CPS 296.1
Application of Linear and Integer Programming: Automated Mechanism Design

Guest Lecture by Mingyu Guo

Mechanism design: setting

- The center has a set of outcomes O that she can choose from
  - Allocations of tasks/resources, joint plans, …
- Each agent i draws a type θ_i from Θ_i
  - usually, but not necessarily, according to some probability distribution
- Each agent has a (commonly known) valuation function $v_i: \Theta_i \times O \rightarrow \Re$
  - Note: depends on θ_i, which is not commonly known
- The center has some objective function $g: \Theta \times O \rightarrow \Re$
  - $\Theta = \Theta_1 \times ... \times \Theta_n$
  - E.g., efficiency ($\Sigma_i v_i(\theta_i, o)$)
  - May also depend on payments (more on those later)
  - The center does not know the types

What should the center do?

- She would like to know the agents' types to make the best decision
- Why not just ask them for their types?
- Problem: agents might lie
- E.g., an agent that slightly prefers outcome 1 may say that outcome 1 will give him a value of 1,000,000 and everything else will give him a value of 0, to force the decision in his favor
- But maybe, if the center is clever about choosing outcomes and/or requires the agents to make some payments depending on the types they report, the incentive to lie disappears…

Quasilinear utility functions

- For the purposes of mechanism design, we will assume that an agent's utility for
  - his type being θ_i,
  - outcome o being chosen,
  - and having to pay π_i,

  can be written as $v_i(\theta_i, o) - \pi_i$
- Such utility functions are called quasilinear
- Some of the results that we will see can be generalized beyond such utility functions, but we will not do so

Definition of a (direct-revelation) mechanism

- A deterministic mechanism without payments is a mapping $o: \Theta \rightarrow O$
- A randomized mechanism without payments is a mapping $o: \Theta \rightarrow \Delta(O)$
  - $\Delta(O)$ is the set of all probability distributions over O
- Mechanisms with payments additionally specify, for each agent i, a payment function $\pi_i : \Theta \rightarrow \Re$ (specifying the payment that that agent must make)
- Each mechanism specifies a Bayesian game for the agents, where i's set of actions $A_i = \Theta_i$
  - We would like agents to use the truth-telling strategy defined by $s(\theta_i) = \theta_i$

The Clarke (aka. VCG) mechanism [Clarke 71]

- The Clarke mechanism chooses some outcome o that maximizes $\Sigma_i v_i(\theta_i', o)$
  - θ_i' = the type that i reports
- To determine the payment that agent j must make:
  - Pretend j does not exist, and choose o_{-j} that maximizes $\Sigma_{i \neq j} v_i(\theta_i', o_{-j})$
  - j pays $\Sigma_{i \neq j} v_i(\theta_i', o_{-j}) - \Sigma_{i \neq j} v_i(\theta_i', o) = \Sigma_{i \neq j} (v_i(\theta_i', o_{-j}) - v_i(\theta_i', o))$

- We say that each agent pays the externality that she imposes on the other agents

- (VCG = Vickrey, Clarke, Groves)

Incentive compatibility

- Incentive compatibility (aka. truthfulness) = there is never an incentive to lie about one's type
- A mechanism is dominant-strategies incentive compatible (aka. strategy-proof) if for any i, for any type vector $\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_n$, and for any alternative type θ_i', we have

  $v_i(\theta_i, o(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_n)) - \pi_i(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_n) \geq$
  $v_i(\theta_i, o(\theta_1, \theta_2, \ldots, \theta_i', \ldots, \theta_n)) - \pi_i(\theta_1, \theta_2, \ldots, \theta_i', \ldots, \theta_n)$
- A mechanism is Bayes-Nash equilibrium (BNE) incentive compatible if telling the truth is a BNE, that is, for any i, for any types θ_i, θ_i',

  $\Sigma_{\theta_{-i}} P(\theta_{-i}) [v_i(\theta_i, o(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_n)) - \pi_i(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_n)] \geq$
  $\Sigma_{\theta_{-i}} P(\theta_{-i}) [v_i(\theta_i, o(\theta_1, \theta_2, \ldots, \theta_i', \ldots, \theta_n)) - \pi_i(\theta_1, \theta_2, \ldots, \theta_i', \ldots, \theta_n)]$

The Clarke mechanism is strategy-proof

- Total utility for agent j is

  $v_j(\theta_j, o) - \Sigma_{i \neq j} (v_i(\theta_i', o_{-j}) - v_i(\theta_i', o)) =$

  $v_j(\theta_j, o) + \Sigma_{i \neq j} v_i(\theta_i', o) - \Sigma_{i \neq j} v_i(\theta_i', o_{-j})$
- But agent j cannot affect the choice of o_{-j}
- Hence, j can focus on maximizing $v_j(\theta_j, o) + \Sigma_{i \neq j} v_i(\theta_i', o)$
- But mechanism chooses o to maximize $\Sigma_i v_i(\theta_i', o)$
- Hence, if $\theta_j' = \theta_j$, j's utility will be maximized!

