Modular probabilistic inference by program transformations

Chung-chieh Shan · QAPL, 2–3 April 2016
On the Criteria To Be Used in Decomposing Systems into Modules

D.L. Parnas
Carnegie-Mellon University

Introduction

A lucid statement of the philosophy of modular programming can be found in a 1970 textbook on the design of system programs by Gouthier and Pont [1, ¶10.23], which we quote below:

A well-defined segmentation of the project effort ensures system modularity. Each task forms a separate, distinct program module. At implementation time each module and its inputs and outputs are well-defined, there is no confusion in the intended interface with other system modules. At checkout time the integrity of the module is tested independently; there are few scheduling problems in synchronizing the completion of several tasks before checkout can begin. Finally, the system is maintained in modular fashion; system errors and deficiencies can be traced to specific system modules, thus limiting the scope of detailed error searching.

Usually nothing is said about the criteria to be used...
Probabilistic Programming for Advancing Machine Learning (PPAML)

Dr. Suresh Jagannathan

Machine learning – the ability of computers to understand data, manage results and infer insights from uncertain information – is the force behind many recent revolutions in computing. Email spam filters, smartphone personal assistants and self-driving vehicles are all based on research advances in machine learning. Unfortunately, even as the demand for these capabilities is accelerating, every new application requires a Herculean effort. Teams of hard-to-find experts must build expensive, custom tools that are often painfully slow and can perform unpredictably against large, complex data sets.
Major features

- BNT supports many types of **conditional probability distributions** (nodes), and it is easy to add more.
  - Tabular (multinomial)
  - Gaussian
  - Softmax (logistic/sigmoid)
  - Multi-layer perceptron (neural network)
  - Noisy-or
  - Deterministic

- BNT supports **decision and utility nodes**, as well as chance nodes, i.e., influence diagrams as well as Bayes nets.

- BNT supports static and dynamic BNs (useful for modelling dynamical systems and sequence data).

- BNT supports many different **inference algorithms**, and it is easy to add more.
  - Exact inference for static BNs:
    - junction tree
    - variable elimination
    - brute force enumeration (for discrete nets)
    - linear algebra (for Gaussian nets)
Sorting analogy

Ordering + Sorting technique = Sorting procedure
- Numeric
- Alphabetical
- Case folding
- Reverse
- ...
- Merge sort
- Quick sort
- Insertion sort
- Radix sort
- ...

Distribution + Inference technique = Inference procedure
- Generative story
- Interpreter/compiler
- Probabilistic programming
Beam Sampling for the Infinite Hidden Markov Model

We start this section by describing the finite HMM, then taking the infinite limit to obtain an intuition for the infinite HMM, followed by a more precise definition. A finite HMM consists of a hidden state sequence \( s = (s_1, s_2, \ldots, s_T) \) and a corresponding observation sequence \( y = (y_1, y_2, \ldots, y_T) \). Each state variable \( s_t \) can take on a finite number of states, say \( 1 \ldots K \). Transitions between states are governed by Markov dynamics parameterized by the transition matrix \( \pi \), where \( \pi_{ij} = p(s_t = j | s_{t-1} = i) \), while the initial state probabilities are \( \pi_{0i} = p(s_1 = i) \). For each state \( s_t \in \{1, \ldots, K\} \), there is a parameter \( \phi_t \), which parameterizes the emission likelihood of state \( s_t \) given the observation \( y_t \). We describe how \( \pi, \phi \) can be written (with \( \alpha, \beta, \gamma \)) = \( \text{Dirichlet}(\alpha, \beta) \), \( \text{Multinomial}(\gamma) \), \( \text{Gamma}(\alpha, \beta) \). It is common to choose \( \phi \) and \( \gamma \) to have strong beliefs about the hyperparameters, is to use gamma hyperpriors: \( \alpha \sim \text{Gamma}(a_\alpha, b_\alpha) \) and \( \gamma \sim \text{Gamma}(a_\gamma, b_\gamma) \). (Teh et al., 2006) describe how these hyperparameters can be sampled efficiently, and we will use this in the experiments to follow.

