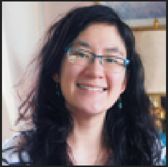
Verifying Probabilistic Properties with Probabilistic Couplings

Justin Hsu
UW–Madison
Computer Sciences

Work with brilliant collaborators





What Are Probabilistic “Relational Properties”?

Today's target properties

Probabilistic

- Programs can take random samples (flip coins)
- Map (single) input value to a distribution over outputs

Relational

- Compare two executions of a program (or: two programs)
- Describe outputs (distributions) from two related inputs
- Also known as 2-properties, or hyperproperties

Examples throughout computer science...

Security and privacy

- Indistinguishability
- Differential privacy

Machine learning

- Uniform stability

... and beyond

- Incentive properties (game theory/mechanism design)
- Convergence and mixing (probability theory)

Challenges for formal verification

Reason about two sources of randomness

- Two executions may behave very differently
- Completely different control flow (even for same program!)

Quantitative reasoning

- Target properties describe distributions
- Probabilities, expected values, etc.
- Very messy for formal reasoning

Today: Combine two ingredients

Probabilistic Couplings

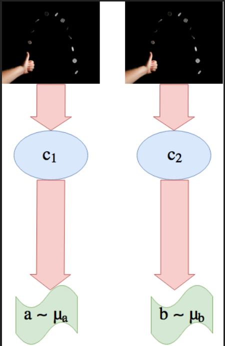
$+$

Relational Program Logics

Probabilistic Couplings and “Proof by Coupling”

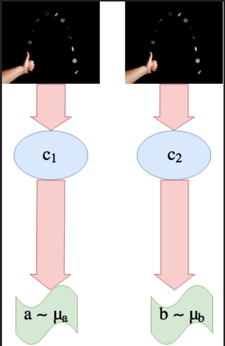
Given: programs c_1 and c_2, each taking 10 coin flips

Experiment #1

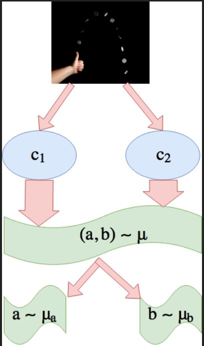


Given: programs c_1 and c_2, each taking 10 coin flips

Experiment #1

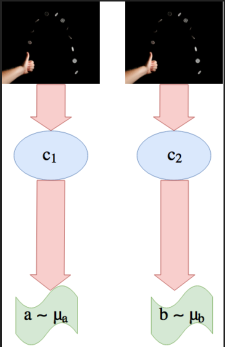


Experiment #2

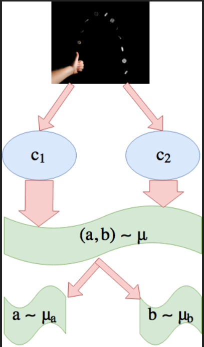


Given: programs c_1 and c_2, each taking 10 coin flips

Experiment #1



Experiment #2



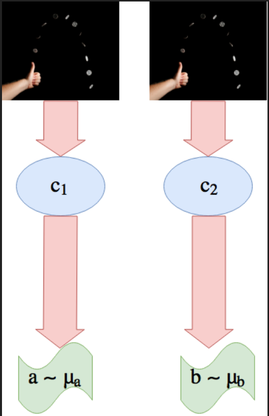
Distributions equal in Experiment #1

$\Longleftrightarrow$

Distributions equal in Experiment #2

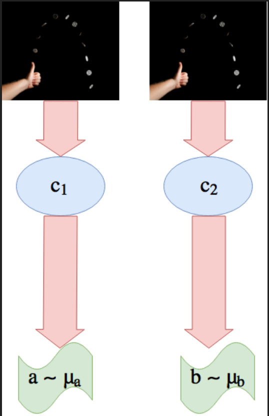
Given: programs c_1 and c_2, each taking 10 coin flips

Experiment #1

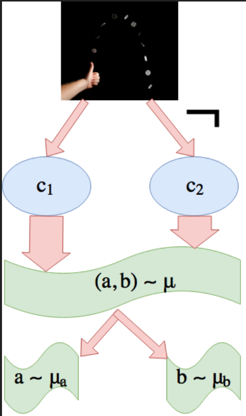


Given: programs c_1 and c_2, each taking 10 coin flips

Experiment #1

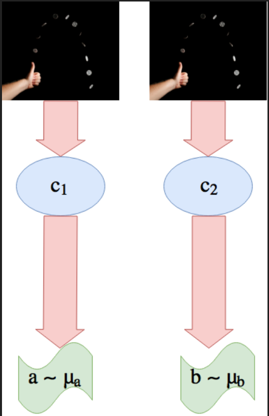


Experiment #2

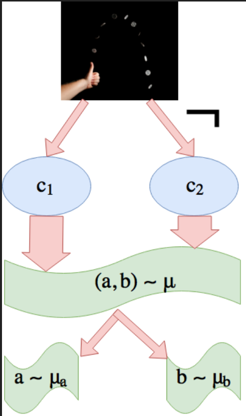


Given: programs c_1 and c_2, each taking 10 coin flips

Experiment #1



Experiment #2



Distributions equal in Experiment #1

$\iff$

Distributions equal in Experiment #2

Why “pretend” two executions share randomness?

Easier to reason about one source of randomness

- Fewer possible executions
- Pairs of coordinated executions follow similar control flow

Reduce quantitative reasoning

- Reason on (non-probabilistic) relations between samples
- Don’t need to work with raw probabilities (messy)

A bit more precisely...

