Programmable MCMC with Soundly Composed Guide Programs

Long Pham[1], Di Wang[2], Feras Saad[1], Jan Hoffmann[1]

[1]Carnegie Mellon University

[2]Peking University

OOPSLA 2024

October 25, 2024

Probabilistic Programming

1. Probabilistic model $p(\boldsymbol{\theta}, \boldsymbol{y})$ as a (model) program

- $\boldsymbol{\theta}$: latent variable
- $\boldsymbol{y}$: observed variable

May contain conditional statements and loops

2. Posterior distribution by Bayes' rule:

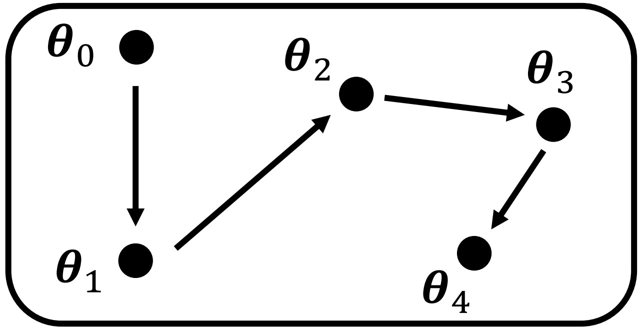
$$p(\,\boldsymbol{\theta} \mid \boldsymbol{y} = D\,) = \frac{p(\boldsymbol{\theta}, \boldsymbol{y} = D)}{\int p(\boldsymbol{\theta}, \boldsymbol{y} = D)\, d\boldsymbol{\theta}}$$

Integral over the space of $\boldsymbol{\theta}$ is difficult

Sampling-based inference algorithms (e.g., Markov-chain Monte Carlo)

$$\boldsymbol{\theta}_1^* \sim \overbrace{q(\boldsymbol{\theta} \mid \boldsymbol{\theta}_0)}^{\text{Proposal distribution}}$$

$$\boldsymbol{\theta}_1 \coloneqq \begin{cases} \boldsymbol{\theta}_1^* & \text{some probability} \\ \boldsymbol{\theta}_0 & \text{otherwise} \end{cases}$$



Example: Bayesian Polynomial Regression

Model program probabilistically generates a polynomial degree and coefficients

$d \sim \text{Categorical}(0.3, 0.5, 0.2)$
$c_0 \sim \text{Normal}(\ldots)$
if $d \geq 1$ then
  $c_1 \sim \text{Normal}(\ldots)$
  if $d \geq 2$ then
    $c_2 \sim \text{Normal}(\ldots)$
$y \sim \mathrm{polynomial}_{\vec{c}}(x)$

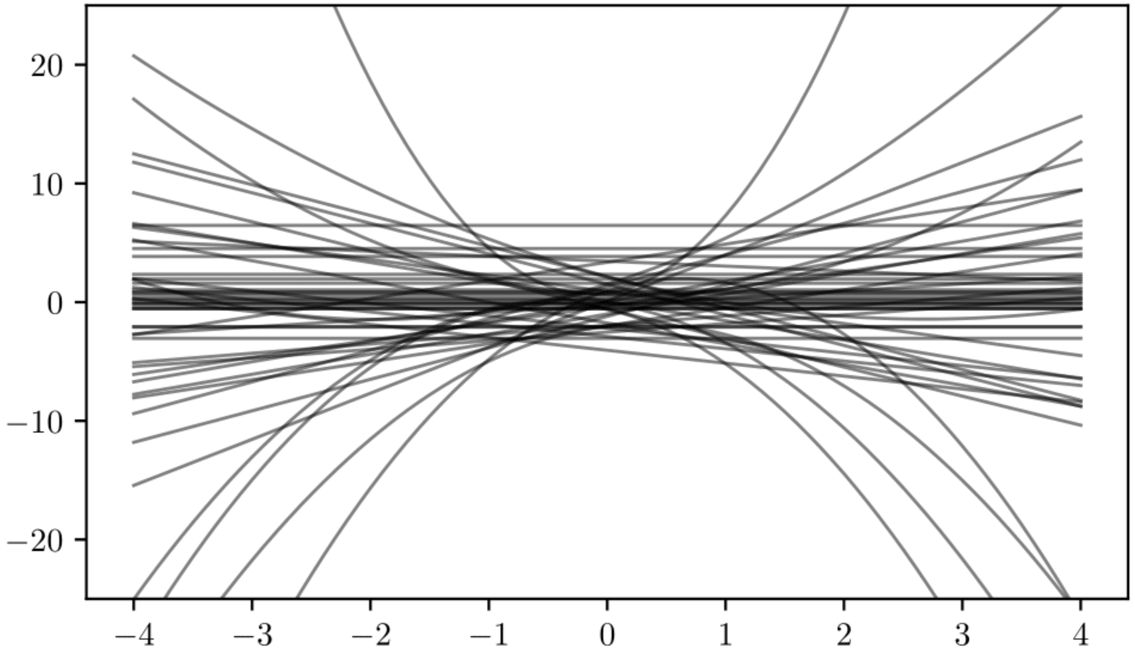
1. Polynomial degree d
2. Constant coefficient c_0
3. Generate coefficients c_1 and c_2 if necessary
4. Generate observed variable y