3. The Gibbs Sampler

The Gibbs sampler was the first sampling algorithm for the iHMM that converges to the true posterior. One proposal builds on the direct assignment sampling scheme for the HDP in (Teh et al., 2006) by marginalizing out the hidden variables \( \pi, \phi \) from (2), (3) and ignoring the ordering of states implicit in \( \beta \). Thus we only need to sample the hidden trajectory \( s \), the base

2. The Infinite Hidden Markov Model

parametric Bayesian extension of the HMM with an infinite number of hidden states. Exact Bayesian inference for the iHMM is intractable. Specifically, given a particular setting of the parameters the forward-backward algorithm cannot be applied since the number of states \( K \) is infinite, while with the parameters marginalized out all hidden state variables will be coupled and the forward-backward algorithm cannot be applied either. Currently the only approximate inference algorithm available is Gibbs sampling, where individual hidden state variables are resampled conditioned on all other variables (Teh et al., 2006). Unfortunately convergence of Gibbs sampling is notoriously slow for the iHMM since the strong dependencies on all previous states make it difficult to converge to the posterior. Another common solution, when we do not have strong beliefs about the hyperparameters, is to use symmetric priors for transition and initial probabilities leading into state \( s \), and concentration parameter \( \alpha > 0 \), which governs variability around \( G_0 \).

4. The Beam Sampler

The forward-backward algorithm does not apply to the iHMM because the number of states, and hence the number of potential state trajectories, are infinite. The idea of beam sampling is to introduce auxiliary variables \( u \) such that conditioned on \( u \) the number of trajectories with positive probability is finite. Now dynamic programming can be used to compute the conditional probabilities of each of these trajectories and thus sample whole trajectories efficiently. These
Hakaru: meaningful and reusable, from clear to fast

Model

disintegrate

Posterior

simplify

expect

Simplified posterior

1321 ms

Idiomatic WebPPL → 1078 ms

267 ms

Handwritten Hakaru → 207 ms

FLOPS 2016 system description paper
“Probabilistic inference by program transformation in Hakaru”

NIPS 2015 workshop poster
“Building blocks for exact and approximate inference”

Transition kernel

simplify

expect

1078 ms

Simplified kernel

207 ms

Simplified kernel

Transition kernel

Disintegrate

expect

Gibbs sampling

Disintegrate

expect

Simplifying posterior

expect

Simplifying posterior

expect

Simplifying posterior

expect

Simplifying posterior

expect

Simplifying posterior

expect
Hakaru: meaningful and reusable, from clear to fast

Model

\[ \text{disintegrate} \]

Posterior

\[ \text{simplify} \]
\[ \text{expect} \]

Simplified posterior

\[ \text{disintegrate} \]

\[ \text{expect} \]

Transition kernel

\[ \text{simplify} \]
\[ \text{expect} \]

Simplified kernel

\[ \text{1321 ms} \]

\[ \text{1078 ms} \]

\[ \text{267 ms} \]

\[ \text{207 ms} \]

FLOPS 2016 system description paper

“Probabilistic inference by program transformation in Hakaru”

NIPS 2015 workshop poster

“Building blocks for exact and approximate inference”

Idiomatic WebPPL

Handwritten Hakaru
Hakaru: meaningful and reusable, from clear to fast

FLOPS 2016 system description paper
“Probabilistic inference by program transformation in Hakaru”

NIPS 2015 workshop poster
“Building blocks for exact and approximate inference”

Idiomatic WebPPL → 1078 ms
Handwritten Hakaru → 207 ms
Diseases A and B are equally prevalent. Disease A causes one of symptoms 1, 2, 3 with equal probability. Disease B causes one of symptoms 1, 2 with equal probability.

\[ A \mapsto \frac{1}{2}, \quad B \mapsto \frac{1}{2} \] : \text{Medical Diagnosis}
Disintegration for medical diagnosis

Diseases $A$ and $B$ are equally prevalent.
Disease $A$ causes one of symptoms $1, 2, 3$ with equal probability.
Disease $B$ causes one of symptoms $1, 2$ with equal probability.

\[
\text{do } \{ \text{disease } \sim \{ A \mapsto 1/2, B \mapsto 1/2 \}; \\
\text{symptom } \sim \text{ case disease of} \\
\quad A \mapsto \{ 1 \mapsto 1/3, 2 \mapsto 1/3, 3 \mapsto 1/3 \} \\
\quad B \mapsto \{ 1 \mapsto 1/2, 2 \mapsto 1/2 \}; \\
\text{return } (\text{symptom, disease}) \} : \mathbb{M} (\text{Symptom } \times \text{Disease})
\]
Disintegration for medical diagnosis

