R2: An Efficient MCMC Sampler for Probabilistic Programs

Aditya V. Nori, Chung-Kil Hur, Sriram K. Rajamani, Selva Samuel

October 23rd, 2014

http://research.microsoft.com/apps/pubs/?id=211941

Outline

- Inference in Probabilistic Programs.
- R2: source code transformation combined with MH sampling.
- R2 Semantics and correctness.
- Empricial evaluation.

Probabilistic Programs

- Usual programs with:
  1. the ability to draw values at random from distributions;
  2. the ability to condition values of variables via observations.
- Inference — computing an explicit representation of the distribution implicitly specified by the program.
- Inference is conveniently performed by sampling.
- Sampling is repeated execution of the program.

R2: The Novel approach

1. Propagation of observations back through the program using the pre-image operation $\textsc{Pre}$ (Dijkstra 1976) and placing an observe statement immediately next to every probabilistic assignment.
2. Modified Metropolis-Hastings (MH) sampling over the transformed probabilistic program.

From Dijkstra: A Discipline of Programming

Page 16:

If the system (machine, mechanism) is denoted by "S" and the desired post-condition by "R", then we denote the corresponding weakest pre-condition by

$$\mathrm{wp}(S, R)$$

Page 17:

If the initial state satisfies wp(S, R), the mechanism is certain to establish eventually the truth of R. Because wp(S, R) is the weakest pre-condition, we also know that if the initial state does not satisfy wp(S, R), this guarantee cannot be given, i.e. the happening may end in a final state not satisfying R or the happening may even fail to reach a final state at all (as we shall see, either because the system finds itself engaged in an endless task or because the system has got stuck).

Related Work

- BLOG (Milch and Russell 2006) uses program structure to come up with good proposals for MCMC.
- Wingate et al (Nonstandard interpretation ..., 2011) use non-standard interpretations of probabilistic programs to improve MCMC efficiency.
- Moldovan et al. (Moldovan et al. 2013) introduce an MCMC algorithm for estimating conditional probabilities from an AND/OR tree.
- Pfeffer (Pfeffer 2007b) presents several structural heuristics to make choices that are less likely to get rejected.
- Chaganty et al. (Chaganty, Nori, and Rajamani 2013 — self-citation) use **pre-image** transformations on observations to perform efficient importance sampling for straight-line programs.

This work — global transformation semantics and MH sampling.

Transformation Example — Original Code

```
1:  bool earthquake, burglary, alarm, phoneWorking,
    maryWakes,called;
2:  earthquake = Bernoulli(0.001);
3:  burglary = Bernoulli(0.01);
4:  alarm = earthquake || burglary;
5:  if (earthquake)
6:    phoneWorking = Bernoulli(0.6);
7:  else
8:    phoneWorking = Bernoulli(0.99);
9:  if (alarm && earthquake)
10:   maryWakes = Bernoulli(0.8);
11: else if (alarm)
12:   maryWakes = Bernoulli(0.6);
13: else
14:   maryWakes = Bernoulli(0.2);
15: called = maryWakes && phoneWorking;
16: observe(called);
17: return burglary;
```

Transformation Example — Transformed Code

```
1:  bool earthquake, burglary, alarm, phoneWorking,
    maryWakes,called;
2:  earthquake = Bernoulli(0.001);
3:  burglary = Bernoulli(0.01);
4:  alarm = earthquake || burglary;
5:  if (earthquake) {
6:    phoneWorking = Bernoulli(0.6);
7:    observe(phoneWorking);
8:  }
9:  else {
10:   phoneWorking = Bernoulli(0.99);
11:   observe(phoneWorking);
12: }
13: if (alarm && earthquake){
14:   maryWakes = Bernoulli(0.8);
15:   observe(maryWakes && phoneWorking);
16: }
17: else if (alarm){
19:   maryWakes = Bernoulli(0.6);
20:   observe(maryWakes && phoneWorking);
21: }
22: else {
23:   maryWakes = Bernoulli(0.2);
24:   observe(maryWakes && phoneWorking);
25: }
26: called = maryWakes && phoneWorking;
27: return burglary;
```

Transformation Example — R2 Steps

- Performs $\textsc{Pre}$ analysis to perform the transformed program — *backward* analysis.
- Places next to each probabilistic statement the propagated pre-image as an observe statement.
- Samples using modified MH with truncated distributions in each probabilistic statement (both the proposal and the target distributions).

