Language-based Cryptographic Proofs in Coq
or
Coq for Probabilistic Programs

FEDERICO OLMEDO
UNIVERSITY OF CHILE

ICSEC KICK-OFF WORKSHOP
SANTIAGO, CHILE — MARCH 2018

Motivation

Why certified cryptographic proofs?

Rigor crisis **in the cryptographic community**

In our opinion, many ***proofs in cryptography have become essentially unverifiable****. Our field may be approaching a crisis of rigor.*

Bellare & Rogaway (2006)

Do we have a problem with cryptographic proofs?
Yes, we do. The problem is that as a community, ***we generate more proofs than we carefully verify*** *(and as a consequence some of our published proofs are incorrect).*

Halevi (2005)

The rigor crisis of the cryptographic community

The case of OAEP encryption scheme

1994 — Introduction and security proof

Since 1994 — Worldwide industrial standard

Standard

- PKCS#1 v2
- IEEE P1363
- ISO 18033-2

The rigor crisis of the cryptographic community

The case of OAEP encryption scheme

1994 — Introduction and security proof

Since 1994 — Worldwide industrial standard

Standard

PKCS#1 v2
IEEE P1363
ISO 18033-2

2001 — Security proof is flawed

And 7 years later…

> ***There appears to be a <u>non-trivial gap in the OAEP security proof</u> [and] this gap cannot be filled.***
>
> **Shoup (2001)**

The rigor crisis of the cryptographic community

The case of BONEH-FRANKLIN encryption scheme

2001 — Introduction and security proof

Since 2001 — Used as subcomponent of several cryptographic protocols

The rigor crisis of the cryptographic community

The case of BONEH-FRANKLIN encryption scheme

2001 — Introduction and security proof

Since 2001 — Used as subcomponent of several cryptographic protocols

2005 — Security proof is flawed

> ***This is just another example in which a well-known and widely used construction turns out to have an unnoticed flawed security reduction.***
>
> ***Galindo (2005)***

CertiCrypt:

Framework for constructing certified cryptographic proofs in Coq

http://certicrypt.gforge.inria.fr/

CertiCrypt:
Framework for constructing certified cryptographic proofs in Coq

http://certicrypt.gforge.inria.fr/

Substantial effort

- 30.000 lines
- 4-6 years
- 6 people

CertiCrypt:
Framework for constructing certified cryptographic proofs in Coq

http://certicrypt.gforge.inria.fr/

Substantial effort

- 30.000 lines
- 4-6 years
- 6 people

High impact

- Formalization of several encryption schemes, digital signatures, hash functions, zero-knowledge protocols, etc
- 12 publications

Basics about CertiCrypt

What is a secure cryptographic scheme?

What is a secure cryptographic scheme?

A cryptographic scheme is secure if an efficient adversary can break it only with negligible probability

What is a secure cryptographic scheme?

A cryptographic scheme is secure if an efficient adversary can break it only with negligible *probability*

What is a secure cryptographic scheme?

A cryptographic scheme is secure if an efficient adversary can break it only with negligible *probability*

➡ Cryptographic schemes *must* be probabilistic (Goldwasser & Micali, '82)

What is a secure cryptographic scheme?

A cryptographic scheme is secure if an *efficient* adversary can break it only with negligible *probability*

➡ Cryptographic schemes *must* be probabilistic (Goldwasser & Micali, '82)

What is a secure cryptographic scheme?

A cryptographic scheme is secure if an *efficient* adversary can break it only with negligible *probability*

- Cryptographic schemes *must* be probabilistic (Goldwasser & Micali, '82)
- Adversaries should run in *probabilistic polynomial time* (PPT)

What is a secure cryptographic scheme?

A cryptographic scheme is secure if an *efficient* adversary can *break* it only with negligible *probability*

- Cryptographic schemes *must* be probabilistic (Goldwasser & Micali, ’82)

- Adversaries should run in *probabilistic polynomial time* (PPT)

What is a secure cryptographic scheme?

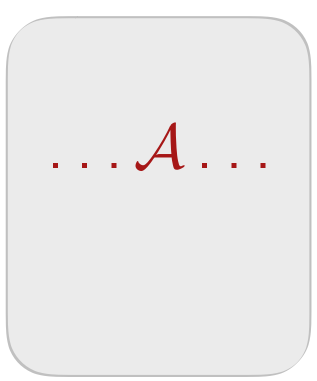
A cryptographic scheme is secure if an *efficient* adversary can *break* it only with negligible *probability*

- Cryptographic schemes *must* be probabilistic (Goldwasser & Micali, ’82)
- Adversaries should run in *probabilistic polynomial time* (PPT)
- There exists a *standard security notion* for each kind of cryptographic scheme

What is a secure cryptographic scheme?

A cryptographic scheme is secure if an *efficient* adversary can *break* it only with negligible *probability*

- Cryptographic schemes *must* be probabilistic (Goldwasser & Micali, '82)
- Adversaries should run in *probabilistic polynomial time* (PPT)
- There exists a *standard security notion* for each kind of cryptographic scheme



Attack game

What is a secure cryptographic scheme?

A cryptographic scheme is secure if an *efficient* adversary can *break* it only with negligible *probability*

- Cryptographic schemes *must* be probabilistic (Goldwasser & Micali, ’82)
- Adversaries should run in *probabilistic polynomial time* (PPT)
- There exists a *standard security notion* for each kind of cryptographic scheme

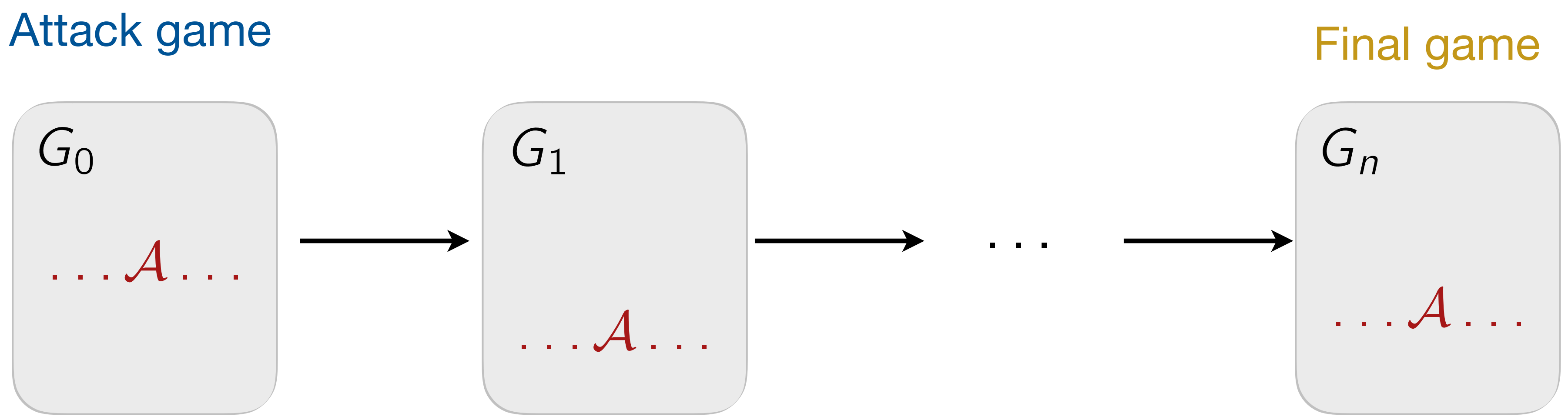
$\ldots\mathcal{A}\ldots$

Attack game

$$\Pr\left[\begin{array}{l}\mathcal{A}\text{ breaks}\\ \text{the scheme}\end{array}\right] \leq \epsilon$$

How do security proof proceed?

By stepwise transformation of the attack game, towards a "simpler" game

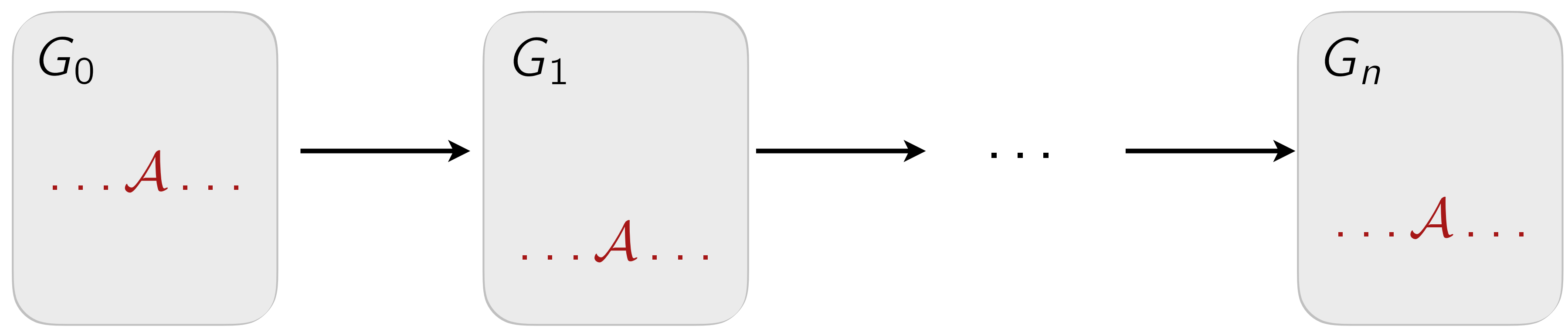


How do security proof proceed?

