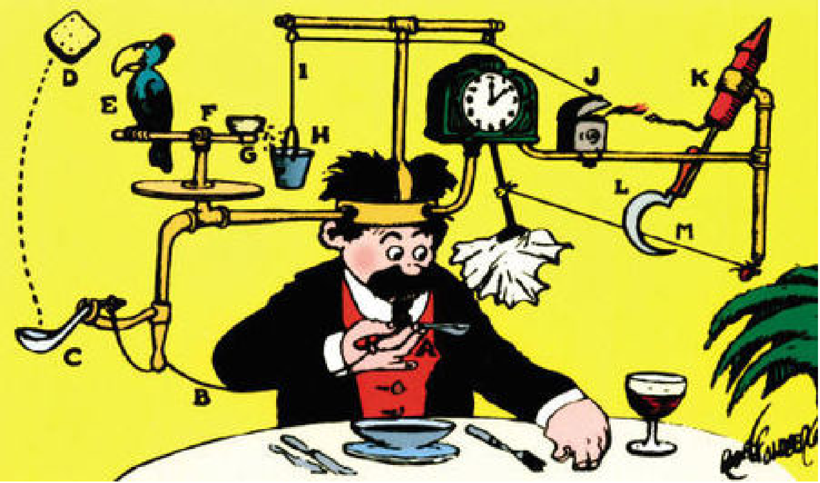
Higher-Order Relational Refinement Types for Mechanism Design and Differential Privacy

Gilles Barthe[1], Marco Gaboardi[2],
Emilio Jesús Gallego Arias[3,4], Justin Hsu[4],
Aaron Roth[4], Pierre-Yves Strub[1]

[1]IMDEA Software, [2]University of Dundee,
[3]CRI Mines–ParisTech, [4]University of Pennsylvania

January 15th, 2015

The Application

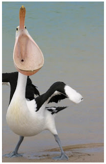


Mechanism Design

One painting for sale

One painting for sale

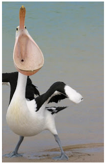
How much will you pay?



One painting for sale

How much will you pay?

$10 million!



$50 million!

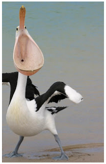
$3

A story

One painting for sale

How much will you pay?

$10 million!



$50 million!

$3

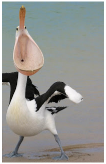
Who wins, and for how much?

How much will you pay?

Top bid pays top price?

- Simple rule
- Can encourage manipulation...

$10 million!



$50 million!

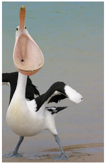
$3

How much will you pay?

Top bid pays top price?

- Simple rule
- Can encourage manipulation...

$10 million!



~~$50 million!~~
$10.1 million?

$3

Algorithm design with strategic inputs

Algorithm design with strategic inputs

Rational agents

- Report data
- Care about output
- May lie, strategize

Algorithm design with strategic inputs

Rational agents

- Report data
- Care about output
- May lie, strategize

Goal: encourage "good" behavior

Designing auctions

- Bidders each have personal value $v : \mathbb{R}$ for the item

Designing auctions

- Bidders each have personal value $v : \mathbb{R}$ for the item
- Bidder's happiness is function of price, v, whether they win

Designing auctions

- Bidders each have personal value $v : \mathbb{R}$ for the item
- Bidder's happiness is function of price, v, whether they win
- Bidder reports a bid $b : \mathbb{R}$ to the mechanism

Designing auctions

- Bidders each have personal value $v : \mathbb{R}$ for the item
- Bidder's happiness is function of price, v, whether they win
- Bidder reports a bid $b : \mathbb{R}$ to the mechanism

$\boxed{\text{Property: agent always maximizes happiness with } b = v}$

A (very) simple auction

Fixed price auction

- Given a fixed price `price`
- Bidder bids `bid`, buys item if higher than `price`

A (very) simple auction

Fixed price auction

- Given a fixed price `price`
- Bidder bids `bid`, buys item if higher than `price`

What is the happiness function for a bidder?

```
fixedprice price value bid =
  if bid > price then
    value - price
  else
    0
```

Consider bidder's happiness function…

- First run: bidder bids $b = v$ (honest)

Consider bidder's happiness function...

- First run: bidder bids $b = v$ (honest)
- Second run: bidder bids arbitrarily (maybe not honest)

The verification strategy

Consider bidder's happiness function...

- First run: bidder bids $b = v$ (honest)
- Second run: bidder bids arbitrarily (maybe not honest)
- Verify: happiness in first run is higher than in second run

Consider bidder's happiness function...