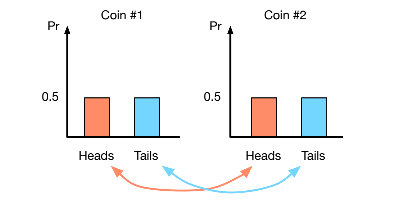
A coupling of two distributions $\mu_1, \mu_2 \in \mathsf{Distr}(A)$ is a joint distribution $\mu \in \mathsf{Distr}(A \times A)$ with $\pi_1(\mu) = \mu_1$ and $\pi_2(\mu) = \mu_2$.

A bit more precisely...

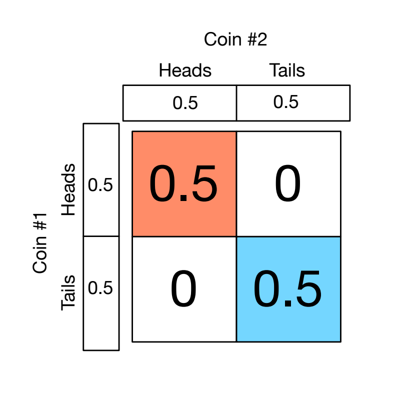
A coupling of two distributions $\mu_1, \mu_2 \in \mathsf{Distr}(A)$ is a joint distribution $\mu \in \mathsf{Distr}(A \times A)$ with $\pi_1(\mu) = \mu_1$ and $\pi_2(\mu) = \mu_2$.

A coupling models two distributions
sharing one source of randomness

For example

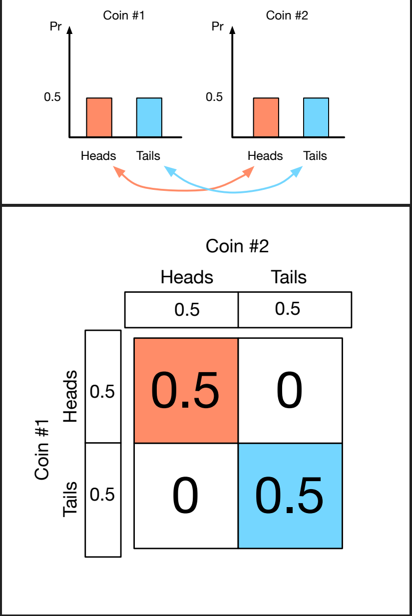


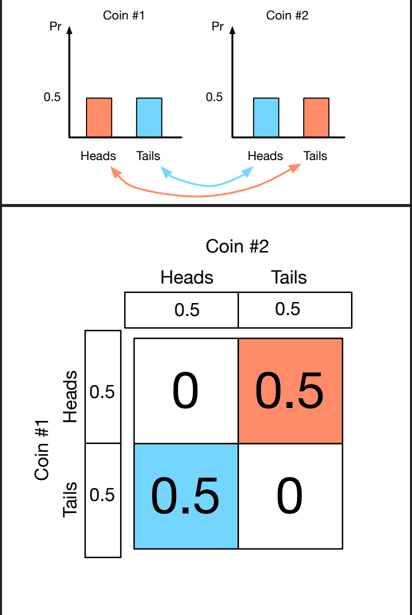
Coin #1
Pr
0.5
Heads
Tails
Coin #2
Pr
0.5
Heads
Tails



Coin #2
Heads
Tails
0.5
0.5
Coin #1
Heads
Tails
0.5
0.5
0.5
0
0
0.5

For example


Coin #1
Pr
0.5
Heads
Tails
Coin #2
Pr
0.5
Heads
Tails
Coin #2
Heads
Tails
0.5
0.5
Coin #1
Heads
Tails
0.5
0.5
0.5
0
0
0.5


Coin #1
Pr
0.5
Heads
Tails
Coin #2
Pr
0.5
Heads
Tails
Coin #2
Heads
Tails
0.5
0.5
Coin #1
Heads
Tails
0.5
0.5
0
0.5
0.5
0

Why are couplings interesting for verification?

Existence of a coupling* can imply a property of two distributions

If there exists a coupling of (μ_1, μ_2) where:

then:

Two coupled samples differ with small probability

μ_1 is “close” to μ_2

If there exists a coupling of (μ_1, μ_2) where:	then:
Two coupled samples differ with small probability	μ_1 is “close” to μ_2
Two coupled samples are always equal	μ_1 is “equal” to μ_2

If there exists a coupling of (μ_1, μ_2) where:	then:
Two coupled samples differ with small probability	μ_1 is "close" to μ_2
Two coupled samples are always equal	μ_1 is "equal" to μ_2
First coupled sample is always larger than second sample	μ_1 "dominates" μ_2

Our plan to verify these properties

Three easy steps

1. Start from two given programs
2. Show that for two related inputs, there exists a coupling of the output distributions with certain properties
3. Conclude relational property of program(s)