Diseases $A$ and $B$ are equally prevalent.
Disease $A$ causes one of symptoms 1, 2, 3 with equal probability.
Disease $B$ causes one of symptoms 1, 2 with equal probability.

\[
\text{do} \ \{ \text{disease} \sim \{ A \mapsto 1/2, B \mapsto 1/2 \}; \text{symptom} \sim \text{case disease of} \\
\quad A \mapsto \{ 1 \mapsto 1/3, 2 \mapsto 1/3, 3 \mapsto 1/3 \} \\
\quad B \mapsto \{ 1 \mapsto 1/2, 2 \mapsto 1/2 \}; \text{return (symptom, disease)} \} : \mathbb{M} (\text{Symptom} \times \text{Disease})
\]

\[
= \{ (1, A) \mapsto 1/6, (2, A) \mapsto 1/6, (3, A) \mapsto 1/6, (1, B) \mapsto 1/4, (2, B) \mapsto 1/4 \}
\]

\[
\begin{array}{c|ccc}
\text{A} & 1/6 & 1/6 & 1/6 \\
\text{B} & 1/4 & 1/4 & 0 \\
\hline
1 & 2 & 3
\end{array}
\]
Disintegration for medical diagnosis

Diseases A and B are equally prevalent.
Disease A causes one of symptoms 1, 2, 3 with equal probability.
Disease B causes one of symptoms 1, 2 with equal probability.

\[
\text{do } \{\text{disease } \sim \{A \mapsto 1/2, B \mapsto 1/2\}; \\
\text{symptom } \sim \text{ case disease of} \\
A \mapsto \{1 \mapsto 1/3, 2 \mapsto 1/3, 3 \mapsto 1/3\} \\
B \mapsto \{1 \mapsto 1/2, 2 \mapsto 1/2\}; \\
\text{return (symptom, disease)}\} : \mathbb{M} (\text{Symptom } \times \text{ Disease})
\]

\[
\lambda \text{symptom. case symptom of} \\
1 \mapsto \{A \mapsto 1/6, B \mapsto 1/4\} \\
2 \mapsto \{A \mapsto 1/6, B \mapsto 1/4\} \\
3 \mapsto \{A \mapsto 1/6\} : \text{Symptom } \rightarrow \mathbb{M} \text{ Disease}
\]
Disintegration on a zero-probability observation

Diseases $A$ and $B$ are equally prevalent.
Disease $A$ causes a symptom chosen uniformly from $[0, 3] \subset \mathbb{R}$.
Disease $B$ causes a symptom chosen uniformly from $[0, 2] \subset \mathbb{R}$.

$$\left\{ A \mapsto 1/2, B \mapsto 1/2 \right\} : \mathbb{M} \text{ Disease}$$
Disintegration on a zero-probability observation

Diseases $A$ and $B$ are equally prevalent.

Disease $A$ causes a symptom chosen uniformly from $[0, 3] \subseteq \mathbb{R}$.
Disease $B$ causes a symptom chosen uniformly from $[0, 2] \subseteq \mathbb{R}$.

\[
\text{do} \; \{ \text{disease} \sim \{ A \mapsto 1/2, B \mapsto 1/2 \}; \\
\text{symptom} \sim \text{case} \; \text{disease} \; \text{of} \\
\quad A \rightarrow \text{uniform} \; 0 \; 3 \\
\quad B \rightarrow \text{uniform} \; 0 \; 2; \\
\text{return} \; (\text{symptom}, \text{disease}) \} : \mathbb{M} \; (\text{Symptom} \times \text{Disease})
\]
Disintegration on a zero-probability observation

Diseases $A$ and $B$ are equally prevalent. Disease $A$ causes a symptom chosen uniformly from $[0, 3] \subseteq \mathbb{R}$. Disease $B$ causes a symptom chosen uniformly from $[0, 2] \subseteq \mathbb{R}$.

\[
\text{do } \{ \text{disease } \sim \{A \mapsto 1/2, B \mapsto 1/2\}; \\
\text{symptom } \sim \text{ case disease of} \\
\quad A \rightarrow \text{uniform } 0 3 \\
\quad B \rightarrow \text{uniform } 0 2; \\
\text{return } (\text{symptom, disease}) \} : \mathcal{M} (\text{Symptom } \times \text{Disease})
\]
Disintegration on a zero-probability observation