The set of random variables depends on the execution path

Example: Bayesian Polynomial Regression

Model program probabilistically generates a polynomial degree and coefficients

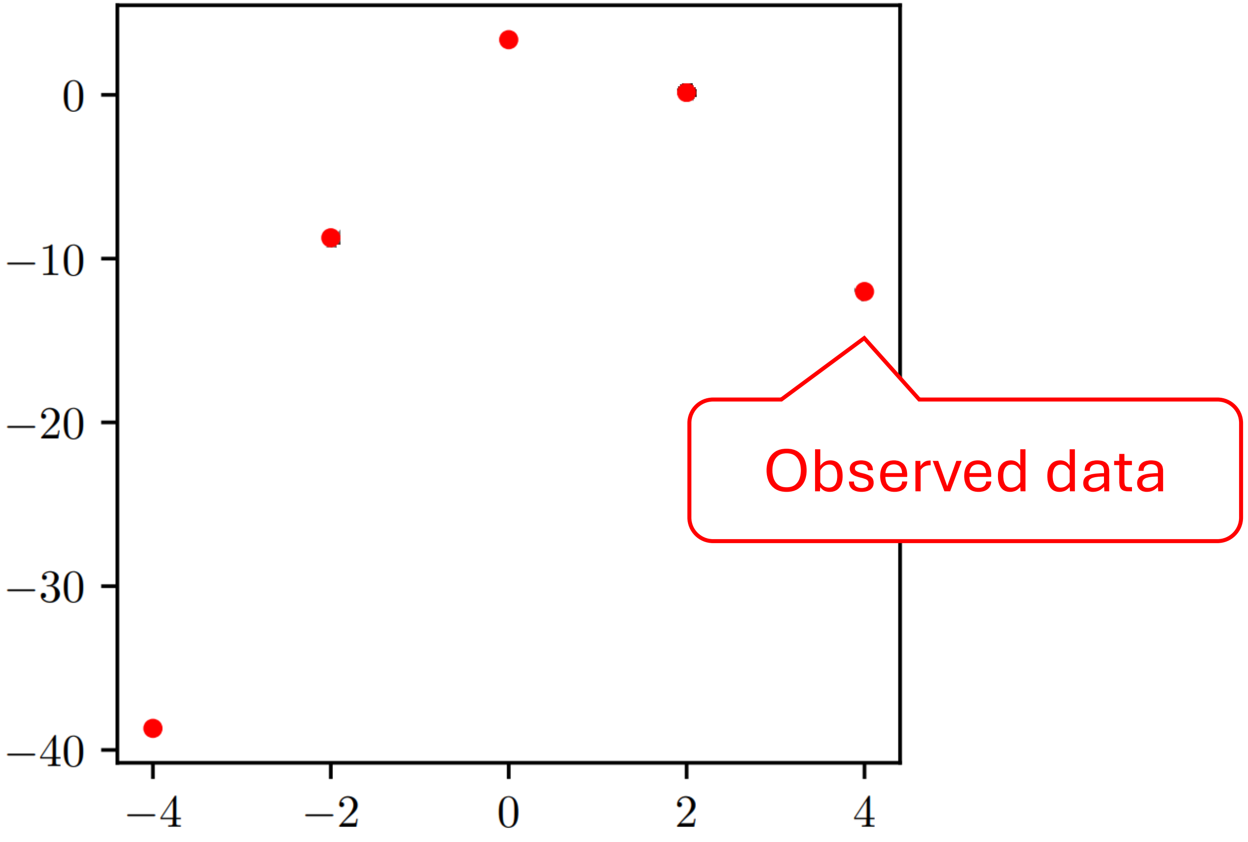
d ~ Categorical(0.3, 0.5, 0.2)
c_0 ~ Normal(…)
if $d \geq 1$ then
  c_1 ~ Normal(…)
  if $d \geq 2$ then
    c_2 ~ Normal(…)
$y \sim \mathrm{polynomial}_{\vec{c}}(x)$



