Sound Probabilistic Inference via Guide Types

Di Wang[1], Jan Hoffmann[1], Thomas Reps[2]

[1] Carnegie Mellon University
[2] University of Wisconsin-Madison

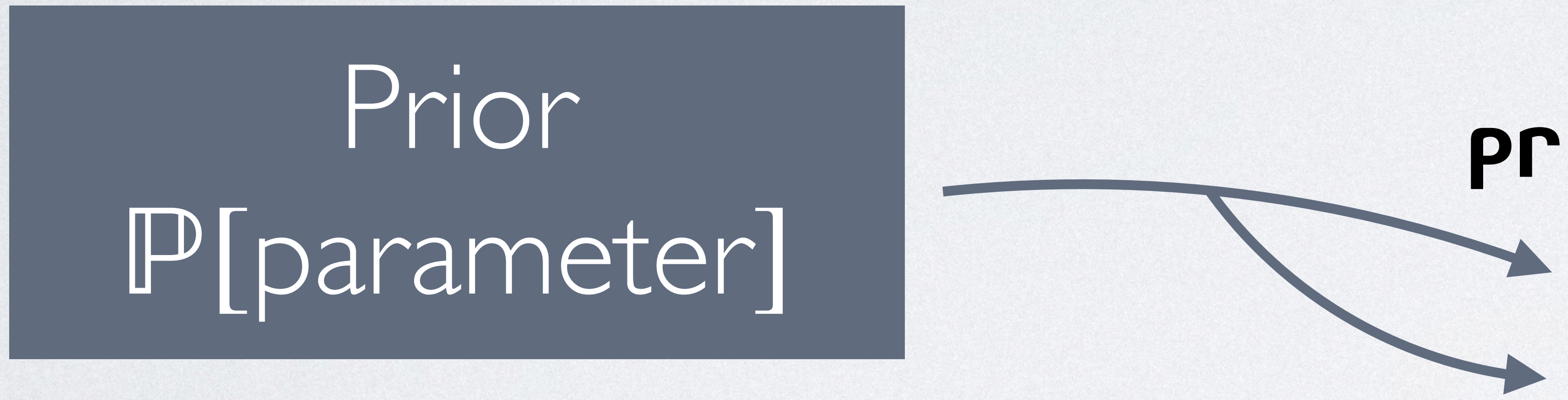
Probabilistic Programming

A Flexible Way of Describing Statistical Models

```
proc model() {
  param1 <- sample(Normal(2, 1));
  param2 <- sample(Normal(-2, 1));
  data <- sample(Normal(param1 * param2, 10));
  return
}
```

Probabilistic Programming

A Flexible Way of Describing **Statistical Models**



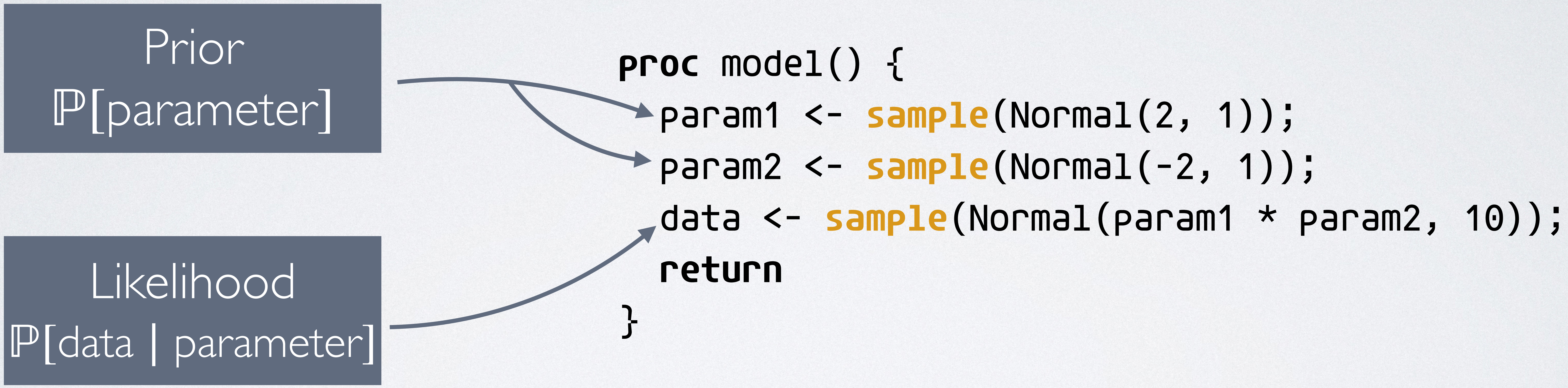
Prior

ℙ[parameter]

```
proc model() {
  param1 <- sample(Normal(2, 1));
  param2 <- sample(Normal(-2, 1));
  data <- sample(Normal(param1 * param2, 10));
  return
}
```

Probabilistic Programming

A Flexible Way of Describing **Statistical Models**



Prior

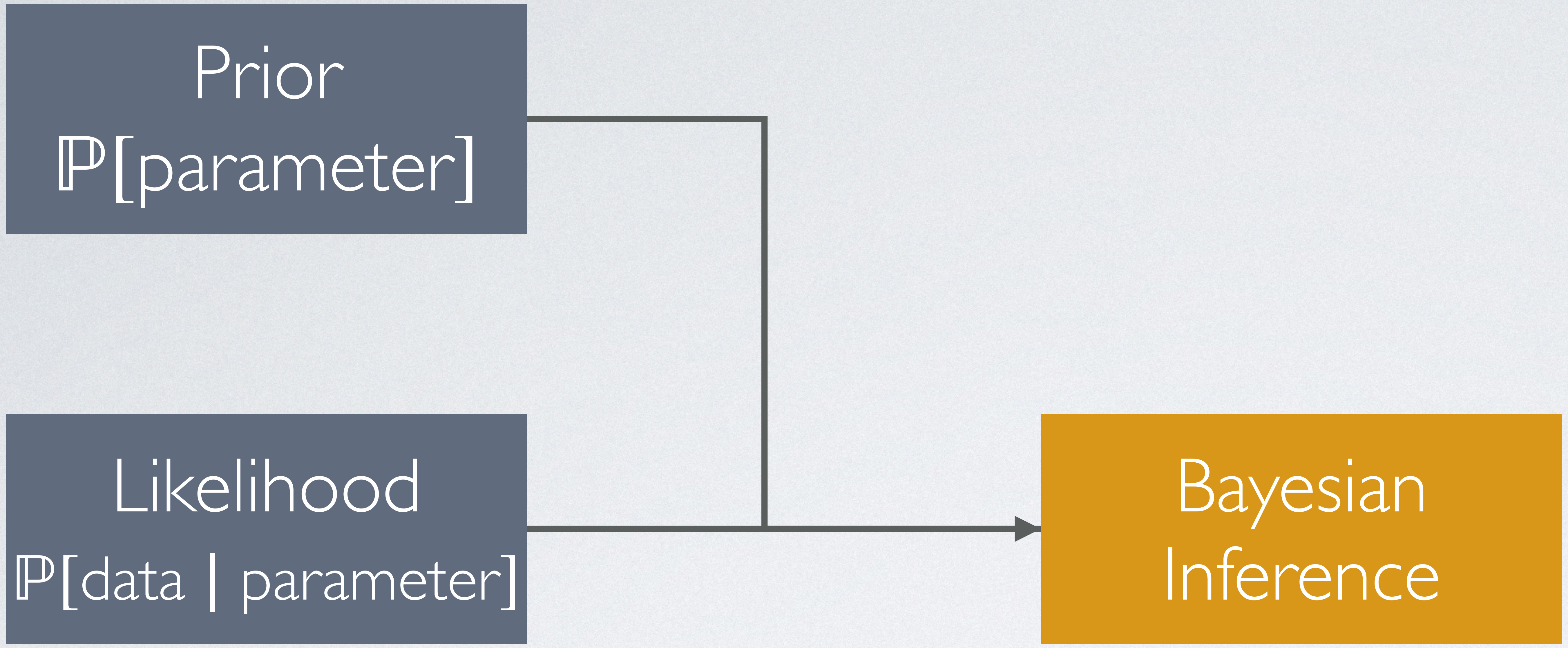
$\mathbb{P}$[parameter]

Likelihood

$\mathbb{P}$[data | parameter]

```
proc model() {
  param1 <- sample(Normal(2, 1));
  param2 <- sample(Normal(-2, 1));
  data <- sample(Normal(param1 * param2, 10));
  return
}
```