- First run: bidder bids $b = v$ (honest)
- Second run: bidder bids arbitrarily (maybe not honest)
- Verify: happiness in first run is higher than in second run

The verification strategy

Consider bidder's happiness function...

- First run: bidder bids $b = v$ (honest)
- Second run: bidder bids arbitrarily (maybe not honest)
- Verify: happiness in first run is higher than in second run

```
fixedprice p v v =
  if v > p then
    v - p
  else
    0
```

$$\geqslant$$

```
fixedprice p v b =
  if b > p then
    v - p
  else
    0
```

Consider bidder's happiness function...

- First run: bidder bids $b = v$ (honest)
- Second run: bidder bids arbitrarily (maybe not honest)
- Verify: happiness in first run is higher than in second run

```
fixedprice p v v =
  if v > p then
    v - p
  else
    0
```

$$\geqslant$$

```
fixedprice p v b =
  if b > p then
    v - p
  else
    0
```

This is a relational property

Introducing HOARe^2

A type system with relational refinement types

Judgment

$$\Gamma \vdash e : \{x : \ T \ \mid \ \phi(x) \ \}$$

Judgment

$$\Gamma \vdash e : \{x : \overset{\text{type}}{\boxed{T}} \mid \phi(x)\}$$

Judgment

type

predicate

$$\Gamma \vdash e : \{x : \boxed{T} \mid \boxed{\phi(x)}\}$$

Judgment

type

predicate

$$\Gamma \vdash e : \{x : T \mid \phi(x)\}$$

"e is a program of type T such that $\phi(e)$ holds"

Example

$$\Gamma \vdash 3 : \{x : \mathbb{Z} \mid x \geqslant 0\}$$

Example

$$\Gamma \vdash 3 : \{x : \mathbb{Z} \mid x \geqslant 0\}$$

"3 is a non-negative integer"

Relational Judgment

$$\Gamma \vdash e :: \{x :: T \mid \phi(\ x_{\lhd}, x_{\rhd}\ )\}$$

Relational Judgment

$$\Gamma \vdash e :: \{x :: T \mid \phi(\boxed{x_{\lhd}, x_{\rhd}})\}$$

Relational Judgment

$$\Gamma \vdash e :: \{x :: T \mid \phi(\boxed{x_{\triangleleft}, x_{\triangleright}})\}$$

ϕ mentions two runs of program e via $x_{\triangleleft}$ and $x_{\triangleright}$

Relational Judgment

$$\Gamma \vdash e :: \{x :: T \mid \phi(\boxed{x_{\lhd}, x_{\rhd}})\}$$

ϕ mentions two runs of program e via $x_{\lhd}$ and $x_{\rhd}$

Example

$$\{y :: \mathbb{Z} \mid y_{\lhd} \leqslant y_{\rhd}\} \vdash e :: \{x :: \mathbb{Z} \mid x_{\lhd} \leqslant x_{\rhd}\}$$

Relational Judgment

$$\Gamma \vdash e :: \{x :: T \mid \phi(\boxed{x_{\triangleleft}, x_{\triangleright}})\}$$

ϕ mentions two runs of program e via $x_{\triangleleft}$ and $x_{\triangleright}$

Example

$$\{y :: \mathbb{Z} \mid y_{\triangleleft} \leqslant y_{\triangleright}\} \vdash e :: \{x :: \mathbb{Z} \mid x_{\triangleleft} \leqslant x_{\triangleright}\}$$

"If y increases, then e increases."

Relational Judgment

$$\Gamma \vdash e :: \{x :: T \mid \phi(\boxed{x_\triangleleft, x_\triangleright})\}$$

ϕ mentions two runs of program e via $x_\triangleleft$ and $x_\triangleright$

Example

$$\{y :: \mathbb{Z} \mid y_\triangleleft \leqslant y_\triangleright\} \vdash e :: \{x :: \mathbb{Z} \mid x_\triangleleft \leqslant x_\triangleright\}$$

"If y increases, then e increases."