Example: Bayesian Polynomial Regression

Model program probabilistically generates a polynomial degree and coefficients

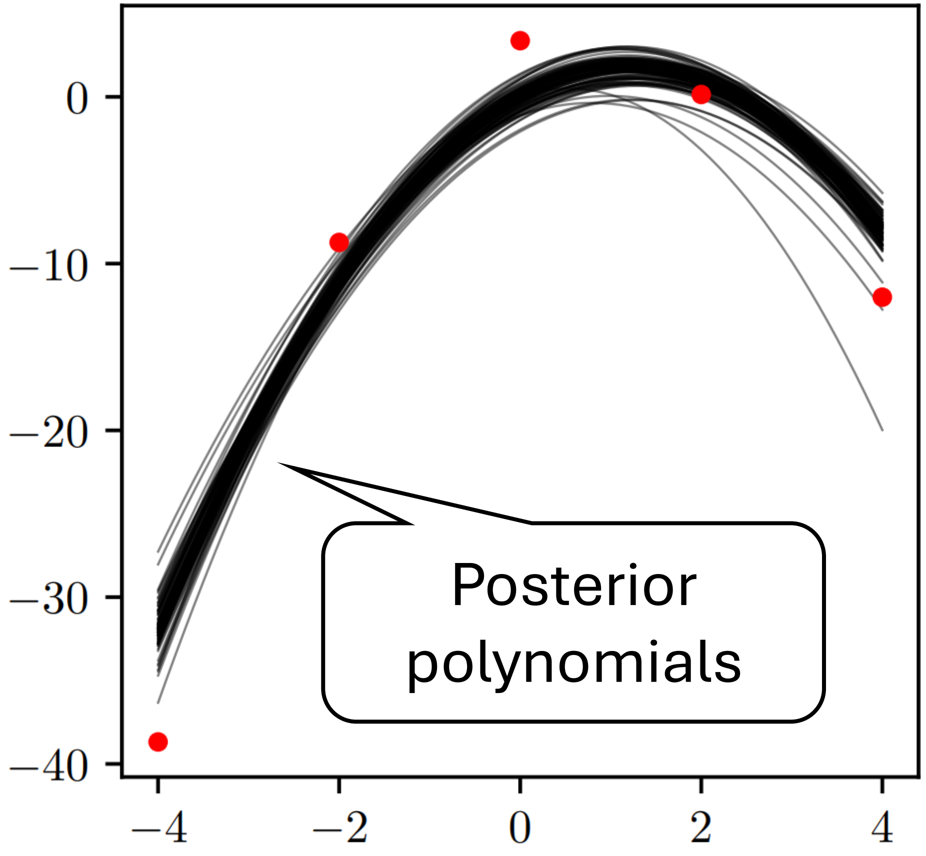
$d \sim \text{Categorical}(0.3, 0.5, 0.2)$
$c_0 \sim \text{Normal}(\ldots)$
if $d \geq 1$ then
  $c_1 \sim \text{Normal}(\ldots)$
  if $d \geq 2$ then
    $c_2 \sim \text{Normal}(\ldots)$
$y \sim \text{polynomial}_{\vec{c}}(x)$