Bayesian Inference



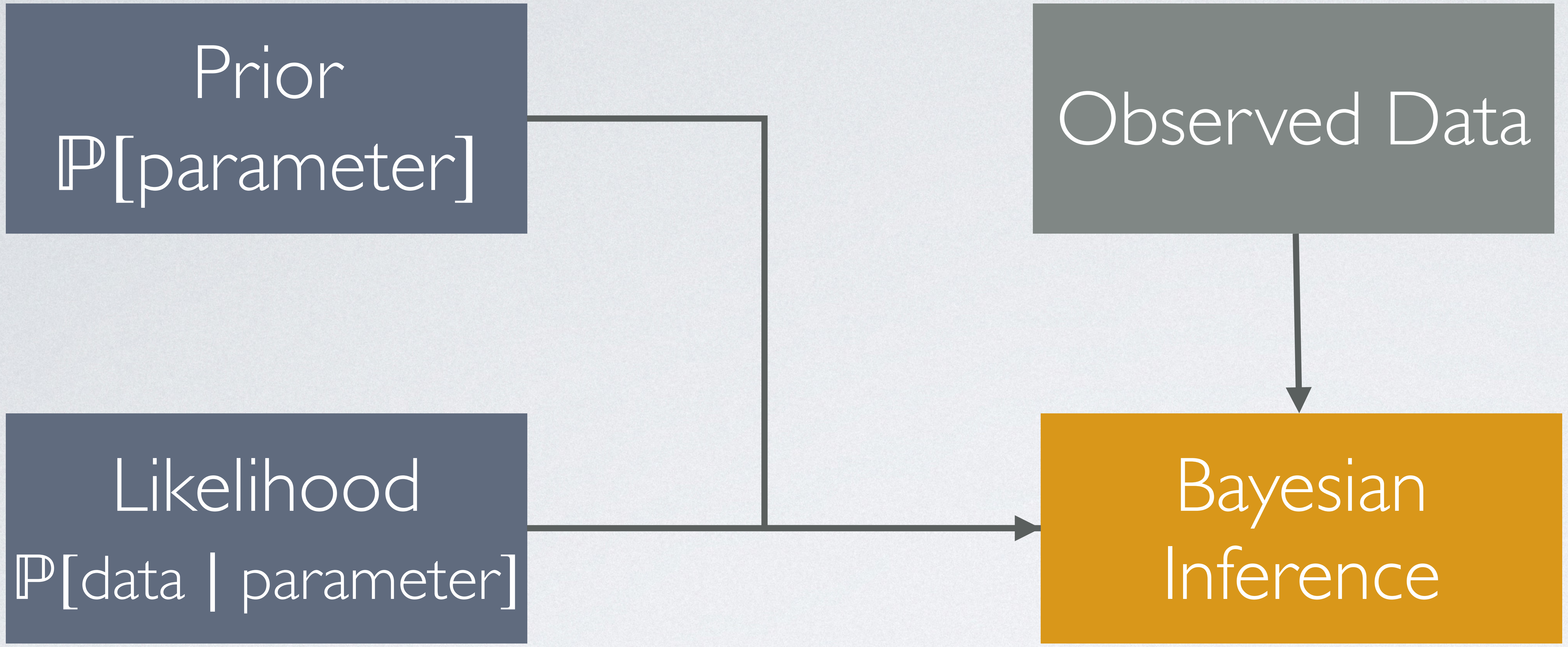
Prior

$\mathbb{P}$[parameter]

Likelihood

$\mathbb{P}$[data | parameter]

Bayesian Inference



Prior

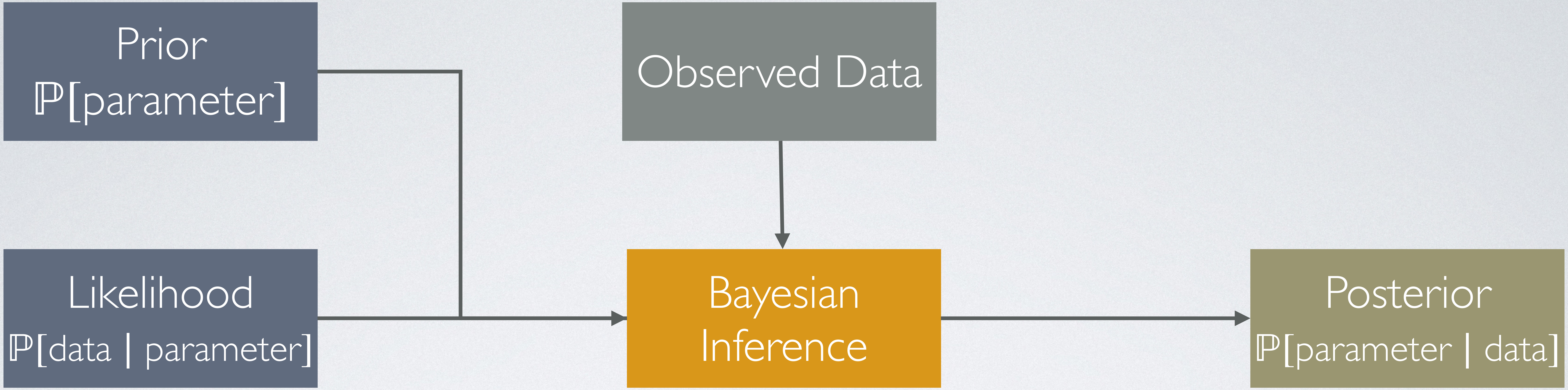
$\mathbb{P}[\text{parameter}]$

Likelihood

$\mathbb{P}[\text{data} \mid \text{parameter}]$

Observed Data

Bayesian Inference



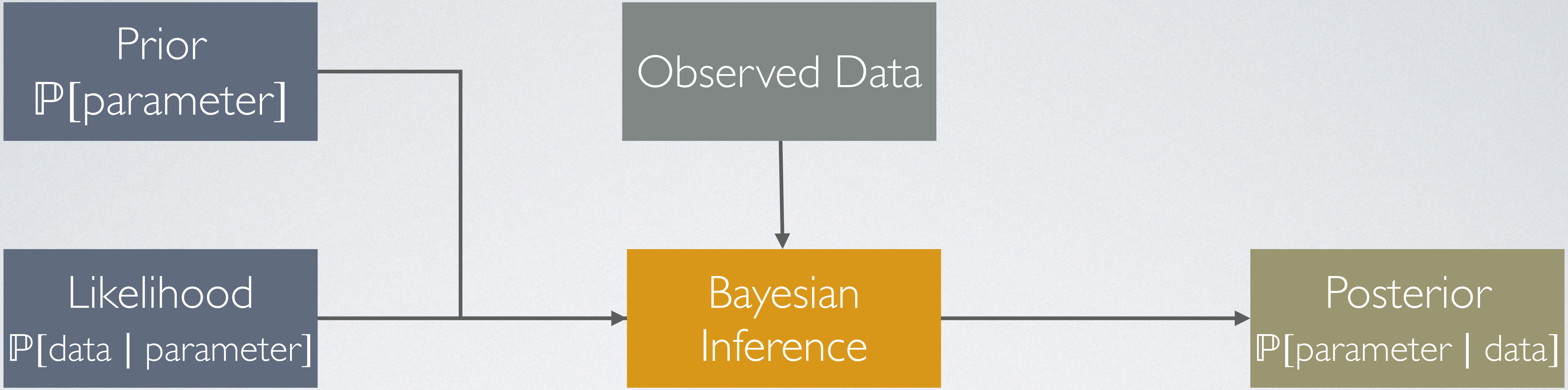
Prior
$\mathbb{P}[\text{parameter}]$

Observed Data

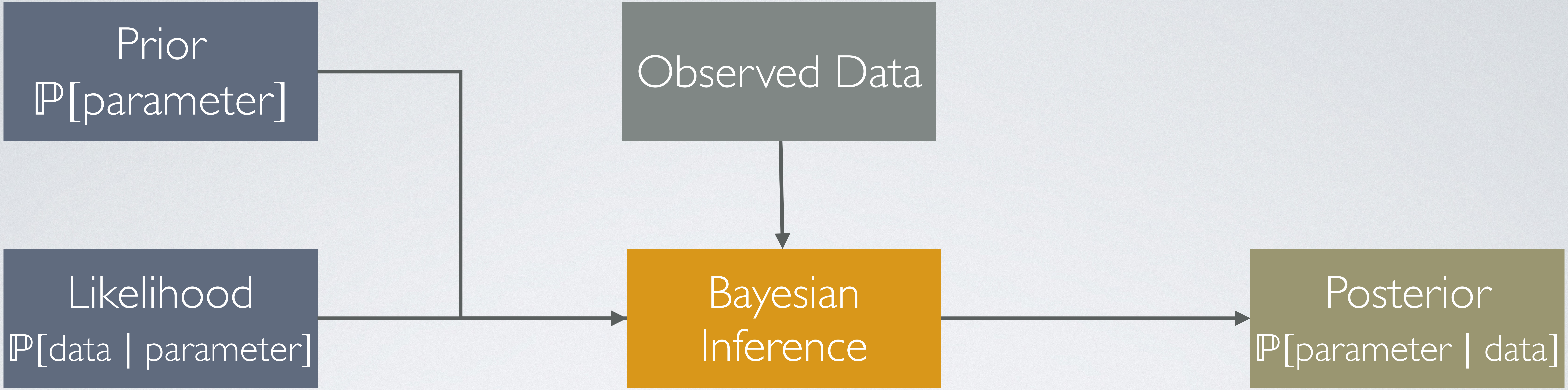
Likelihood
$\mathbb{P}[\text{data} \mid \text{parameter}]$