Our plan to verify these properties

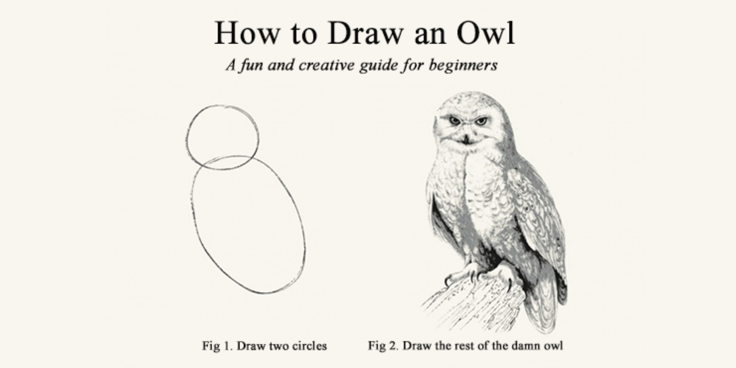
Three easy steps

1. Start from two given programs
2. Show that for two related inputs, there exists a coupling of the output distributions with certain properties
3. Conclude relational property of program(s)

Our plan to verify these properties

Three easy steps

1. Start from two given programs
2. Show that for two related inputs, there exists a coupling of the output distributions with certain properties
3. Conclude relational property of program(s)



Show existence of a coupling by constructing it

A coupling proof is a recipe
for constructing a coupling

Show existence of a coupling by constructing it

A coupling proof is a recipe for constructing a coupling

1. Specify: How to couple pairs of intermediate samples
2. Deduce: Relation between final coupled samples
3. Conclude: Property about two original distributions

Probabilistic Relational Program Logics

Make statements about imperative programs

Imperative language WHILE

$$c ::= \mathsf{skip} \mid x \leftarrow e \mid \mathsf{if}\ b\ \mathsf{then}\ c\ \mathsf{else}\ c' \mid c;\ c' \mid \mathsf{while}\ b\ \mathsf{do}\ c$$

Make statements about imperative programs

Imperative language WHILE

$$c ::= \mathsf{skip} \mid x \leftarrow e \mid \mathsf{if}\ b\ \mathsf{then}\ c\ \mathsf{else}\ c' \mid c;\ c' \mid \mathsf{while}\ b\ \mathsf{do}\ c$$

Semantics: WHILE programs transform memories

- Variables: Fixed set $\mathcal{X}$ of program variable names
- Memories $\mathcal{M}$: functions from $\mathcal{X}$ to values $\mathcal{V}$ (e.g., 42)
- Interpret each command c as a memory transformer:

$$[\![c]\!] : \mathcal{M} \to \mathcal{M}$$

Program logics (Floyd-Hoare logics)

Logical judgments look like this

$$\{P\} \;\; c \;\; \{Q\}$$

Interpretation

- Program c, WHILE program (e.g., $x \leftarrow y; y \leftarrow y+1$)
- Precondition P, formula over $\mathcal{X}$ (e.g., $y \geq 0$)
- Postcondition Q, formula over $\mathcal{X}$ (e.g., $x \geq 0 \land y \geq 0$)

If P holds before running c, then Q holds after running c

Probabilistic Relational Hoare Logic (pRHL) [BGZ-B]

Previously

- Inspired by Benton's Relational Hoare Logic
- Foundation of the EasyCrypt system
- Verified security of many cryptographic schemes

Probabilistic Relational Hoare Logic (pRHL) [BGZ-B]

Previously

- Inspired by Benton's Relational Hoare Logic
- Foundation of the EasyCrypt system
- Verified security of many cryptographic schemes

New interpretation

pRHL is a logic for formal proofs by coupling

Language and judgments

The pWhile imperative language

$$c ::= \mathsf{skip} \mid x \leftarrow e \mid x \xleftarrow{\$} d \mid \mathsf{if}\ e\ \mathsf{then}\ c\ \mathsf{else}\ c \mid c; c \mid \mathsf{while}\ e\ \mathsf{do}\ c$$

Language and judgments

The pWhile imperative language

$$c ::= \mathsf{skip} \mid x \leftarrow e \mid \boxed{x \xleftarrow{\$} d} \mid \mathsf{if}\ e\ \mathsf{then}\ c\ \mathsf{else}\ c \mid c; c \mid \mathsf{while}\ e\ \mathsf{do}\ c$$

Semantics of pWhile programs

- Input: a single memory (assignment to variables)
- Output: a distribution over memories
- Interpret each command c as:

$$[\![c]\!] : \mathcal{M} \to \mathsf{Distr}(\mathcal{M})$$

Basic pRHL judgments

$$\{P\} \;\; c_1 \sim c_2 \;\; \{Q\}$$

- P and Q are formulas over program variables
- Labeled program variables: x_1, x_2
- P is precondition, Q is postcondition

Interpreting the judgment

Logical judgments in pRHL look like this

$$\{P\} \;\; c_1 \sim c_2 \;\; \{Q\}$$

Interpreting the judgment

Logical judgments in pRHL look like this

$$\{P\}\ c_1 \sim c_2\ \{Q\}$$

Interpreting pre- and post-conditions

- As usual, P is a relation on two memories
- Q interpreted as a relation $\langle Q \rangle$ on memory distributions

Interpreting the judgment

Logical judgments in pRHL look like this

$$\{P\}\ c_1 \sim c_2\ \{Q\}$$

Interpreting pre- and post-conditions

- As usual, P is a relation on two memories
- Q interpreted as a relation $\langle Q \rangle$ on memory distributions

Definition (Valid pRHL judgment)

For any pair of related inputs $(m_1, m_2) \in [\![P]\!]$, there exists a coupling $\mu \in \mathsf{Distr}(\mathcal{M} \times \mathcal{M})$ of the output distributions $([\![c_1]\!]m_1, [\![c_2]\!]m_2)$ such that $\mathsf{supp}(\mu) \subseteq [\![Q]\!]$.

