped', s), t_i) else M.do {
8     subtrace ← μ(s).traced.sampler
9     return ((false, μ(s).returnValue(subtrace)), Cons subtrace t_i)
10   }
11  })
12  return {l = (reverse res)}
13 }

15 traced.density(t) = let
16   (ρ, s) = fold t.l (1, s_0) (λ ((ρ, s), t).(ρ * μ(s).traced.density(t), μ(s).returnValue(t)))
17  in ρ * min(p(s), p_max)

19 returnValue(t) = fold t.l s_0 (λ (s, t).μ(s).returnValue(t))
```

$$\textbf{forEach}_l\ x\ \textbf{in}\ v\ \mu(x)$$

```
1  traced.sampler = M.do {
2   v_t ← mapM (λ x. μ(x).traced.sampler) v
3   return {l = v_t}
4  }

6  traced.density(t) = ∏_{(x,t_x)∈zipv v t.l} μ(x).traced.density(t_x)
```

8 $returnValue(t) = \mathbf{mapv}\ (\lambda(x, t_x).\mu(x).returnValue(t_x))\ (\mathbf{zipv}\ v\ t.l)$

## B PROOF NOTES: CORRECTNESS OF DENSITIES AND DISINTEGRATIONS (THEOREM 1)

*Correctness of* $\mathcal{P}.\mathbf{return}$. First, note that for $\mathcal{P}.\mathbf{return}\ x$, we have $[\![traced.sampler]\!] = [\![\mathbf{return}\ \{\}]\!] = \mathcal{B}[\![\{\}]\!] = [\![\mathcal{M}.\mathbf{do}\ \{t \leftarrow \mathcal{B}[\![\{\}]\!]; \mathbf{score}\ 1; \mathbf{return}\ t\}]\!]$, so $\lambda t.1$ is a valid (and clearly strictly positive) density. Because its trace type is empty, it has only the trivial empty trace disintegration.

*Correctness of* $\mathbf{sample}_l\ d$. Let $d : \mathcal{D}\ \tau$. We can use the definition of density to rewrite $d.sampler$ in the implementation of *traced.sampler*, to obtain

$$[\![\mathcal{M}.\mathbf{do}\{x \leftarrow \mathcal{B}[\![\tau]\!]; \mathbf{score}\ d.density(x); \mathbf{return}\ \{l = x\}\}]\!].$$

By noting that $\mathcal{B}[\![\tau]\!]$ is equal to $\mathcal{M}.\mathbf{do}\ \{t \leftarrow \mathcal{B}[\![\{l : \tau\}]\!]; \mathbf{return}\ t.l\}\}$, then repeatedly applying the monad laws, we can further rewrite *traced.sampler* as

$$[\![\mathcal{M}.\mathbf{do}\{t \leftarrow \mathcal{B}[\![\{l : \tau\}]\!]; \mathbf{score}\ d.density(t.l); \mathbf{return}\ t\}]\!],$$

thus justifying the density $\lambda t.d.density(t.l)$. The strict positivity of this density follows from that of $d.density$. This equation, combined with the fact that $\{\} +\!\!+ t = t$, also justifies the formula for $observe_{\{l\}}\ t$.

*Correctness of* $\mathcal{P}.\mathbf{do}$. First, note that if we can show them to be correct, both density formulas (for the trace distribution and for the disintegrations) are strictly positive by induction, because pointwise multiplication preserves strict positivity, and these densities are both products of other strictly positive densities.

We now show that *traced.density* is a density for *traced.sampler*. We begin by recalling the implementation for the sampler:

$traced.sampler = \mathcal{M}.\mathbf{do}\ \{$
$\quad t_\tau \leftarrow \mu.traced.sampler$
$\quad \mathbf{let}\ x = \mu.returnValue(t_\tau)$
$\quad t_\sigma \leftarrow (\nu\ x).traced.sampler$
$\quad \mathbf{return}\ t_\tau +\!\!+ t_\sigma$
$\}$

We use the definition of density to rewrite the sampling lines, and also get rid of **let** expression, performing the substitution for $x$:

$traced.sampler = \mathcal{M}.\mathbf{do}\ \{$
$\quad t_\tau \leftarrow \mathcal{B}[\![\tau]\!]$
$\quad \mathbf{score}\ \mu.traced.density(t_\tau)$
$\quad t_\sigma \leftarrow \mathcal{B}[\![\sigma]\!]$
$\quad \mathbf{score}\ (\nu\ (\mu.returnValue(t_\tau))).traced.density(t_\sigma)$
$\quad \mathbf{return}\ t_\tau +\!\!+ t_\sigma$
$\}$

By the commutativity of the measure monad, we can swap the second line of the body with the third. We can also consolidate the two stock measures into one joint stock measure, and rewrite references to $t_\tau$ and $t_\sigma$ accordingly:

$$
\begin{aligned}
&\mathit{traced.sampler} = \mathcal{M}.\mathbf{do}\,\{\\