Background

- First used in the RF* language, POPL 2014

Happiness function

```
fixedprice price value bid =
  if bid > price then
    value - price
  else
    0
```

Happiness function

```
fixedprice price value bid =
  if bid > price then
    value - price
  else
    0
```

Truthfulness in a type

Happiness function

```
fixedprice price value bid =
  if bid > price then
    value - price
  else
    0
```

Truthfulness in a type

$$\{p :: \mathbb{R} \mid p_{\lhd} = p_{\rhd}\} \qquad \text{(Fixed price)}$$

Typing truthfulness

Happiness function

```
fixedprice price value bid =
  if bid > price then
    value - price
  else
    0
```

Truthfulness in a type

$$\{p :: \mathbb{R} \mid p_{\lhd} = p_{\rhd}\} \qquad \text{(Fixed price)}$$

$$\rightarrow \{v :: \mathbb{R} \mid v_{\lhd} = v_{\rhd}\} \qquad \text{(Bidder value fixed)}$$

Happiness function

```
fixedprice price value bid =
  if bid > price then
    value - price
  else
    0
```

Truthfulness in a type

$$
\begin{aligned}
&\{p :: \mathbb{R} \mid p_{\lhd} = p_{\rhd}\} && \text{(Fixed price)} \\
&\to \{v :: \mathbb{R} \mid v_{\lhd} = v_{\rhd}\} && \text{(Bidder value fixed)} \\
&\to \{b :: \mathbb{R} \mid b_{\lhd} = v_{\lhd}\} && \text{(Bid} = \text{value on} \lhd \text{ run)}
\end{aligned}
$$

Happiness function

```
fixedprice price value bid =
  if bid > price then
    value - price
  else
    0
```

Truthfulness in a type

$$\begin{aligned}
&\{p :: \mathbb{R} \mid p_{\lhd} = p_{\rhd}\} && \text{(Fixed price)}\\
&\to \{v :: \mathbb{R} \mid v_{\lhd} = v_{\rhd}\} && \text{(Bidder value fixed)}\\
&\to \{b :: \mathbb{R} \mid b_{\lhd} = v_{\lhd}\} && \text{(Bid = value on } \lhd \text{ run)}\\
&\to \{u :: \mathbb{R} \mid u_{\lhd} \geqslant u_{\rhd}\} && \text{(Truthful)}
\end{aligned}$$

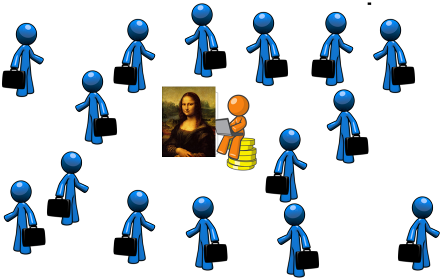
A more complex auction

- Unlimited supply of items (e.g., music files)
- Want to use `fixedprice`, but for what price?

A more complex auction

- Unlimited supply of items (e.g., music files)
- Want to use `fixedprice`, but for what price?

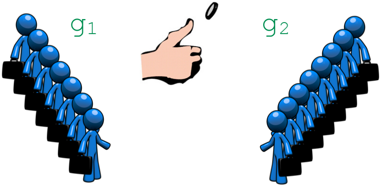
Randomize!



A more complex auction

- Unlimited supply of items (e.g., music files)
- Want to use `fixedprice`, but for what price?

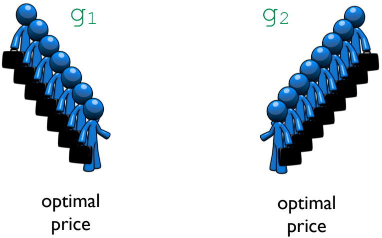
Randomize!



A more complex auction

- Unlimited supply of items (e.g., music files)
- Want to use `fixedprice`, but for what price?

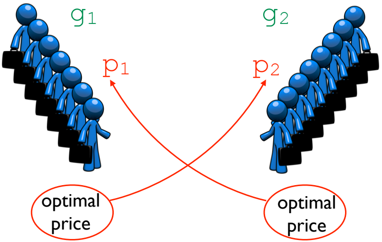
Randomize!



A more complex auction

- Unlimited supply of items (e.g., music files)
- Want to use `fixedprice`, but for what price?

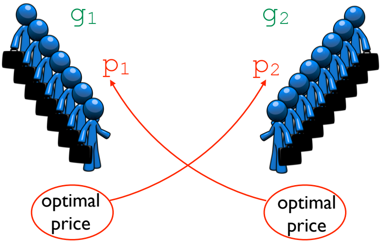
Randomize!



A more complex auction

- Unlimited supply of items (e.g., music files)
- Want to use `fixedprice`, but for what price?

Randomize!

