Proving Uniformity and Independence by Self-Composition and Coupling

Gilles Barthe
Thomas Espitau
Benjamin Grégoire
Justin Hsu*
Pierre-Yves Strub
A puzzle

A random walk on a cycle

- Start at position $s \in \{0, 1, \ldots, n - 1\}$
- Each iteration, flip a fair coin
  - Heads: increment position (modulo $n$)
  - Tails decrement position (modulo $n$)
- Return: last edge $(r, r + 1)$ to be traversed

A question

What is the distribution of the returned edge, and how does it depend on the starting position $s$?
A puzzle

Somewhat surprisingly, the distribution of the final edge is uniform: starting position \( s \) doesn't matter!
A puzzle

Somewhat surprisingly, the distribution of the final edge is uniform: the starting position $s$ doesn't matter!
A puzzle

Somewhat surprisingly, the distribution of the final edge is uniform: the starting position $s$ doesn't matter!
A puzzle

Somewhat surprisingly, the distribution of the final edge is uniform: starting position $s$ doesn't matter!
A puzzle

Somewhat surprisingly, the distribution of the final edge is uniform. Starting position $s$ doesn't matter!
Somewhat surprisingly, the distribution of the final edge is uniform: starting position $s$ doesn't matter!
A puzzle

Somewhat surprisingly, the distribution of the final edge is uniform: starting position doesn’t matter!
A puzzle

Somewhat surprisingly, the distribution of the final edge is uniform. Starting position doesn't matter!
A puzzle

Somewhat surprisingly

Distribution of final edge is **uniform**: Starting position $s$ doesn’t matter!
Basic properties of probabilistic programs

Uniformity of a variable $X$

For any two values $w, v$ in the (finite) range of $X$, we have:

$$\Pr[X = w] = \Pr[X = v]$$

in output distribution.
Basic properties of probabilistic programs

Uniformity of a variable $X$
For any two values $w, v$ in the (finite) range of $X$, we have:
\[
\Pr[X = w] = \Pr[X = v]
\]
in output distribution.

Independence of two variables $X, Y$
For any two values $w, v$, we have:
\[
\Pr[X = w \land Y = v] = \Pr[X = w] \cdot \Pr[Y = v]
\]
in output distribution.
Basic properties of probabilistic programs

Uniformity of a variable \( X \)
For any two values \( w, v \) in the (finite) range of \( X \), we have:

\[
\Pr[X = w] = \Pr[X = v]
\]

in output distribution.

Independence of two variables \( X, Y \)
For any two values \( w, v \), we have:

\[
\Pr[X = w \land Y = v] = \Pr[X = w] \cdot \Pr[Y = v]
\]

in output distribution.

Can be quite subtle to verify!
The idea today

Use logic for relational verification to verify uniformity and independence
A crash course: the relational logic pRHL
A curious program logic: pRHL [Barthe, Grégoire, Zanella-Béguelin]

pWhile: An imperative language with random sampling

\[ c ::= x \leftarrow e \mid x \leftarrow \text{flip}(p) \mid \text{if } e \text{ then } c \text{ else } c \mid \text{while } e \text{ do } c \mid \text{skip} \mid c; c \]
A curious program logic: pRHL  [Barthe, Grégoire, Zanella-Béguelin]

pWhile: An imperative language with random sampling

\[ c ::= x \leftarrow e \mid x \leftarrow \text{flip}(p) \mid \text{if } e \text{ then } c \text{ else } c \mid \text{while } e \text{ do } c \mid \text{skip} \mid c; c \]

pRHL is a program logic that is:

- Probabilistic: Programs can draw samples
A curious program logic: pRHL [Barthe, Grégoire, Zanella-Béguelin]

pWhile: An imperative language with random sampling

\[ c ::= x \leftarrow e \mid x \leftarrow \text{flip}(p) \mid \text{if } e \text{ then } c \text{ else } c \mid \text{while } e \text{ do } c \mid \text{skip} \mid c; c \]

pRHL is a program logic that is:

- Probabilistic: Programs can draw samples
- Relational: Describe executions of two programs
Judgments in pRHL

\{ P(in\langle 1 \rangle, in\langle 2 \rangle) \} \ c \sim \ c' \ \{ Q(out\langle 1 \rangle, out\langle 2 \rangle) \}
J udgments in pRHL

\{ P(\langle 1 \rangle, \langle 2 \rangle) \} \ c \sim \ c' \ \{ Q(\langle 1 \rangle, \langle 2 \rangle) \}

A ssertions

- Non-probabilistic
- FO formulas over program variables tagged with \langle 1 \rangle or \langle 2 \rangle
Judgments in pRHL

\[ \{ P(\text{in} \langle 1 \rangle, \text{in} \langle 2 \rangle) \} \ c \sim c' \ \{ Q(\text{out} \langle 1 \rangle, \text{out} \langle 2 \rangle) \} \]

Assertions

- Non-probabilistic
- FO formulas over program variables tagged with \( \langle 1 \rangle \) or \( \langle 2 \rangle \)
Judgments in pRHL

\{ P(\text{in}\langle 1 \rangle, \text{in}\langle 2 \rangle) \} \ c \sim c' \ \{ Q(\text{out}\langle 1 \rangle, \text{out}\langle 2 \rangle) \} 

Assertions

- Non-probabilistic
- FO formulas over program variables tagged with \langle 1 \rangle or \langle 2 \rangle

Deep connection to probabilistic couplings

- Proofs specify how to correlate random samplings in runs
- Reduce sources of randomness, simplify verification
For our purposes today: equality of distributions

If this is provable:

\[ \vdash \{ P \} \; c \sim c' \; \{ e\langle 1 \rangle = e'\langle 2 \rangle \} \]

Then:

On any two input memories related by \( P \), the distribution of \( e \) in the first output is equal to the distribution of \( e' \) in the second output.
In particular: express equality of probabilities

If this is provable for booleans $b, b'$:

$$\vdash \{P\} \ c \sim c' \ \{b\langle 1 \rangle = b'\langle 2 \rangle\}$$

Then:

On any two input memories related by $P$, the probability of $b$ in the first output is equal to the probability of $b'$ in the second output.
Random sampling rules in pRHL

Simplified version

\[ \text{FlipEq} \quad \Downarrow \{ \top \} \quad x \overset{\$}{\leftarrow} \text{flip}(p) \sim x' \overset{\$}{\leftarrow} \text{flip}(p) \quad \{x(1) = x'(2)\} \]

\[ \text{FlipNeg} \quad \Downarrow \{ \top \} \quad x \overset{\$}{\leftarrow} \text{flip}(p) \sim x' \overset{\$}{\leftarrow} \text{flip}(1 - p) \quad \{x(1) = \neg x'(2)\} \]
Random sampling rules in pRHL

**Simplified version**

**FlipEq**

\[
\begin{array}{l}
\vdash \{\top\} \quad x \leftarrow \text{flip}(p) \sim x' \leftarrow \text{flip}(p) \quad \{x(1) = x'(2)\}
\end{array}
\]

**FlipNeg**

\[
\begin{array}{l}
\vdash \{\top\} \quad x \leftarrow \text{flip}(p) \sim x' \leftarrow \text{flip}(1 - p) \quad \{x(1) = \neg x'(2)\}
\end{array}
\]

**Reading:** for any \(p \in [0, 1]\),

1. **[FlipEq]**: Distributions of \(\text{flip}(p)\) and \(\text{flip}(p)\) are equal
2. **[FlipNeg]**: Distributions of \(\text{flip}(p)\) and negated \(\text{flip}(1 - p)\) are equal
Rest of rules are standard (≈ Hoare logic)

Assignments

\[\vdash \{Q[e(1), e'(2)/x(1), x'(2)]\} \quad x \leftarrow e_1 \sim x' \leftarrow e_2 \quad \{Q\}\]

