

# Computer-aided Verification in Mechanism Design

Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias,  
Justin Hsu<sup>\*</sup>, Aaron Roth<sup>\*</sup>, Pierre-Yves Strub

IMDEA Software, École Polytechnique, University at Buffalo, <sup>\*</sup>University of  
Pennsylvania

December 14, 2016

Mechanism design

=

Algorithm design

+

Strategic inputs

\*In computer science

# Encourage agents to behave simply

## Benefits

- ▶ For the agents: easy to decide what to do
- ▶ For the designer: easy to reason about what agents will do

# Best case: truthfulness

## Model

- ▶ Agents have private type  $t_i \in T$
- ▶ Mechanism inputs: agents report  $s_i \in T$
- ▶ Mechanism outputs: outcome  $o \in O$  and payments  $p_i \in \mathbb{R}$

# Best case: truthfulness

## Model

- ▶ Agents have private type  $t_i \in T$
- ▶ Mechanism inputs: agents report  $s_i \in T$
- ▶ Mechanism outputs: outcome  $o \in O$  and payments  $p_i \in \mathbb{R}$

## Definition (Complete information)

A mechanism is **truthful** (DSIC) if each agent maximizes their utility by reporting  $s_i = t_i$ , no matter what other agents do.

## Definition (Incomplete information)

A mechanism is **Bayesian Incentive Compatible** (BIC) if each agent maximizes their expected utility by reporting  $s_i = t_i$ , when other agents report their true type drawn from a known prior  $\mu$ .

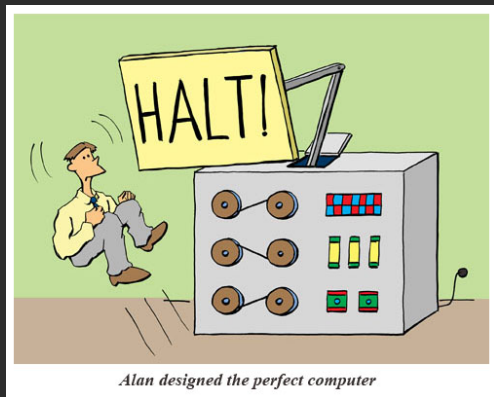
Mechanism  $\approx$  Program

Truthfulness  $\approx$  Property

---

Program verification  
for incentive properties

But isn't this really hard?



## Divide the task

- ▶ Proof **construction**: hard
- ▶ Proof **checking**: easy

# Why verify properties? Check correctness

```

3 import random
4
5 class GSP:
6     """
7     Implements the generalized second price auction mechanism.
8     """
9     @staticmethod
10     def compute(slot_clicks, reserve, bids):
11         """
12         Given info about the setting (clicks for each slot, and reserve price),
13         and bids (list of (id, bid) tuples), compute the following:
14         allocation: list of the occupancies in each slot
15             len(allocation) = n * len(bids), len(slot_clicks)
16         per_click_payments: list of payments for each slot
17             len(per_click_payments) = len(allocation)
18
19         If any bids are below the reserve price, they are ignored.
20
21         Returns a pair of lists (allocation, per_click_payments):
22         - allocation is a list of the ids of the bidders in each slot
23           [in order]
24         - per_click_payments is the corresponding payments.
25         """
26         valid = lambda (e, bid): bid >= reserve
27         valid_bids = filter(valid, bids)
28
29         rev_cmp_bids = lambda (a1, b1), (a2, b2): cmp(b2, b1)
30         # shuffle first to make sure we don't have any bias for lower or
31         # higher ids
32         random.shuffle(valid_bids)
33         valid_bids.sort(reverse=True)
34
35         num_slots = len(slot_clicks)
36         allocated_bids = valid_bids[num_slots:]
37         if len(allocated_bids) > num_slots:
38             return ((), ())
39
40         (allocation, just_bids) = zip(*allocated_bids)
41
42         # Each pays the bid below them, or the reserve
43         per_click_payments = [slot[just_bids[i]] for i in range(num_slots - 1)]
44         # figure out whether the last slot payment is set by the reserve or
45         # the first non-allocated bidder
46         if len(valid_bids) > num_slots:
47             last_payment = valid_bids[num_slots][1]
48         else:
49             last_payment = reserve
50         per_click_payments.append(last_payment)
51         return (allocation, per_click_payments)
52
53     @staticmethod
54     def bid_range_for_slot(slot, slot_clicks, reserve, bids):
55         """
56         Compute the range of bids that would result in the bidder ending up
57         in slot, given that the other bidders submit bids.