PROB — the probabilistic programming language

x	$\in$	Vars	
uop	::=	$\cdots$	C unary operators
bop	::=	$\cdots$	C binary operators
φ, ψ	::=	$\cdots$	logical formula
$\mathcal{E}$	::=		expressions
		$\mid x$	variable
		$\mid c$	constant
		$\mid \mathcal{E}_1$ **bop** $\mathcal{E}_2$	binary operation
		$\mid$ **uop** $\mathcal{E}_1$	unary operation

$\mathcal{S}$	::=		statements
		$\mid$ `skip`	skip
		$\mid x = \mathcal{E}$	deterministic assignment
		$\mid x \sim \texttt{Dist}(\bar{\theta})$	probabilistic assignment
		$\mid$ `observe` (φ)	observe
		$\mid \mathcal{S}_1; \mathcal{S}_2$	sequential composition
		$\mid$ `if` $\mathcal{E}$ `then` $\mathcal{S}_1$ `else` $\mathcal{S}_2$	conditional composition
		$\mid$ `while` $\mathcal{E}$ `do` $\mathcal{S}_1$	loop
$\mathcal{P}$	::=	$\mathcal{S}$ `return` $\mathcal{E}$	program

- Language semantics defined rigorously.
- PRE preserves program semantics.

The R2 Algorithm — Pre-Image Analysis

$$
\begin{aligned}
&\textsc{Pre}(x = \mathcal{E}, \varphi) && = (x = \mathcal{E}, \varphi[\mathcal{E}/x]) \\
&\textsc{Pre}(\texttt{skip}, \varphi) && = (\texttt{skip}, \varphi) \\
&\textsc{Pre}(\mathcal{S}_1; \mathcal{S}_2, \varphi) && = \textbf{let } (\mathcal{S}_2', \varphi') = \textsc{Pre}(\mathcal{S}_2, \varphi) \textbf{ and} \\
& && \quad \textbf{let } (\mathcal{S}_1', \varphi'') = \textsc{Pre}(\mathcal{S}_1, \varphi') \textbf{ in} \\
& && \quad (\mathcal{S}_1'; \mathcal{S}_2', \varphi'') \\
&\textsc{Pre}(\texttt{if } \mathcal{E} \texttt{ then } \mathcal{S}_1 && = \textbf{let } (\mathcal{S}_1', \varphi_1) = \textsc{Pre}(\mathcal{S}_1, \varphi) \textbf{ and} \\
&\qquad \texttt{else } \mathcal{S}_2, \varphi) && \quad \textbf{let } (\mathcal{S}_2', \varphi_2) = \textsc{Pre}(\mathcal{S}_2, \varphi) \textbf{ in} \\
& && \quad (\texttt{if } \mathcal{E} \texttt{ then } \mathcal{S}_1' \texttt{ else } \mathcal{S}_2', \\
& && \quad (\mathcal{E} \wedge \varphi_1) \vee (\neg\mathcal{E} \wedge \varphi_2)) \\
&\textsc{Pre}(\texttt{observe } (\psi), \varphi) && = (\texttt{observe}(\psi), \varphi \wedge \psi) \\
&\textsc{Pre}(x \sim \texttt{Dist}(\bar{\theta}), \varphi) && = (x \sim \texttt{Dist}(\bar{\theta}); \texttt{observe}(\varphi), \exists x.\varphi) \\
&\textsc{Pre}(\texttt{while } \{\psi\}\ \mathcal{E} \texttt{ do } \mathcal{S}, \varphi) && = \textbf{let } (\mathcal{S}', \psi') = \textsc{Pre}(\mathcal{S}, \psi) \textbf{ in} \\
& && \quad \textbf{assert}(\mathcal{E} \wedge \psi' \implies \psi); \\
& && \quad \textbf{assert}(\neg\mathcal{E} \wedge \varphi \implies \psi); \\
& && \quad (\texttt{while } \{\psi\}\ \mathcal{E} \texttt{ do } \mathcal{S}', \psi)
\end{aligned}
$$