By stepwise transformation of the attack game, towards a "simpler" game

Attack game

Final game

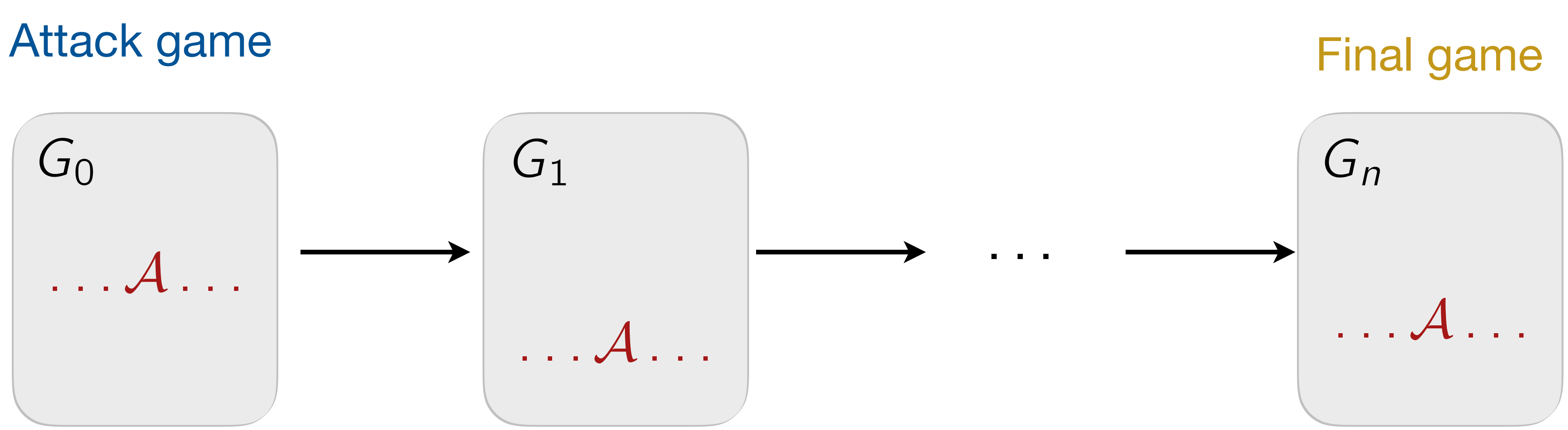


$$\Pr_{G_0}[E_0] \quad \leq \quad f_1\left(\Pr_{G_1}[E_1]\right)$$

Probability of breaking the scheme

How do security proof proceed?

By stepwise transformation of the attack game, towards a "simpler" game

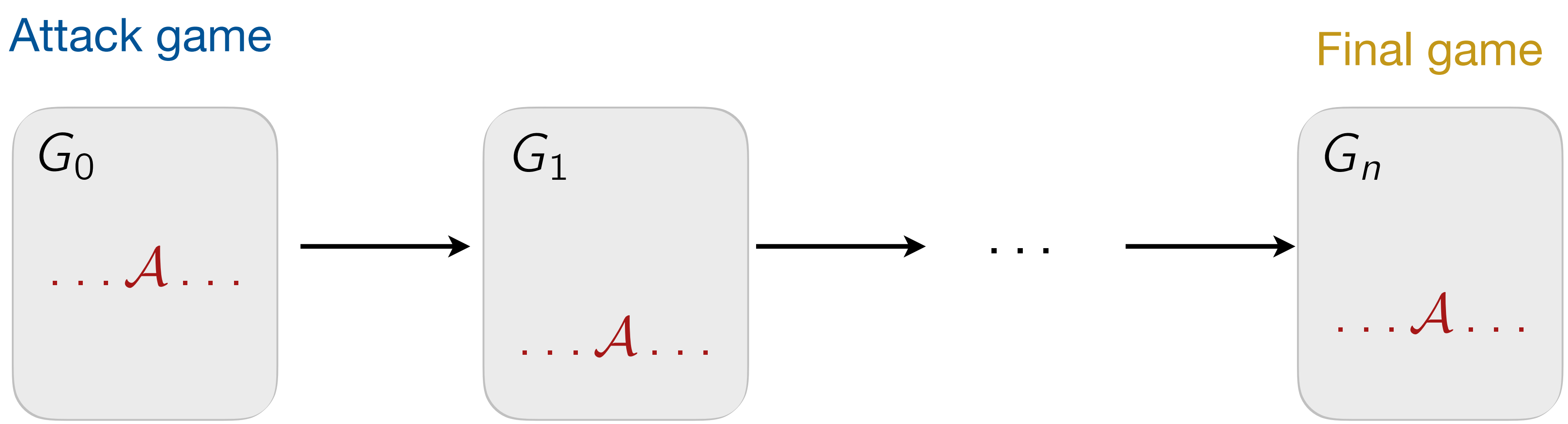


$$\Pr_{G_0}[E_0] \leq f_1\left(\Pr_{G_1}[E_1]\right) \leq \cdots \leq f_n\left(\Pr_{G_n}[E_n]\right)$$

Probability of breaking the scheme

How do security proof proceed?

By stepwise transformation of the attack game, towards a "simpler" game



$$\Pr_{G_0}[E_0] \quad \leq \quad f_1\left(\Pr_{G_1}[E_1]\right) \quad \leq \quad \cdots \quad \leq \quad f_n\left(\Pr_{G_n}[E_n]\right)$$

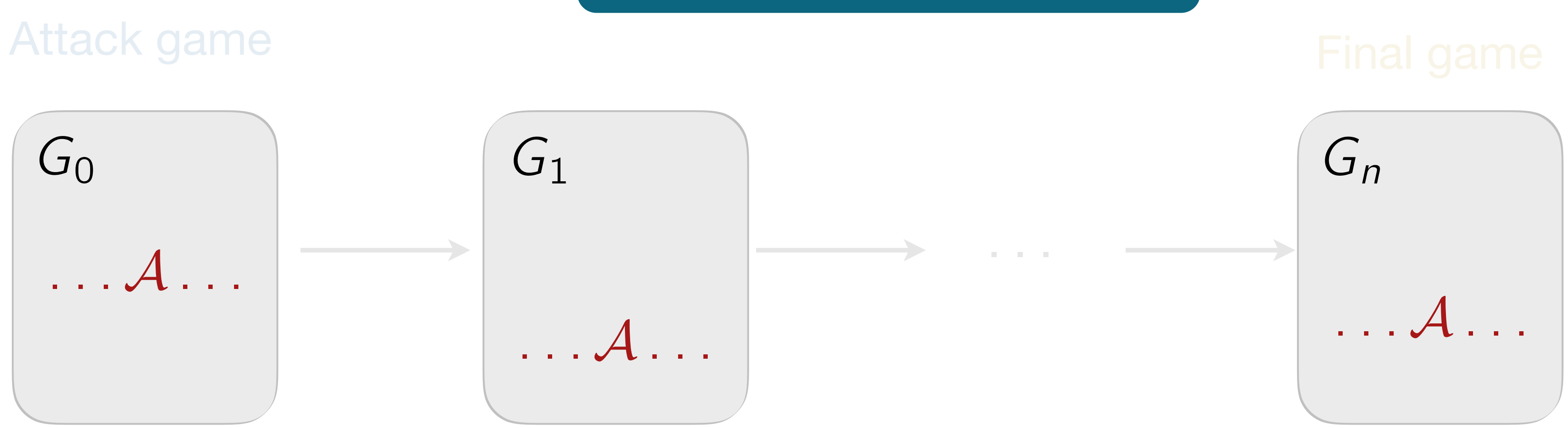
Probability of breaking the scheme

$$\Pr_{G_0}[E_0] \quad \leq \quad f\left(\Pr_{G_n}[E_n]\right) \quad \leq \quad \epsilon$$

How do security proof proceed?

By stepwise transformation of the attack game, towards a “simpler” game

How do we represent games?



$$\Pr_{G_0}[E_0] \quad \leq \quad f_1\left(\Pr_{G_1}[E_1]\right) \quad \leq \quad \cdots \quad \leq \quad f_n\left(\Pr_{G_n}[E_n]\right)$$

Probability of breaking the scheme

$$\Pr_{G_0}[E_0] \quad \leq \quad f\left(\Pr_{G_n}[E_n]\right) \quad \leq \quad \epsilon$$

How do security proof proceed?

By stepwise transformation of the attack game, towards a "simpler" game

$$\Pr_{G_0}[E_0] \quad \leq \quad f_1\left(\Pr_{G_1}[E_1]\right) \quad \leq \quad \cdots \quad \leq \quad f_n\left(\Pr_{G_n}[E_n]\right)$$

Probability of breaking the scheme

How do we relate the probabilities of events between consecutive games?