Bayesian Inference

Posterior
$\mathbb{P}[\text{parameter} \mid \text{data}]$

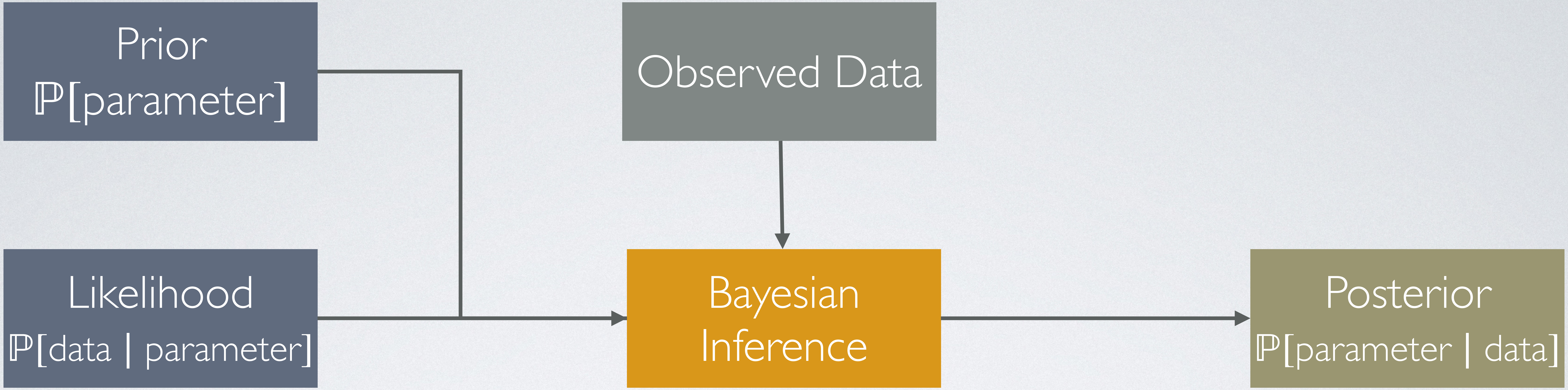


- Bayesian inference is good at reasoning about **uncertainty** in model parameters



- Bayesian inference is good at reasoning about **uncertainty** in model parameters

- **Downside:** No single algorithm works well for all models



- Bayesian inference is good at reasoning about **uncertainty** in model parameters

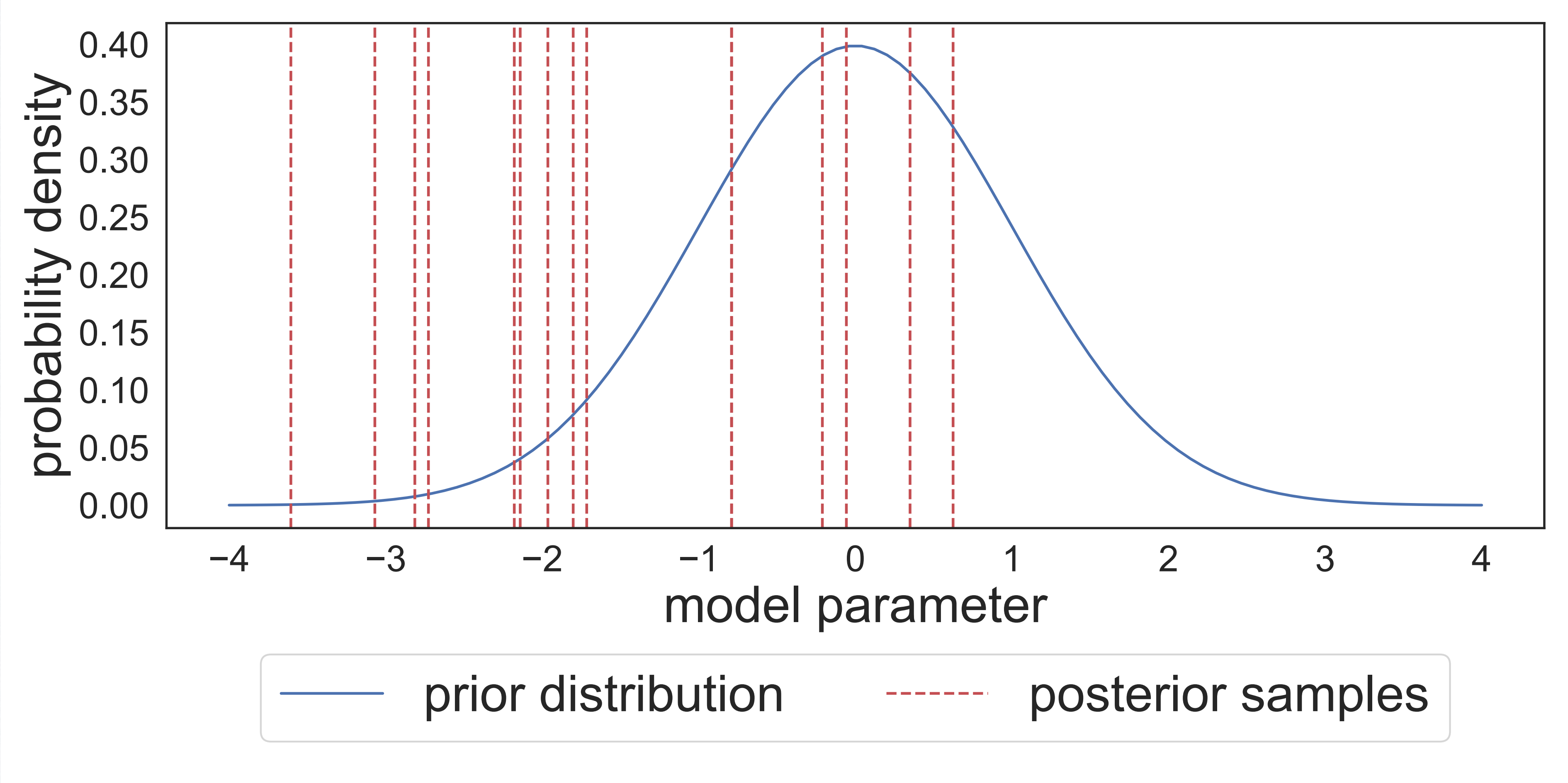
- **Downside:** No single algorithm works well for all models

- **This paper:** **Customize** Bayesian inference while maintaining **soundness**