Sequencing

\[\vdash \{P\} \quad c_1 \sim c'_1 \quad \{Q\} \quad \vdash \{Q\} \quad c_2 \sim c'_2 \quad \{R\}\]

\[\vdash \{P\} \quad c_1; c_2 \sim c'_1; c'_2 \quad \{R\}\]

Loops

\[\vdash \{P \land b(1)\} \quad c \sim c' \quad \{P\} \quad \models P \implies b(1) = b'(2)\]

\[\vdash \{P\} \quad \text{while } b \text{ do } c \sim \text{while } b' \text{ do } c' \quad \{P \land \neg b(1)\}\]
Rest of rules are standard (≈ Hoare logic)

**Assignments**

\[
\frac{\infer{\{Q[e'\langle 2\rangle/x'\langle 1\rangle, x'\langle 2\rangle]\}}{\text{ASSN}}}{x \leftarrow e_1 \sim x' \leftarrow e_2 \{Q}\}
\]

**Sequencing**

\[
\frac{\infer{\{P\}}{\text{SEQ}}}{c_1 \sim c'_1 \{Q\} \quad \infer{\{Q\}}{c_2 \sim c'_2 \{R\}} \quad \infer{\{P\}}{c_1; c_2 \sim c'_1; c'_2 \{R\}}}
\]

**Loops**

\[
\frac{\infer{\{P\}}{\text{WHILE}}}{\{P \land b'\langle 1\rangle\} \quad c \sim c' \{P\} \quad \Longrightarrow \quad b'\langle 2\rangle \quad \infer{\{P\}}{\text{while } b \text{ do } c \sim \text{while } b' \text{ do } c' \{P \land \neg b'\langle 1\rangle\}}}
\]
Benefits of pRHL

Probabilistic properties without probabilistic reasoning

- Abstract away all probabilities
- All reasoning is about relation between samples

Highly similar to Hoare logic

- Most things “just work”
- Compositional reasoning
Benefits of pRHL

Probabilistic properties without probabilistic reasoning

- Abstract away all probabilities
- All reasoning is about relation between samples

Highly similar to Hoare logic

- Most things “just work”
- Compositional reasoning

Apply to non-relational properties, like uniformity and independence.
Verifying uniformity: simulating a fair coin
The algorithm

**Goal**
Generate one fair coin flip, using only coin flips with a fixed bias $p \in (0, 1)$.

**Procedure**
1. Flip two coins with bias $p$
2. Re-flip as long as they are equal
3. Return the first coin flip the first time they are different
Consider the program $fair$:

\[
x \leftarrow tt;
y \leftarrow tt;
\]

while $x = y$ do

\[
x \leftarrow \text{flip}(p);
y \leftarrow \text{flip}(p);
\]

return($x$)

To show: generates fair coin flip

---

Distribution of return value is uniform
Observation: uniformity can be proved in pRHL

For every two booleans $w, v$, show:

\[
\vdash \{ p(1) = p(2) \} \quad \text{fair} \sim \text{fair} \quad \{(x(1) = w) \iff (x(2) = v)\}
\]

Reading: for every two booleans $w, v$,

\[
\Pr[x = w] = \Pr[x = v] \quad \text{in the output of fair.}
\]

Four choices in all for $w, v$

- We show the cases with $w \neq v$
Step 1: rearrange program

Two equivalent programs: $fair$ and $fair'$

\[
\begin{align*}
x &\leftarrow tt; \\
y &\leftarrow tt; \\
\text{while } x = y \text{ do} \\
&\quad x \leftarrow \text{flip}(p); \\
&\quad y \leftarrow \text{flip}(p); \\
&\quad \text{return}(x)
\end{align*}
\]

\[
\begin{align*}
x &\leftarrow tt; \\
y &\leftarrow tt; \\
\text{while } x = y \text{ do} \\
&\quad y \leftarrow \text{flip}(p); \\
&\quad x \leftarrow \text{flip}(p); \\
&\quad \text{return}(x)
\end{align*}
\]
Step 1: rearrange program