Language-based cryptographic proofs

Language-based cryptographic proofs

Games $\Longrightarrow$ (probabilistic) programs

Language-based cryptographic proofs

Games	$\Longrightarrow$	(probabilistic) programs
Probability space	$\Longrightarrow$	
Probability of event	$\Longrightarrow$	
Game transformations	$\Longrightarrow$	
Generic adversary	$\Longrightarrow$	

Language-based cryptographic proofs

Games	$\Longrightarrow$	(probabilistic) programs
Probability space	$\Longrightarrow$	program denotation
Probability of event	$\Longrightarrow$	
Game transformations	$\Longrightarrow$	
Generic adversary	$\Longrightarrow$	

Language-based cryptographic proofs

Games	$\Longrightarrow$	(probabilistic) programs
Probability space	$\Longrightarrow$	program denotation
Probability of event	$\Longrightarrow$	probability of postcondition
Game transformations	$\Longrightarrow$	
Generic adversary	$\Longrightarrow$	

Language-based cryptographic proofs

Games	$\Longrightarrow$	(probabilistic) programs
Probability space	$\Longrightarrow$	program denotation
Probability of event	$\Longrightarrow$	probability of postcondition
Game transformations	$\Longrightarrow$	program transformations
Generic adversary	$\Longrightarrow$	

Language-based cryptographic proofs

Games	$\Longrightarrow$	(probabilistic) programs
Probability space	$\Longrightarrow$	program denotation
Probability of event	$\Longrightarrow$	probability of postcondition
Game transformations	$\Longrightarrow$	program transformations
Generic adversary	$\Longrightarrow$	unspecified procedure

The probabilistic language

$$
\begin{array}{rcll}
\mathcal{C} & ::= & \mathsf{skip} & \text{nop} \\
& \mid & \mathcal{C};\ \mathcal{C} & \text{sequence} \\
& \mid & \mathcal{V} \leftarrow \mathcal{E} & \text{assignment} \\
& \mid & \mathcal{V} \xleftarrow{\$} \mathcal{D}\mathcal{E} & \text{random sampling} \\
& \mid & \mathsf{if}\ \mathcal{E}\ \mathsf{then}\ \mathcal{C}\ \mathsf{else}\ \mathcal{C} & \text{conditional} \\
& \mid & \mathsf{while}\ \mathcal{E}\ \mathsf{do}\ \mathcal{C} & \text{while loop} \\
& \mid & \mathcal{V} \leftarrow \mathcal{P}(\mathcal{E}, \ldots, \mathcal{E}) & \text{procedure call}
\end{array}
$$

The probabilistic language

$\mathcal{C}$	$::=$	skip	nop
	$\mid$	$\mathcal{C};\ \mathcal{C}$	sequence
	$\mid$	$\mathcal{V} \leftarrow \mathcal{E}$	assignment
	$\mid$	$\mathcal{V} \stackrel{\$}{\leftarrow} \mathcal{D}\mathcal{E}$	random sampling
	$\mid$	if $\mathcal{E}$ then $\mathcal{C}$ else $\mathcal{C}$	conditional
	$\mid$	while $\mathcal{E}$ do $\mathcal{C}$	while loop
	$\mid$	$\mathcal{V} \leftarrow \mathcal{P}(\mathcal{E}, \ldots, \mathcal{E})$	procedure call

$$[\![c]\!] \ : \ \mathbb{S} \to \mathcal{D}(\mathbb{S})$$

The probabilistic language

$\mathcal{C}$	$::=$	skip	nop
	$\mid$	$\mathcal{C};\ \mathcal{C}$	sequence
	$\mid$	$\mathcal{V} \leftarrow \mathcal{E}$	assignment
	$\mid$	$\mathcal{V} \xleftarrow{\$} \mathcal{D}\mathcal{E}$	random sampling
	$\mid$	if $\mathcal{E}$ then $\mathcal{C}$ else $\mathcal{C}$	conditional
	$\mid$	while $\mathcal{E}$ do $\mathcal{C}$	while loop
	$\mid$	$\mathcal{V} \leftarrow \mathcal{P}(\mathcal{E}, \ldots, \mathcal{E})$	procedure call

$$[\![c]\!] \ : \ \forall (k : \mathbb{N}).\ \mathbb{S}_k \to \mathcal{D}(\mathbb{S}_k)$$

security parameter

How do we relate the probability of program?

How do we relate the probability of program?

We need to prove claims of the form

$$\Pr_{c_1(s_1)}[E_1] \quad \leq \quad f\big(\Pr_{c_2(s_2)}[E_2]\big)$$

How do we relate the probability of program?

We need to prove claims of the form

$$\Pr_{c_1(s_1)}[E_1] \quad \leq \quad f\big(\Pr_{c_2(s_2)}[E_2]\big)$$

But usually, it suffices proving claims of the form

$$\Pr_{c_1(s_1)}[E] \quad = \quad \Pr_{c_2(s_2)}[E]$$

How do we relate the probability of program?

We need to prove claims of the form

$$\Pr_{c_1(s_1)}[E_1] \quad \leq \quad f\big(\Pr_{c_2(s_2)}[E_2]\big)$$

But usually, it suffices proving claims of the form

$$\Pr_{c_1(s_1)}[E] \quad = \quad \Pr_{c_2(s_2)}[E]$$

for which we can rely on **observational equivalence** between programs:

$$\{I\}\ c_1 \ \sim\ c_2\ \{O\}$$

How do we relate the probability of program?

We need to prove claims of the form

$$\Pr_{c_1(s_1)}[E_1] \quad \leq \quad f\big(\Pr_{c_2(s_2)}[E_2]\big)$$

But usually, it suffices proving claims of the form

$$\Pr_{c_1(s_1)}[E] \quad = \quad \Pr_{c_2(s_2)}[E]$$

for which we can rely on **observational equivalence** between programs:

Input set of variables

$$\{I\}\ c_1 \ \sim\ c_2\ \{O\}$$

How do we relate the probability of program?

We need to prove claims of the form

$$\Pr_{c_1(s_1)}[E_1] \quad \leq \quad f\big(\Pr_{c_2(s_2)}[E_2]\big)$$

But usually, it suffices proving claims of the form

$$\Pr_{c_1(s_1)}[E] \quad = \quad \Pr_{c_2(s_2)}[E]$$

for which we can rely on **observational equivalence** between programs:

Input set of variables

Output set of variables

$$\{I\}\ c_1 \ \sim \ c_2\ \{O\}$$

How do we relate the probability of program?

We need to prove claims of the form

$$\Pr_{c_1(s_1)}[E_1] \quad \leq \quad f\big(\Pr_{c_2(s_2)}[E_2]\big)$$

But usually, it suffices proving claims of the form

$$\Pr_{c_1(s_1)}[E] \quad = \quad \Pr_{c_2(s_2)}[E]$$

for which we can rely on **observational equivalence** between programs:

Input set of variables

Output set of variables

$$\frac{\{I\}\ c_1 \ \sim\ c_2\ \{O\}}{\Pr_{c_1(s_1)}[E] \ = \ \Pr_{c_2(s_2)}[E]}$$

How do we relate the probability of program?

We need to prove claims of the form

$$\Pr_{c_1(s_1)}[E_1] \quad \leq \quad f\big(\Pr_{c_2(s_2)}[E_2]\big)$$

But usually, it suffices proving claims of the form

$$\Pr_{c_1(s_1)}[E] \quad = \quad \Pr_{c_2(s_2)}[E]$$

for which we can rely on **observational equivalence** between programs:

Input set of variables

Output set of variables

$$\frac{fv(E) \subseteq O \qquad \{I\}\ c_1 \ \sim\ c_2\ \{O\}}{\Pr_{c_1(s_1)}[E] \ = \ \Pr_{c_2(s_2)}[E]}$$

How do we relate the probability of program?