Encoding couplings with PRHL theorems

$$\{P\} \ c_1 \sim c_2 \ \{o_1 = o_2\}$$

Interpretation

If two inputs satisfy P, there exists a coupling of the output distributions where the coupled samples have equal o

Encoding couplings with PRHL theorems

$$\{P\} \;\; c_1 \sim c_2 \;\; \{o_1 = o_2\}$$

Interpretation

If two inputs satisfy P, there exists a coupling of the output distributions where the coupled samples have equal o

This implies:

If two inputs satisfy P, the distributions of o are equal

Encoding couplings with pRHL theorems

$$\{P\}\ c_1 \sim c_2\ \{o_1 \geq o_2\}$$

This implies:

If two inputs satisfy P, then the first distribution of o stochastically dominates the second distribution of o

Proving Judgments:

The Proof System of pRHL

More convenient way to prove judgments

Inference rules describe:

- Judgments that are always true (axioms)
- How to prove judgment for a program by combining judgments for components

More convenient way to prove judgments

Inference rules describe:

- Judgments that are always true (axioms)
- How to prove judgment for a program by combining judgments for components

Example: sequential composition rule

Given: $\{P\}\ c_1\ \{Q\}$ and $\{Q\}\ c_2\ \{R\}$

More convenient way to prove judgments

Inference rules describe:

- Judgments that are always true (axioms)
- How to prove judgment for a program by combining judgments for components

Example: sequential composition rule

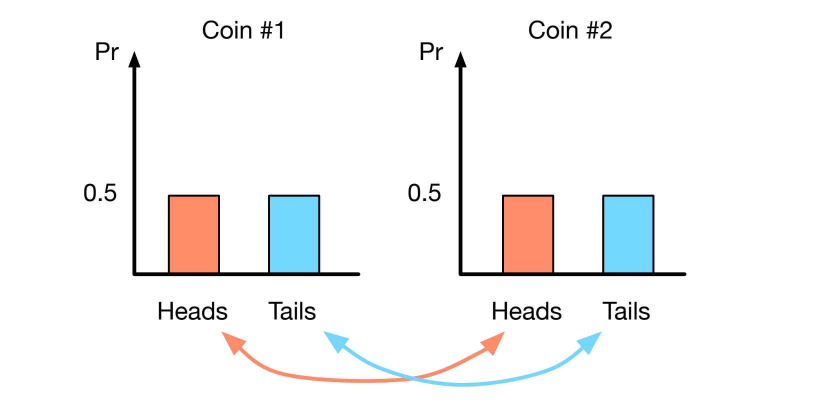
$$\frac{\text{Given:} \quad \{P\}\ c_1\ \{Q\} \quad \text{and} \quad \{Q\}\ c_2\ \{R\}}{\text{Conclude:} \quad \{P\}\ c_1\ ;\ c_2\ \{R\}}$$

Reading the rules: introduce couplings

$$\frac{}{\vdash \{\ \}\ \ x_1 \xleftarrow{\$} \mathit{flip} \sim x_2 \xleftarrow{\$} \mathit{flip}\ \ \{x_1 = x_2\}}$$

Reading the rules: introduce couplings

$$\frac{}{\vdash \{\ \}\ x_1 \xleftarrow{\$} \mathit{flip} \sim x_2 \xleftarrow{\$} \mathit{flip}\ \{x_1 = x_2\}}$$

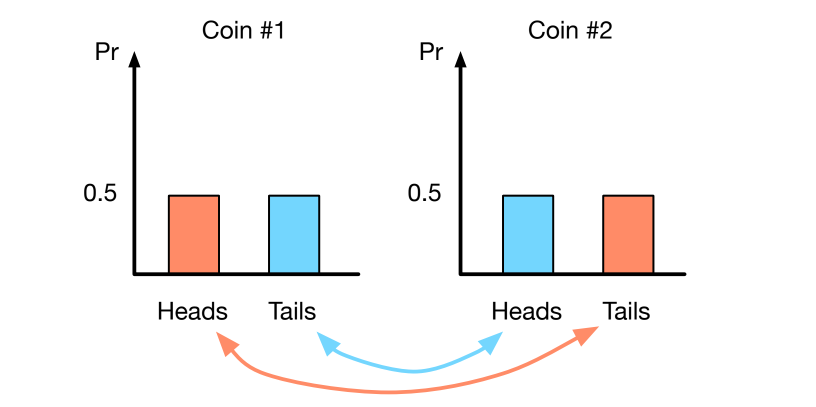


Reading the rules: introduce couplings

$$\frac{}{\vdash \{\ \}\ \ x_1 \xleftarrow{\$} \mathit{flip} \sim x_2 \xleftarrow{\$} \mathit{flip}\ \ \{x_1 \neq x_2\}}$$

Reading the rules: introduce couplings

$$\frac{}{\vdash \{\,\}\ x_1 \xleftarrow{\$} \mathit{flip} \sim x_2 \xleftarrow{\$} \mathit{flip}\ \{x_1 \neq x_2\}}$$



Reading the rules: combine couplings

$$\frac{\vdash \{P\}\ c_1 \sim c_2\ \{Q\} \qquad \vdash \{Q\}\ c_1' \sim c_2'\ \{R\}}{\vdash \{P\}\ c_1; c_1' \sim c_2; c_2'\ \{R\}}$$