- Extension of idea: add any term to agent j's payment that does not depend on j's reported type
- This is the family of Groves mechanisms [Groves 73]

Individual rationality

- A selfish center: “All agents must give me all their money.” – but the agents would simply not participate
  - If an agent would not participate, we say that the mechanism is not individually rational
- A mechanism is ex-post individually rational if for any i, for any type vector $\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_n$, we have

  $v_i(\theta_i, o(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_n)) - \pi_i(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_n) \geq 0$
- A mechanism is ex-interim individually rational if for any i, for any type θ_i,

  $\Sigma_{\theta_{-i}} P(\theta_{-i}) [v_i(\theta_i, o(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_n)) - \pi_i(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_n)] \geq 0$
  - i.e., an agent will want to participate given that he is uncertain about others’ types (not used as often)

Additional nice properties of the Clarke mechanism

- Ex-post individually rational, assuming:
  - An agent's presence never makes it impossible to choose an outcome that could have been chosen if the agent had not been present, and
  - No agent ever has a negative value for an outcome that would be selected if that agent were not present
- Weakly budget balanced - that is, the sum of the payments is always nonnegative - assuming:
  - If an agent leaves, this never makes the combined welfare of the other agents (not considering payments) smaller

Generalized Vickrey Auction (GVA)
(= VCG applied to combinatorial auctions)

- Example:
  - Bidder 1 bids ({A, B}, 5)
  - Bidder 2 bids ({B, C}, 7)
  - Bidder 3 bids ({C}, 3)
- Bidders 1 and 3 win, total value is 8
- Without bidder 1, bidder 2 would have won
  - Bidder 1 pays 7 - 3 = 4
- Without bidder 3, bidder 2 would have won
  - Bidder 3 pays 7 - 5 = 2
- Strategy-proof, ex-post IR, weakly budget balanced
- Vulnerable to collusion (more so than 1-item Vickrey auction)
  - E.g., add two bidders ({B}, 100), ({A, C}, 100)
  - What happens?
  - More on collusion in GVA in [Ausubel & Milgrom 06, Conitzer & Sandholm 06]

Clarke mechanism is not perfect

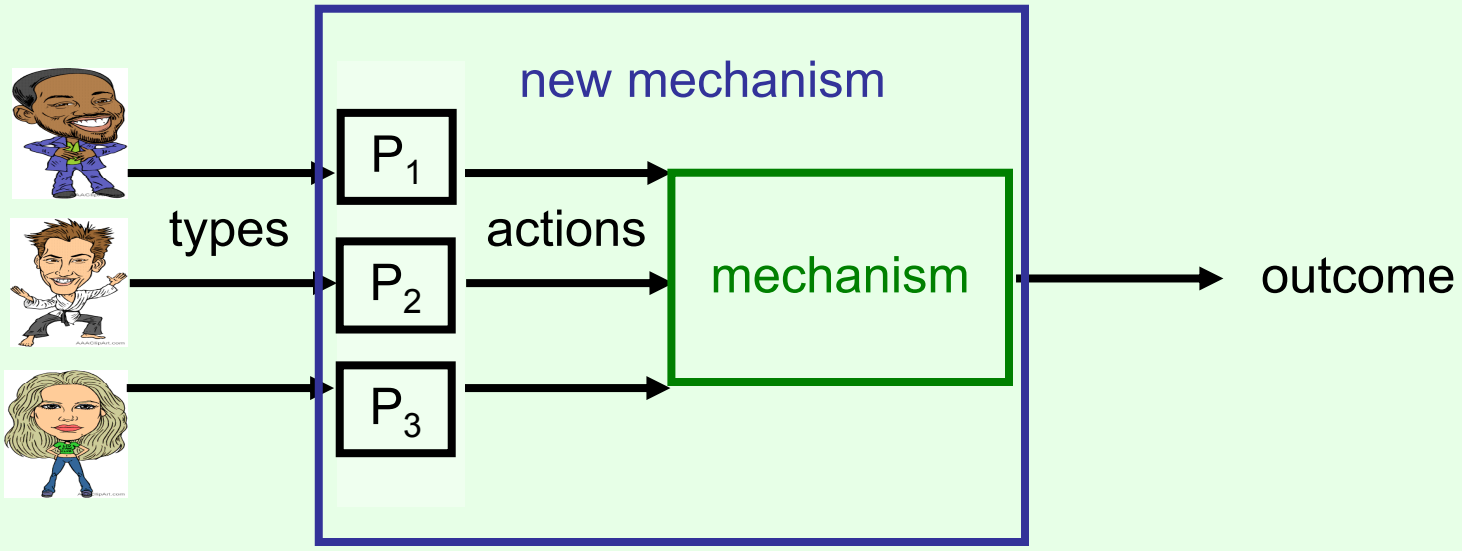
- Requires payments + quasilinear utility functions
- In general money needs to flow away from the system
  - Strong budget balance = payments sum to 0
  - In general, this is impossible to obtain in addition to the other nice properties [Green & Laffont 77]
- Vulnerable to collusion
  - E.g., suppose two agents both declare a ridiculously large value (say, $1,000,000) for some outcome, and 0 for everything else. What will happen?
- Maximizes sum of agents' utilities (if we do not count payments), but sometimes the center is not interested in this
  - E.g., sometimes the center wants to maximize revenue

Why restrict attention to truthful direct-revelation mechanisms?

- Bob has an incredibly complicated mechanism in which agents do not report types, but do all sorts of other strange things
- E.g.: Bob: “In my mechanism, first agents 1 and 2 play a round of rock-paper-scissors. If agent 1 wins, she gets to choose the outcome. Otherwise, agents 2, 3 and 4 vote over the other outcomes using the Borda rule. If there is a tie, everyone pays $100, and…”
- Bob: “The equilibria of my mechanism produce better results than any truthful direct revelation mechanism.”
- Could Bob be right?