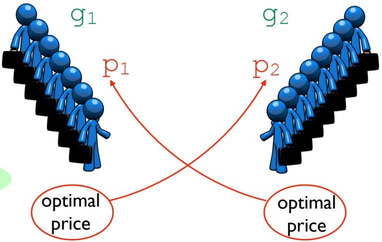


Verify: happiness higher when bid is true value

A more complex auction

- Unlimited supply of items (e.g., music files)
- Want to use `fixedprice`, but for what price?

Randomize!



on average

Verify: happiness higher when bid is true value

Monotonicity of expectation

- (One) Distribution μ over A

Monotonicity of expectation

- (One) Distribution μ over A
- Two functions $f_1, f_2 : A \to \mathbb{R}$ with

$$f_1\ x \geqslant f_2\ x \qquad \text{for all } x : A$$

Monotonicity of expectation

- (One) Distribution μ over A
- Two functions $f_1, f_2 : A \to \mathbb{R}$ with

$$f_1\ x \geqslant f_2\ x \qquad \text{for all } x : A$$

- Then, fact about expected values:

$$\mathbb{E}_\mu[f_1] \geqslant \mathbb{E}_\mu[f_2]$$

One key ingredient

Monotonicity of expectation

- (One) Distribution μ over A
- Two functions $f_1, f_2 : A \to \mathbb{R}$ with

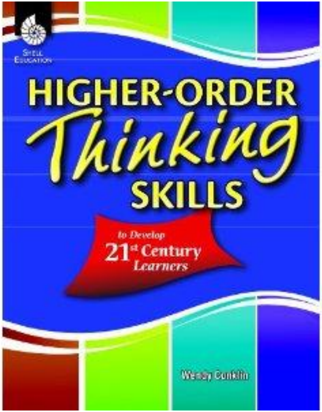
$$f_1 \; x \geqslant f_2 \; x \qquad \text{for all } x : A$$

- Then, fact about expected values:

$$\mathbb{E}_\mu[f_1] \geqslant \mathbb{E}_\mu[f_2]$$

f_1 bigger than
f_2 on average

Extending HOARe^2



Distributions and Higher-order refinements

Relating Distributions

Probabilistic programs

- Reason about two runs of a probabilistic program
- Use type of probability distributions

Probabilistic programs

- Reason about two runs of a probabilistic program
- Use type of probability distributions

Typing distributions

$$\Gamma \vdash e :: \mathfrak{M}_{0,0}[\{x :: T \mid \phi(x_{\triangleleft}, x_{\triangleright})\}]$$

Probabilistic programs

- Reason about two runs of a probabilistic program
- Use type of probability distributions

Typing distributions

$$\Gamma \vdash e :: \mathfrak{M}_{0,0}[\{x :: T \mid \phi(x_{\triangleleft}, x_{\triangleright})\}]$$

"e is a distribution over T, with two runs related by ϕ "

Probabilistic programs

- Reason about two runs of a probabilistic program
- Use type of probability distributions

Typing distributions

$$\Gamma \vdash e :: \mathfrak{M}_{0,0}[\{x :: T \mid \phi(x_{\triangleleft}, x_{\triangleright})\}]$$

"e is a distribution over T, with two runs related by ϕ "

???

Equivalence of Distributions

$$\Gamma \vdash e :: \mathfrak{M}_{0,0}[\{x :: T \mid \phi(x_\triangleleft, x_\triangleright)\}]$$

What does this mean?

- Convert relation ϕ to a relation $\phi^\#$ on distributions over T
- Two runs of e related by $\phi^\#$ (as distributions!)

Example

$$\Gamma \vdash e :: \mathfrak{M}_{0,0}[\{x :: T \mid \ x_{\triangleleft} = x_{\triangleright}\ \}]$$

Example

$$\Gamma \vdash e :: \mathfrak{M}_{0,0}[\{x :: T \mid \boxed{x_{\triangleleft} = x_{\triangleright}}\}]$$

Example

$$\Gamma \vdash e :: \mathfrak{M}_{0,0}[\{x :: T \mid \boxed{x_\lhd = x_\rhd}\}]$$

"e is a distribution over T that is identical in both runs"

Example

$$\Gamma \vdash e :: \mathfrak{M}_{0,0}[\{x :: T \mid \boxed{x_{\lhd} = x_{\rhd}}\}]$$

"e is a distribution over T that is identical in both runs"

Background

- Proposed by Barthe, Köpf, Olmedo, Zanella
- Generalizing $0, 0$ to ε, δ models differential privacy

Example

$$\Gamma \vdash e :: \mathfrak{M}_{0,0}\big[\{x :: T \mid \boxed{x_{\triangleleft} = x_{\triangleright}}\}\big]$$