GUIDE-BASED INFERENCE

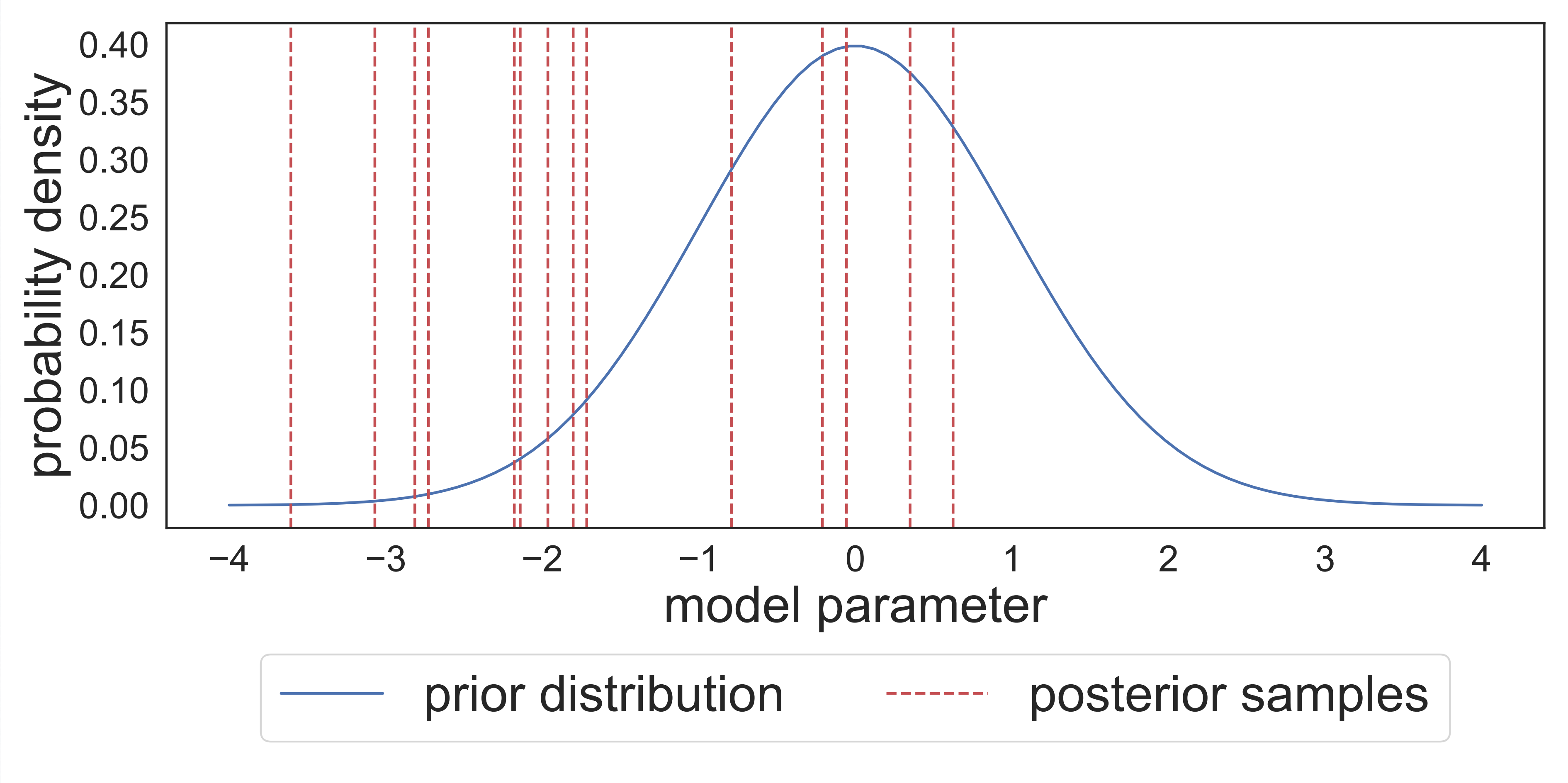
GUIDE-BASED INFERENCE

Monte-Carlo Methods



GUIDE-BASED INFERENCE

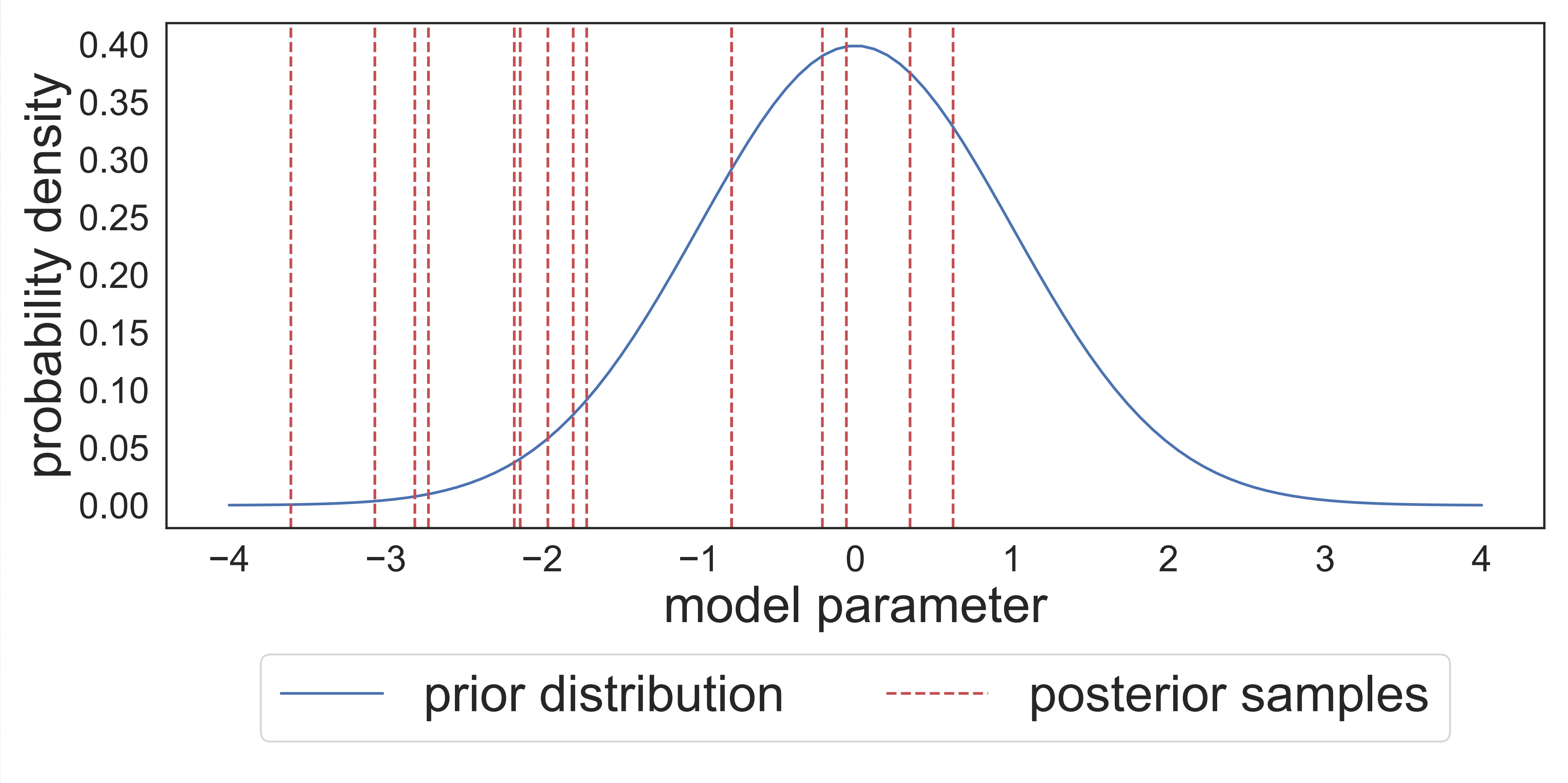
Monte-Carlo Methods



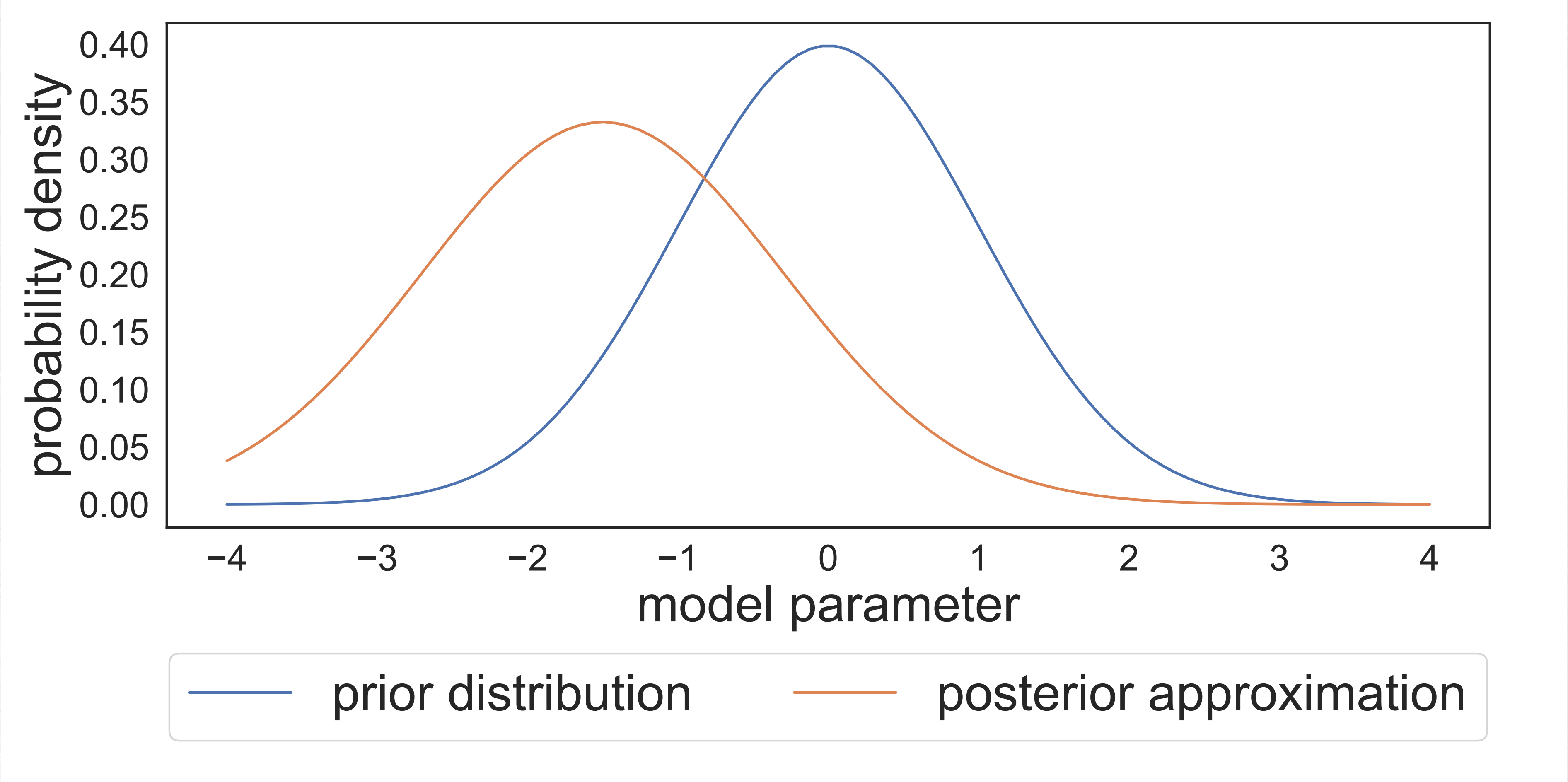
How to **guide** Monte-Carlo methods to explore the sample space for parameters?

GUIDE-BASED INFERENCE

Monte-Carlo Methods



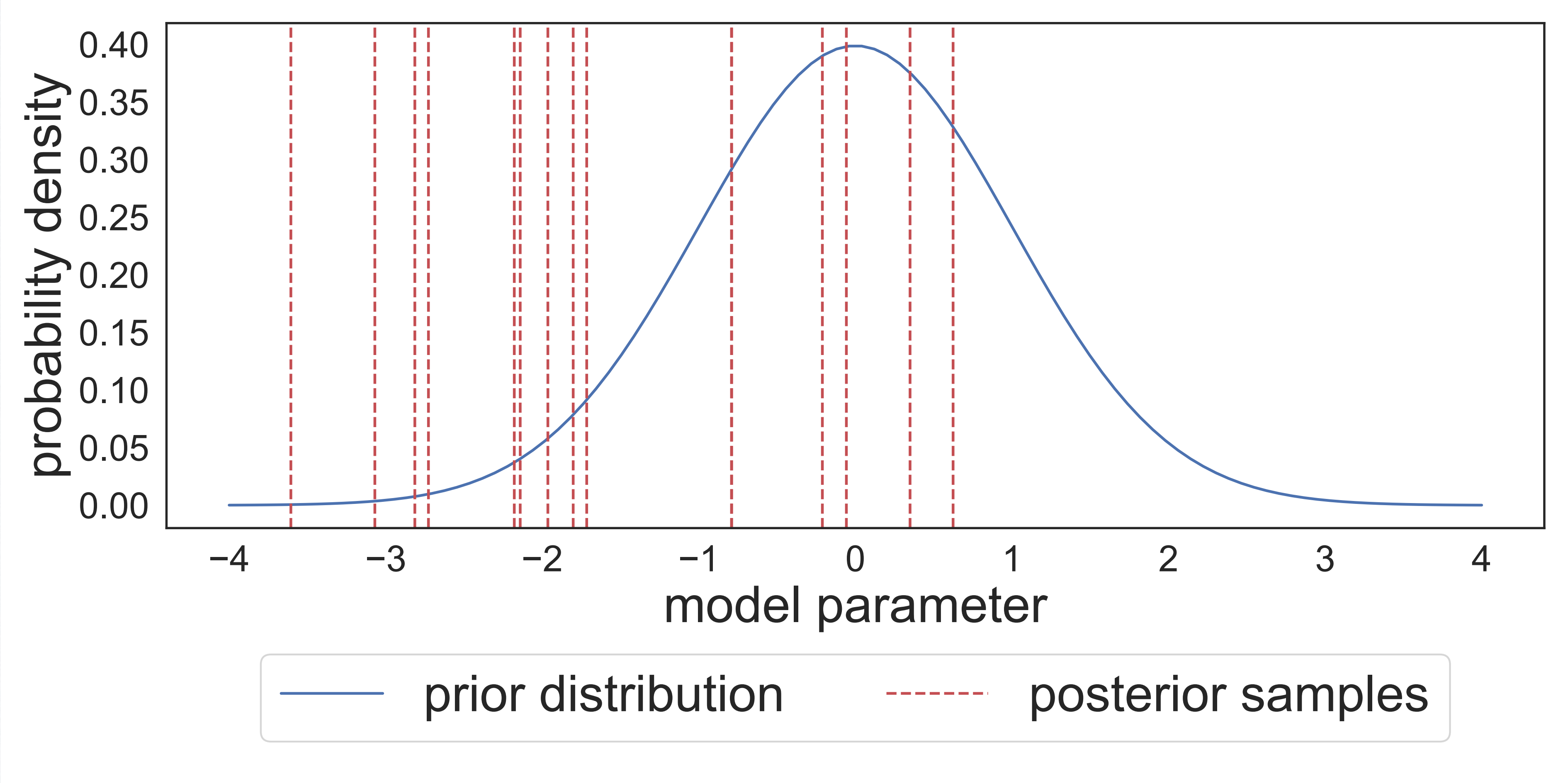
Variational Methods



How to **guide** Monte-Carlo methods to explore the sample space for parameters?

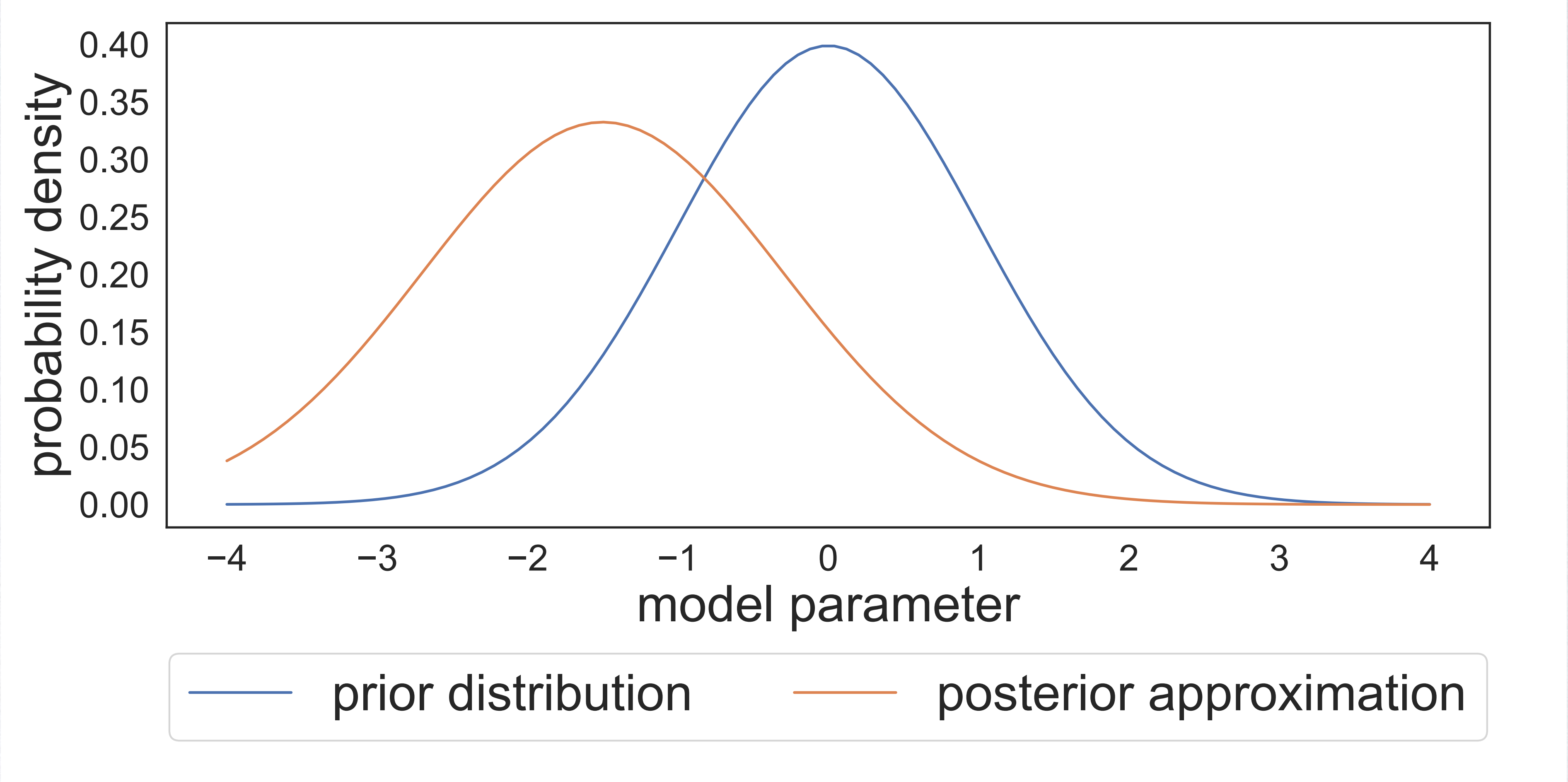
GUIDE-BASED INFERENCE

Monte-Carlo Methods



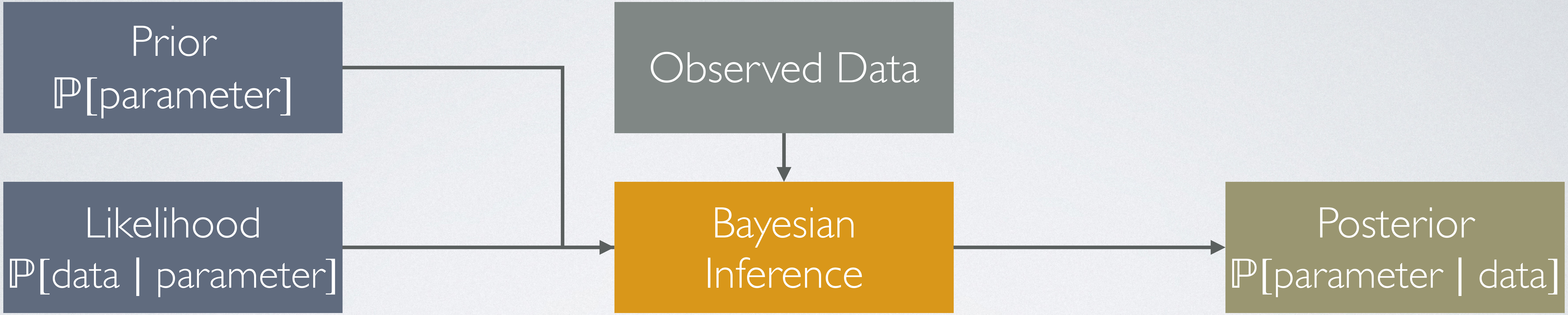
How to **guide** Monte-Carlo methods to explore the sample space for parameters?

Variational Methods



How to specify the **guide** distributions used by variational methods for approximation?

Guide-Based Inference



Prior
ℙ[parameter]

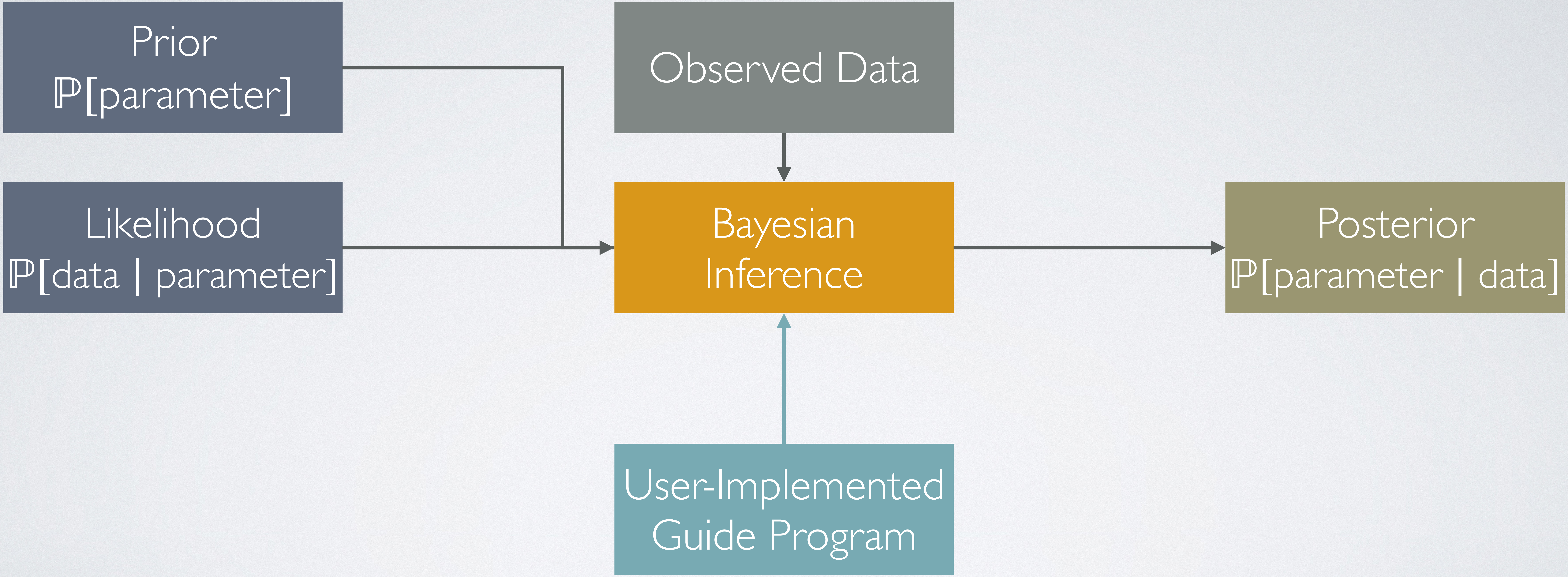
Likelihood
ℙ[data | parameter]

Observed Data

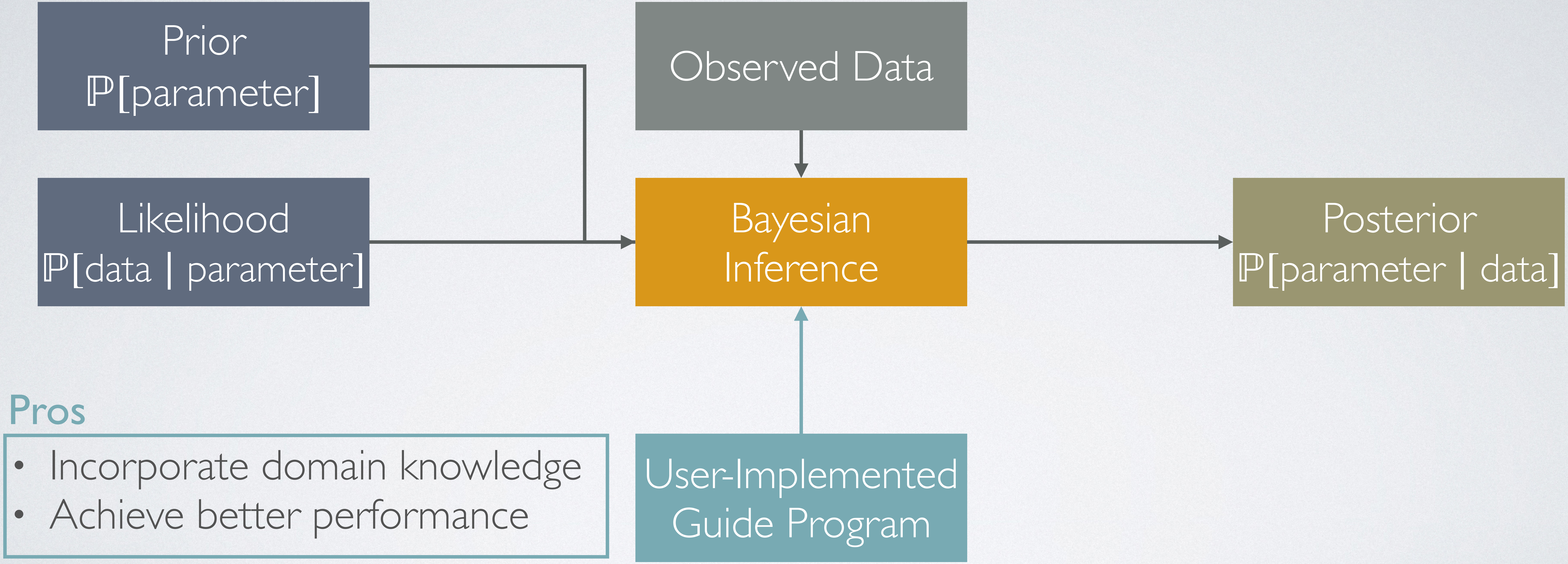
Bayesian Inference

Posterior
ℙ[parameter | data]

GUIDE-BASED INFERENCE



GUIDE-BASED INFERENCE



Prior
$\mathbb{P}$[parameter]

Likelihood
$\mathbb{P}$[data | parameter]

Observed Data

Bayesian Inference

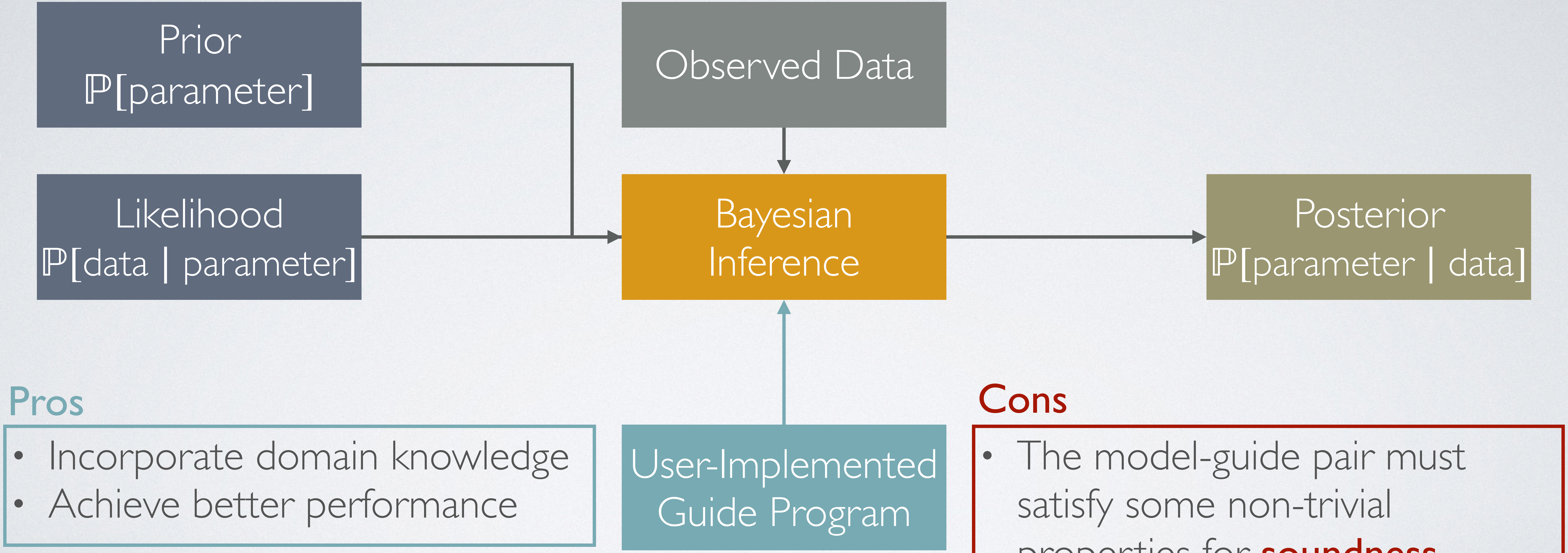
Posterior
$\mathbb{P}$[parameter | data]