We need to prove claims of the form

$$\Pr_{c_1(s_1)}[E_1] \quad \leq \quad f\big(\Pr_{c_2(s_2)}[E_2]\big)$$

But usually, it suffices proving claims of the form

$$\Pr_{c_1(s_1)}[E] \quad = \quad \Pr_{c_2(s_2)}[E]$$

for which we can rely on **observational equivalence** between programs:

Input set of variables

Output set of variables

$$\frac{fv(E) \subseteq O \qquad \{I\}\ c_1 \ \sim\ c_2\ \{O\} \qquad s_1 =_I s_2}{\Pr_{c_1(s_1)}[E] \ = \ \Pr_{c_2(s_2)}[E]}$$

Proving observational equivalence

CertiCrypt provides several *mechanised program transformations* for establishing observational equivalence

Proving observational equivalence

CertiCrypt provides several *mechanised program transformations* for establishing observational equivalence

Program Transformation: $\mathcal{T}(c_1, c_2, I, O) = (c_1', c_2', I', O')$

Proving observational equivalence

CertiCrypt provides several *mechanised program transformations* for establishing observational equivalence

PROGRAM TRANSFORMATION: $\mathcal{T}(c_1, c_2, I, O) = (c_1', c_2', I', O')$

SOUNDNESS RESULT:

Proving observational equivalence

CertiCrypt provides several *mechanised program transformations* for establishing observational equivalence

PROGRAM TRANSFORMATION: $\mathcal{T}(c_1, c_2, I, O) = (c_1', c_2', I', O')$

SOUNDNESS RESULT:

$$\{I\}\ c_1 \sim c_2\ \{O\}$$

Proving observational equivalence

CertiCrypt provides several *mechanised program transformations* for establishing observational equivalence

PROGRAM TRANSFORMATION: $\mathcal{T}(c_1, c_2, I, O) = (c_1', c_2', I', O')$

SOUNDNESS RESULT:

$$\frac{\mathcal{T}(c_1, c_2, I, O) = (c_1', c_2', I', O')}{\{I\}\ c_1 \ \sim\ c_2\ \{O\}}$$

Proving observational equivalence

CertiCrypt provides several *mechanised program transformations* for establishing observational equivalence

PROGRAM TRANSFORMATION: $\mathcal{T}(c_1, c_2, I, O) = (c_1', c_2', I', O')$

SOUNDNESS RESULT:

$$\frac{\mathcal{T}(c_1, c_2, I, O) = (c_1', c_2', I', O') \qquad \{I'\}\ c_1' \ \sim\ c_2'\ \{O'\}}{\{I\}\ c_1 \ \sim\ c_2\ \{O\}}$$

Proving observational equivalence

CertiCrypt provides several *mechanised program transformations* for establishing observational equivalence

Program Transformation: $\mathcal{T}(c_1, c_2, I, O) = (c_1', c_2', I', O')$

Soundness Result:

$$\frac{\mathcal{T}(c_1, c_2, I, O) = (c_1', c_2', I', O') \qquad \{I'\}\ c_1' \sim c_2'\ \{O'\}}{\{I\}\ c_1 \sim c_2\ \{O\}}$$

Some Instances:

- Deadcode elimination
- Constant propagation
- Procedure call inlining
- Common prefix/suffix elimination

Proving observational equivalence

CertiCrypt provides an (incomplete) tactic for proving *self-equivalence*

Does $\{I\}\ c \ \sim\ c\ \{O\}$ hold?

Proving observational equivalence

CertiCrypt provides an (incomplete) tactic for proving *self-equivalence*

Does $\{I\}\ c \ \sim\ c\ \{O\}$ hold?

- Analyse dependencies to compute I' such that $\{I'\}\ c \ \sim\ c\ \{O\}$

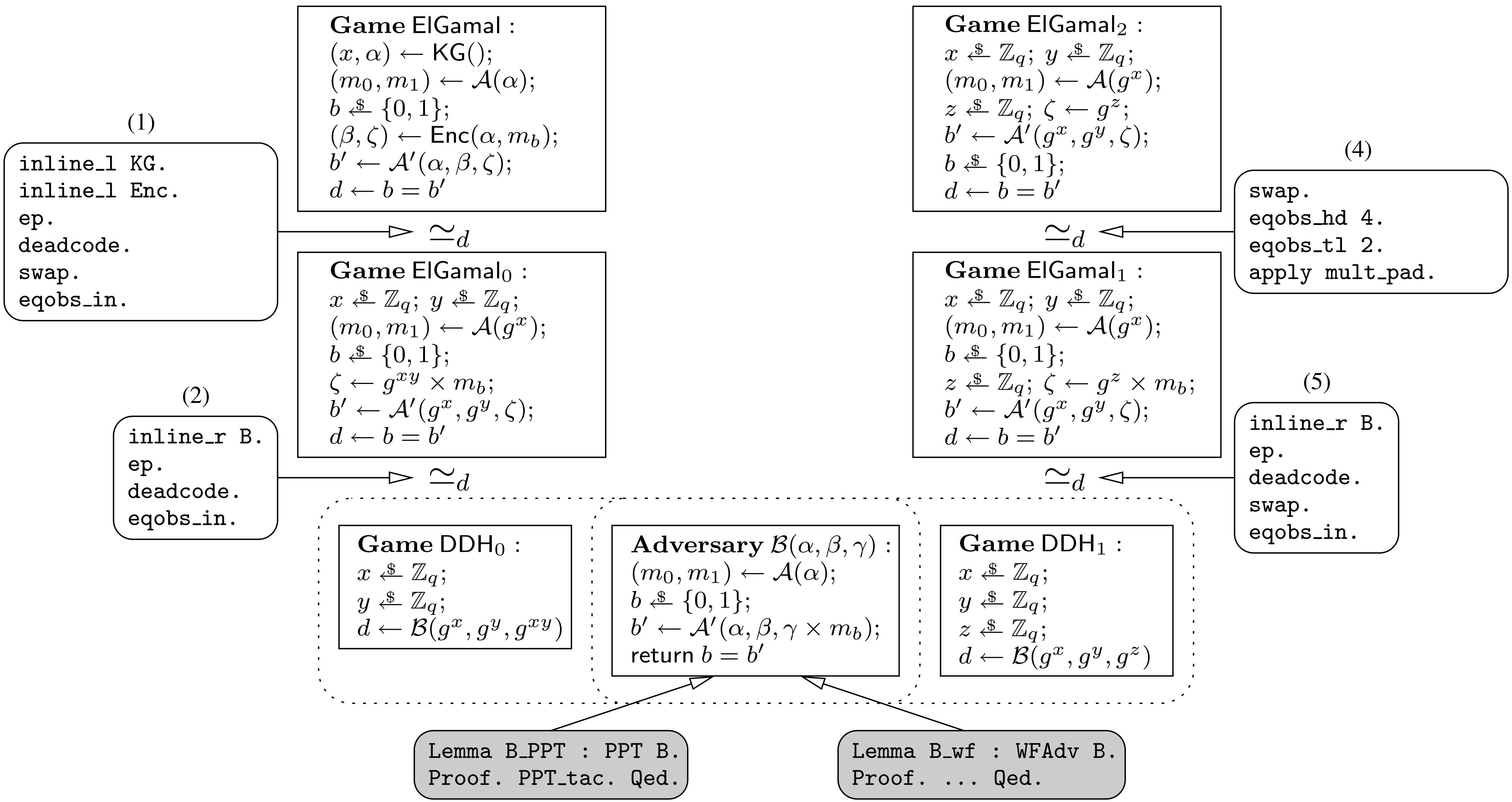
Proving observational equivalence

CertiCrypt provides an (incomplete) tactic for proving *self-equivalence*

Does $\{I\}\ c \ \sim\ c\ \{O\}$ hold?

- Analyse dependencies to compute I' such that $\{I'\}\ c \ \sim\ c\ \{O\}$
- Check that $I' \subseteq I$

Security proof of ElGamal encryption scheme



Security proof of ElGamal encryption scheme

Game $\mathsf{ElGamal}_2$:
$x \xleftarrow{\$} \mathbb{Z}_q;\ y \xleftarrow{\$} \mathbb{Z}_q;$
$(m_0, m_1) \leftarrow \mathcal{A}(g^x);$
$z \xleftarrow{\$} \mathbb{Z}_q;\ \zeta \leftarrow g^z;$
$b' \leftarrow \mathcal{A}'(g^x, g^y, \zeta);$
$b \xleftarrow{\$} \{0,1\};$
$d \leftarrow b = b'$

$\simeq_d$

```
swap.
eqobs_hd 4.
eqobs_tl 2.
apply mult_pad.
```

Game $\mathsf{ElGamal}_1$:
$x \xleftarrow{\$} \mathbb{Z}_q;\ y \xleftarrow{\$} \mathbb{Z}_q;$
$(m_0, m_1) \leftarrow \mathcal{A}(g^x);$
$b \xleftarrow{\$} \{0,1\};$
$z \xleftarrow{\$} \mathbb{Z}_q;\ \zeta \leftarrow g^z \times m_b;$
$b' \leftarrow \mathcal{A}'(g^x, g^y, \zeta);$
$d \leftarrow b = b'$

Observational equivalence is not enough

Observational equivalence is not enough

$$\frac{???}{\{x\}\ \ \textsf{if}\ (x{=}0)\ \textsf{then}\ y{\leftarrow}x\ \textsf{else}\ y{\leftarrow}1\ \ \sim\ \ \textsf{if}\ (x{=}0)\ \textsf{then}\ y{\leftarrow}0\ \textsf{else}\ y{\leftarrow}1\ \ \{x, y\}}$$

Observational equivalence is not enough

- Establishing observational equivalence may require additional contextual information

$$\frac{???}{\{x\}\ \ \textsf{if}\ (x{=}0)\ \textsf{then}\ y{\leftarrow}x\ \textsf{else}\ y{\leftarrow}1\ \ \sim\ \ \textsf{if}\ (x{=}0)\ \textsf{then}\ y{\leftarrow}0\ \textsf{else}\ y{\leftarrow}1\ \ \{x,y\}}$$