```

$$\begin{aligned}
 \mathbb{E}[\bar{A}(t_2) \mid w] &= \mathbb{E}[\bar{A}(t_1) \mid w, \beta = k] \cdot \Pr[\beta = k \mid w] \\
 &\quad + \mathbb{E}[\bar{A}(t_1) \mid w, \beta = \ell] \cdot \Pr[\beta = \ell \mid w] \\
 &= w \Pr[\beta = k \mid w] + w \Pr[\beta = \ell \mid w] \\
 \mathbb{E}[\bar{A}(t_2) \mid w] &= \mathbb{E}[\bar{A}(t_1) \mid w, \beta = k] \cdot \Pr[\beta = k \mid w] \\
 &\quad + \mathbb{E}[\bar{A}(t_1) \mid w, \beta = \ell] \cdot \Pr[\beta = \ell \mid w] \\
 &= w \Pr[\beta = k \mid w] + w \Pr[\beta = \ell \mid w]
 \end{aligned}$$

$$\begin{aligned}
 \mathbb{E}[\bar{A}(t_2) - \bar{A}(t_1) \mid w] &= (w - w) \cdot (\Pr[\beta = k \mid w] - \Pr[\beta = \ell \mid w]) \\
 &= \frac{w - w}{\Pr(W = w)} \cdot \left( \Pr \left[ \begin{array}{c} \beta = k, \\ W = w \end{array} \right] - \Pr \left[ \begin{array}{c} \beta = \ell, \\ W = w \end{array} \right] \right).
 \end{aligned}$$

Every factor in the last line is non-negative, except possibly the probability difference  $\Pr[\beta = k, W = w] - \Pr[\beta = \ell, W = w]$ . To prove that this difference is in fact positive, we will in fact prove that

$$\Pr[\beta = k, W = w \mid a, b, n] \geq \Pr[\beta = \ell, W = w \mid a, b, n],$$

for all possible values of the random variables  $a, b$ , and  $n$ . Note that when we condition on  $a, b, n$ , the values of  $\beta, W$  determine the value of the vector  $\bar{X} = (\bar{X}_1, \dots, \bar{X}_n)$ , and vice-versa. Specifically, since  $\bar{X}$  is the vector obtained from  $n/L$  by rearranging its entries using  $\sigma^{-1}$ ,  $W$  constrains  $\bar{X}$  to be one of two possible vectors  $x, x'$  that differ by interchanging their  $k^{\text{th}}$  and  $\ell^{\text{th}}$  components. Assume without loss of generality that  $x_k \geq x_\ell$ . (Otherwise, simply rename  $x$  to  $x'$  and vice-versa.) Then

$$\Pr[\beta = k, W = w \mid a, b, n] = \Pr[\bar{X} = x \mid a, b, n]$$

$$(A.1) \quad = \prod_{j=1}^n \Pr[L \cdot \bar{X}_j = x_j]$$

$$\Pr[\beta = \ell, W = w \mid a, b, n] = \Pr[\bar{X} = x' \mid a, b, n]$$

**A.2 Proof of Lemma 4.13.** In this section we restate and prove Lemma 4.13.

**LEMMA A.1.** For any real numbers  $a, b, z$  in the interval  $(0, 1)$  such that  $a \leq b/2$ , if  $q$  is any integer greater than  $1/a$ , then

$$\min(a, 1 - z) < b(1 - z^q).$$

*Proof.* As  $q > 1/a$  we have

$$(1 - a)^q < e^{-aq} < e^{-1} < \frac{1}{2}.$$

The proof consists of applying this inequality in two cases.

**Case 1:**  $a < 1 - z$ . In this case we have  $z < 1 - a$ , hence

$$1 - z^q > 1 - (1 - a)^q > \frac{1}{2}$$

$$\min(a, 1 - z) = a \leq \frac{b}{2} < b(1 - z^q),$$

as claimed.