Reading the rules: combine couplings

$$\frac{\vdash \{P\}\ c_1 \sim c_2\ \{Q\} \qquad \vdash \{Q\}\ c_1' \sim c_2'\ \{R\}}{\vdash \{P\}\ c_1; c_1' \sim c_2; c_2'\ \{R\}}$$

$$\boxed{\text{Sequence couplings}}$$

Reading the rules: combine couplings

$$\frac{\vdash \{P \wedge S\}\ c_1 \sim c_2\ \{Q\} \qquad \vdash \{P \wedge \neg S\}\ c_1 \sim c_2\ \{Q\}}{\vdash \{P\}\ c_1 \sim c_2\ \{Q\}}$$

Reading the rules: combine couplings

$$\frac{\vdash \{P \wedge S\}\ c_1 \sim c_2\ \{Q\} \qquad \vdash \{P \wedge \neg S\}\ c_1 \sim c_2\ \{Q\}}{\vdash \{P\}\ c_1 \sim c_2\ \{Q\}}$$

$\boxed{\text{Select couplings}}$

Reading the rules: combine couplings

$$\frac{\vdash \{P \wedge e_1 \wedge e_2\}\ \ c_1 \sim c_2\ \ \{P\} \qquad \models P \to e_1 = e_2}{\vdash \{P\}\ \ \textsf{while}\ e_1\ \textsf{do}\ c_1 \sim \textsf{while}\ e_2\ \textsf{do}\ c_2\ \ \{P \wedge (\neg e_1 \wedge \neg e_2)\}}$$

Reading the rules: combine couplings

$$\frac{\vdash \{P \land e_1 \land e_2\}\ c_1 \sim c_2\ \{P\} \qquad \models P \to e_1 = e_2}{\vdash \{P\}\ \text{while } e_1 \text{ do } c_1 \sim \text{while } e_2 \text{ do } c_2\ \{P \land (\neg e_1 \land \neg e_2)\}}$$

Repeat couplings

Reading the rules: combine couplings

$$\frac{\vdash \{P \wedge e_1 \wedge e_2\}\ c_1 \sim c_2\ \{P\} \qquad \models P \to e_1 = e_2}{\vdash \{P\}\ \text{while } e_1 \text{ do } c_1 \sim \text{while } e_2 \text{ do } c_2\ \{P \wedge (\neg e_1 \wedge \neg e_2)\}}$$

Repeat couplings

Not a rule: conjunction

$$\frac{\vdash \{P\}\ c_1 \sim c_2\ \{Q\} \qquad \vdash \{P\}\ c_1 \sim c_2\ \{R\}}{\vdash \{P\}\ c_1 \sim c_2\ \{Q \wedge R\}}$$

Not a rule: conjunction

$$\frac{\vdash \{P\}\ c_1 \sim c_2\ \{Q\} \qquad \vdash \{P\}\ c_1 \sim c_2\ \{R\}}{\vdash \{P\}\ c_1 \sim c_2\ \{Q \wedge R\}}$$

Can't compose this way

Is this just bisimulation?

More general property

- Relation need not be equivalence (bisimulation)
- Relation need not be preorder (simulation)

More general model of computation

- Probabilistic imperative programs
- State space can be infinite/parametrized

More flexible construction

- No fixed notion of a transition
- Coupling can be constructed “asynchronously”

Formal Proofs by Coupling

Ex. 1: Equivalence

Target property: equivalence

P's output distribution is the same for any two inputs

- Shows: output distribution is the same for any input
- Security: input is secret, output is encrypted

Warmup example: secrecy of one-time-pad (OTP)

The program

- Program input: a secret boolean sec
- Program output: an encrypted version of the secret

$$
\begin{array}{ll}
key \xleftarrow{\$} \mathit{flip}; & \text{// draw random key} \\
enc \leftarrow sec \oplus key; & \text{// exclusive or} \\
\text{return}(enc) & \text{// return encrypted}
\end{array}
$$

Proof by coupling

- Either sec_1, sec_2 are equal, or unequal
  1. If equal: couple sampling for key to be equal in both runs
  2. If unequal: couple sampling for key to be unequal in both runs
- Coupling ensures $enc_1 = enc_2$, hence distributions equal

Formalizing the proof in pRHL

Case 1: $sec_1 = sec_2$

- By applying identity coupling rule (general version):

$$\begin{array}{l} \{sec_1 = sec_2\} \\ key \xleftarrow{\$} flip; \\ \{key_1 = key_2\} \\ enc \leftarrow sec \oplus key \\ \{enc_1 = enc_2\} \end{array}$$

- Hence:

$$\{sec_1 = sec_2\}\ \ otp \sim otp\ \ \{enc_1 = enc_2\}$$

Formalizing the proof in pRHL

Case 2: $sec_1 \neq sec_2$

- By applying negation coupling rule (general version):

$$\begin{array}{l} \{sec_1 \neq sec_2\} \\ key \xleftarrow{\$} flip; \\ \{key_1 \neq key_2\} \\ enc \leftarrow sec \oplus key \\ \{enc_1 = enc_2\} \end{array}$$

- Hence:

$$\{sec_1 \neq sec_2\}\ \ otp \sim otp\ \ \{enc_1 = enc_2\}$$

Formalizing the proof in pRHL

Combining the cases:

$$\frac{\{sec_1 = sec_2\}\ otp \sim otp\ \{enc_1 = enc_2\} \quad \{sec_1 \neq sec_2\}\ otp \sim otp\ \{enc_1 = enc_2\}}{\{\top\}\ otp \sim otp\ \{enc_1 = enc_2\}}$$

and we are done!