User-Implemented Guide Program

Pros

- Incorporate domain knowledge
- Achieve better performance

GUIDE-BASED INFERENCE



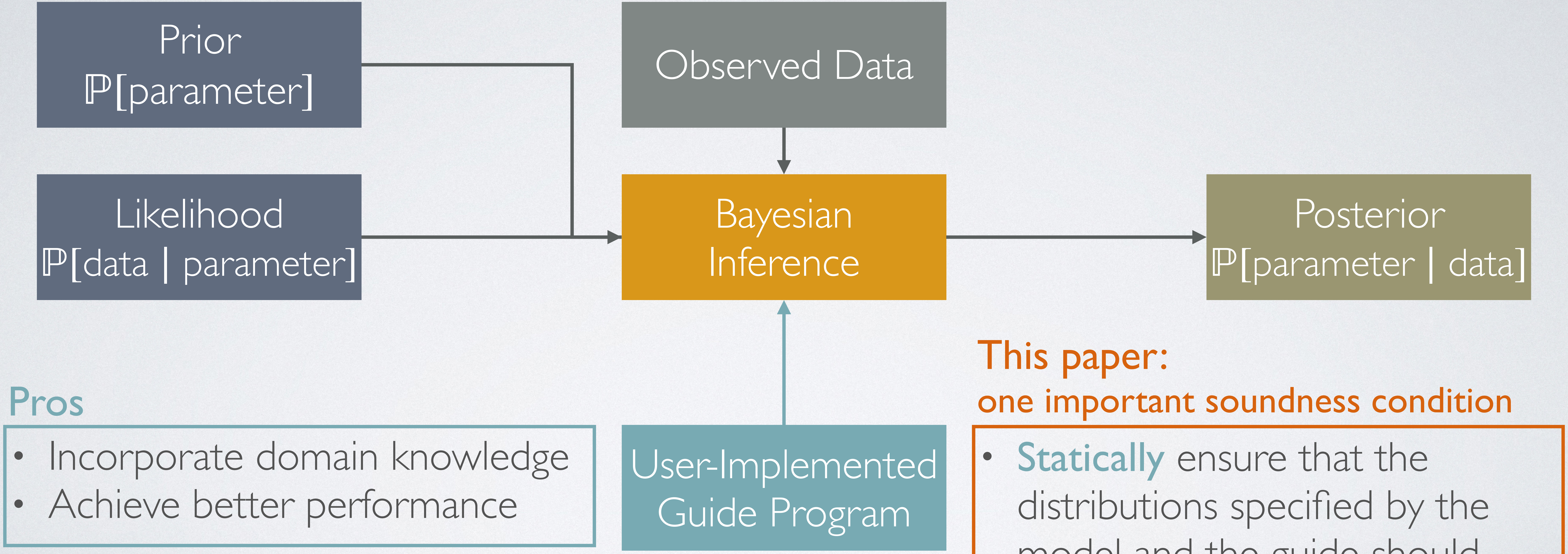
Pros

- Incorporate domain knowledge
- Achieve better performance

Cons

- The model-guide pair must satisfy some non-trivial properties for **soundness**

GUIDE-BASED INFERENCE



Prior
$\mathbb{P}$[parameter]

Likelihood
$\mathbb{P}$[data | parameter]

Observed Data

Bayesian Inference

Posterior
$\mathbb{P}$[parameter | data]

User-Implemented Guide Program

Pros

- Incorporate domain knowledge
- Achieve better performance

This paper:

one important soundness condition

- **Statically** ensure that the distributions specified by the model and the guide should **have the same support**