Example: Bayesian Polynomial Regression

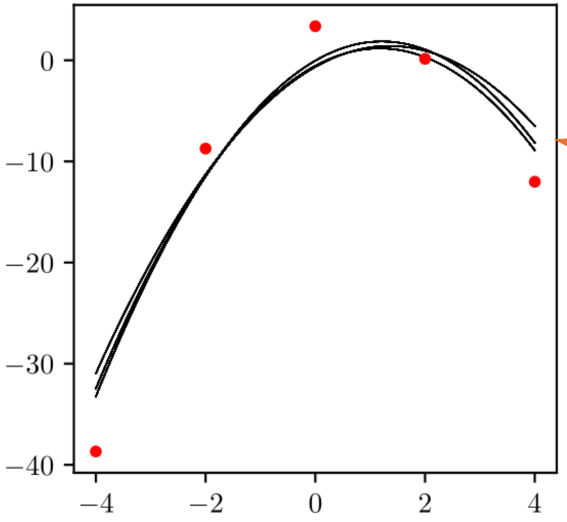
Model program probabilistically generates a polynomial degree and coefficients

$d \sim$ Categorical(0.3, 0.5, 0.2)
$c_0 \sim$ Normal(…)
if $d \geq 1$ then
  $c_1 \sim$ Normal(…)
  if $d \geq 2$ then
    $c_2 \sim$ Normal(…)
$y \sim \mathrm{polynomial}_{\vec{c}}(x)$



Programmable Inference: Guide Programs

Traditional PPLs use a generic inference algorithm, but it may perform poorly



Little variance within 50 posterior samples

Modern PPLs (e.g., Gen and Pyro) support programmable inference where the inference algorithm is customized by a user-written guide program

It customizes a proposal distribution for MCMC

Programmable Inference: BMH

Multiple-Block Metropolis-Hastings (BMH) samples r.v.s block by block

- Split latent variables into four blocks: d, c_0, c_1, c_2
- Specify proposal distributions for each block

Guide programs

Guide G_d

$\xrightarrow{\sigma_0}$

d ~ Categorical(...)
if $d \geq \sigma[d]$
  sample new
  coefficients

Sample d

Reuses old samples of existing r.v.s

$\xrightarrow{\sigma_1^*}$ MH update $\xrightarrow{\sigma_1}$

Guide G_{c_0}

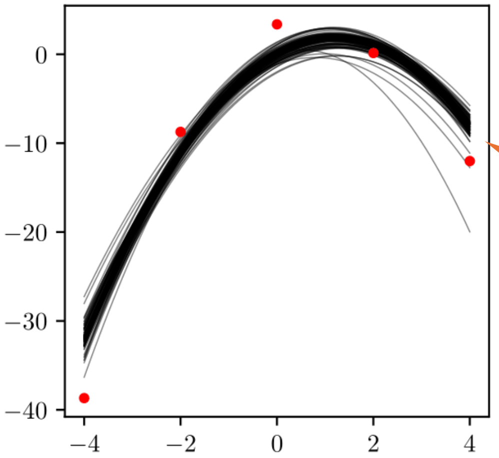
if $\sigma[d] \geq 0$
  c_0~ Normal(...)

...

Guide G_{c_2}

if $\sigma[d] \geq 2$
  c_2~ Normal(...)

$$\sigma_1 \coloneqq \begin{cases} \sigma_1^* & \text{with some probability} \\ \sigma_0 & \text{otherwise} \end{cases}$$



More variance within 50 posterior samples

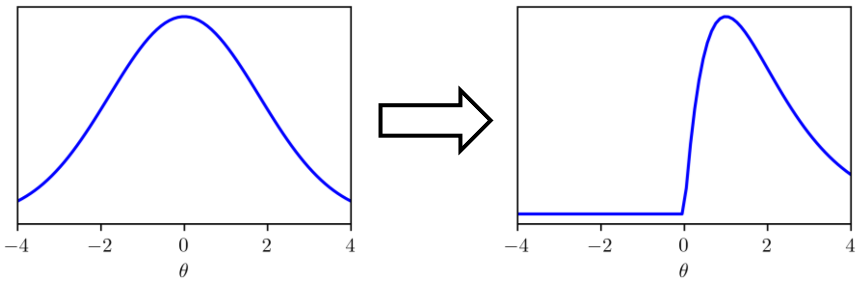
Contribution: Soundness Verification of BMH

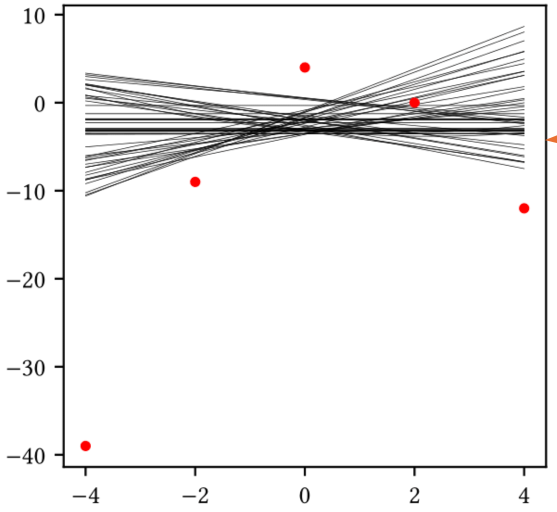
<u>Sufficient soundness condition of BMH</u>

Model and guide must have the same support

<u>Unsound BMH</u>: replace a normal distribution with a gamma distribution

Normal distribution Gamma distribution





Guide is unable to sample negative values for coefficient c_2

<u>Our contribution</u>

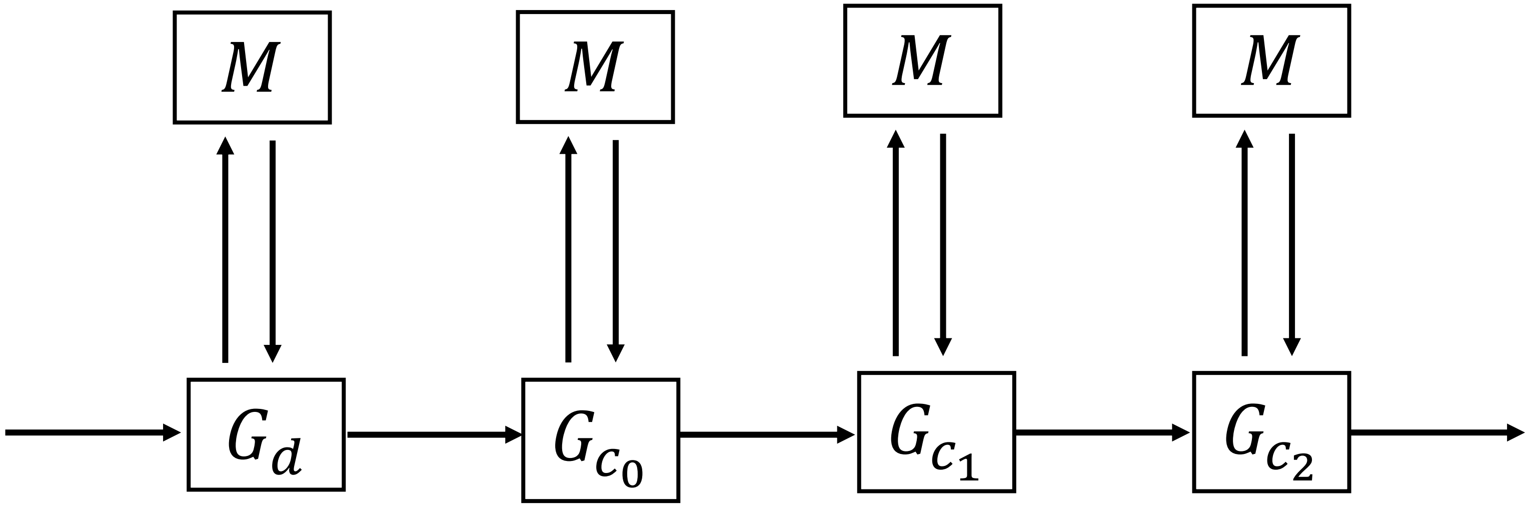
Develop a type-based framework for verifying that the model and guide have the same support in BMH

Outline

- ❑Motivation for soundness verification of BMH
- ❑Formulation of BMH
- ❑Polynomial-time decidability of structural guide-type equality
- ❑Coverage checking
- ❑Implementation and evaluation