Two equivalent programs: \textit{fair} and \textit{fair}'

\begin{align*}
  x & \leftarrow \texttt{tt}; \\
  y & \leftarrow \texttt{tt}; \\
  \text{while } x = y \text{ do} \\
  & \quad x \leftarrow \text{flip}(p); \\
  & \quad y \leftarrow \text{flip}(p); \\
  & \quad \text{return}(x)
\end{align*}

\begin{align*}
  x & \leftarrow \texttt{tt}; \\
  y & \leftarrow \texttt{tt}; \\
  \text{while } x = y \text{ do} \\
  & \quad y \leftarrow \text{flip}(p); \\
  & \quad x \leftarrow \text{flip}(p); \\
  & \quad \text{return}(x)
\end{align*}
Step 1: rearrange program

Two equivalent programs: *fair* and *fair'*

\[
\begin{align*}
x & \leftarrow \text{tt}; \\
y & \leftarrow \text{tt}; \\
\text{while } x = y \text{ do} \\
x & \leftarrow \text{flip}(p); \\
y & \leftarrow \text{flip}(p); \\
\text{return}(x)
\end{align*}
\]

\[
\begin{align*}
x & \leftarrow \text{tt}; \\
y & \leftarrow \text{tt}; \\
\text{while } x = y \text{ do} \\
y & \leftarrow \text{flip}(p); \\
x & \leftarrow \text{flip}(p); \\
\text{return}(x)
\end{align*}
\]
Step 1: rearrange program

Two equivalent programs: \textit{fair} and \textit{fair'}

\[
\begin{align*}
x & \leftarrow tt; \\
y & \leftarrow tt; \\
\text{while } x = y \text{ do} \\
& \quad x \leftarrow \text{flip}(p); \\
& \quad y \leftarrow \text{flip}(p); \\
& \quad \text{return}(x)
\end{align*}
\]

For the cases \(w \neq v\), suffices to show:

\[
\vdash \{p(1) = p(2)\} \quad \text{fair} \sim \text{fair'} \quad \{x(1) = \overline{x(2)}\}
\]
Step 2: apply the loop rule

\[
\text{while } x = y \text{ do } \\
\quad x \leftarrow \text{flip}(p); \\
\quad y \leftarrow \text{flip}(p); \\
\text{return}(x)
\]
Step 2: apply the loop rule

while \( x = y \) do
\[
\begin{align*}
& x \leftarrow \text{flip}(p); \\
& y \leftarrow \text{flip}(p); \\
& \text{return}(x)
\end{align*}
\]

while \( x = y \) do
\[
\begin{align*}
& y \leftarrow \text{flip}(p); \\
& x \leftarrow \text{flip}(p); \\
& \text{return}(x)
\end{align*}
\]

In the body: apply [FLIPEQ] for both pairs of samples
Step 2: apply the loop rule

while $x = y$ do
  $x \leftarrow \text{flip}(p)$;
  $y \leftarrow \text{flip}(p)$;
return($x$)

while $x = y$ do
  $y \leftarrow \text{flip}(p)$;
  $x \leftarrow \text{flip}(p)$;
return($x$)

In the body: apply [FLIP_EQ] for both pairs of samples

- We have: $x\langle 1 \rangle = y\langle 2 \rangle$
Step 2: apply the loop rule

while $x = y$ do
  $x \leftarrow \text{flip}(p)$;
  $y \leftarrow \text{flip}(p)$;
return($x$)

while $x = y$ do
  $y \leftarrow \text{flip}(p)$;
  $x \leftarrow \text{flip}(p)$;
return($x$)

In the body: apply \texttt{[FLIPEQ]} for both pairs of samples

- We have: $x\langle 1 \rangle = y\langle 2 \rangle$
- And: $x\langle 2 \rangle = y\langle 1 \rangle$
Step 2: apply the loop rule

while $x = y$ do
  $x \leftarrow \text{flip}(p);$  
  $y \leftarrow \text{flip}(p);$  
  return($x$)