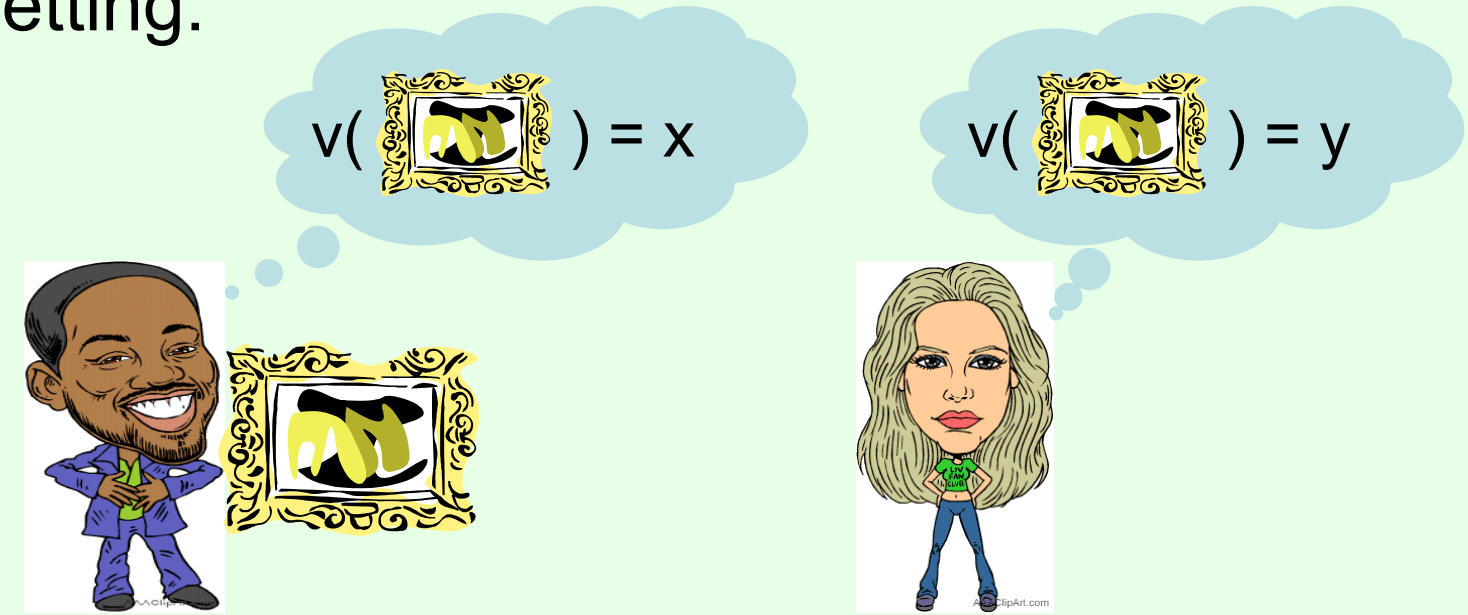
The revelation principle

- For any (complex, strange) mechanism that produces certain outcomes under strategic behavior (dominant strategies, BNE)…
- … there exists a (dominant-strategies, BNE) incentive compatible direct revelation mechanism that produces the same outcomes!



Myerson-Satterthwaite impossibility [1983]

- Simple setting:



- We would like a mechanism that:
  - is efficient (trade if and only if y > x),
  - is budget-balanced (seller receives what buyer pays),
  - is BNE incentive compatible, and
  - is ex-interim individually rational
- This is impossible!

A few computational issues in mechanism design

- Algorithmic mechanism design
  - Sometimes standard mechanisms are too hard to execute computationally (e.g., Clarke requires computing optimal outcome)
  - Try to find mechanisms that are easy to execute computationally (and nice in other ways), together with algorithms for executing them
- Automated mechanism design
  - Given the specific setting (agents, outcomes, types, priors over types, …) and the objective, have a computer solve for the best mechanism for this particular setting
- When agents have computational limitations, they will not necessarily play in a game-theoretically optimal way
  - Revelation principle can collapse; need to look at nontruthful mechanisms
- Many other things (computing the outcomes in a distributed manner; what if the agents come in over time (online setting); …)

General vs. specific mechanisms

- Mechanisms such as Clarke (VCG) mechanism are very general…
- … but will instantiate to something specific in any specific setting
  - This is what we care about

Example: Divorce arbitration

- Outcomes:



- Each agent is of *high* type w.p. .2 and *low* type w.p. .8
  - Preferences of *high* type:
    - u(get the painting) = 11,000
    - u(museum) = 6,000
    - u(other gets the painting) = 1,000
    - u(burn) = 0
  - Preferences of *low* type:
    - u(get the painting) = 1,200
    - u(museum) = 1,100
    - u(other gets the painting) = 1,000
    - u(burn) = 0

Clarke (VCG) mechanism

high

low

high

low

 Both pay 5,000	 Husband pays 200
 Wife pays 200	 Both pay 100

Expected sum of divorcees' utilities = 5,136

“Manual” mechanism design has yielded

- some positive results:
  - “Mechanism x achieves properties P in any setting that belongs to class C”
- some impossibility results:
  - “There is no mechanism that achieves properties P for all settings in class C”

Difficulties with manual mechanism design

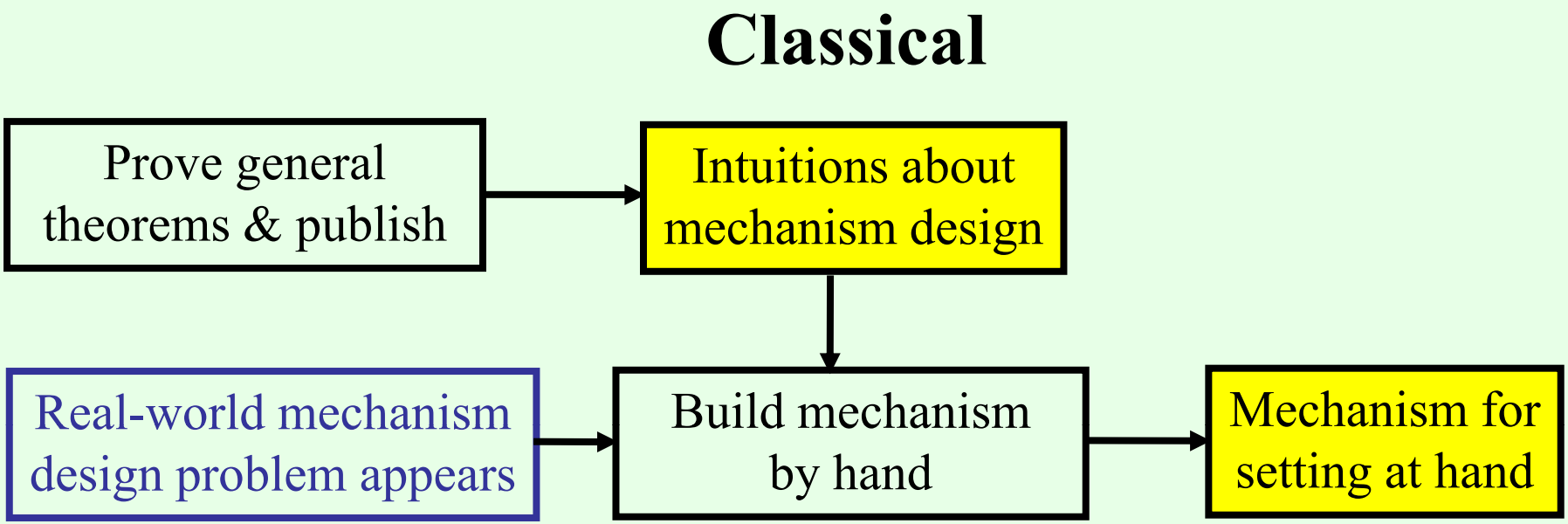
- Design problem instance comes along
  - Set of outcomes, agents, set of possible types for each agent, prior over types, …
- What if no canonical mechanism covers this instance?
  - Unusual objective, or payments not possible, or …
  - Impossibility results may exist for the general class of settings
    - But instance may have additional structure (restricted preferences or prior) so good mechanisms exist (but unknown)
- What if a canonical mechanism does cover the setting?
  - Can we use instance's structure to get higher objective value?
  - Can we get stronger nonmanipulability/participation properties?
- Manual design for every instance is prohibitively slow

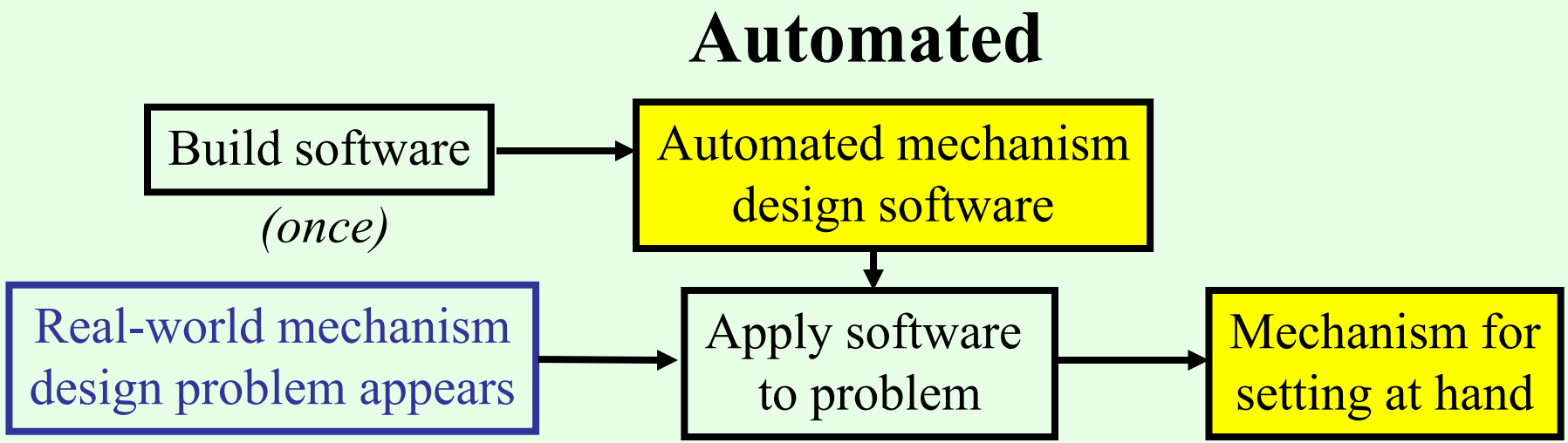
Automated mechanism design (AMD)

[Conitzer & Sandholm UAI-02, later papers]

- Idea: Solve mechanism design as optimization problem automatically
- Create a mechanism for the specific setting at hand rather than a class of settings
- Advantages:
  - Can lead to greater value of designer's objective than known mechanisms
  - Sometimes circumvents economic impossibility results & always minimizes the pain implied by them
  - Can be used in new settings & for unusual objectives
  - Can yield stronger incentive compatibility & participation properties
  - Shifts the burden of design from human to machine

Classical vs. automated mechanism design





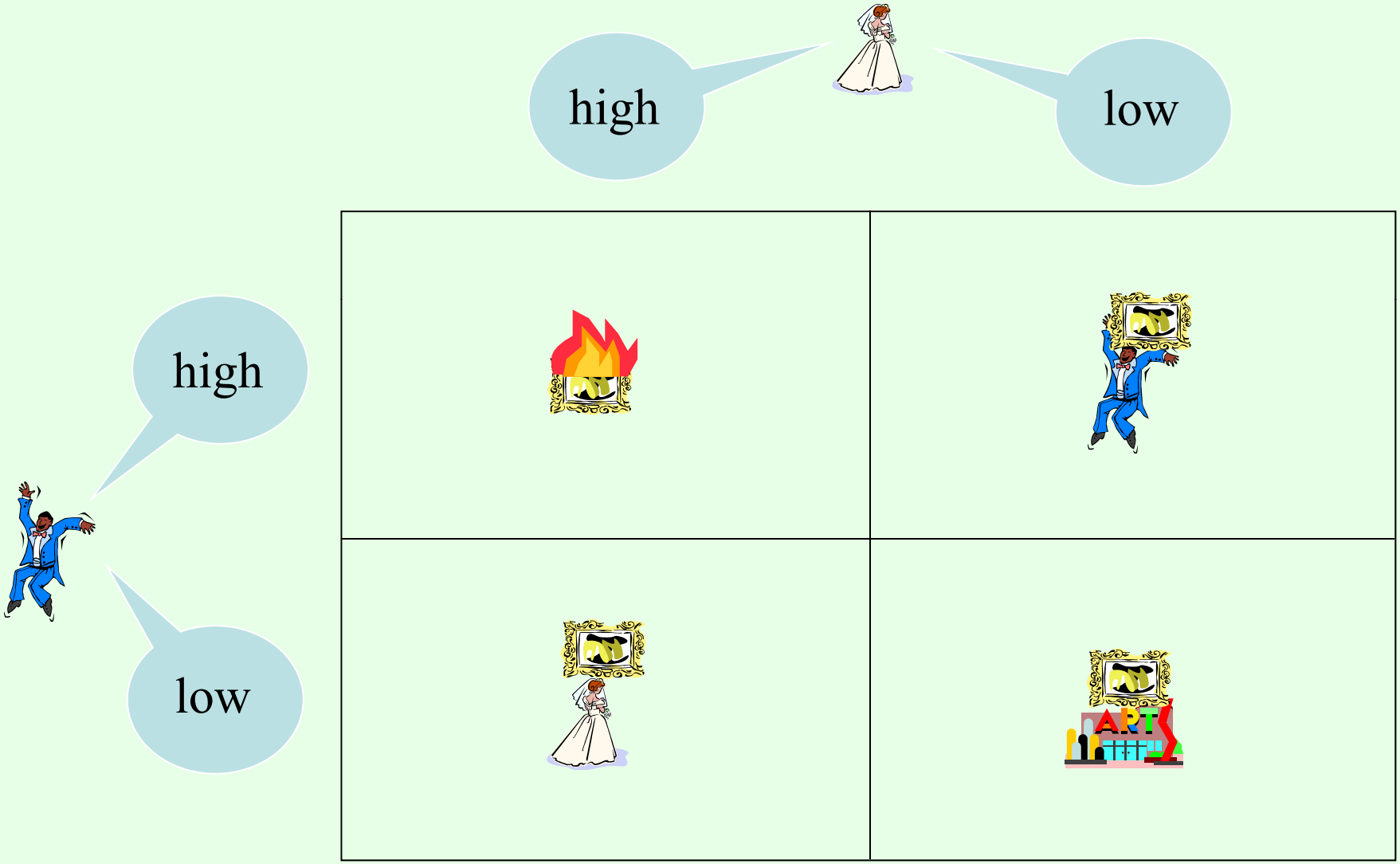
Input

- Instance is given by
  - Set of possible *outcomes*
  - Set of *agents*
    - For each agent
      - set of possible *types*
      - *probability distribution* over these types
  - *Objective function*
    - Gives a value for each outcome for each combination of agents' types
    - E.g. social welfare, payment maximization
  - Restrictions on the mechanism
    - Are payments allowed?
    - Is randomization over outcomes allowed?
    - What versions of incentive compatibility (IC) & individual rationality (IR) are used?

Output

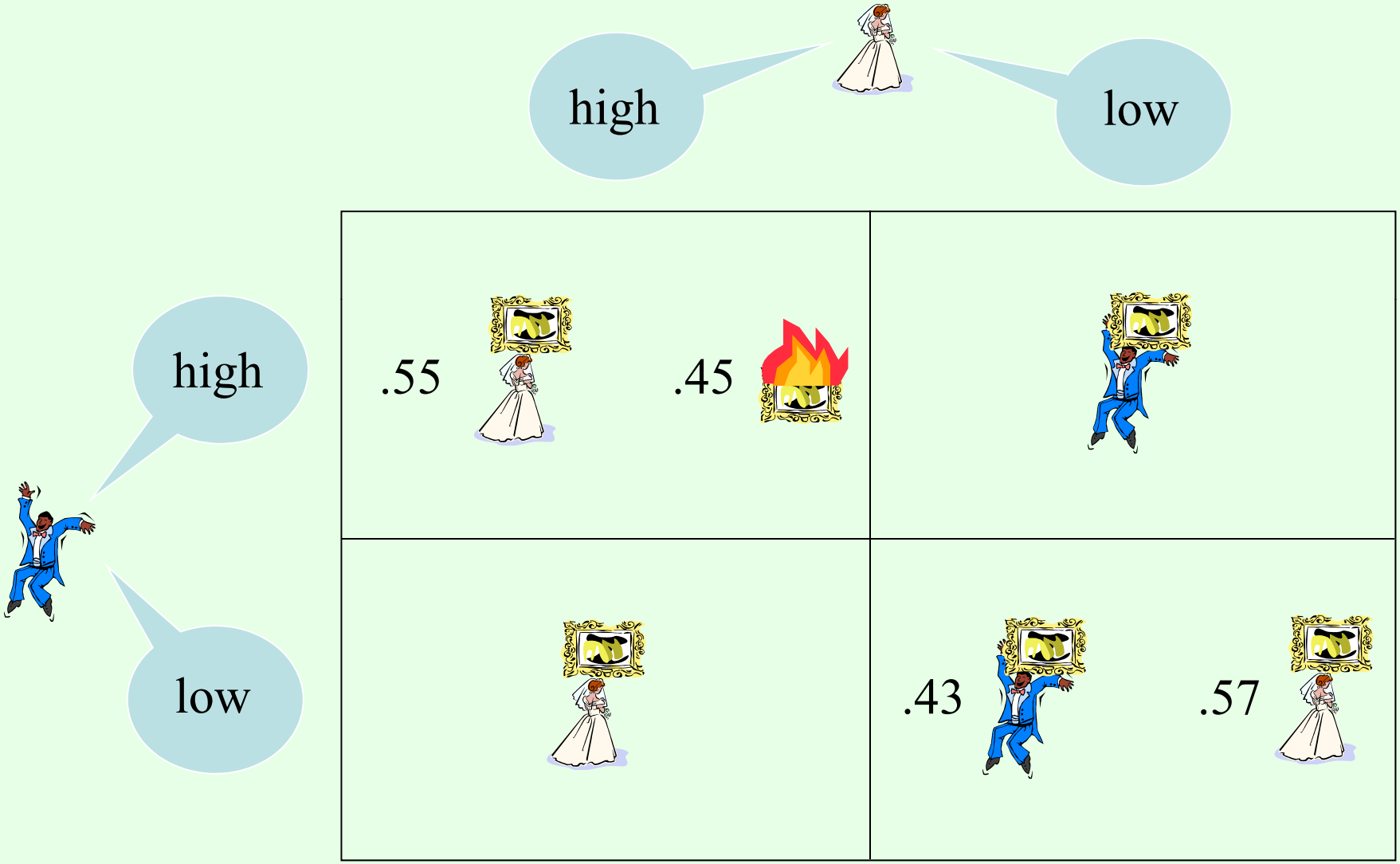
- *Mechanism*
  - A mechanism maps combinations of agents' revealed types to outcomes
    - *Randomized mechanism* maps to probability distributions over outcomes
    - Also specifies payments by agents (if payments allowed)
- … which
  - satisfies the IR and IC constraints
  - maximizes the expectation of the objective function

Optimal BNE incentive compatible deterministic mechanism without payments for maximizing sum of divorcees' utilities



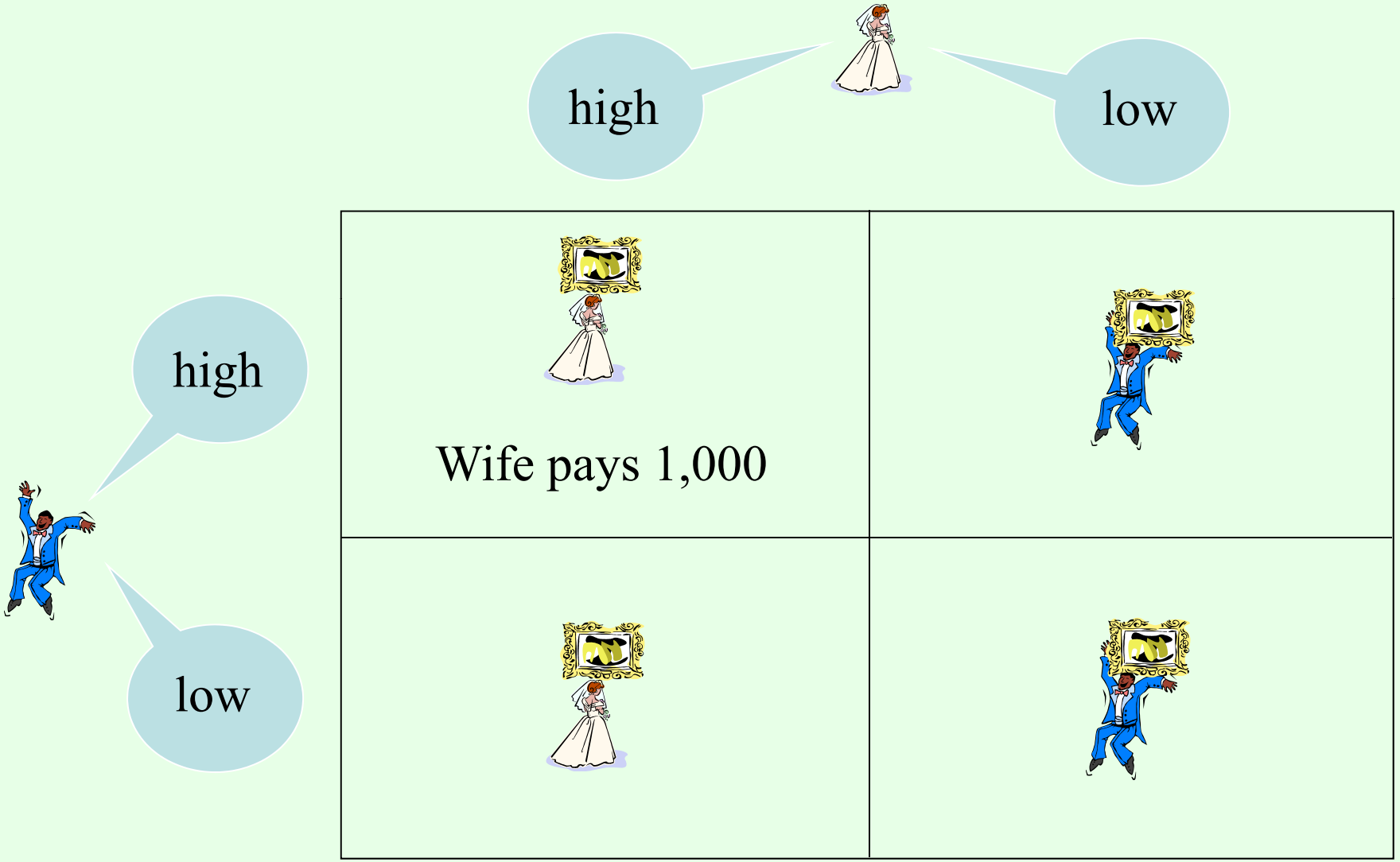
Expected sum of divorcees' utilities = 5,248

Optimal BNE incentive compatible *randomized* mechanism without payments for maximizing sum of divorcees' utilities



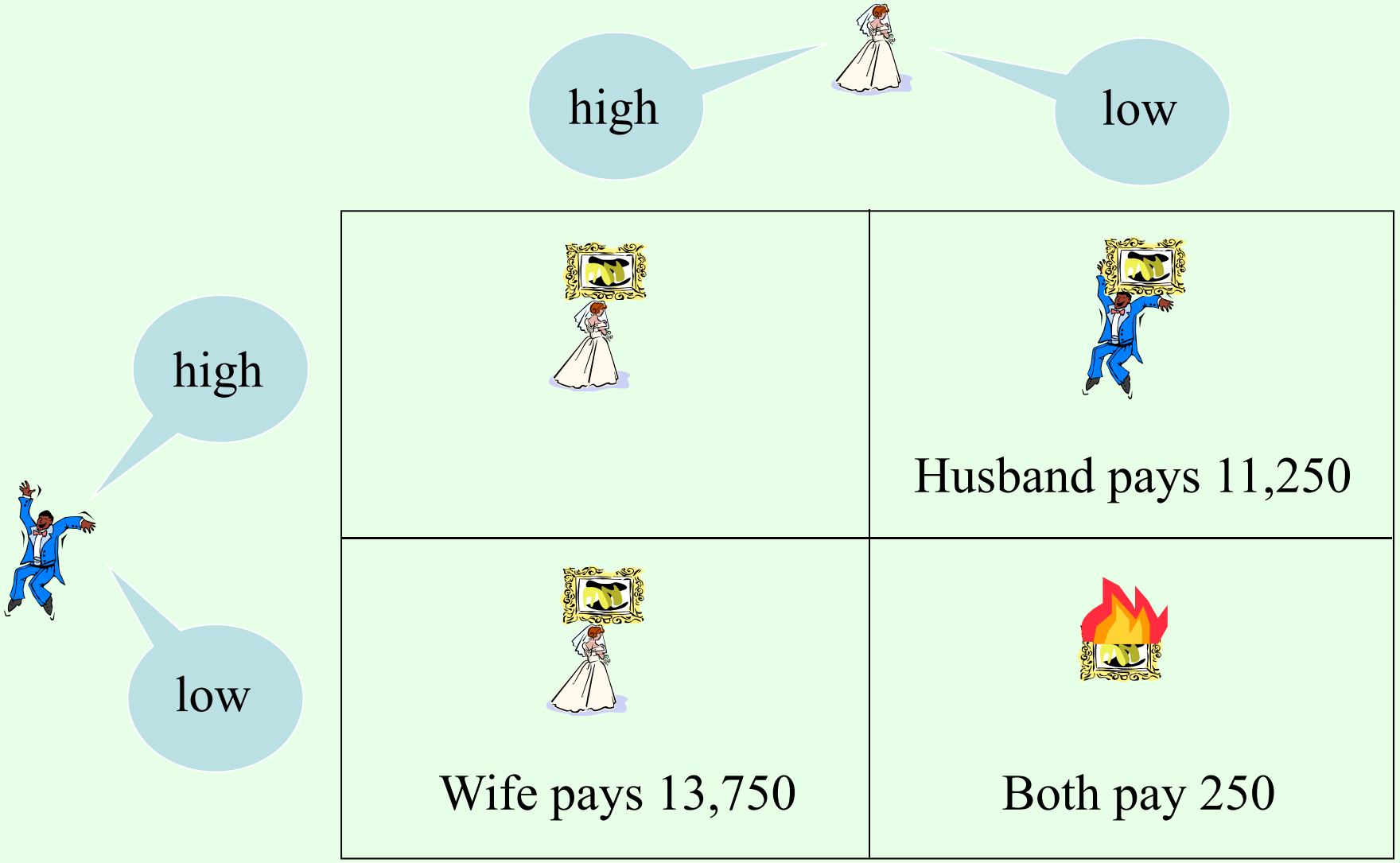
Expected sum of divorcees' utilities = 5,510

Optimal BNE incentive compatible randomized mechanism *with payments* for maximizing sum of divorcees' utilities



Expected sum of divorcees' utilities = 5,688

Optimal BNE incentive compatible randomized mechanism with payments for *maximizing arbitrator's revenue*



Expected sum of divorcees' utilities = 0

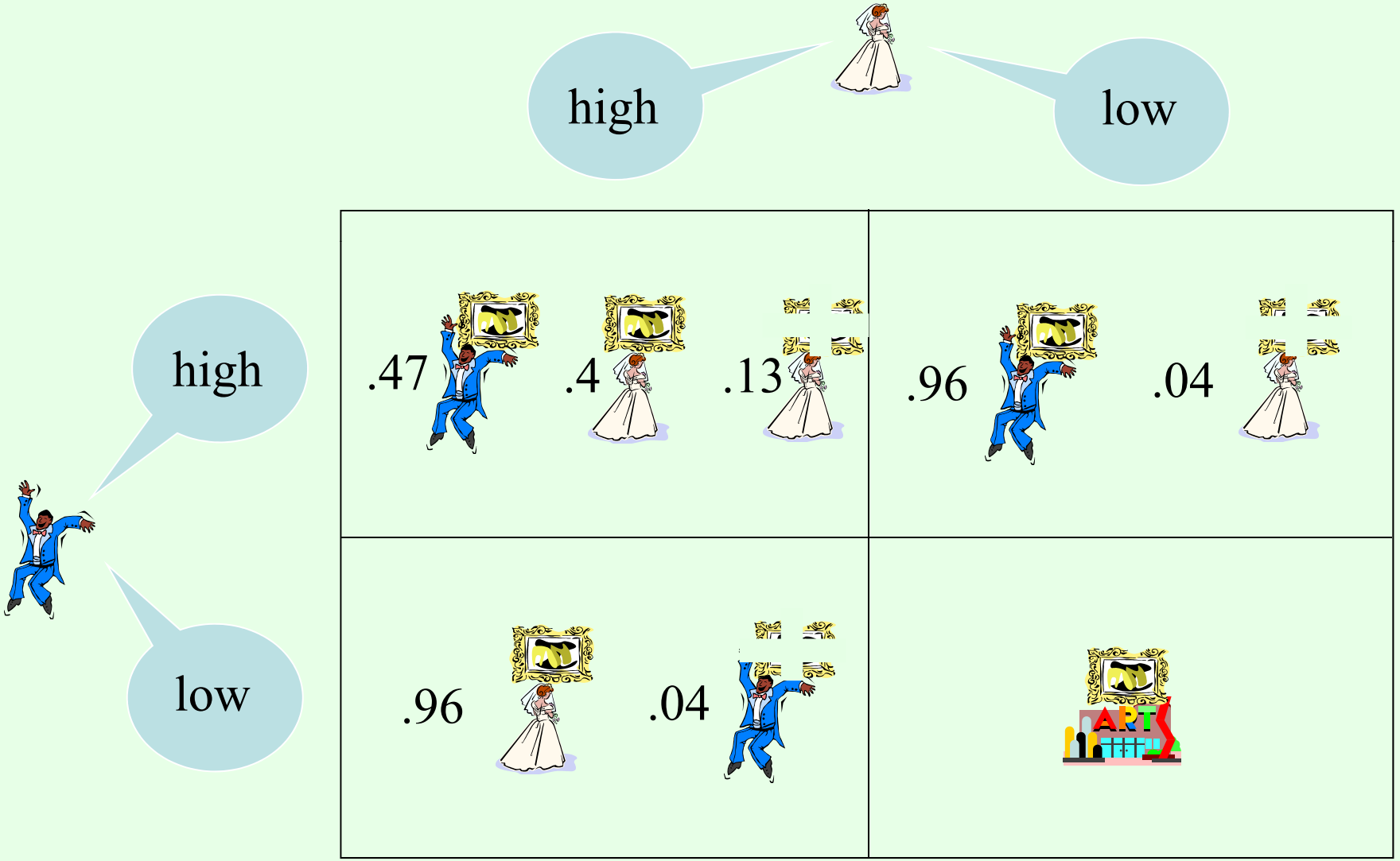
Arbitrator expects 4,320

Modified divorce arbitration example

- Outcomes: 

- