Coupling Proofs
Are Probabilistic Product Programs

Gilles Barthe, Benjmain Grégoire, Justin Hsu*, Pierre-Yves Strub

IMDEA Software, Inria, University of Pennsylvania*, École Polytechnique

January 18, 2017
A simple card-flipping process

Setup

- **Input**: position in \{1, \ldots, 9\}
- **Repeat**: Draw uniformly random card \( \in \{1, \ldots, 9\} \)
  - Go forward that many steps
- **Output** last position before crossing 100
In pictures

![Diagram showing a sequence of numbers with a blue arrow pointing to the number 4.]

- 3
- 1
- 5
- ...
- 4
Output last position: 99
Starting at a different position
Starting at a different position

How close are the two output distributions?
Starting at a different position

How close are the two output distributions?
Starting at a different position

How close are the two output distributions?
Starting at a different position

How close are the two output distributions?
Combine first process and second process
Combine first process and **second process**
Combine first process and second process
Combine first process and second process
Combine first process and second process

Product program: One program simulating two programs

/five.osf
Combine first process and second process

Product program: One program simulating two programs
Combine first process and second process

Product program: One program simulating two programs

/one.osf
Combine first process and second process

Product program: One program simulating two programs

/three.osf
Combine first process and second process

Product program: One program simulating two programs

/\five.osf
Combine first process and second process
Combine first process and second process

Product program: One program simulating two programs
Combine first process and second process

Product program: One program simulating two programs

/five.osf
Combine first process and second process

Product program: One program simulating two programs

/5.osf
Combine first process and second process

Product program: One program simulating two programs
Why is this interesting?
In general

Property $P$ of product program

$\Downarrow$

Property $P'$ of two programs
Our construction

Two simulated programs can share randomness
Distance between output distributions
Distance between output distributions
Distance between output distributions \(\leq\) Probability that outputs differ
Distance between output distributions $\leq$ Probability that outputs differ
Our technical contributions

A probabilistic product construction with shared randomness

A probabilistic program logic \( \times \text{pRHL} \): a proof-relevant version of pRHL
A crash course: Probabilistic Relational Hoare Logic [BGZ-B]
Imperative language

\[ c ::= x \leftarrow e \mid c ; c \mid \text{if } e \text{ then } c \text{ else } c \mid \text{while } e \text{ do } c \]
Imperative language

\[ c ::= x \leftarrow e \mid c ; c \mid \text{if } e \text{ then } c \text{ else } c \mid \text{while } e \text{ do } c \mid x \leftarrow S \ [S] \]

Uniform sampling from finite set \([S]\)

- coin flip: \([\text{heads, tails}]\)
- random card: \([1, \ldots, 9]\)
Imperative language

\[ c ::= x \leftarrow e \mid c ; c \mid \text{if } e \text{ then } c \text{ else } c \mid \text{while } e \text{ do } c \mid x \leftarrow \mathcal{S} \]

Uniform sampling from finite set \([\mathcal{S}]\)

- coin flip: [ heads, tails ]
- random card: [ 1, ..., 9 ]

Command semantics \([c]\)

- **Input**: memory
- **Output**: distribution over memories
Judgments: similar to Hoare logic

\{P\} \ c \ \{Q\}

Assertions: binary relation on memories

Can refer to tagged program variables: \(x^{\langle 1 \rangle}\) and \(x^{\langle 2 \rangle}\)

First order formulas, non-probabilistic

If the two inputs satisfy \(P\), we can share the randomness on two runs of \(c\) so that the two outputs satisfy \(Q\).
Judgments: similar to Hoare logic

\{ P \} \ c \ \{ Q \}

Assertions: binary relation on memories

- Can refer to tagged program variables: $x(1)$ and $x(2)$
- First order formulas, non-probabilistic

If the two inputs satisfy $P$, we can share the randomness on two runs of $c$ so that the two outputs satisfy $Q$. 