In the body: apply [FLIPEQ] for both pairs of samples

- We have: $x\langle 1 \rangle = y\langle 2 \rangle$
- And: $x\langle 2 \rangle = y\langle 1 \rangle$

Establishes main invariant:

\[
x\langle 2 \rangle = (\text{if } x\langle 1 \rangle = y\langle 1 \rangle \text{ then } y\langle 2 \rangle \text{ else } \neg x\langle 1 \rangle)
\]
Step 3: putting it all together

Applying [ASSN], [SEQ] shows:

\[ \vdash \{ p'(1) = p'(2) \} \quad \text{fair} \sim \text{fair} \quad \{(x'(1) = w) \iff (x'(2) = v)\} \]

when \( w \neq v \); can also show same judgment when \( w = v \).

Conclude

\textit{fair} returns a uniform boolean
Extensions:
verifying independence
Verifying independence: the easier way

Observation: reduce independence to uniformity

\[(x, y)\] is uniform over pairs \[\Downarrow\] \[x\] and \[y\] are independent

Limitation

- Only can show independence for uniform variables
Verifying independence: the harder way

Use self-composition

- Let $c[1], c[2]$ be two copies of $c$ with disjoint variables
- Prove a pRHL judgment relating

\[ c \sim c[1]; c[2] \]
Verifying independence: the harder way

Use self-composition

- Let $c[1], c[2]$ be two copies of $c$ with disjoint variables
- Prove a pRHL judgment relating

$$c \sim c[1] ; c[2]$$

Independence of two variables $X, Y$

For any two values $w, v$, we have:

$$\Pr[X = w \land Y = v] = \Pr[X = w] \cdot \Pr[Y = v]$$

in output distribution.
Verifying independence: the harder way

Use self-composition

- Let \( c[1], c[2] \) be two copies of \( c \) with disjoint variables
- Prove a pRHL judgment relating

\[
c \sim c[1]; c[2]
\]

Independence of two variables \( X, Y \)

For any two values \( w, v \), we have:

\[
\Pr[X = w \land Y = v] = \Pr[X = w] \cdot \Pr[Y = v]
\]

in output distribution.
Verifying independence: the harder way

Use self-composition

- Let \( c[1], c[2] \) be two copies of \( c \) with disjoint variables
- Prove a \( \text{pRHL} \) judgment relating

\[
c \sim c[1]; c[2]
\]

Independence of two variables \( X, Y \)
For any two values \( w, v \), we have:

\[
\Pr[X = w \land Y = v] = \Pr[X = w] \cdot \Pr[Y = v]
\]

in output distribution.
Verifying independence: the harder way

Use self-composition

- Let $c[1], c[2]$ be two copies of $c$ with disjoint variables
- Prove a pRHL judgment relating

$$c \sim c[1]; c[2]$$

Independence of two variables $X, Y$

For any two values $w, v$, we have:

$$\Pr[X = w \land Y = v] = \Pr[X = w] \cdot \Pr[Y = v]$$

in output distribution.
Verifying independence: the harder way

Use self-composition

- Let $c[1], c[2]$ be two copies of $c$ with disjoint variables
- Prove a pRHL judgment relating

$$c \sim c[1]; c[2]$$

Independence of two variables $X, Y$

For any two values $w, v$, we have:

$$\Pr[X = w \land Y = v] = \Pr[X = w] \cdot \Pr[Y = v]$$

in output distribution.

Benefits

- Can prove independence for non-uniform variables
- Similar ideas can cover conditional independence
Summing up
See the paper for

Lots more examples

- Cycle random walk
- Pairwise and $k$-wise independence
- Bayesian network
- Ballot theorem

Details about the implementation

- Most examples formalized in EasyCrypt framework
Future directions

• Automate this approach
• Explore relational verification for non-relational properties
• Integrate with more general probabilistic verification tools
Proving Uniformity and Independence by Self-Composition and Coupling

Gilles Barthe
Thomas Espitau
Benjamin Grégoire
Justin Hsu*
Pierre-Yves Strub