Observational equivalence is not enough

- Establishing observational equivalence may require additional contextual information

$$\frac{???}{\{x\}\ \ \text{if } (x{=}0) \text{ then } y \leftarrow x \text{ else } y \leftarrow 1 \ \sim\ \text{if } (x{=}0) \text{ then } y \leftarrow 0 \text{ else } y \leftarrow 1\ \ \{x, y\}}$$

- Cryptographic proofs may involve weaker relationships between consecutive games, e.g.

$$\Pr_{c_1(s_1)}[E_1] \;\leq\; \Pr_{c_2(s_2)}[E_2]$$

Relational Hoare logic

Relational Hoare logic

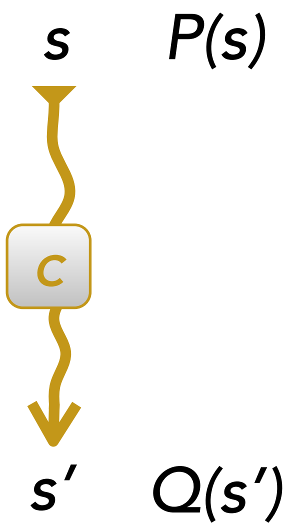
Standard Hoare Logic (HL)

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

Relational Hoare logic

Standard Hoare Logic (HL)

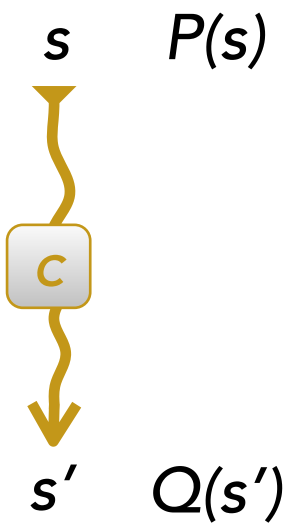
$$\{P\}\ c\ \{Q\}$$



Relational Hoare logic

Standard Hoare Logic (HL)

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



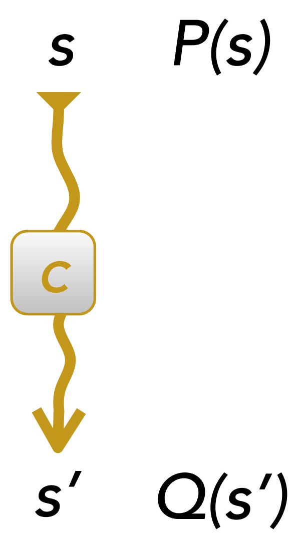
Relational Hoare Logic (RHL)

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

Relational Hoare logic

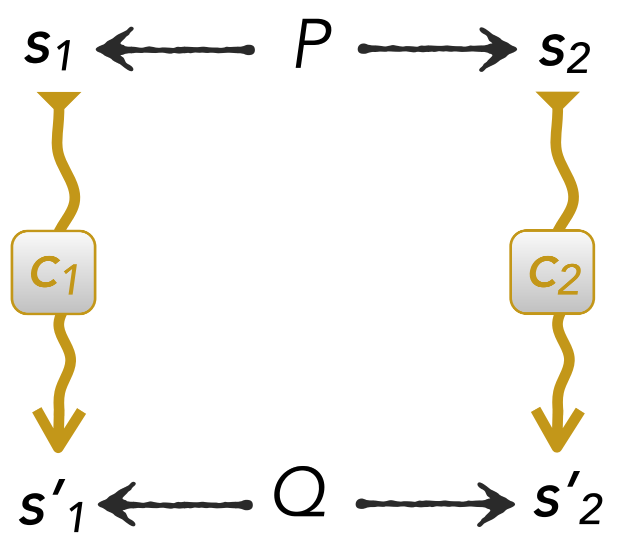
Standard Hoare Logic (HL)

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



Relational Hoare Logic (RHL)

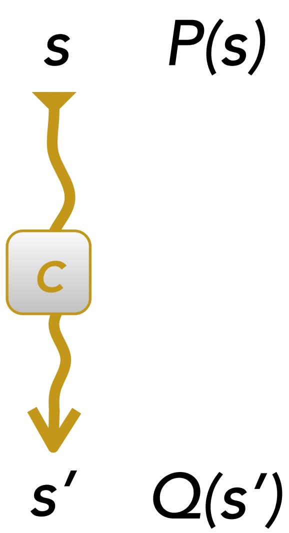
$$\{P\}\ c_1 \sim c_2\ \{Q\}$$



Relational Hoare logic

Standard Hoare Logic (HL)

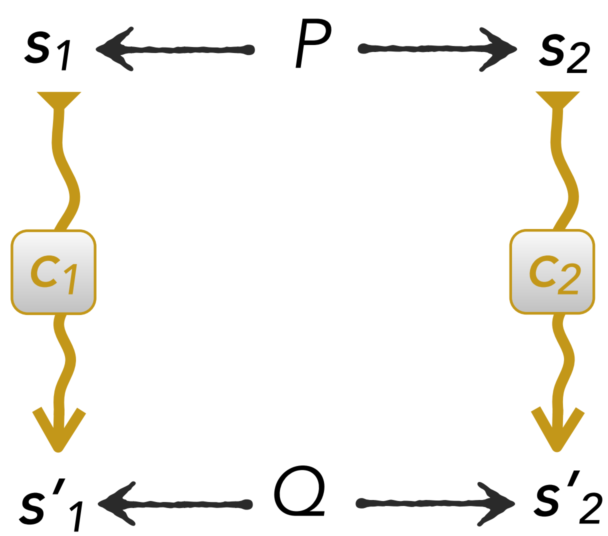
$$\{P\}\ c\ \{Q\}$$



Relational Hoare Logic (RHL)

probabilistic programs

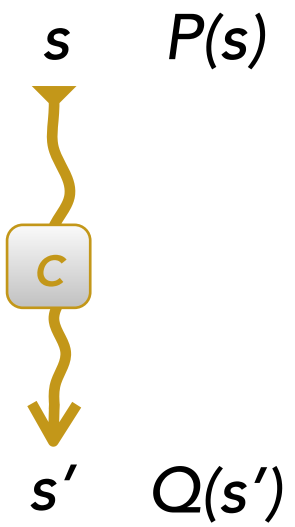
$$\{P\}\ c_1 \sim c_2\ \{Q\}$$



Relational Hoare logic

Standard Hoare Logic (HL)

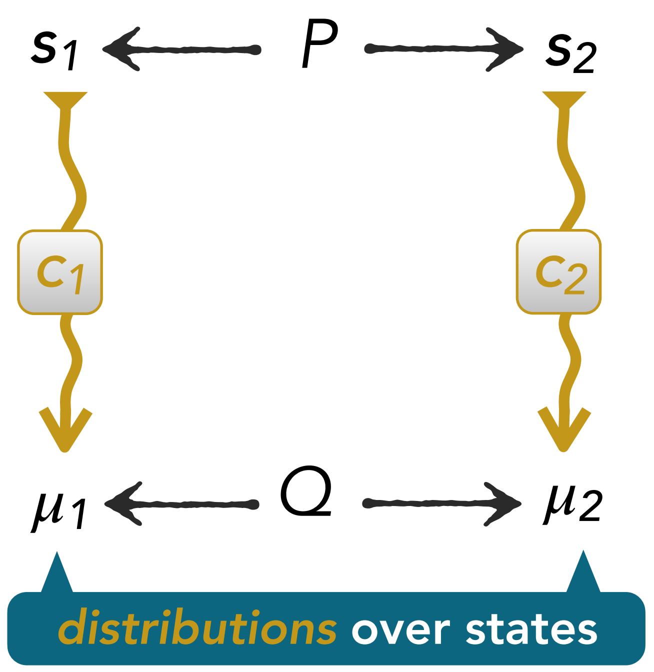
$$\{P\}\ c\ \{Q\}$$



Relational Hoare Logic (RHL)

probabilistic programs

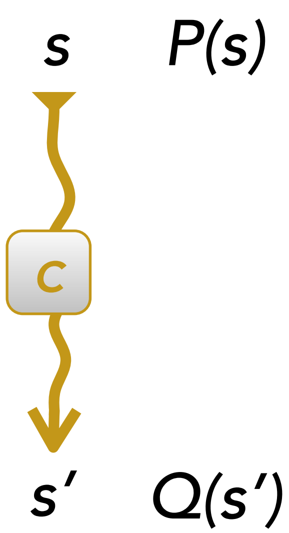
$$\{P\}\ c_1 \sim c_2\ \{Q\}$$



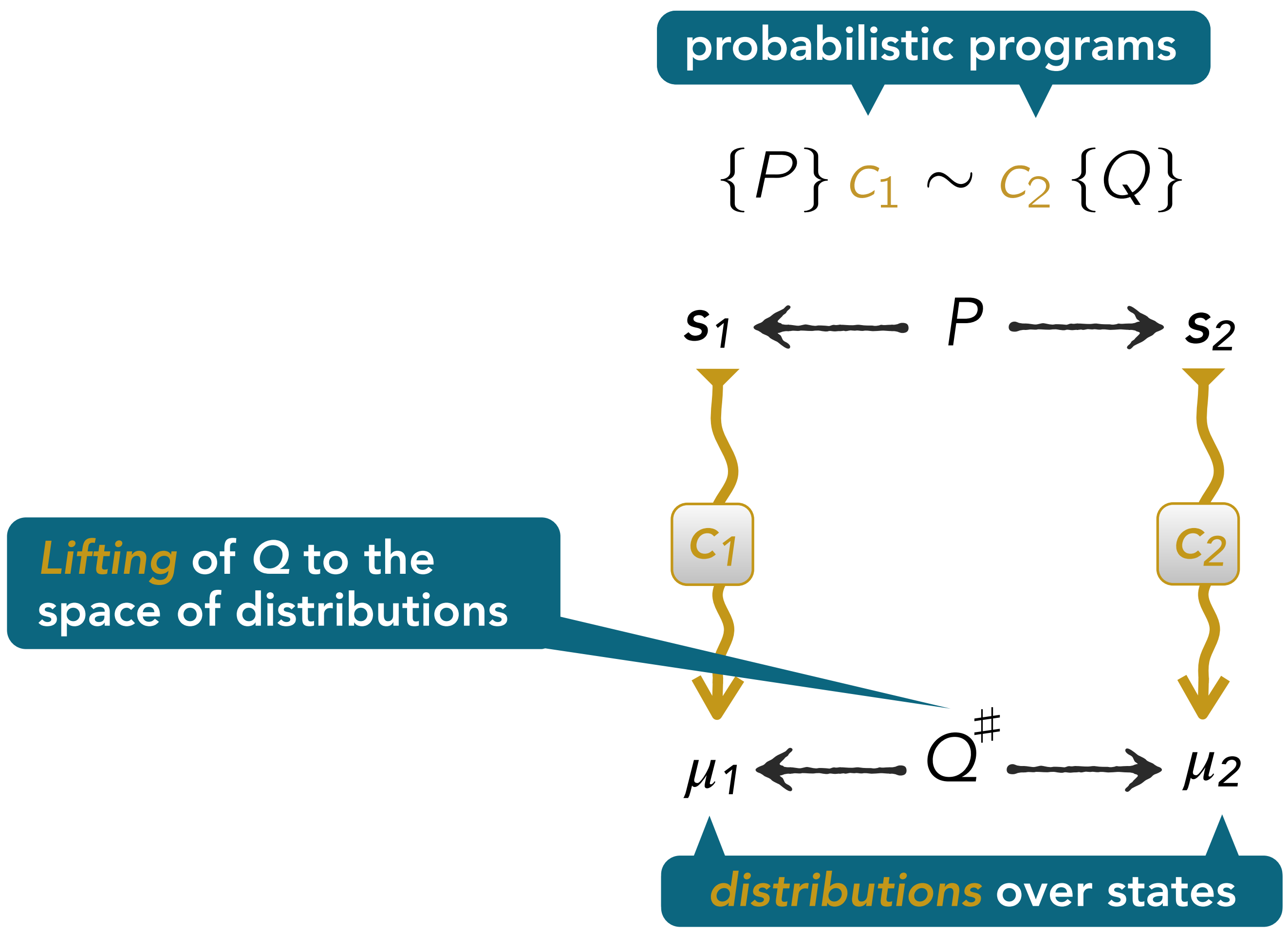
Relational Hoare logic

Standard Hoare Logic (HL)

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



Relational Hoare Logic (RHL)



Relational Hoare logic — Judgment examples

- $$z := y{+}1 \;\sim\; z := x$$

Relational Hoare logic — Judgment examples

- $\models \{y\langle 1\rangle + 1 = x\langle 2\rangle\}\ \ z := y+1 \ \sim\ z := x$

Relational Hoare logic — Judgment examples

- $\models \{y\langle 1\rangle{+}1 = x\langle 2\rangle\}\ z := y{+}1 \ \sim\ z := x\ \{z\langle 1\rangle = z\langle 2\rangle\}$

Relational Hoare logic — Judgment examples

- $\models \{y\langle 1\rangle{+}1 = x\langle 2\rangle\}\ \ z := y{+}1 \ \sim\ z := x \ \ \{z\langle 1\rangle = z\langle 2\rangle\}$

- $$\begin{array}{r}\texttt{if}\ b\ \texttt{then}\ x := 0\\ \texttt{else}\ x := 1\end{array} \sim \begin{array}{r}\texttt{if}\ b\ \texttt{then}\ x := 1\\ \texttt{else}\ x := 0\end{array}$$

Relational Hoare logic — Judgment examples

- $\models \{y\langle 1\rangle + 1 = x\langle 2\rangle\}\ z := y+1 \sim z := x\ \{z\langle 1\rangle = z\langle 2\rangle\}$

- $\models \{b\langle 1\rangle = b\langle 2\rangle\}\ \begin{array}{l}\texttt{if}\ b\ \texttt{then}\ x := 0\\ \quad\texttt{else}\ x := 1\end{array} \sim \begin{array}{l}\texttt{if}\ b\ \texttt{then}\ x := 1\\ \quad\texttt{else}\ x := 0\end{array}$

Relational Hoare logic — Judgment examples

- $\models \{y\langle 1\rangle{+}1 = x\langle 2\rangle\}\ z := y{+}1 \ \sim\ z := x\ \{z\langle 1\rangle = z\langle 2\rangle\}$

- $\models \{b\langle 1\rangle = b\langle 2\rangle\}\ \begin{array}{r}\texttt{if}\ b\ \texttt{then}\ x := 0\\ \texttt{else}\ x := 1\end{array} \ \sim\ \begin{array}{r}\texttt{if}\ b\ \texttt{then}\ x := 1\\ \texttt{else}\ x := 0\end{array}\ \{x\langle 1\rangle = 1 - x\langle 2\rangle\}$

Proof system

Proof system

- Most rules are direct adaptations of traditional HL rules

Proof system

- Most rules are direct adaptations of traditional HL rules

$$(\vdash \{P\}\, \texttt{skip}\, \{P\})$$

Proof system

- Most rules are direct adaptations of traditional HL rules

$$\vdash \{P\}\, \texttt{skip} \sim \texttt{skip}\, \{P\} \qquad (\vdash \{P\}\, \texttt{skip}\, \{P\})$$

Proof system

- Most rules are direct adaptations of traditional HL rules

$\vdash \{P\}\,\texttt{skip} \sim \texttt{skip}\,\{P\}$ $\qquad (\vdash \{P\}\,\texttt{skip}\,\{P\})$

$$\left(\frac{\vdash \{P\}\; c\; \{Q'\} \qquad \vdash \{Q'\}\; c'\; \{Q\}}{\vdash \{P\}\; c; c'\; \{Q\}}\right)$$

Proof system

- Most rules are direct adaptations of traditional HL rules

$$\vdash \{P\}\ \texttt{skip} \sim \texttt{skip}\ \{P\} \qquad (\vdash \{P\}\ \texttt{skip}\ \{P\})$$

$$\frac{\vdash \{P\}\ c_1 \sim c_2\ \{Q'\} \quad \vdash \{Q'\}\ c_1' \sim c_2'\ \{Q\}}{\vdash \{P\}\ c_1; c_1' \sim c_2; c_2'\ \{Q\}} \qquad \left(\frac{\vdash \{P\}\ c\ \{Q'\} \quad \vdash \{Q'\}\ c'\ \{Q\}}{\vdash \{P\}\ c; c'\ \{Q\}}\right)$$

Proof system

- Most rules are direct adaptations of traditional HL rules

$$\vdash \{P\}\, \texttt{skip} \sim \texttt{skip}\, \{P\} \qquad (\vdash \{P\}\, \texttt{skip}\, \{P\})$$

$$\frac{\vdash \{P\}\, c_1 \sim c_2\, \{Q'\} \quad \vdash \{Q'\}\, c_1' \sim c_2'\, \{Q\}}{\vdash \{P\}\, c_1; c_1' \sim c_2; c_2'\, \{Q\}} \qquad \left(\frac{\vdash \{P\}\, c\, \{Q'\} \quad \vdash \{Q'\}\, c'\, \{Q\}}{\vdash \{P\}\, c; c'\, \{Q\}}\right)$$

- Requires programs to execute lockstep

Proof system

- Most rules are direct adaptations of traditional HL rules

$$\vdash \{P\}\ \texttt{skip} \sim \texttt{skip}\ \{P\} \qquad (\vdash \{P\}\ \texttt{skip}\ \{P\})$$

$$\frac{\vdash \{P\}\ c_1 \sim c_2\ \{Q'\} \quad \vdash \{Q'\}\ c_1' \sim c_2'\ \{Q\}}{\vdash \{P\}\ c_1; c_1' \sim c_2; c_2'\ \{Q\}} \qquad \left(\frac{\vdash \{P\}\ c\ \{Q'\} \quad \vdash \{Q'\}\ c'\ \{Q\}}{\vdash \{P\}\ c; c'\ \{Q\}}\right)$$