The R2 Algorithm — MH Sampling

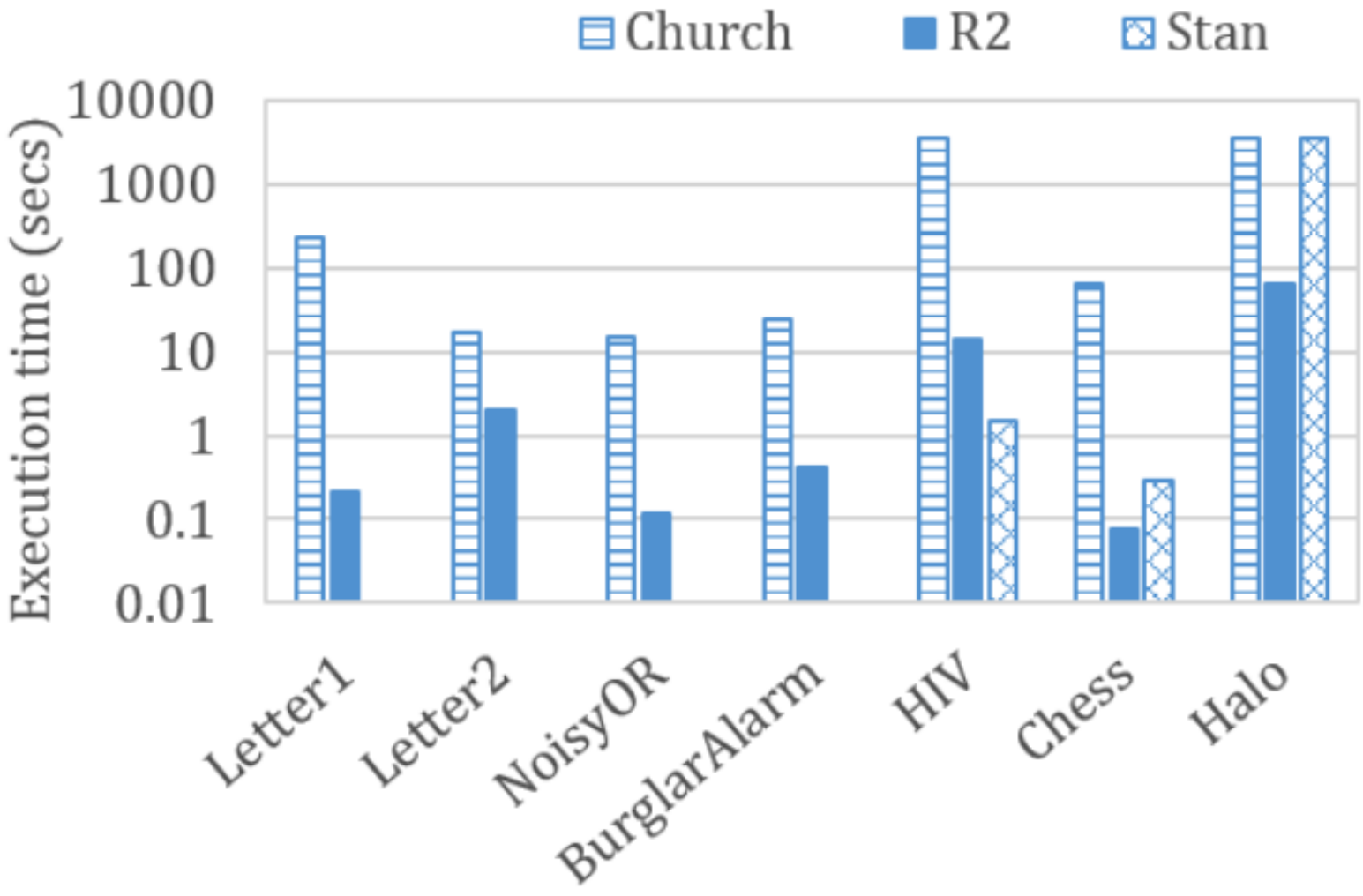
For any P: $P_{|\varphi}$ — truncated distribution:

$$\text{if } \varphi(x) \text{ is false, then } P_{|\varphi} = 0, \text{ else } \frac{P}{Z}$$

1. For every sample statement and observation "$x\ P; \mathit{observe}(\varphi)$", the proposal distribution $Q_{|\varphi}(x_{old} \to x_{new})$ is truncated according to the condition φ.
2. The acceptance probability β is defined in terms of $P_{|\varphi}$ and $Q_{|\varphi}$:

$$\beta = \min\left(1, \frac{P_{|\varphi}(x_{new})\,Q_{|\varphi}(x_{new} \to x_{old})}{P_{|\varphi}(x_{old})\,Q_{|\varphi}(x_{old} \to x_{new})}\right)$$

Empirical Evaluation



Empirical Evaluation — Examples

- `Letter1, Letter2:` probabilistic model for a student getting a good reference letter (Koller and Friedman 2009).
- `NoisyOR:` given a directed acyclic graph, each node is a noisy-or of its parents. Find the posterior probability of a node, given observations (Kiselyov and Shan 2009).
- `BurglarAlarm:` estimate the probability of a burglary having observed a phone call.
- `HIV:` a multi-level linear model with interaction and varying slope and intercept (Hoffman and Gelman 2013).
- `Chess:` a skill rating system for a chess tournament consisting of 77 players and 2926 games (Herbrich, Minka, and Graepel 2006).
- `Halo:` a skill rating system for a tournament consisting of 31 teams, with at most 4 players per team, and 465 games played between teams (Herbrich, Minka, and Graepel 2006).

Conclusion

- Unique algorithm that uses source code analysis to avoid rejection due to conditioning.
- Formal semantic of probabilistic programs.
- Experimental results — R2 is much faster than Church or Stan.

Thank You