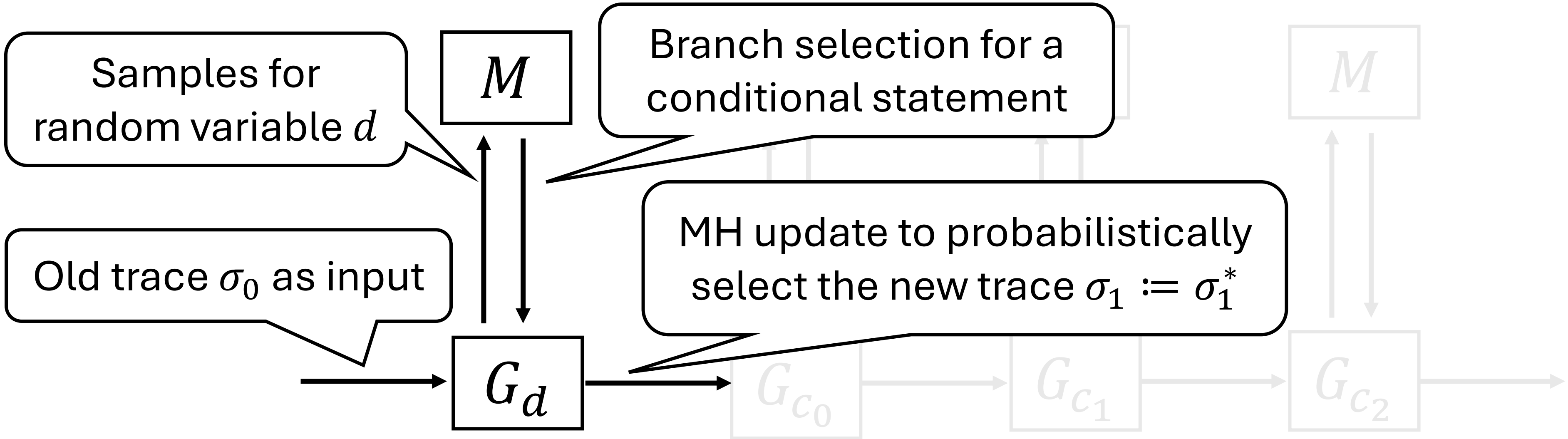
Contribution 1: Formulation of BMH

BMH is a sequential composition of guide programs



Contribution 1: Formulation of BMH

BMH is a sequential composition of guide program



The communication protocol is described by a guide type:

$$\underbrace{\mathbb{N}_3}_{\text{Distribution type of } d} \wedge \mathbb{R} \overbrace{\wedge \,\&}^{\text{Branch selection}} \begin{cases} 1, & \text{Termination} \\ \mathbb{R} \wedge \& \begin{cases} 1, \\ \mathbb{R} \wedge 1 \end{cases} \end{cases}$$

Contribution 2: Structural Guide-Type Equality

Soundness ingredient

The model and guide must have an equal guide type

Context-free guide types

Types indexed by type parameters

$$\underbrace{T[X]}_{\substack{\text{Run a protocol } T\text{, followed} \\ \text{by a continuation } X}} \coloneqq \mathbb{R} \wedge \left(X \mathbin{\&} \underbrace{T\big[T[X]\big]}_{\substack{\text{The recursive call is a} \\ \text{sequential composition of } T}}\right)$$

Guide types may have infinite state spaces: $T[X]$, $T[T[X]]$, $T[T[T[X]]]$, …

Contribution 2: Structural Guide-Type Equality

Structural guide-type equality is decidable in polynomial time

1. Translate guide types to context-free processes

Context-free grammars viewed as processes

2. Require the norms (i.e., #of steps to reach termination) to be finite

3. Theorem (Hirshfeld et al., 1994). Bisimilarity of two context-free processes (i.e., structural guide-type equality) with finite norms is decidable in polynomial time.