- Requires programs to execute lockstep

$$\frac{\vdash \{I \wedge G_{1\langle 1\rangle}\}\ c_1 \sim c_2\ \{I\} \qquad \boxed{\models (I \Longrightarrow G_{1\langle 1\rangle} = G_{2\langle 2\rangle})}}{\vdash \{I\}\ \texttt{while}\ G_1\ \texttt{do}\ c_1 \sim \texttt{while}\ G_2\ \texttt{do}\ c_2\ \{I \wedge \neg G_{1\langle 1\rangle}\}}\,[\text{while}]$$

Proof system

- Most rules are direct adaptations of traditional HL rules

$$\vdash \{P\}\ \texttt{skip} \sim \texttt{skip}\ \{P\}$$

$$(\vdash \{P\}\ \texttt{skip}\ \{P\})$$

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

$$\left(\frac{\vdash \{P\}\ c\ \{Q'\} \quad \vdash \{Q'\}\ c'\ \{Q\}}{\vdash \{P\}\ c; c'\ \{Q\}}\right)$$

- Requires programs to execute lockstep

$$\frac{\vdash \{I \wedge G_{1\langle 1\rangle}\}\ c_1 \sim c_2\ \{I\} \qquad \boxed{\models (I \implies G_{1\langle 1\rangle} = G_{2\langle 2\rangle})}}{\vdash \{I\}\ \texttt{while}\ G_1\ \texttt{do}\ c_1 \sim \texttt{while}\ G_2\ \texttt{do}\ c_2\ \{I \wedge \neg G_{1\langle 1\rangle}\}}\ [\text{while}]$$

- (The classic fragment) only relates programs that are structurally equal.

Proof system

- Most rules are direct adaptations of traditional HL rules

$$\vdash \{P\}\ \texttt{skip} \sim \texttt{skip}\ \{P\}$$

$$(\vdash \{P\}\ \texttt{skip}\ \{P\})$$

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

$$\left(\frac{\vdash \{P\}\ c\ \{Q'\} \qquad \vdash \{Q'\}\ c'\ \{Q\}}{\vdash \{P\}\ c; c'\ \{Q\}}\right)$$

- Requires programs to execute lockstep

$$\frac{\vdash \{I \wedge G_{1\langle 1\rangle}\}\ c_1 \sim c_2\ \{I\} \qquad \boxed{\models (I \Longrightarrow G_{1\langle 1\rangle} = G_{2\langle 2\rangle})}}{\vdash \{I\}\ \texttt{while}\ G_1\ \texttt{do}\ c_1 \sim \texttt{while}\ G_2\ \texttt{do}\ c_2\ \{I \wedge \neg G_{1\langle 1\rangle}\}}\ \text{[while]}$$

- (The classic fragment) only relates programs that are structurally equal.
  But the logic can be extended with "one-sided" rules, e.g.

Proof system

- Most rules are direct adaptations of traditional HL rules

$$\vdash \{P\}\ \texttt{skip} \sim \texttt{skip}\ \{P\}$$ $(\vdash \{P\}\ \texttt{skip}\ \{P\})$

$$\frac{\vdash \{P\}\ c_1 \sim c_2\ \{Q'\} \qquad \vdash \{Q'\}\ c_1' \sim c_2'\ \{Q\}}{\vdash \{P\}\ c_1; c_1' \sim c_2; c_2'\ \{Q\}}$$ $\left(\frac{\vdash \{P\}\ c\ \{Q'\} \qquad \vdash \{Q'\}\ c'\ \{Q\}}{\vdash \{P\}\ c; c'\ \{Q\}}\right)$

- Requires programs to execute lockstep

$$\frac{\vdash \{I \land G_{1\langle 1\rangle}\}\ c_1 \sim c_2\ \{I\} \qquad \boxed{\models (I \implies G_{1\langle 1\rangle} = G_{2\langle 2\rangle})}}{\vdash \{I\}\ \texttt{while}\ G_1\ \texttt{do}\ c_1 \sim \texttt{while}\ G_2\ \texttt{do}\ c_2\ \{I \land \neg G_{1\langle 1\rangle}\}}\ [\text{while}]$$

- (The classic fragment) only relates programs that are structurally equal. But the logic can be extended with "one-sided" rules, e.g.

$$\frac{}{\vdash \{P\}\ \texttt{if}\ G\ \texttt{then}\ c_1\ \texttt{else}\ c_1' \sim c_2\ \{Q\}}\ [\text{c-branch}]$$

Proof system

- Most rules are direct adaptations of traditional HL rules

$$\vdash \{P\}\ \texttt{skip} \sim \texttt{skip}\ \{P\}$$

$(\vdash \{P\}\ \texttt{skip}\ \{P\})$

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

$\left(\frac{\vdash \{P\}\ c\ \{Q'\} \quad \vdash \{Q'\}\ c'\ \{Q\}}{\vdash \{P\}\ c; c'\ \{Q\}}\right)$

- Requires programs to execute lockstep

$$\frac{\vdash \{I \wedge G_{1\langle 1\rangle}\}\ c_1 \sim c_2\ \{I\} \qquad \boxed{\models (I \Longrightarrow G_{1\langle 1\rangle} = G_{2\langle 2\rangle})}}{\vdash \{I\}\ \texttt{while}\ G_1\ \texttt{do}\ c_1 \sim \texttt{while}\ G_2\ \texttt{do}\ c_2\ \{I \wedge \neg G_{1\langle 1\rangle}\}}\text{[while]}$$

- (The classic fragment) only relates programs that are structurally equal. But the logic can be extended with "one-sided" rules, e.g.

$$\frac{\vdash \{P \wedge G_{\langle 1\rangle}\}\ c_1 \sim c_2\ \{Q\} \quad \vdash \{P \wedge \neg G_{\langle 1\rangle}\}\ c_1' \sim c_2\ \{Q\}}{\vdash \{P\}\ \texttt{if}\ G\ \texttt{then}\ c_1\ \texttt{else}\ c_1' \sim c_2\ \{Q\}}\text{[c-branch]}$$

From the logic to probability claims

From the logic to probability claims

$$\frac{}{\Pr[c_1(s_1) : A] = \Pr[c_2(s_2) : B]} \text{ [Pr-Eq]}$$

From the logic to probability claims

$$\frac{\models \{P\}\ c_1 \sim c_2\ \{Q\}}{\Pr[c_1(s_1) : A] = \Pr[c_2(s_2) : B]}\ \text{[Pr-Eq]}$$

From the logic to probability claims

$$\frac{\models \{P\}\ c_1 \sim c_2\ \{Q\} \qquad Q \implies (A\langle 1\rangle \iff B\langle 2\rangle)}{\Pr[c_1(s_1) : A] = \Pr[c_2(s_2) : B]} \text{ [Pr-Eq]}$$

From the logic to probability claims

$$\frac{s_1 \; P \; s_2 \qquad \models \{P\}\; c_1 \sim c_2 \;\{Q\} \qquad Q \implies (A\langle 1\rangle \iff B\langle 2\rangle)}{\Pr[c_1(s_1) : A] = \Pr[c_2(s_2) : B]} \text{ [Pr-Eq]}$$

From the logic to probability claims

$$\frac{s_1 \, P \, s_2 \qquad \models \{P\}\ c_1 \sim c_2\ \{Q\} \qquad Q \implies (A\langle 1\rangle \iff B\langle 2\rangle)}{\Pr[c_1(s_1) : A] = \Pr[c_2(s_2) : B]} \ [\text{Pr-Eq}]$$

$$\frac{}{\Pr[c_1(s_1) : A] \leq \Pr[c_2(s_2) : B]} \ [\text{Pr-Le}]$$

From the logic to probability claims

$$\frac{s_1 \; P \; s_2 \qquad \models \{P\}\, c_1 \sim c_2 \,\{Q\} \qquad Q \implies (A\langle 1\rangle \iff B\langle 2\rangle)}{\Pr[c_1(s_1) : A] = \Pr[c_2(s_2) : B]} \; \text{[Pr-Eq]}$$

$$\frac{s_1 \; P \; s_2 \qquad \models \{P\}\, c_1 \sim c_2 \,\{Q\} \qquad Q \implies (A\langle 1\rangle \implies B\langle 2\rangle)}{\Pr[c_1(s_1) : A] \leq \Pr[c_2(s_2) : B]} \; \text{[Pr-Le]}$$

Wrapping up

Conclusion

Conclusion

Successful application of machine-checked proofs to the field of cryptography

Conclusion

Successful application of machine-checked proofs to the field of cryptography

- Formal semantics of probabilistic language
- A probabilistic relational Hoare logic
- Mechanised program transformations
- Formalization of emblematic schemes: OAEP, ElGammal, FDH, etc.

Conclusion

Successful application of machine-checked proofs to the field of cryptography

- Formal semantics of probabilistic language
- A probabilistic relational Hoare logic
- Mechanised program transformations
- Formalization of emblematic schemes: OAEP, ElGammal, FDH, etc.