"e is a distribution over T that is identical in both runs"

Background

- Proposed by Barthe, Köpf, Olmedo, Zanella
- Generalizing $0, 0$ to ε, δ models differential privacy

Our contribution

- Simplify and build into a type system

Refinements on functions

$$\Gamma \vdash e :: \{f :: T \to U \mid \phi\}$$

Refinements on functions

$$\Gamma \vdash e :: \{f :: T \to U \mid \phi\}$$

"e is a function from T to U that satisfies ϕ"

Refinements on functions

$$\Gamma \vdash e :: \{f :: T \to U \mid \phi\}$$

"e is a function from T to U that satisfies ϕ"

Our contribution

- Consistency by carefully handling termination

Refinements on functions

$$\Gamma \vdash e :: \{f :: T \to U \mid \phi\}$$

"e is a function from T to U that satisfies ϕ"

Our contribution

- Consistency by carefully handling termination
- Show naïve treatment leads to inconsistency

Want to show

$$\mathbb{E}\ \mu\ f_1 \geqslant \mathbb{E}\ \mu\ f_2$$

In HOARe^2, type $\mathbb{E}$ as...

$$\mathfrak{M}_{0,0}[\{x :: A \mid x_\lhd = x_\rhd\}] \qquad \text{(Same distributions)}$$

Expressing monotonicity of expectations

Want to show

$$\mathbb{E}\ \mu\ f_1 \geqslant \mathbb{E}\ \mu\ f_2$$

In HOARe^2, type $\mathbb{E}$ as...

$$\mathfrak{M}_{0,0}[\{x :: A \mid x_\triangleleft = x_\triangleright\}] \qquad \text{(Same distributions)}$$
$$\to \{f :: A \to \mathbb{R} \mid \forall x.\ f_\triangleleft\ x \geqslant f_\triangleright\ x\} \qquad \text{(Higher-order)}$$

Expressing monotonicity of expectations

Want to show

$$\mathbb{E}\ \mu\ f_1 \geqslant \mathbb{E}\ \mu\ f_2$$

In HOARe^2, type $\mathbb{E}$ as...

$$\begin{aligned} &\mathfrak{M}_{0,0}[\{x :: A \mid x_\lhd = x_\rhd\}] && \text{(Same distributions)} \\ &\to \{f :: A \to \mathbb{R} \mid \forall x.\ f_\lhd\ x \geqslant f_\rhd\ x\} && \text{(Higher-order)} \\ &\to \{e :: \mathbb{R} \mid e_\lhd \geqslant e_\rhd\} && \text{(Monotonic)} \end{aligned}$$

Expressing monotonicity of expectations

Want to show

$$\mathbb{E}\ \mu\ f_1 \geqslant \mathbb{E}\ \mu\ f_2$$

In HOARe^2, type $\mathbb{E}$ as...

$$\begin{aligned}
\mathbb{E} :: \mathfrak{M}_{0,0}[\{x :: A \mid x_\lhd = x_\rhd\}] & \qquad \text{(Same distributions)} \\
\to \{f :: A \to \mathbb{R} \mid \forall x.\ f_\lhd\ x \geqslant f_\rhd\ x\} & \qquad \text{(Higher-order)} \\
\to \{e :: \mathbb{R} \mid e_\lhd \geqslant e_\rhd\} & \qquad \text{(Monotonic)}
\end{aligned}$$

Much more in the paper

Semantics

- Soundness of the system
- Requires termination

Implementation

- Automated, low annotation burden
- Why3 and SMT solvers

Translation

- Embedding of DFuzz, a language for differential privacy

More complex examples

- Verify differential privacy
- Verify MD properties beyond truthfulness

Takeaway points

Four features, one system

- HOARe^2: relational properties for randomized programs
- Combine features in a clean, usable way

Four features, one system

- HOARe^2: relational properties for randomized programs
- Combine features in a clean, usable way

Formal verification for mechanism design!

- Exciting, under-explored area for verification
- Tons of interesting properties, mechanisms
- Strong motivation besides (mere) correctness

Higher-Order Relational Refinement Types for Mechanism Design and Differential Privacy

Gilles Barthe[1], Marco Gaboardi[2],
Emilio Jesús Gallego Arias[3,4], Justin Hsu[4],
Aaron Roth[4], Pierre-Yves Strub[1]

[1]IMDEA Software, [2]University of Dundee,
[3]CRI Mines–ParisTech, [4]University of Pennsylvania

January 15th, 2015