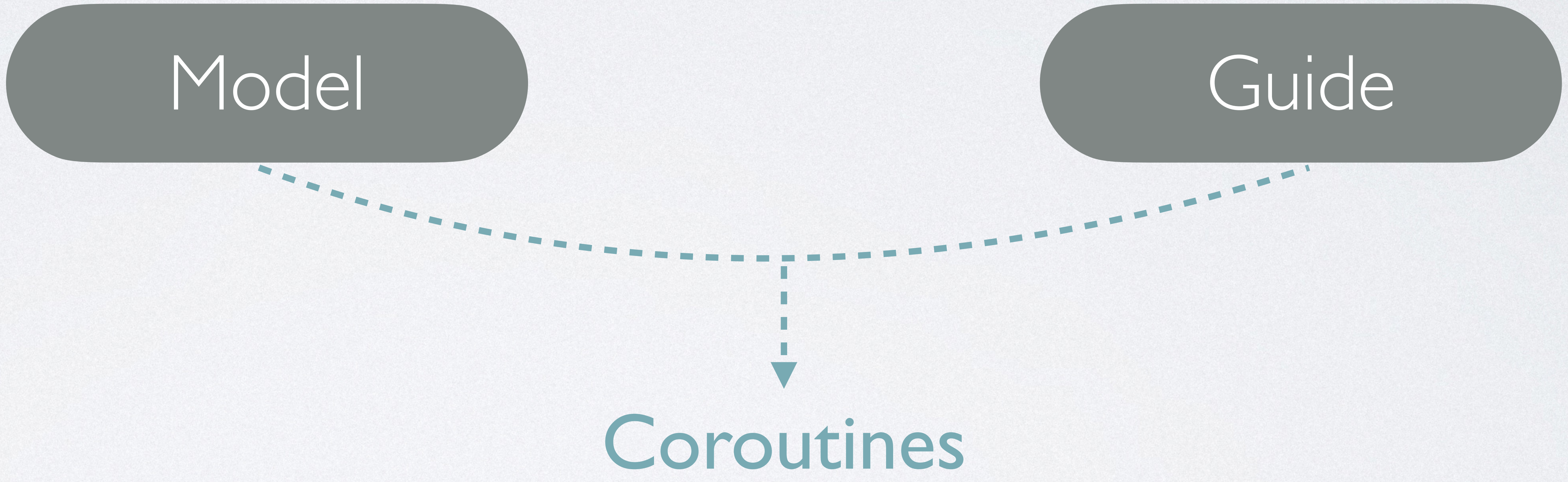
Our Approach

Our Approach

Model

Guide

Our Approach

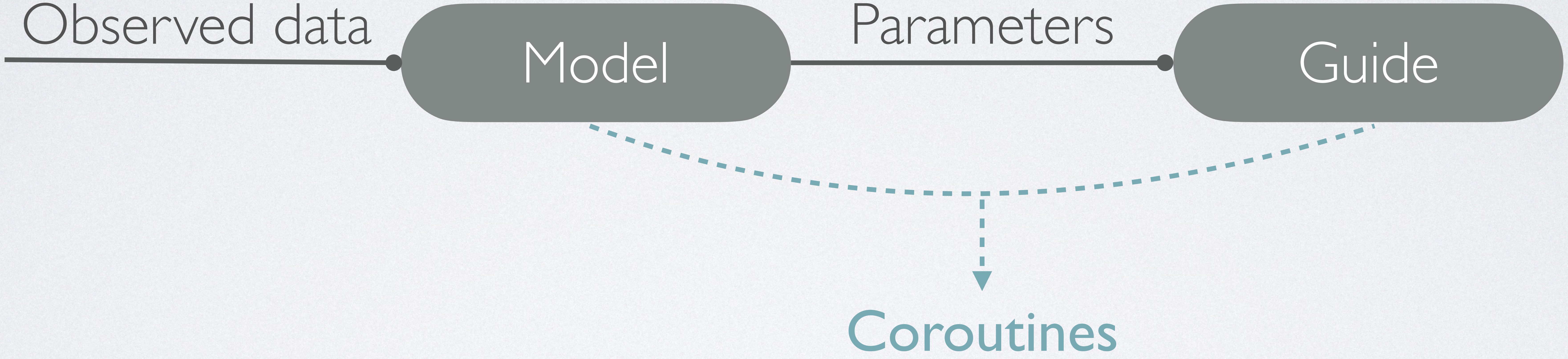


Model

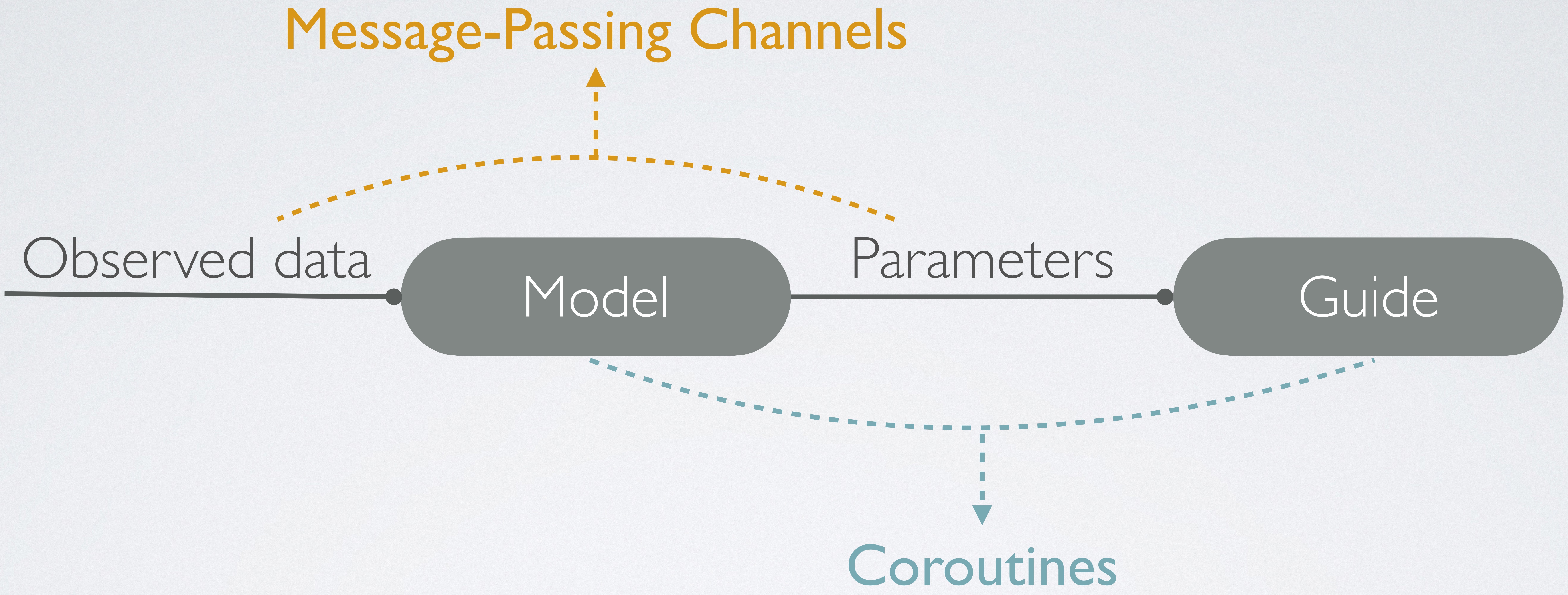
Guide

Coroutines

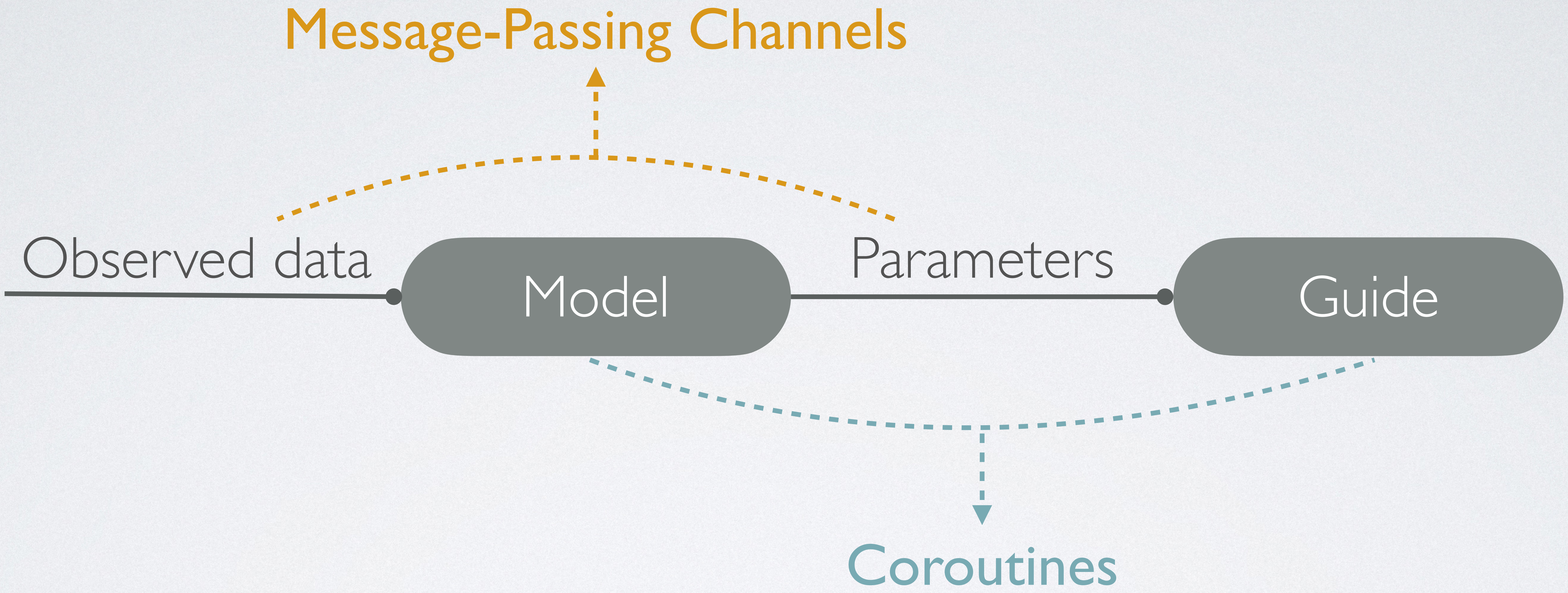
Our Approach



Our Approach



Our Approach



Key idea: Use communication to exchange random samples and control-flow branches, and impose type-based constraints on communication to guarantee inference soundness

Sample-site compatibility:
Corresponding sample sites should have the same support

Sample-site compatibility:

Corresponding sample sites should have the same support

Model

```
param1 <- sample_recv{param}(Gamma(2, 1))
```

Guide

```
param1 <- sample_send{param}(Gamma(1, 1))
```

Sample-site compatibility: Corresponding sample sites should have the same support

Model

```
param1 <- sample_recv{param}(Gamma(2, 1))
```

Guide

```
param1 <- sample_send{param}(Gamma(1, 1))
```

Sample-site compatibility:

Corresponding sample sites should have the same support

Model

```
param1 <- sample_recv{param}(Gamma(2, 1))
```

Guide

```
param1 <- sample_send{param}(Gamma(1, 1))
```

Control-flow compatibility:

Corresponding program paths should sample the same set of model parameters

Sample-site compatibility:

Corresponding sample sites should have the same support

Model

```
param1 <- sample_recv{param}(Gamma(2, 1))
```

Guide

```
param1 <- sample_send{param}(Gamma(1, 1))
```

Control-flow compatibility:

Corresponding program paths should sample the same set of model parameters

Model

```
if_send{param} param1 < 2 then … else …
```

Guide

```
if_recv{param} then … else …
```

Sample-site compatibility:

Corresponding sample sites should have the same support

Model

```
param1 <- sample_recv{param}(Gamma(2, 1))
```

Guide

```
param1 <- sample_send{param}(Gamma(1, 1))
```

Control-flow compatibility:

Corresponding program paths should sample the same set of model parameters

Model

```
if_send{param} param1 < 2 then … else …
```

Guide

```
if_recv{param} then … else …
```

GUIDE TYPES

Key idea: We take inspiration from session types and develop a type system to prescribe communication protocols

GUIDE TYPES

Key idea: We take inspiration from session types and develop a type system to prescribe communication protocols

```
proc sound_guide() {
  param1 <- sample_send{param}(Gamma(1, 1));
  if_recv{param} {
    return
  } else {
    param2 <- sample_send{param}(Uniform(0, 1));
    return
  }
}
```

Guide Types

Key idea: We take inspiration from session types and develop a type system to prescribe communication protocols

```
proc sound_guide() {
  param1 <- sample_send{param}(Gamma(1, 1));
  if_recv{param} {
    return
  } else {
    param2 <- sample_send{param}(Uniform(0, 1));
    return
  }
}
```

The guide type for channel `param`:

$$\mathbb{R}_+ \wedge (\mathbf{1} \,\&\, (\mathbb{R}_{(0,1)} \wedge \mathbf{1}))$$

Guide Types

Key idea: We take inspiration from session types and develop a type system to prescribe communication protocols

```
proc sound_guide() {
  param1 <- sample_send{param}(Gamma(1, 1));
  if_recv{param} {
    return
  } else {
    param2 <- sample_send{param}(Uniform(0, 1));
    return
  }
}
```

A nonnegative sample

The guide type for channel **param**:

$$\boxed{\mathbb{R}_+} \wedge (\mathbf{1} \mathbin{\&} (\mathbb{R}_{(0,1)} \wedge \mathbf{1}))$$

GUIDE TYPES

Key idea: We take inspiration from session types and develop a type system to prescribe communication protocols

```
proc sound_guide() {
  param1 <- sample_send{param}(Gamma(1, 1));
  if_recv{param} {
    return
  } else {
    param2 <- sample_send{param}(Uniform(0, 1));
    return
  }
}
```

The guide type for channel `param`:

$$\mathbb{R}_+ \wedge (\mathbf{1} \boxed{\&} (\mathbb{R}_{(0,1)} \wedge \mathbf{1}))$$

Receive a branch selection

GUIDE TYPES

Key idea: We take inspiration from session types and develop a type system to prescribe communication protocols

```
proc sound_guide() {
  param1 <- sample_send{param}(Gamma(1, 1));
  if_recv{param} {
    return
  } else {
    param2 <- sample_send{param}(Uniform(0, 1));
    return
  }
}
```

The guide type for channel **param**:

$$\mathbb{R}_+ \land (\mathbf{1} \,\&\, (\boxed{\mathbb{R}_{(0,1)}} \land \mathbf{1}))$$

A [0,1]-valued sample

MORE IN THE PAPER

- How our system supports recursion, control-flow divergence, and type reconstruction
- Full formalism of the coroutine-based semantics and the system of guide types
- Proof of type safety and inference soundness
- A prototype implementation and experiments on expressibility and performance
- Comparison with prior work by Lew et al. [1] and Lee et al. [2]

[1] A. K. Lew, M. F. Cusumano-Towner, B. Sherman, M. Carbin, and V. K. Mansinghka. *Trace Types and Denotational Semantics for Sound Programmable Inference in Probabilistic Languages*. POPL'20.
[2] W. Lee, H. Yu, X. Rival, and H. Yang. *Towards Verified Stochastic Variational Inference for Probabilistic Programs*. POPL'20.