KEY INSIGHT:

View cryptographic proofs as a problem of (relational) probabilistic program verification

Conclusion

Successful application of machine-checked proofs to the field of cryptography

- Formal semantics of probabilistic language
- A probabilistic relational Hoare logic
- Mechanised program transformations
- Formalization of emblematic schemes: OAEP, ElGammal, FDH, etc.

KEY INSIGHT:

View cryptographic proofs as a problem of (relational) probabilistic program verification

Thanks!

Backup Slides

Language semantics

$$
\begin{aligned}
[\![\mathsf{skip}]\!]\ m &= \mathsf{unit}\ m \\
[\![c;\ c']\!]\ m &= \mathsf{bind}\ ([\![c]\!]\ m)\ [\![c']\!] \\
[\![x \leftarrow e]\!]\ m &= \mathsf{unit}\ (m\,\{[\![e]\!]_{\mathcal{E}}\ m/x\}) \\
[\![x \xleftarrow{\$} d]\!]\ m &= \mathsf{bind}\ ([\![d]\!]_{\mathcal{DE}}\ m)\ (\lambda v.\ \mathsf{unit}\ (m\,\{v/x\})) \\
[\![\mathsf{assert}\ e]\!]\ m &= \textbf{if}\ ([\![e]\!]_{\mathcal{E}}\ m = \mathsf{true})\ \textbf{then}\ (\mathsf{unit}\ m)\ \textbf{else}\ \mu_0 \\
[\![\mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2]\!]\ m &= \textbf{if}\ ([\![e]\!]_{\mathcal{E}}\ m = \mathsf{true})\ \textbf{then}\ ([\![c_1]\!]\ m)\ \textbf{else}\ ([\![c_2]\!]\ m) \\
[\![\mathsf{while}\ e\ \mathsf{do}\ c]\!]\ m &= \lambda f.\ \mathrm{lub}\,(\lambda n.\ ([\![[\mathsf{while}\ e\ \mathsf{do}\ c]_n]\!]\ m)(f))
\end{aligned}
$$

$$
\text{where}\quad
\begin{aligned}
[\mathsf{while}\ e\ \mathsf{do}\ c]_0 &= \mathsf{assert}\ \neg e \\
[\mathsf{while}\ e\ \mathsf{do}\ c]_{n+1} &= \mathsf{if}\ e\ \mathsf{then}\ c;\ [\mathsf{while}\ e\ \mathsf{do}\ c]_n
\end{aligned}
$$

The measure monad (ALEA library)

$$\mathcal{D}(A) \triangleq (A \to [0,1]) \to [0,1]$$

$$\mu(f) = \text{"expected value of } f \text{ wrt } \mu\text{"}$$

$$\begin{array}{rcl} \text{unit} & : & A \to \mathcal{D}(A) \\ & \stackrel{\text{def}}{=} & \lambda x.\ \lambda f.\ f(x) \\ \text{bind} & : & \mathcal{D}(A) \to (A \to \mathcal{D}(B)) \to \mathcal{D}(B) \\ & \stackrel{\text{def}}{=} & \lambda \mu.\ \lambda M.\ \lambda f.\ \mu(\lambda x.\ M(x)(f)). \end{array}$$

Example

$$[\![b_1 \stackrel{\$}{\leftarrow} \{t, f\};\ b_2 \stackrel{\$}{\leftarrow} \{t, f\}]\!]\ s = \lambda f.\ \tfrac{1}{4} f(s[b_1, b_2/t, t]) + \tfrac{1}{4} f(s[b_1, b_2/t, f]) \\ \tfrac{1}{4} f(s[b_1, b_2/f, t]) + \tfrac{1}{4} f(s[b_1, b_2/f, f])$$

Lifting relations to distributions via couplings

Lifting relations to distributions via couplings

Q LIFTING $Q^\sharp$

relation over program states

relation over *distributions* on program states

$$(\mu_1, \mu_2) \models Q^\sharp \quad \triangleq \quad \exists \mu \in \mathcal{D}(\mathbb{S} \times \mathbb{S}). \begin{cases} \pi_1(\mu) = \mu_1 \ \wedge \ \pi_2(\mu) = \mu_2, \ \text{and} \\ \Pr_\mu[\neg Q] = 0 \end{cases}$$

coupling between μ_1 and μ_2

* See *Logical, Metric, and Algorithmic Characterisations of Probabilistic Bisimulation*, Deng & Du.

Proof system (two-sided rules)

$$\frac{}{\vdash \{P\}\ \texttt{skip} \sim \texttt{skip}\ \{P\}}\ [\text{skip}] \qquad \frac{}{\vdash \{Q[x_{\langle 1\rangle}/A_{\langle 1\rangle}, y_{\langle 2\rangle}/B_{\langle 2\rangle}]\}\ x := A \sim y := B\ \{Q\}}\ [\text{assgn}]$$

$$\frac{}{\vdash \{\underline{\text{true}}\}\ \texttt{abort} \sim \texttt{abort}\ \{Q\}}\ [\text{abort}] \qquad \frac{\vdash \{P\}\ c_1 \sim c_2\ \{Q'\} \quad \vdash \{Q'\}\ c_1' \sim c_2'\ \{Q\}}{\vdash \{P\}\ c_1; c_1' \sim c_2; c_2'\ \{Q\}}\ [\text{seq}]$$

$$\frac{\models (P \Longrightarrow P') \quad \vdash \{P'\}\ c_1 \sim c_2\ \{Q'\} \quad \models (Q' \Longrightarrow Q)}{\vdash \{P\}\ c_1 \sim c_2\ \{Q\}}\ [\text{cons}]$$

$$\frac{\begin{array}{c}\models (P \Longrightarrow G_{1\langle 1\rangle} = G_{2\langle 2\rangle}) \\ \vdash \{P \wedge G_{1\langle 1\rangle}\}\ c_1 \sim c_2\ \{Q\} \quad \vdash \{P \wedge \neg G_{1\langle 1\rangle}\}\ c_1' \sim c_2'\ \{Q\}\end{array}}{\vdash \{P\}\ \texttt{if}\ G_1\ \texttt{then}\ c_1\ \texttt{else}\ c_1' \sim \texttt{if}\ G_2\ \texttt{then}\ c_2\ \texttt{else}\ c_2'\ \{Q\}}\ [\text{if}]$$

$$\frac{\vdash \{I \wedge G_{1\langle 1\rangle}\}\ c_1 \sim c_2\ \{I\} \qquad \models (I \Longrightarrow G_{1\langle 1\rangle} = G_{2\langle 2\rangle})}{\vdash \{I\}\ \texttt{while}\ G_1\ \texttt{do}\ c_1 \sim \texttt{while}\ G_2\ \texttt{do}\ c_2\ \{I \wedge \neg G_{1\langle 1\rangle}\}}\ [\text{while}]$$

$$\frac{\vdash \{P^{-1}\}\ c_2 \sim c_1\ \{Q^{-1}\}}{\vdash \{P\}\ c_1 \sim c_2\ \{Q\}}\ [\text{inv}] \qquad \frac{\vdash \{P\}\ c_1 \sim c_2\ \{Q\} \quad \vdash \{P'\}\ c_2 \sim c_3\ \{Q'\}}{\vdash \{P \circ P'\}\ c_1 \sim c_3\ \{Q \circ Q'\}}\ [\text{comp}]$$

$$\frac{s_1\ P\ s_2 \ \triangleq\ (\mu_1 \blacktriangleright \lambda v \bullet\ \eta_{s_1[x_1/v]})\ \mathcal{L}(Q)\ (\mu_2 \blacktriangleright \lambda v \bullet\ \eta_{s_2[x_2/v]})}{\vdash \{P\}\ x_1 \stackrel{\$}{:=} \mu_1 \ \sim\ x_2 \stackrel{\$}{:=} \mu_2\ \{Q\}}\ [\text{rand}]$$

Proof system (one-sided rules)

$$\frac{}{\vdash \{\underline{\mathsf{false}}\}\ c_1 \sim c_2\ \{Q\}}\ [\text{contr}]$$

$$\frac{}{\vdash \{Q[x_{\langle 1\rangle}/A_{\langle 1\rangle}]\}\ x \coloneqq A \sim \texttt{skip}\ \{Q\}}\ [\text{d-assgn}]$$

$$\frac{\vdash \{P \wedge G_{\langle 1\rangle}\}\ c_1 \sim c_2\ \{Q\} \quad \vdash \{P \wedge \neg G_{\langle 1\rangle}\}\ c_1' \sim c_2\ \{Q\}}{\vdash \{P\}\ \texttt{if}\ G\ \texttt{then}\ c_1\ \texttt{else}\ c_1' \sim c_2\ \{Q\}}\ [\text{c-branch}]$$

$$\frac{}{\vdash \{P \wedge \neg G_{\langle 1\rangle}\}\ \texttt{while}\ G\ \texttt{do}\ c \sim \texttt{skip}\ \{P \wedge \neg G_{\langle 1\rangle}\}}\ [\text{d-while}]$$