Formal Proofs by Coupling

Ex. 2: Stochastic Domination

Target property: stochastic domination

Order relation on distributions

- Given: ordered set $(A, \leq_A)$
- Lift to ordering on distributions $(\text{Distr}(A), \leq_{sd})$

For naturals $(\mathbb{N}, \leq)$ ...

Two distributions $\mu_1, \mu_2 \in \text{Distr}(\mathbb{N})$ satisfy $\mu_1 \leq_{sd} \mu_2$ if

$$\text{for all } k \in \mathbb{N},\ \mu_1(\{n \mid k \leq n\}) \leq \mu_2(\{n \mid k \leq n\})$$

Proof by coupling

$ct \leftarrow 0;$
for i$= 1, \ldots, T_1$ do
  $r \xleftarrow{\$} \mathit{flip};$
  if $r =$ heads then
    $ct \leftarrow ct + 1;$
return(ct)

$ct \leftarrow 0;$
for i$= 1, \ldots, T_2$ do
  $r \xleftarrow{\$} \mathit{flip};$
  if $r =$ heads then
    $ct \leftarrow ct + 1;$
return(ct)

Proof by coupling

$ct \leftarrow 0;$
for i$= 1, \ldots, T_1$ do
  $r \xleftarrow{\$} \mathit{flip};$
  if $r =$ heads then
    $ct \leftarrow ct + 1;$
return(ct)

$ct \leftarrow 0;$
for i$= 1, \ldots, T_2$ do
  $r \xleftarrow{\$} \mathit{flip};$
  if $r =$ heads then
    $ct \leftarrow ct + 1;$
return(ct)

Suppose $T_1 \geq T_2$: first loop runs more

- Want to prove $\mu_1 \geq_{sd} \mu_2$

Proof by coupling

$ct \leftarrow 0;$
for i$= 1, \ldots, T_1$ do
  $r \xleftarrow{\$} \mathit{flip};$
  if $r =$ heads then
    $ct \leftarrow ct + 1;$
return(ct)

$ct \leftarrow 0;$
for i$= 1, \ldots, T_2$ do
  $r \xleftarrow{\$} \mathit{flip};$
  if $r =$ heads then
    $ct \leftarrow ct + 1;$
return(ct)

Suppose $T_1 \geq T_2$: first loop runs more

- Want to prove $\mu_1 \geq_{sd} \mu_2$

Suffices to construct a coupling where $ct_1 \geq ct_2$

- Couple the first T_2 samples to be equal across both runs; establishes $ct_1 = ct_2$
- Take the remaining $T_1 - T_2$ samples (in the first run) to be arbitrary; preserves $ct_1 \geq ct_2$

Formalizing the proof in pRHL

$$
\begin{array}{l}
ct \leftarrow 0; \\
\text{for i} = 1, \ldots, T_1 \text{ do} \\
\quad r \xleftarrow{\$} \mathit{flip}; \\
\quad \text{if } r = \text{heads then} \\
\quad\quad ct \leftarrow ct + 1; \\
\text{return}(ct)
\end{array}
\qquad\Bigg|\qquad
\begin{array}{l}
ct \leftarrow 0; \\
\text{for i} = 1, \ldots, T_2 \text{ do} \\
\quad r \xleftarrow{\$} \mathit{flip}; \\
\quad \text{if } r = \text{heads then} \\
\quad\quad ct \leftarrow ct + 1; \\
\text{return}(ct)
\end{array}
$$

Goal: prove

$$\boxed{\vdash \{T_1 \geq T_2\} \ c_1 \sim c_2 \ \{ct_1 \geq ct_2\}}$$

Proof sketch

Step 1: Rewrite

$ct \leftarrow 0$;
for i$= 1, \dots, T_2$ do
  $r \xleftarrow{\$} \mathit{flip}$;
  if $r =$ heads then
    $ct \leftarrow ct + 1$;
for i$= T_2 + 1, \dots, T_1$ do
  $r \xleftarrow{\$} \mathit{flip}$;
  if $r =$ heads then
    $ct \leftarrow ct + 1$;
return(ct)
	$ct \leftarrow 0$;
	for i$= 1, \dots, T_2$ do
	  $r \xleftarrow{\$} \mathit{flip}$;
	  if $r =$ heads then
	    $ct \leftarrow ct + 1$;
	return(ct)

Proof sketch

$ct \leftarrow 0$;
for i$= 1, \ldots, T_2$ do
  $r \xleftarrow{\$} \mathit{flip}$;
  if $r =$ heads then
    $ct \leftarrow ct + 1$

$ct \leftarrow 0$;
for i$= 1, \ldots, T_2$ do
  $r \xleftarrow{\$} \mathit{flip}$;
  if $r =$ heads then
    $ct \leftarrow ct + 1$

Proof sketch

$ct \leftarrow 0;$
for i$= 1, \ldots, T_2$ do
  $r \xleftarrow{\$} \mathit{flip};$
  if $r =$ heads then
    $ct \leftarrow ct + 1$
	$ct \leftarrow 0;$
	for i$= 1, \ldots, T_2$ do
	  $r \xleftarrow{\$} \mathit{flip};$
	  if $r =$ heads then
	    $ct \leftarrow ct + 1$