Outline

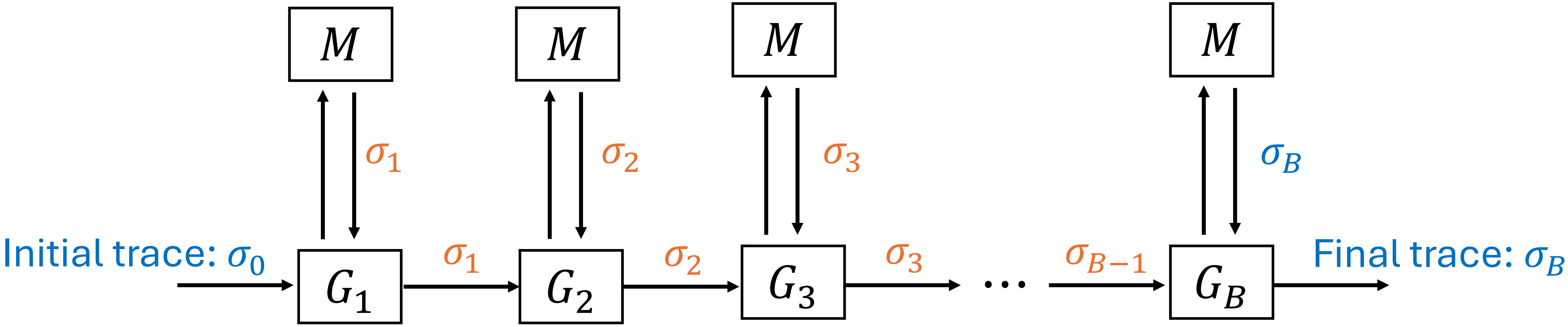
- ❑Motivation for soundness verification of BMH
- ❑Formulation of BMH
- ❑Polynomial-time decidability of structural guide-type equality
- ❑Coverage checking
- ❑Implementation and evaluation

Contribution 3: Coverage Checking of Composed Guides

Every random variable must be covered (i.e., sampled freshly by at least one guide) on any execution path

<u>Problem statement of coverage checking</u>

For an arbitrary initial trace σ_0 and final trace σ_B,
do there exist intermediate traces $\sigma_1, \ldots, \sigma_{B-1}$ such that σ_i can be generated from σ_{i-1} $(i = 1, \ldots B)$?



Contribution 3: Coverage Checking of Composed Guides

Coverage-annotated guide types

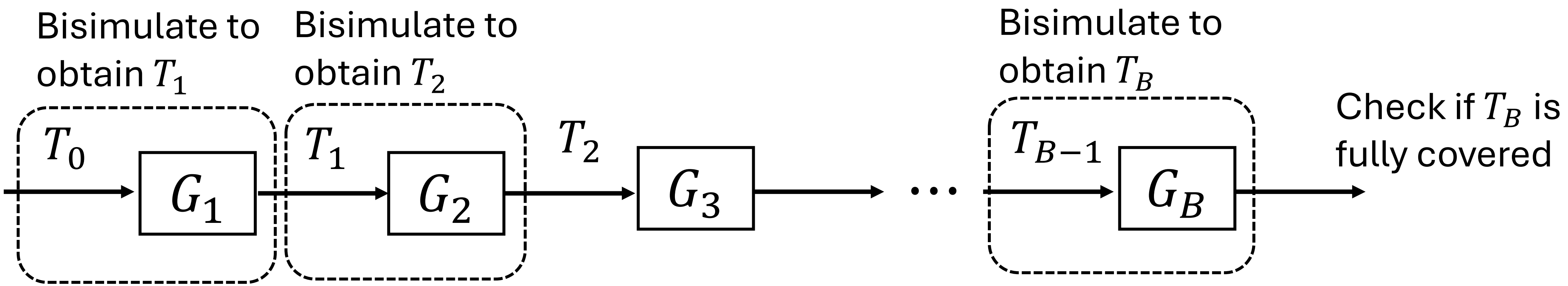
$$\mathbb{R}_u \wedge \& \begin{cases} 1, \\ \mathbb{R}_c \wedge 1 \end{cases}$$

Subscript u means the sample is reused

Subscript c means the sample is freshly sampled

Coverage-checking algorithm

Starting with a fully uncovered guide type T_0, we bisimulate a guide type T_{i-1} alongside G_i to update coverage annotations



Implementation and Evaluation

Table 1. Experiment results of guide-type inference and coverage checking of 28 benchmark programs.