**Case 2:**  $a \geq 1 - z$ . The equation

$$\frac{1 - z^q}{1 - z} = 1 + z + \dots + z^{q-1}$$

reveals that  $\frac{1 - z^q}{1 - z}$  is an increasing function of  $z \in (0, 1)$ . As  $z \geq 1 - a$ , we may conclude that

$$\begin{aligned} \frac{b \left( \frac{1 - z^q}{1 - z} \right)}{b} &\geq b \left( \frac{1 - (1 - a)^q}{1 - (1 - a)} \right) \\ &= \frac{b}{a} (1 - (1 - a)^q) > \frac{b}{a} \cdot \frac{1}{2} \geq 1, \end{aligned}$$

whence  $b(1 - z^q) \geq 1 - z = \min(a, 1 - z)$ , as desired.  $\square$



# Why verify incentive properties? Convince agents

What if agents don't believe incentive property?

- ▶ Incentive properties often not obvious
- ▶ Read the proof (?)

# Why verify incentive properties? Convince agents

What if agents don't believe incentive property?

- ▶ Incentive properties often not obvious
- ▶ Read the proof (?)

A possible model

- ▶ Designer constructs formal proof of incentive property
- ▶ Agents check it automatically

# Our work: A case study

## Target

- ▶ Replica-surrogate-matching mechanism (HKM)
- ▶ To prove: BIC

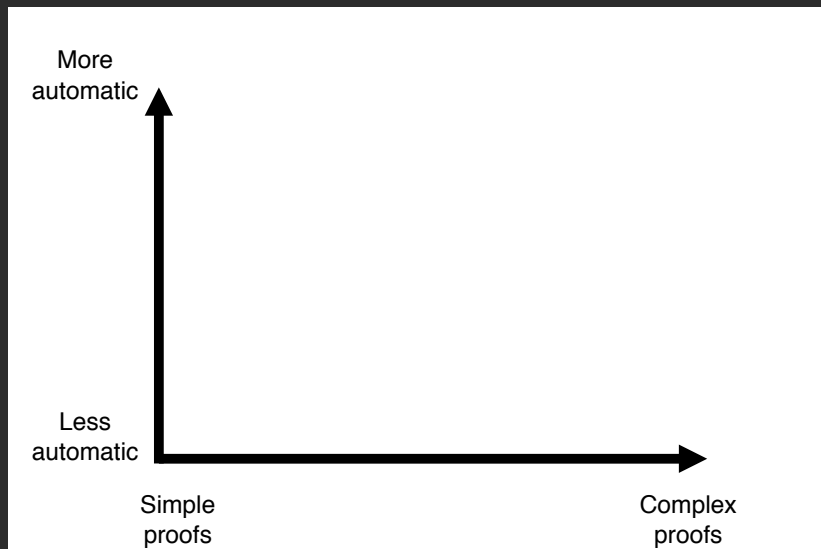
## Proof is non-trivial

- ▶ Lots of reasoning about randomization
- ▶ Need incentive property for VCG mechanism

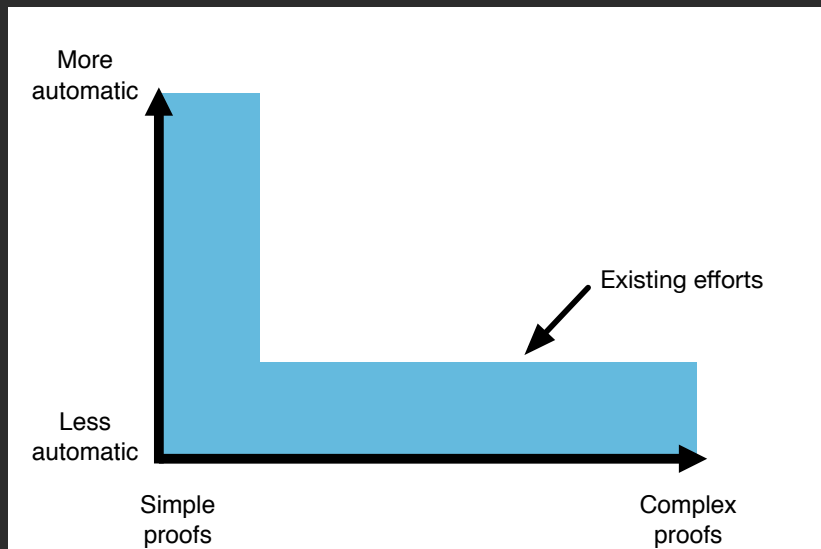
## Proof construction approaches: basic tradeoff