Step 2: First loop

- Use sampling rule with identity coupling: $r_1 = r_2$

Proof sketch

$ct \leftarrow 0;$
for i$= 1, \ldots, T_2$ do
  $r \xleftarrow{\$} \mathit{flip};$
  if $r =$ heads then
    $ct \leftarrow ct + 1$

$ct \leftarrow 0;$
for i$= 1, \ldots, T_2$ do
  $r \xleftarrow{\$} \mathit{flip};$
  if $r =$ heads then
    $ct \leftarrow ct + 1$

Step 2: First loop

- Use sampling rule with identity coupling: $r_1 = r_2$

Proof sketch

$ct \leftarrow 0;$
for i$= 1, \ldots, T_2$ do
  $r \xleftarrow{\$} flip;$
  if $r =$ heads then
    $ct \leftarrow ct + 1$

$ct \leftarrow 0;$
for i$= 1, \ldots, T_2$ do
  $r \xleftarrow{\$} flip;$
  if $r =$ heads then
    $ct \leftarrow ct + 1$

Step 2: First loop

- Use sampling rule with identity coupling: $r_1 = r_2$

Proof sketch

$ct \leftarrow 0;$
for i$= 1, \dots, T_2$ do
  $r \xleftarrow{\$} \mathit{flip};$
  if $r =$ heads then
    $ct \leftarrow ct + 1$
	$ct \leftarrow 0;$
	for i$= 1, \dots, T_2$ do
	  $r \xleftarrow{\$} \mathit{flip};$
	  if $r =$ heads then
	    $ct \leftarrow ct + 1$

Step 2: First loop

- Use sampling rule with identity coupling: $r_1 = r_2$
- Establish loop invariant $ct_1 = ct_2$

Proof sketch

```
for i= $T_2 + 1, \ldots, T_1$ do
    $r \xleftarrow{\$} flip$;
    if $r$ = heads then
        $ct \leftarrow ct + 1$;
return($ct$)
```

return(ct)

Step 3: Second loop

- Use “one-sided” sampling rule
- Apply “one-sided” loop rule to show invariant $ct_1 \geq ct_2$

Formal Proofs by Coupling

Ex. 3: Uniformity

Simulating a fair coin flip from a biased coin

Problem setting

- Given: ability to draw biased coin flips $\mathit{flip}(p)$, $p \neq 1/2$
- Goal: simulate a fair coin flip $\mathit{flip}(1/2)$

Simulating a fair coin flip from a biased coin

Problem setting

- Given: ability to draw biased coin flips $\mathit{flip}(p)$, $p \neq 1/2$
- Goal: simulate a fair coin flip $\mathit{flip}(1/2)$

Algorithm ("von Neumann's trick")

$x \leftarrow \mathit{true}; y \leftarrow \mathit{true};$	// initialize $x = y$
while $x = y$ do	// if equal, repeat
$\quad x \xleftarrow{\$} \mathit{flip}(p);$	// flip biased coin
$\quad y \xleftarrow{\$} \mathit{flip}(p);$	// flip biased coin
return(x)	// if not equal, return x

Simulating a fair coin flip from a biased coin

Problem setting

- Given: ability to draw biased coin flips $\mathit{flip}(p)$, $p \neq 1/2$
- Goal: simulate a fair coin flip $\mathit{flip}(1/2)$

Algorithm ("von Neumann's trick")

$x \leftarrow \mathit{true}; y \leftarrow \mathit{true};$	// initialize $x = y$
while $x = y$ do	// if equal, repeat
$\quad x \xleftarrow{\$} \mathit{flip}(p);$	// flip biased coin
$\quad y \xleftarrow{\$} \mathit{flip}(p);$	// flip biased coin
return(x)	// if not equal, return x

How to prove that the result x is unbiased (uniform)?

From existence of coupling, to uniformity

Suppose that we know there exist two couplings:

1. Under first coupling, $x_1 = \mathit{true}$ implies $x_2 = \mathit{false}$
2. Under second coupling, $x_1 = \mathit{false}$ implies $x_2 = \mathit{true}$

From existence of coupling, to uniformity

Suppose that we know there exist two couplings:

1. Under first coupling, $x_1 = \mathit{true}$ implies $x_2 = \mathit{false}$
2. Under second coupling, $x_1 = \mathit{false}$ implies $x_2 = \mathit{true}$

As a consequence:

- By (1), $\Pr[x_1 = \mathit{true}] \leq \Pr[x_2 = \mathit{false}]$
- By (2), $\Pr[x_1 = \mathit{false}] \leq \Pr[x_2 = \mathit{true}]$

From existence of coupling, to uniformity

Suppose that we know there exist two couplings:

1. Under first coupling, $x_1 = \mathit{true}$ implies $x_2 = \mathit{false}$
2. Under second coupling, $x_1 = \mathit{false}$ implies $x_2 = \mathit{true}$

As a consequence:

- By (1), $\Pr[x_1 = \mathit{true}] \leq \Pr[x_2 = \mathit{false}]$
- By (2), $\Pr[x_1 = \mathit{false}] \leq \Pr[x_2 = \mathit{true}]$

But x_1 and x_2 have same distribution

- By (1), $\Pr[x_1 = \mathit{true}] \leq \Pr[x_1 = \mathit{false}]$
- By (2), $\Pr[x_1 = \mathit{false}] \leq \Pr[x_1 = \mathit{true}]$
- Hence uniform: $\Pr[x_1 = \mathit{true}] = \Pr[x_1 = \mathit{false}]$

Proof by coupling

Algorithm (“von Neumann’s trick”)

```
x ← true; y ← true;          // initialize x = y
while x = y do               // if equal, repeat
    x ←$ flip(p);            // flip biased coin
    y ←$ flip(p);            // flip biased coin
return(x)                    // if not equal, return x
```

Construct couplings such that:

1. Under first coupling, $x_1 = \mathit{true}$ implies $x_2 = \mathit{false}$
2. Under second coupling, $x_1 = \mathit{false}$ implies $x_2 = \mathit{true}$

Consider the following coupling:

- Couple sampling of x_1 to be equal to sampling of y_2
- Couple sampling of x_2 to be equal to sampling of y_1
- Resulting coupling satisfies both (1) and (2)!

Formalizing the proof in pRHL

Relate two (equivalent) versions of the program:

$x \leftarrow true; y \leftarrow true;$	$x \leftarrow true; y \leftarrow true;$
while $x = y$ do	while $x = y$ do
$\quad x \xleftarrow{\$} \mathit{flip}(p);$	$\quad y \xleftarrow{\$} \mathit{flip}(p);$
$\quad y \xleftarrow{\$} \mathit{flip}(p);$	$\quad x \xleftarrow{\$} \mathit{flip}(p);$
return(x)	return(x)

Formalizing the proof in pRHL

Relate two (equivalent) versions of the program:

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

Build coupling for loop bodies, then loops

- Use sampling rule with identity coupling: $x_1 = y_2$

Formalizing the proof in pRHL

Relate two (equivalent) versions of the program:

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

Build coupling for loop bodies, then loops

- Use sampling rule with identity coupling: $x_1 = y_2$

Formalizing the proof in pRHL

Relate two (equivalent) versions of the program:

$$
\begin{array}{l|l}
\begin{array}{l}
x \leftarrow \mathit{true}; y \leftarrow \mathit{true}; \\
\text{while } x = y \text{ do} \\
\quad x \xleftarrow{\$} \mathit{flip}(p); \\
\quad y \xleftarrow{\$} \mathit{flip}(p); \\
\text{return}(x)
\end{array}
&
\begin{array}{l}
x \leftarrow \mathit{true}; y \leftarrow \mathit{true}; \\
\text{while } x = y \text{ do} \\
\quad y \xleftarrow{\$} \mathit{flip}(p); \\
\quad x \xleftarrow{\$} \mathit{flip}(p); \\
\text{return}(x)
\end{array}
\end{array}
$$

Build coupling for loop bodies, then loops

- Use sampling rule with identity coupling: $x_1 = y_2$
- Use sampling rule with identity coupling: $y_1 = x_2$

Formalizing the proof in pRHL

Relate two (equivalent) versions of the program:

$x \leftarrow \mathit{true}; y \leftarrow \mathit{true};$	$x \leftarrow \mathit{true}; y \leftarrow \mathit{true};$
while $x = y$ do	while $x = y$ do
$\quad x \xleftarrow{\$} \mathit{flip}(p);$	$\quad y \xleftarrow{\$} \mathit{flip}(p);$
$\quad y \xleftarrow{\$} \mathit{flip}(p);$	$\quad x \xleftarrow{\$} \mathit{flip}(p);$
return(x)	return(x)

Build coupling for loop bodies, then loops

- Use sampling rule with identity coupling: $x_1 = y_2$
- Use sampling rule with identity coupling: $y_1 = x_2$

Formalizing the proof in pRHL

Relate two (equivalent) versions of the program:

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

Build coupling for loop bodies, then loops

- Use sampling rule with identity coupling: $x_1 = y_2$
- Use sampling rule with identity coupling: $y_1 = x_2$
- Use loop rule with invariant:

$$(x_1 = y_1 \rightarrow x_1 = y_2) \wedge (x_1 \neq y_1 \rightarrow x_1 \neq x_2)$$

Formalizing the proof in pRHL

Relate two (equivalent) versions of the program:

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

Build coupling for loop bodies, then loops

- Use sampling rule with identity coupling: $x_1 = y_2$
- Use sampling rule with identity coupling: $y_1 = x_2$
- Use loop rule with invariant:

$$(x_1 = y_1 \to x_1 = y_2) \wedge (x_1 \neq y_1 \to x_1 \neq x_2)$$

Wrapping Up

Variations and extensions

Approximate couplings

- Prove differential privacy as approximate equivalence
- Coming up next in Marco's tutorial!

Expectation couplings

- Prove quantitative bounds on distance b/t distributions
- MC convergence, stability of ML, path coupling, ...
- Program logic: https://arxiv.org/abs/1708.02537
- Pre-expectation calculus: https://arxiv.org/abs/1901.06540

Automation

- Encode search for coupling proofs as a synthesis problem
- Coupling proofs: https://arxiv.org/abs/1804.04052
- Approximate couplings: https://arxiv.org/abs/1709.05361

References

Relational reasoning via probabilistic coupling

- Initial connection between couplings and pRHL (LPAR 2015)
- arXiv: https://arxiv.org/abs/1509.03476

Coupling proofs are probabilistic product programs

- Extract product programs from pRHL proofs (POPL 2016)
- arXiv: https://arxiv.org/abs/1607.03455

Proving uniformity and independence by self-composition and coupling

- Coupling proofs for non-relational properties (LPAR 2017)
- arXiv: https://arxiv.org/abs/1701.06477

Verifying Probabilistic Properties with Probabilistic Couplings

Justin Hsu
UW–Madison
Computer Sciences