/one.osf/three.osf
Judgments: similar to Hoare logic

\[
\{P\} \ c \ \{Q\}
\]

Assertions: binary relation on memories

- Can refer to tagged program variables: \(x^{(1)}\) and \(x^{(2)}\)
- First order formulas, non-probabilistic

If the two inputs satisfy \(P\), we can share the randomness on two runs of \(c\) so that the two outputs satisfy \(Q\).
Proof rules in pRHL: mostly similar to Hoare logic

\[
\begin{align*}
\text{ASSN} & \quad \frac{Q \{e(1), e(2)/x(1), x(2)\}}{x \leftarrow e \{Q\}} \\
\text{SEQ} & \quad \frac{\{P\} c \{Q\} \quad \{Q\} c' \{R\}}{\{P\} c ; c' \{R\}} \\
\text{COND} & \quad \frac{P \quad P \land e(1)}{\{P\} \text{ if } e \text{ then } c \text{ else } c' \{Q\}} \\
\text{LOOP} & \quad \frac{\{P \land e(1) \land e(2)\} c \{P \land e(1) = e(2)\}}{\{P \land e(1) = e(2)\} \text{ while } e \text{ do } c \{P \land \neg e(1) \land \neg e(2)\}} \\
\text{CONSEQ} & \quad \frac{\{P\} c \{Q\}}{\Rightarrow P' \Rightarrow P \land Q \Rightarrow Q'} \\
\text{CASE} & \quad \frac{\{P \land R\} c \{Q\} \quad \{P \land \neg R\} c \{Q\}}{\{P\} c \{Q\}}
\end{align*}
\]
Proof rules in pRHL: mostly similar to Hoare logic
Proof rules in pRHL: Random sampling

\[ f : S \rightarrow S \text{ bijection} \]

\[
\{ \top \} \quad x \leftarrow [S] \quad \{ x^{\langle 2 \rangle} = f(x^{\langle 1 \rangle}) \}
\]
Proof rules in pRHL: Random sampling

\[ f : S \rightarrow S \text{ bijection} \]

\[ \{ \top \} \quad x \leftarrow [S] \quad \{ x\langle 2 \rangle = f(x\langle 1 \rangle) \} \]

Select how to share randomness
Introducing \( \times \) pRHL
Idea: Product program $c^\times$ simulates two processes

$$\{P\} \ c \ \{Q\}$$
Idea: Product program $c^\times$ simulates two processes

$$\{P\} \ c \ \{Q\} \sim \ c^\times$$
Idea: Product program $c^\times$ simulates two processes

$\{P\} \ c \ \{Q\} \rightsquigarrow \ c^\times$

Runs in combined memory

- Two separate copies of single memory
- Duplicate program variables: $x\langle 1 \rangle$ and $x\langle 2 \rangle$
Idea: Product program $c^\times$ simulates two processes

\[
\{ P \} \ c \ \{ Q \} \xrightarrow{~} c^\times
\]

Runs in combined memory

- Two separate copies of single memory
- Duplicate program variables: $x\langle 1 \rangle$ and $x\langle 2 \rangle$

Property of $c^\times \implies$ property of two runs of $c$
A tour of $\times pRHL$ rules: [Seq]

In $pRHL$:

$$\{P\} \ c \ \{Q\} \ \Rightarrow \ \{Q\} \ c' \ \{R\}$$

$$\{P\} \ c \ ; \ c' \ \{R\}$$
A tour of $\times pRHL$ rules: [Seq]

In $\times pRHL$:

\[
\begin{align*}
\{P\} \ c \ \{Q\} & \rightsquigarrow c^\times & \{Q\} \ c' \ \{R\} & \rightsquigarrow c'^\times \\
\{P\} \ c ; c' \ \{R\} & \rightsquigarrow c^\times ; c'^\times
\end{align*}
\]
A tour of \( \times \text{pRHL} \) rules: [Seq]