&\quad t \leftarrow \mathcal{B}[\![\tau +\!\!+ \sigma]\!]\\
&\quad \mathbf{score}\ \mu.\mathit{traced.density}(\mathbf{restrict}_\tau\, t)\\
&\quad \mathbf{score}\ (\nu\ (\mu.\mathit{returnValue}\,(\mathbf{restrict}_\tau\, t))).\mathit{traced.density}(\mathbf{restrict}_\sigma\, t)\\
&\quad \mathbf{return}\ t\\
&\}
\end{aligned}
$$

Adjacent **score** statements can be combined via multiplication:

$$
\begin{aligned}
&\mathit{traced.sampler} = \mathcal{M}.\mathbf{do}\,\{\\
&\quad t \leftarrow \mathcal{B}[\![\tau +\!\!+ \sigma]\!]\\
&\quad \mathbf{score}\ \mu.\mathit{traced.density}(\mathbf{restrict}_\tau\, t) * (\nu\ (\mu.\mathit{returnValue}\,(\mathbf{restrict}_\tau\, t))).\mathit{traced.density}(\mathbf{restrict}_\sigma\, t)\\
&\quad \mathbf{return}\ t\\
&\}
\end{aligned}
$$

And we can invoke the definition of density to conclude.

Next, we show that $\mathit{observe}_K\,(t).\mathit{sampler}$ is a valid trace disintegration. Our goal is to show that the following expression is equal to *traced.sampler*:

$$
\begin{aligned}
&\mathcal{M}.\mathbf{do}\,\{\\
&\quad t \leftarrow \mathcal{B}[\![\textstyle\prod_{k\in K}\tau_k]\!]\\
&\quad \mathbf{let}\ (t_\tau, t_\sigma) = (\mathbf{restrict}_{\tau\cap K}(t), \mathbf{restrict}_{\sigma\cap K}(t))\\
&\quad s_\tau \leftarrow (\mu.\mathit{observe}_{K\cap\tau}\,(t_\tau)).\mathit{sampler}\\
&\quad \mathbf{let}\ x = \mu.\mathit{returnValue}(s_\tau +\!\!+ t_\tau)\\
&\quad s_\sigma \leftarrow ((\nu\ x).\mathit{observe}_{K\cap\sigma}(t_\sigma)).\mathit{sampler}\\
&\quad \mathbf{return}\ t +\!\!+ (s_\tau +\!\!+ s_\sigma)\\
&\}
\end{aligned}
$$

We begin by rewriting the first two lines as assigning $t_\tau$ and $t_\sigma$ separately:

$$
\begin{aligned}
&\mathcal{M}.\mathbf{do}\,\{\\
&\quad t_\tau \leftarrow \mathcal{B}[\![\textstyle\prod_{k\in K\cap\tau}\tau_k]\!]\\
&\quad t_\sigma \leftarrow \mathcal{B}[\![\textstyle\prod_{k\in K\cap\sigma}\tau_k]\!]\\
&\quad s_\tau \leftarrow (\mu.\mathit{observe}_{K\cap\tau}\,(t_\tau)).\mathit{sampler}\\
&\quad \mathbf{let}\ x = \mu.\mathit{returnValue}(s_\tau +\!\!+ t_\tau)\\
&\quad s_\sigma \leftarrow ((\nu\ x).\mathit{observe}_{K\cap\sigma}(t_\sigma)).\mathit{sampler}\\
&\quad \mathbf{return}\ (t_\tau +\!\!+ t_\sigma) +\!\!+ (s_\tau +\!\!+ s_\sigma)\\
&\}
\end{aligned}
$$

We now use commutativity (of the monad, and on the last line, of $+\!\!+$) to move the $t$ and $s$ variables next to each other:

$$
\begin{aligned}
&\mathcal{M}.\mathbf{do}\,\{\\
&\quad t_\tau \leftarrow \mathcal{B}[\![\textstyle\prod_{k\in K\cap\tau}\tau_k]\!]\\
&\quad s_\tau \leftarrow (\mu.\mathit{observe}_{K\cap\tau}\,(t_\tau)).\mathit{sampler}\\
&\quad t_\sigma \leftarrow \mathcal{B}[\![\textstyle\prod_{k\in K\cap\sigma}\tau_k]\!]\\
&\quad s_\sigma \leftarrow ((\nu\ (\mu.\mathit{returnValue}(s_\tau +\!\!+ t_\tau))).\mathit{observe}_{K\cap\sigma}(t_\sigma)).\mathit{sampler}\\
&\quad \mathbf{return}\ (t_\tau +\!\!+ s_\tau) +\!\!+ (t_\sigma +\!\!+ s_\sigma)\\
&\}
\end{aligned}
$$

By induction (using the fact that $\mu$ and $\nu$ have valid trace disintegrations), we can rewrite lines 1 and 2, then lines 3 and 4, to get exactly the expression for *traced.sampler*, as required:

$\mathcal{M}$.**do** {
  $t_\tau \leftarrow \mu.traced.sampler$
  $t_\sigma \leftarrow (\nu\ (\mu.returnValue(t_\tau))).traced.sampler$
  **return** $(t_\tau +\!\!+ t_\sigma)$
} = *traced.sampler*

The proof that the density for the disintegration is correct follows the same pattern as the proof that the prior density (for *traced.sampler*) is correct: we expand each of the *sampler* calls using the definition of density, then reorganize to bring the two resulting **score** statements next to one another, resulting in the multiplication that appears in the disintegration's density.