Es macht so glücklich, Computer zu sein:
alle Schererein
verwandeln sich in Rechnerei
und gehn in Millionstel Sekunden vorbei.
In wenigen “bit”
kriegst du die ganze Weltordnung mit
im Grund
heißt die Frage ja immer “Sein oder Nichtsein”,
die erledigst du sogar ohne Und,
den ganzen Moder
mit einem einzigen Oder.

Andreas Okopenko

Contents

4 Graph Tracking in Probabilistic Models

The system described in chapter 3, implemented in a Julia package `IRTracker.jl`, can now be utilized for the analysis of probabilistic models written in `DynamicPPL.jl`, and for posterior inference in `Turing.jl`. This part of the work is realized in another package, `AutoGibbs.jl`, which is available as open-source code[1]. There are two applications provided, built on top of the graph tracking functionality: first, dependency analysis of random variables in a model can be performed. This results in the complete graphical model for static models, and a slice of it for dynamic models. The resulting graph can be plotted for visualization. Second, given the dependency graph, the conditional likelihoods of unobserved variables in static models can be extracted. With these, analytic Gibbs conditionals can be derived and used in `Turing.jl`'s within-Gibbs sampler.

4.1 Dependency Analysis in Dynamic Models

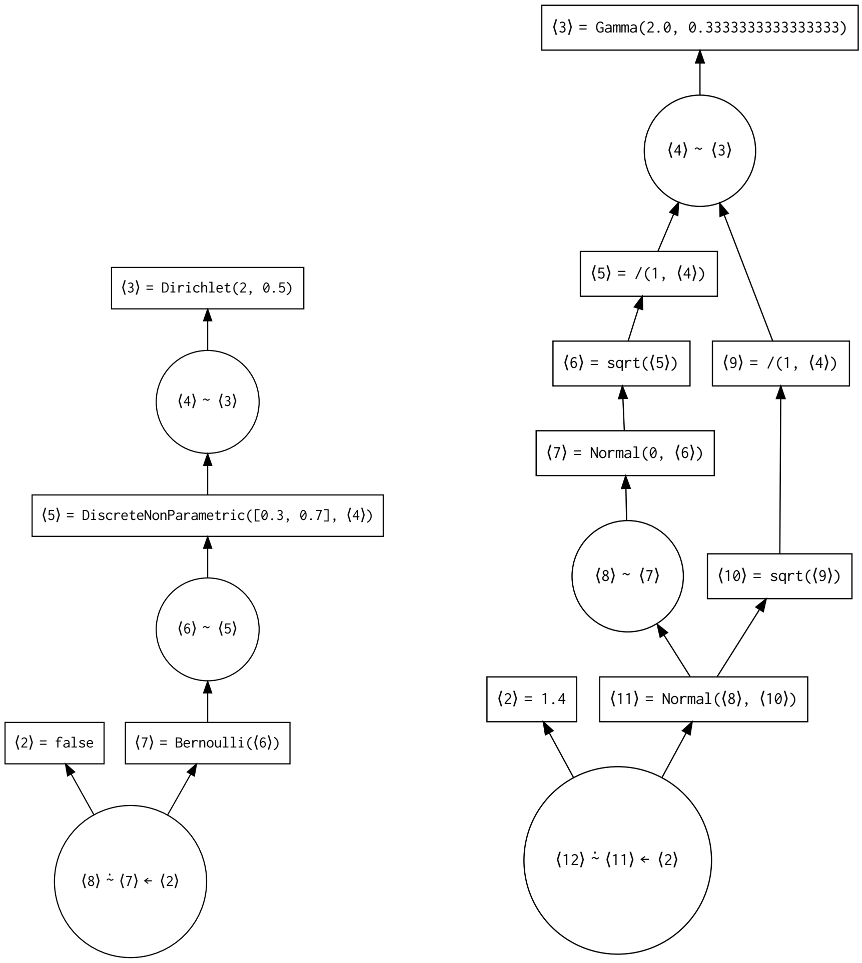
Context for DPPL-models; slicing; graph extraction + new representation; plotting

[1] `https://github.com/phipsgabler/AutoGibbs.jl`

```
@model function bernoulli_mixture(x)
    w ~ Dirichlet(2, 1/2)
    p ~ DiscreteNonParametric([0.3, 0.7], w)
    x ~ Bernoulli(p)
end

@model function hierarchical_gaussian(x)
    λ ~ Gamma(2.0, inv(3.0))
    m ~ Normal(0, sqrt(1 / λ))
    x ~ Normal(m, sqrt(1 / λ))
end
```

Listing 4.1: Simple mixture of two Bernoulli random variables with fixed weights.



(a) bernoulli_mixture (b) hierarchical_gaussian

Figure 4.1: Dependency graphs of the models in ??

4.2 JAGS-Style Automatic Calculation of Gibbs Conditionals

Gibbs sampler implementation for Turing; likelihood closures; conditional likelihood extraction

4.3 Evaluation

$$\begin{aligned} \mu_k &\sim \text{Normal}(0, \sigma_1), \quad k = 1, \dots, K \\ w &\sim \text{Dirichlet(K)} \\ z_n &\sim \text{Discrete}([1, \dots, K], w), \quad n = 1, \dots, N \\ x_n &\sim \text{Normal}(\mu_{z_n}, \sigma_1), \quad n = 1, \dots, N \end{aligned} \tag{4.1}$$

$$\begin{aligned} T_k &\sim \text{Dirichlet}(K), \quad k = 1, \dots, K \\ m_k &\sim \text{Normal}(k, \sigma_1), \quad k = 1, \dots, K \\ s_1 &\sim \text{Categorical}(K) \\ s_k &\sim \text{Categorical}(T_{s_{k-1}}), \quad k = 2, \dots, N \\ x_k &\sim \text{Normal}(m_{s_k}, \sigma_2), \quad k = 1, \dots, N \end{aligned} \tag{4.2}$$

$$\begin{aligned} w &\sim \text{TruncatedStickBreakingProcess}(\alpha, \text{K}) \\ z_n &\sim \text{Categorical}(w), \quad n = 1, \dots, N \\ \mu_k &\sim \text{Normal}(0, \sigma_1), \quad k = 1, \dots, K \\ y_n &\sim \text{Normal}(\mu_{z_n}, \sigma_2), \quad n = 1, \dots, N \end{aligned} \tag{4.3}$$

Algorithms		GMM			HMM			IMM
AG + HMC	Data size	10	25	50	10	25	50	10
	Chains	30	30	30	30	30	30	30
	Compilations	3	3	3	3	3	3	3
PG + HMC, 10 particles	Data size	10	25	50	10	25	50	10
	Chains	10	10	10	10	10	10	10
PG + HMC, 25 particles	Data size	10	25	50	10	25	50	10
	Chains	10	10	10	10	10	10	10
PG + HMC, 50 particles	Data size	10	25	50	10	25	50	10
	Chains	10	10	10	10	10	*x	10

Table 4.1: Experimental combinations that were run. Chains were always of length 5000. The parameters for HMC were a stepsize of 0.1, and 10 leapfrog steps. A new static Gibbs conditional was extracted for each block of 10 chains that was run with the same parameters while Particle Gibbs was varied over the three particle sizes. Particle Gibbs with 50 particles was sometimes killed due to timeouts on the server.

List of Algorithms