In \( \times \text{pRHL} \):

\[
\{ P \} \ x \ \{ Q \} \leadsto c^{\times} \quad \quad \quad \{ Q \} \ x' \ \{ R \} \leadsto c'^{\times}
\]

\[
\{ P \} \ x ; c' \ \{ R \} \leadsto c^{\times} ; c'^{\times}
\]

Sequence product programs
A tour of $\times pRHL$ proof rules: [Rand]

In $pRHL$:

\[
\begin{align*}
& f : S \rightarrow S \text{ bijection} \\
& \{\top\} \xleftarrow{\$} [S] \{x\langle 2 \rangle = f(x\langle 1 \rangle)\}
\end{align*}
\]
A tour of \( \times \)pRHL proof rules: [Rand]

In \( \times \)pRHL:

\[
\begin{align*}
\{ \top \} & \quad x \leftarrow^{\$} [S] \quad \{ x \langle 2 \rangle = f(x \langle 1 \rangle) \} \\
\implies & \quad x \langle 1 \rangle \leftarrow^{\$} [S] ; \quad x \langle 2 \rangle \leftarrow f(x \langle 1 \rangle)
\end{align*}
\]
A tour of \(\times\)pRHL proof rules: [Rand]

In \(\times\)pRHL:

\[
f : S \rightarrow S \text{ bijection}
\]

\[
\{ \top \} \quad x \leftarrow [S] \quad \{ x^{(2)} = f(x^{(1)}) \} \quad \rightsquigarrow \quad x^{(1)} \leftarrow [S] ; x^{(2)} \leftarrow f(x^{(1)})
\]

Sample \(x^{(2)}\) depends on \(x^{(1)}\)
A tour of \( \times \text{pRHL} \) rules: [Case]

In \text{pRHL}:

\[
\begin{align*}
\{P \land Q\} & \; c \; \{R\} & \{P \land \neg Q\} & \; c \; \{R\} \\
\{P\} & \; c \; \{R\} \\
\end{align*}
\]
A tour of $\times pRHL$ rules: [Case]

In $\times pRHL$:

\[
\begin{align*}
\{P \land Q\} & \vdash c \{R\} \implies c^\times \\
\{P \land \neg Q\} & \vdash c \{R\} \implies c_\neg^\times \\
\{P\} & \vdash c \{R\} \implies \text{if } Q \text{ then } c^\times \text{ else } c_\neg^\times
\end{align*}
\]
A tour of $\times$ pRHL rules: [Case]

In $\times$ pRHL:

\[
\begin{align*}
\{P \land Q\} & \xrightarrow{c} c^\times \\
\{P \land \neg Q\} & \xrightarrow{c} c^- \\
\{P\} & \xrightarrow{c} \text{if } Q \text{ then } c^\times \text{ else } c^-
\end{align*}
\]

Case in proof $\xrightarrow{\sim}$ conditional in product
See the paper for ...

Verifying rapid mixing for Markov chains
- Examples from statistical physics
- A cool card trick

Advanced proof rules
- Asynchronous loop rule

Soundness
Our technical contributions

A probabilistic product construction with shared randomness

A probabilistic program logic × pRHL: a proof-relevant version of pRHL
Proof by coupling

A proof technique from probability theory

- **Given**: two processes
- **Specify**: how to coordinate random samplings
- **Analyze**: properties of linked/coupled processes

**Attractive features**

- Compositional
- Reason about relation between samples, not probabilities
- Reduce properties of **two** programs to properties of **one** program
Coupling proofs $\cong$ pRHL proofs
Coupling proofs $\approx$ pRHL proofs

describe

Two coupled processes
Coupling proofs $\approx$ pRHL proofs

describe $\approx$ encode

Two coupled processes $\approx$ Probabilistic product programs
Coupling proofs $\approx$ pRHL proofs

describe

Two coupled processes $\approx$ Probabilistic product programs

encode

Probabilistic product programs are the computational content of coupling proofs