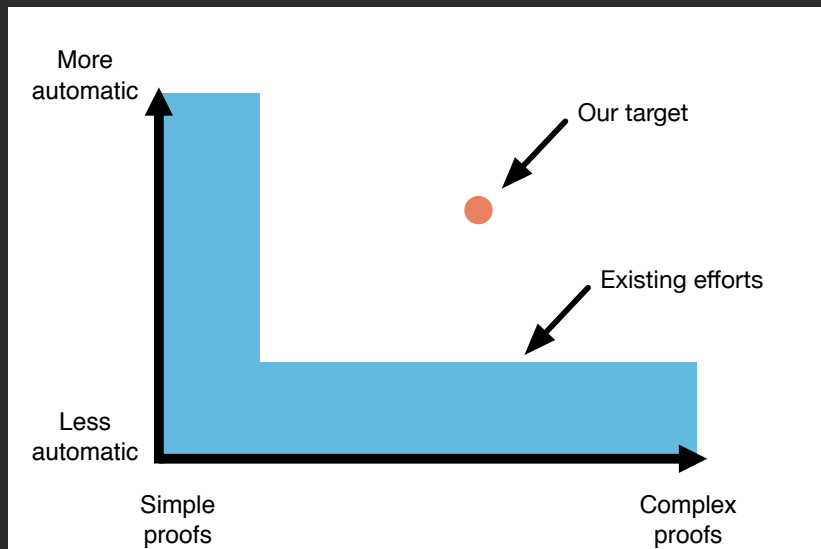
## Proof construction approaches: basic tradeoff



# Proof construction approaches: basic tradeoff



# Proof construction approaches: basic tradeoff



# Idea: incentive properties are relational properties

Program: agent's report  $\rightarrow$  agent's (expected) utility

- ▶ First run: agent report equal to agent type (truthful)
- ▶ Second run: agent report arbitrary (non-truthful)
- ▶ Truthfulness: first utility larger than second utility

Leverage specialized tools

- ▶ HOARe<sup>2</sup>: for probabilistic relational properties



# Formally verifying BIC

## Four main steps

1. Write program
2. Annotate program with assertions
3. Apply solvers to automatically check assertions
4. Fall back to less automated approaches for remaining steps

# Formally verifying BIC

## Four main steps

1. Write program
2. Annotate program with assertions
3. Apply solvers to automatically check assertions
4. Fall back to less automated approaches for remaining steps

# Writing the assertions

Basic form

$$\{prog :: S \mid \Phi(prog_1, prog_2)\}$$

# Writing the assertions

Basic form

$$\{prog :: S \mid \Phi(prog_1, prog_2)\}$$

Incentive Compatibility

$$\{rept :: T \mid rept_1 = type\} \rightarrow \{util :: \mathbb{R} \mid util_1 \geq util_2\}$$

# Applying solvers

Given  $x_1 < x_2$ , prove:

- ▶  $x_1 + 1 < x_2 + 2$  (easy)
- ▶  $f(x_1) < f(x_2)$ , where  $f$  is a program (harder)

# Applying solvers

Given  $x_1 < x_2$ , prove:

- ▶  $x_1 + 1 < x_2 + 2$  (easy)
- ▶  $f(x_1) < f(x_2)$ , where  $f$  is a program (harder)

## Results

- ▶ Almost all assertions ( $\sim 60$ ) automatically proved ( $\sim$  seconds)
- ▶ Solvers run out of time on three assertions

# Formally verifying BIC

## Four main steps

1. Write program
2. Annotate program with assertions
3. Apply solvers to automatically check assertions
4. Fall back to less automated approaches for remaining steps

See paper for details!

# Perspective

## Promising signs: automatic parts

- ▶ Handle complex proofs and mechanisms
- ▶ Solvers usually work, and are fast

## Pain points: manual parts

- ▶ When solvers fail: life is hard
- ▶ Crafting program and assertions



# Needed: more case studies!

Do you have a mechanism that ...

- ▶ has a tedious proof?
- ▶ uses randomization?
- ▶ satisfies a relational property?

# Needed: more case studies!

Do you have a mechanism that ...

- ▶ has a tedious proof?
- ▶ uses randomization?
- ▶ satisfies a relational property?

## We want to know!

For brave souls: <https://github.com/ejgallego/HOARe2>

# Needed: more case studies!

Do you have a mechanism that ...

- ▶ has a tedious proof?
- ▶ uses randomization?
- ▶ satisfies a relational property?

## We want to know!

For brave souls: <https://github.com/ejgallego/HOARe2>

(Also, I am looking for a job ...)

# Computer-aided Verification in Mechanism Design

Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias,  
Justin Hsu\*, Aaron Roth\*, Pierre-Yves Strub

IMDEA Software, École Polytechnique, University at Buffalo, \*University of  
Pennsylvania

December 14, 2016

# Writing the program

## Main program: one agent's utility

- ▶ Input: agent's true type and report
- ▶ Output: agent's expected utility from mechanism
- ▶ Assume: other agents reports drawn from prior (BIC)

## Top level code

```
def Util (othermoves, myty, mybid) =  
  E (c ~ rsmcoins) {  
    (mysur, mypay) = RStrans(c, myty, mybid);  
    myval = E {  
      for i = 1 ... n - 1:  
        sample othersurs[i] ~ (sample otherty ~ mu; othermoves[i](otherty));  
        algInput = (mysur, othersurs);  
        outcome = alg(algInput);  
        return value (myty, outcome)  
    };  
    return (myval - mypay);  
  }
```

# Handling the hard assertions

## Hardest step

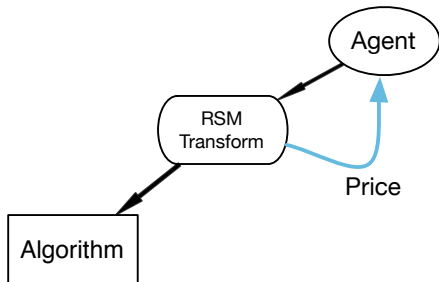
- ▶ Mechanism transforms each report into a “surrogate” report
- ▶ Key lemma: if report  $\sim$  prior, transformation preserves prior
- ▶ Manually construct proof in different system (EasyCrypt),  
~ 190 out of ~ 260 total lines of manual proof

# RSM mechanism (Hartline, Kleinberg, Malekian)

Agent

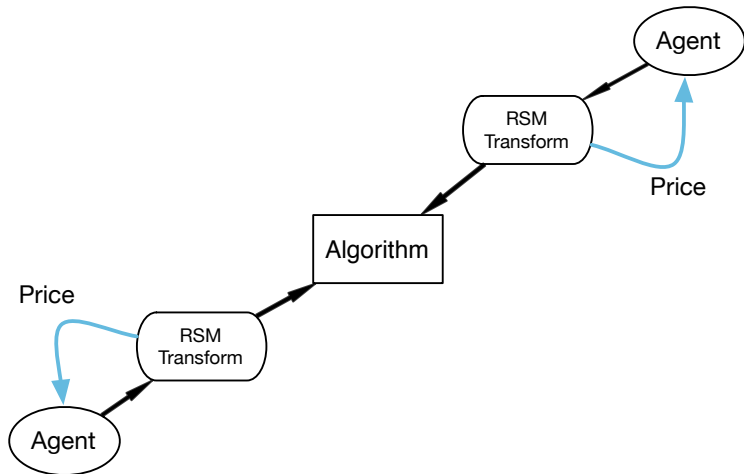
Algorithm

# RSM mechanism (Hartline, Kleinberg, Malekian)





# RSM mechanism (Hartline, Kleinberg, Malekian)



# RSM mechanism (Hartline, Kleinberg, Malekian)