Program	Description	Guide Type	LOC	Type Inference: Time (ms)	Type Inference: Prior Work	Coverage Check: Match	Coverage Check: Mismatch	Coverage Check: Time (ms)
branching	Random control flow [Anglican]	Finite	46	1.33	✓	True Pos.	True Neg.	0.46
coordination	Coordination game [Anglican]	Finite	24	0.19	✓	True Pos.	True Neg.	0.34
drill	Oil wildcatter problem [Anglican]	Finite	56	0.17	✓	True Pos.	True Neg.	0.37
ex-1	Ex. 1 [52]	Finite	42	1.31	✓	True Pos.	True Neg.	0.46
gaussian	Gaussian with unknown means [Anglican]	Finite	20	0.16	✓	True Pos.	True Neg.	0.46
gbm	Geometric Brownian motion [Anglican]	Finite	35	0.25	✓	True Pos.	True Neg.	0.52
gda	Gaussian discriminant analysis [Anglican]	Finite	40	1.86	✓	True Pos.	True Neg.	3.17
gmm	Gaussian mixture model [Anglican]	Finite	75	4.73	✓	True Pos.	True Neg.	7.71
grw	Gaussian random walk [Anglican]	Finite	24	0.17	✓	True Pos.	True Neg.	0.74
hmm	Hidden Markov model [Anglican]	Finite	76	2.56	✓	True Pos.	True Neg.	7.21
kalman	Kalman filter [Anglican]	Finite	72	4.44	✓	True Pos.	True Neg.	7.54
kalman-chaos	Kalman for chaotic attractors [Anglican]	Finite	114	5.86	✓	True Pos.	True Neg.	5.68
lr	Bayesian linear regression [Anglican]	Finite	36	0.19	✓	True Pos.	True Neg.	1.15
run-factory	Beta-binomial model [Anglican]	Finite	20	0.13	✓	True Pos.	True Neg.	0.61
scientists	Posterior estimation with Gaussians [54]	Finite	40	0.27	✓	True Pos.	True Neg.	0.52
seq	Non-recursive sequence [52]	Finite	22	0.23	✓	True Pos.	True Neg.	0.46
sprinkler	Bayesian network [Anglican]	Finite	26	0.14	✓	True Pos.	True Neg.	0.43
user-behavior	Dishonest form filling [Anglican]	Finite	64	1.22	✓	True Pos.	True Neg.	3.17
vae	Variational autoencoder [Pyro]	Finite	48	4.20	✓	True Pos.	True Neg.	22.39
weight	Unreliable weight [Pyro]	Finite	18	0.26	✓	True Pos.	True Neg.	0.70
aircraft	Aircraft detection [Anglican]	Regular	117	6.19	✗	True Pos.	True Neg.	5.96
iter	Regular iteration [52]	Regular	47	2.01	✗	True Pos.	True Neg.	0.54
marsaglia	Marsaglia algorithm [Anglican]	Regular	76	3.51	✗	True Pos.	True Neg.	5.13
ptrace	Poisson trace [Anglican]	Regular	47	1.49	✗	True Pos.	True Neg.	0.40
ex-2	Ex. 2 [52]	Context-Free	78	4.77	✗	True Pos.	True Neg.	4.70
			93	15.48	✗	False Neg.	True Neg.	3.26
diter	Double iteration [52]	Context-Free	52	1.48	✗	True Pos.	True Neg.	0.57
			62	2.09	✗	False Neg.	True Neg.	0.49
gp-dsl	Gaussian process DSL [52]	Context-Free	242	879.53	✗	True Pos.	True Neg.	4.71
			261	2487.91	✗	False Neg.	True Neg.	4.59
recur	Context-free recursion [52]	Context-Free	71	11.53	✗	True Pos.	True Neg.	16.32
			83	15.55	✗	False Neg.	True Neg.	6.35

Type equality and inference

Type inference of some BMH benchmarks require structural guide-type equality

Coverage checking

- Effective in benchmarks with regular guide types (i.e., finite state spaces).
- It can return False Negative for benchmarks with infinite-state guide types

Conclusion

1. Formulated Multiple-Block Metropolis-Hastings (BMH) in guide-based programmable inference
2. Proved polynomial-time decidability of structural guide-type equality
3. Developed a coverage-checking algorithm for verifying that every random variable is freshly sampled at least once