Diseases $A$ and $B$ are equally prevalent.
Disease $A$ causes a symptom chosen uniformly from $[0, 3] \subseteq \mathbb{R}$.
Disease $B$ causes a symptom chosen uniformly from $[0, 2] \subseteq \mathbb{R}$.

\[
\text{do}\ \{\text{disease} \sim \{A \mapsto 1/2, B \mapsto 1/2\}; \text{symptom} \sim \text{case disease of} \ A \rightarrow \text{uniform} \ 0 \ 3 \ B \rightarrow \text{uniform} \ 0 \ 2; \text{return} (\text{symptom}, \text{disease})\}\quad :\ M (\text{Symptom} \times \text{Disease})
\]

\[
\lambda \text{symptom. if symptom} \leq 2 \quad \text{then} \quad \{A \mapsto 1/6, B \mapsto 1/4\};
\]
\[
\text{else} \quad \{A \mapsto 1/6\} : \text{Symptom} \rightarrow M \text{ Disease}
\]
Choose *disease* uniformly from $[1, 3] \subset \mathbb{R}$.
Choose *symptom* uniformly from $[0, \text{disease}] \subset \mathbb{R}$.

**uniform** 1 3 : $\mathbb{M}$ Disease
Disintegration on a zero-probability observation

Choose disease uniformly from \([1, 3] \subset \mathbb{R}\).
Choose symptom uniformly from \([0, \text{disease}] \subset \mathbb{R}\).

\[
\text{do}\{ \text{disease} \sim \text{uniform} \ 1 \ 3; \ 
\text{symptom} \sim \text{uniform} \ 0 \ \text{disease}; 
\text{return} \ (\text{symptom}, \text{disease}) \}\quad : \mathbb{M} \ (\text{Symptom} \times \text{Disease})
\]
Choose \textit{disease} uniformly from \([1, 3] \subset \mathbb{R}\).
Choose \textit{symptom} uniformly from \([0, \textit{disease}] \subset \mathbb{R}\).

\[
\textbf{do} \ \{\textit{disease} \sim \textbf{uniform} \ 1 \ 3; \\
\textit{symptom} \sim \textbf{uniform} \ 0 \ \textit{disease}; \\
\textbf{return} \ (\textit{symptom}, \textit{disease})\} \quad : \ \mathbb{M} \ (\text{Symptom} \times \text{Disease})
\]
Disintegration on a zero-probability observation

Choose disease uniformly from $[1, 3] \subset \mathbb{R}$.
Choose symptom uniformly from $[0, \text{disease}] \subset \mathbb{R}$.

\[
\text{do } \{ \text{disease } \sim \text{ uniform } 1 3; \\
\text{symptom } \sim \text{ uniform } 0 \text{ disease}; \\
\text{return } (\text{symptom, disease}) \} : \mathbb{M} (\text{Symptom } \times \text{ Disease})
\]

\[
\lambda \text{symptom. do } \{ \text{disease } \sim \text{ uniform } 1 3; \\
\text{if } 0 \leq \text{symptom} \leq \text{disease} \\
\text{then } \{ \text{disease } \mapsto 1/\text{disease} \} \\
\text{else } \{ \} \} : \text{Symptom } \rightarrow \mathbb{M} \text{ Disease}
\]
Measure semantics

\[ [M \alpha] = (\alpha \to \mathbb{R}) \to \mathbb{R} \]
Measure semantics

\[
\llbracket M \alpha \rrbracket = (\alpha \to \mathbb{R}) \to \mathbb{R}
\]

\[
\llbracket \{ A \mapsto 1/2, B \mapsto 1/2 \} \rrbracket(f) = \frac{f(A)}{2} + \frac{f(B)}{2}
\]

\[
\llbracket \text{return (symptom, disease)} \rrbracket(f) = f(\text{symptom, disease})
\]

\[
\llbracket \text{uniform 1 3} \rrbracket(f) = \int_{1}^{3} \frac{f(x)}{2} \, dx
\]

\[
\llbracket \text{lebesgue} \rrbracket(f) = \int_{-\infty}^{\infty} f(x) \, dx
\]

\[
\llbracket \text{do \{ } x \sim m; k \text{ \}} \rrbracket(f) = \llbracket m \rrbracket(\lambda x. \llbracket k \rrbracket f)
\]
Measure semantics

\[
\begin{align*}
\left[ M \alpha \right] &= (\alpha \to \mathbb{R}) \to \mathbb{R} \\
\left[ \lambda A \mapsto 1/2, B \mapsto 1/2 \right](f) &= \frac{f(A)}{2} + \frac{f(B)}{2} \\
\left[ \text{return } (\text{symptom, disease}) \right](f) &= f(\text{symptom, disease}) \\
\left[ \text{uniform } 1 \ 3 \right](f) &= \int_{1}^{3} \frac{f(x)}{2} \, dx \\
\left[ \text{lebesgue} \right](f) &= \int_{-\infty}^{\infty} f(x) \, dx \\
\left[ \text{do } \{ x \leftarrow m; k \} \right](f) &= \left[ m \right] (\lambda x. \left[ k \right] f) \\
\left[ \text{do } \{ d \leftarrow \text{uniform } 1 \ 3; \right. \\
& \quad s \leftarrow \text{uniform } 0 \ d; \ \left. \right\} \text{return } (s, d) \} \right](f) &= \int_{1}^{3} \int_{0}^{d} \frac{f(s, d)}{2 \cdot d} \, ds \, dd
\end{align*}
\]

Patently linear in f
Disintegration specification

\[[m] = [\textbf{do}\{s \sim \text{lebesgue}; d \sim k; \textbf{return} (s, d)\}]\]
Disintegration specification

\[
\mathbb{[m]} = \mathbb{[do \{ s \sim \text{lebesgue}; d \sim k; \ \text{return} \ (s, d) \}]}
\]

\[m = \text{do} \{ d \sim \text{uniform} \ 1 \ 3; \]
\[s \sim \text{uniform} \ 0 \ d; \]
\[\text{return} \ (s, d) \}\]

\[k = \text{do} \{ d \sim \text{uniform} \ 1 \ 3; \]
\[\text{if} \ 0 \leq s \leq d \]
\[\text{then} \ \{ \text{d} \mapsto \text{1/d} \}
\[\text{else} \ \} \]

\[\int_{3}^{1} \\text{if} \ 0 \leq s \leq d \]
\[\text{then} \ \text{d} \mapsto \text{1/d}\]
\[\text{else} \ \}

\[\int_{3}^{1} \int_{d}^{0} f(s, d) \cdot d \cdot d\]
\[= \mathbb{J}_{m}[f] \]
Disintegration specification

\[ [m] = \left[ \text{do } \{ s \sim \text{lebesgue}; d \sim k; \text{return } (s, d) \} \right] \]

\[ m = \text{do } \{ d \sim \text{uniform } 1 \ 3; \]
\[ \quad s \sim \text{uniform } 0 \ d; \]
\[ \quad \text{return } (s, d) \} \]

\[ k = \text{do } \{ d \sim \text{uniform } 1 \ 3; \]
\[ \quad \text{if } 0 \leq s \leq d \]
\[ \quad \text{then } \{ d \mapsto 1/d \}
\[ \quad \text{else } \emptyset \} \}

\[ [k](f) = \int_{1}^{3} \frac{\text{if } 0 \leq s \leq d \text{ then } f(d)/d \text{ else } 0}{2} \ d d \]

\[ [\text{do } \{ s \sim \text{lebesgue}; d \sim k; \text{return } (s, d) \}](f) \]
\[ = \int_{-\infty}^{\infty} \int_{1}^{3} \frac{\text{if } 0 \leq s \leq d \text{ then } f(s, d)/d \text{ else } 0}{2} \ d d \ d s \]
\[ = \int_{1}^{3} \int_{0}^{d} \frac{f(s, d)}{2 \cdot d} \ d s \ d d = [m] \]
Useful but unspecified and thus unautomated before

\[ q(x^\star \mid \mid x(i)) = \begin{cases} 
 p(x^\star_j \mid \mid x(i) - j) & \text{if } x^\star - j = x(i) - j \\
 0 & \text{otherwise} \end{cases} \]

The corresponding acceptance probability is:

\[ A(x(i), x^\star) = \min\{1, \frac{p(x^\star) q(x(i))}{p(x^\star - j) q(x(i) - j)}\} = \min\{1, \frac{p(x^\star - j) p(x(i) - j)}{p(x(i) - j)}\} = 1. \]

That is, the acceptance probability for each proposal is one and, hence, the deterministic scan Gibbs sampler algorithm is often presented as shown in Figure 12.

Since the Gibbs sampler can be viewed as a special case of the MH algorithm, it is possible to introduce MH steps into the Gibbs sampler. That is, when the full conditionals are available and belong to the family of standard distributions (Gamma, Gaussian, etc.), we will draw the new samples directly. Otherwise, we can draw samples with MH steps embedded within the Gibbs algorithm. For \( n = 2 \), the Gibbs sampler is also known as the data augmentation algorithm, which is closely related to the expectation maximisation (EM) algorithm (Dempster, Laird, & Rubin, 1977; Tanner & Wong, 1987).

Directed acyclic graphs (DAGS) are one of the best known application areas for Gibbs sampling (Pearl, 1987). Here, a large-dimensional joint distribution is factored into a directed graph that encodes the conditional independencies in the model. In particular, if \( x_{pa}(j) \)

**Figure 12.** Gibbs sampler.

(Borel paradox)
Determinism requires inversion

\[
\text{do } \{d \sim \text{uniform } 0\; 1; \\
\quad s \sim \text{return } (2 \cdot d); \\
\quad \text{return } (s, d)\}
\]
Determinism requires inversion

\[
do \ \{d \sim \text{uniform } 0 \ 1; \\
\quad s \sim \text{return } (2 \cdot d); \\
\quad \text{return } (s, d)\}
\]

\[
do \ \{d_1 \sim \text{uniform } 0 \ 1; \\
\quad d_2 \sim \lfloor 1 \mapsto 1/2, 2 \mapsto 1/2 \rfloor; \\
\quad s \sim \text{return } d_1^{d_2}; \\
\quad \text{return } (s, (d_1, d_2))\}
\]

\text{(Deterministic observable)}
Question: Is it correct that Grigori Grigorievich Grigoriev won a luxury car at the All-Union Championship in Moscow?

Answer: In principle, yes.

But first of all it was not Grigori Grigorievich Grigoriev, but Vassili Vassilievich Vassiliev.

Second, it was not at the All-Union Championship in Moscow, but at a Collective Farm Sports Festival in Smolensk.

Third, it was not a car, but a bicycle.

And fourth he didn’t win it, but rather it was stolen from him.
Automatic disintegrator

Question: Is it correct that our disintegrator is a lazy evaluator?
Answer: In principle, yes.

evaluate : \[ \lceil \alpha \rceil \rightarrow H \rightarrow (\alpha \times H) \]
Automatic disintegrator

Question: Is it correct that our disintegrator is a lazy evaluator?
  
  Answer: In principle, yes.
  
  But first of all it is not an evaluator, but a partial evaluator.

\[
\text{evaluate : } [\alpha] \rightarrow H \rightarrow ([\alpha] \times H)
\]
Automatic disintegrator

Question: Is it correct that our disintegrator is a lazy evaluator?

Answer: In principle, yes.

But first of all it is not an evaluator, but a partial evaluator.

Second, it not only evaluates terms, but also performs random choices.

\[
\begin{align*}
evaluate: \; & \left[ \alpha \right] \rightarrow H \rightarrow (\left[ \alpha \right] \rightarrow H \rightarrow [M \gamma]) \rightarrow [M \gamma] \\
\text{perform:} \; & \left[ M \alpha \right] \rightarrow H \rightarrow (\left[ \alpha \right] \rightarrow H \rightarrow [M \gamma]) \rightarrow [M \gamma]
\end{align*}
\]
Automatic disintegrator

**Question:** Is it correct that our disintegrator is a lazy evaluator?

**Answer:** In principle, yes.

But first of all it is not an evaluator, but a partial evaluator.

Second, it not only evaluates terms, but also performs random choices.

Third, it not only produces outcomes and values, but also constrains them.

\[
\text{evaluate} : \left[ \alpha \right] \rightarrow H \rightarrow (\left[ \alpha \right] \rightarrow H \rightarrow [M \gamma]) \rightarrow [M \gamma]
\]

\[
\text{perform} : \left[ M \alpha \right] \rightarrow H \rightarrow (\left[ \alpha \right] \rightarrow H \rightarrow [M \gamma]) \rightarrow [M \gamma]
\]

\[
\text{constrain-value} : \left[ \alpha \right] \rightarrow [\alpha] \rightarrow H \rightarrow (H \rightarrow [M \gamma]) \rightarrow [M \gamma]
\]

\[
\text{constrain-outcome} : \left[ M \alpha \right] \rightarrow [\alpha] \rightarrow H \rightarrow (H \rightarrow [M \gamma]) \rightarrow [M \gamma]
\]
Question: Is it correct that our disintegrator is a lazy evaluator?

Answer: In principle, yes.

But first of all it is not an evaluator, but a partial evaluator.
Second, it not only evaluates terms, but also performs random choices.
Third, it not only produces outcomes and values, but also constrains them.
And fourth it doesn’t produce one term, but searches for a random variable to constrain.

evaluate : [α] → H → ([α] → H → {[M γ]}) → {[M γ]}
perform : [M α] → H → ([α] → H → {[M γ]}) → {[M γ]}
constrain-value : [α] → [α] → H → (H → {[M γ]}) → {[M γ]}
constrain-outcome : [M α] → [α] → H → (H → {[M γ]}) → {[M γ]}
Automatic disintegrator in action

\[
\begin{align*}
&\text{perform (do \{d \leftarrow \textbf{uniform} \ 1 \ 3; \ s \leftarrow \textbf{uniform} \ 0 \ d; \ \textbf{return} \ (s, \ d)\})} \\
&\quad [d' \leftarrow \textbf{uniform} \ 1 \ 3] \\
&\text{perform (do \{s \leftarrow \textbf{uniform} \ 0 \ d'; \ \textbf{return} \ (s, \ d')\})} \\
&\quad [d' \leftarrow \textbf{uniform} \ 1 \ 3; \ s' \leftarrow \textbf{uniform} \ 0 \ d'] \\
&\text{perform (return \ (s', \ d'))} \\
&\text{evaluate \ (s', \ d') \Rightarrow (s', \ d')} \\
&\text{constrain-value \ s' \ s} \\
&\text{constrain-outcome (\textbf{uniform} \ 0 \ d') \ s}
\end{align*}
\]
Automatic disintegrator in action

perform \( (\text{do } \{d \sim \text{uniform } 1 3; s \sim \text{uniform } 0 d; \text{ return } (s, d)\}) \)[]

perform \( (\text{do } \{s \sim \text{uniform } 0 d'; \text{ return } (s, d')\}) \)

perform \( (\text{return } (s', d')) \)

evaluate \( (s', d') \Rightarrow (s', d') \)

constrain-value \( s' \)

constrain-outcome \( (\text{uniform } 0 d') \)

evaluate \( 0 \Rightarrow 0 \)

evaluate \( d' \)

perform \( \text{(uniform } 1 3) \)

\( \Rightarrow d'' \)

\( \Rightarrow d'' \)

if \( 0 \leq s \leq d'' \) then do \( \{() \sim (() \mapsto 1/d''); \square\} \) else \( \int \)

\( \text{let } d' = d''; \text{ let } s' = s \)
Automatic disintegrator in action

perform \((\text{do}\ \{d \sim \text{uniform} \ 1 \ 3; \ s \sim \text{uniform} \ 0 \ d; \ \text{return} \ (s, d)\})\) 
\[\text{[}d' \sim \text{uniform} \ 1 \ 3\]\n
perform \((\text{do}\ \{s \sim \text{uniform} \ 0 \ d'; \ \text{return} \ (s, d')\}\)) 
\[\text{[}d' \sim \text{uniform} \ 1 \ 3; \ s' \sim \text{uniform} \ 0 \ d'\]\n
evaluate \((s', d') \Rightarrow (s', d')\)

constrain-value \(s'\) \(s\)

constrain-outcome \((\text{uniform} \ 0 \ d')\) \(s\)
evaluate \(0 \Rightarrow 0\)
evaluate \(d'\)

perform \((\text{uniform} \ 1 \ 3)\) 
\[\Rightarrow \ d''\]
\[\Rightarrow \ d''\]

\[\text{if} \ 0 \leq s \leq d'' \ \text{then do} \ \{() \sim \mathcal{L}(\cdot) \mapsto 1/d''; \ \square\} \ \text{else do} \ \{\text{[let} \ d' = d''; \ \text{let} \ s' = s\} \]
Interim summary

- Generate observed symptoms from hidden causes
- **First exact inference algorithm** for continuous distributions
- **Enable modular composition** of inference techniques
- **Lessons for language design** and reasoning
- Ongoing work: arrays
  - more dominating measures
  - more deterministic observables
  - prove correctness
Hakaru: meaningful and reusable, from clear to fast

FLOPS 2016 system description paper
“Probabilistic inference by program transformation in Hakaru”

NIPS 2015 workshop poster
“Building blocks for exact and approximate inference”

Model
- disintegrate

Posterior
- simplify
- expect

Simplified posterior

Transition kernel
- mh sampling
  - disintegrate
  - expect
- gibbs sampling
  - disintegrate
  - expect

Idiomatic WebPPL ➔ 1321 ms
1078 ms
267 ms
207 ms

Handwritten Hakaru ➔ 1321 ms
1078 ms
267 ms
207 ms
Hakaru: meaningful and reusable, from clear to fast

FLOPS 2016 system description paper
“Probabilistic inference by program transformation in Hakaru”

NIPS 2015 workshop poster
“Building blocks for exact and approximate inference”

Model
\[ \text{disintegrate} \]

Posterior
\[ \text{simplify} \]
\[ \text{expect} \]

Simplified posterior

\begin{align*}
\text{Idiomatic WebPPL} & \rightarrow 1078 \text{ ms} \\
\text{Handwritten Hakaru} & \rightarrow 207 \text{ ms} \\
\end{align*}

\[ \text{1321 ms} \]
\[ \text{1078 ms} \]
\[ \text{267 ms} \]
\[ \text{207 ms} \]

Transition kernel
\[ \text{Simplify} \]
\[ \text{expect} \]

Simplified kernel

\[ \text{Simplify} \]
\[ \text{expect} \]

\[ \text{Simplify} \]
\[ \text{expect} \]
Simplifying probabilistic programs via semantics

Probabilistic program

Continuation passing

Abstract integral

Computer algebra

Improved integral

Computer algebra

Simplified program

PADL 2016 paper
“Simplifying probabilistic programs using computer algebra”
Simplifying a discrete distribution

Let $x \sim \text{uniform } 0 1$; $y \sim \text{uniform } 0 1$; if $x < y$ then return true else return false.

Integral expression:

$$\int_0^1 \int_0^1 \left( \begin{array}{l} f(\text{true}) \quad \text{if } x < y \\ f(\text{false}) \quad \text{otherwise} \end{array} \right) \, dy \, dx$$

Symbolic integration:

$$\frac{1}{2} \cdot f(\text{true}) + \frac{1}{2} \cdot f(\text{false})$$

PADL 2016 paper

“Simplifying probabilistic programs using computer algebra”
Simplifying a continuous distribution

PADL 2016 paper
“Simplifying probabilistic programs using computer algebra”

```
\begin{align*}
\text{do } & \{ x \sim \text{normal } 0 1; \\
& \quad y \sim \text{normal } x 1; \\
& \quad \text{return } y \}\}
\end{align*}
```

Symbolic integration

\[
\int_{-\infty}^{\infty} \frac{\exp\left(-\frac{x^2}{2}\right)}{\sqrt{2 \cdot \pi}} \cdot \int_{-\infty}^{\infty} \frac{\exp\left(-\frac{(y-x)^2}{2}\right)}{\sqrt{2 \cdot \pi}} \cdot f(y) \, dy \, dx
\]

Holonomic representation

\[
\int_{-\infty}^{\infty} \frac{\exp\left(-\frac{y^2}{4}\right)}{2 \cdot \sqrt{\pi}} \cdot f(y) \, dy
\]

normal 0 \sqrt{2}
Simplifying a conditional distribution

PADL 2016 paper
“Simplifying probabilistic programs using computer algebra”

\[
\int_{-\infty}^{\infty} \frac{\exp\left(-\frac{x^2}{2}\right)}{\sqrt{2 \cdot \pi}} \cdot \frac{\exp\left(-\frac{(y-x)^2}{2}\right)}{\sqrt{2 \cdot \pi}} \cdot f(x) \, dx
\]

Holonomic representation

Use conjugacies \textbf{(normal, gamma, ...)} without pairwise hard-coding
Summary

General strategy:
- represent concepts
- formal, hence **executable**
- **meaningful**, hence modular

Meaning-preserving **transformations** on probabilistic programs!
- disintegrate
- simplify
- etc.

Example **modularity** payoffs:
- combine exact and approximate techniques without re-coding
- use conjugacies without pairwise hard-coding
- same **performance**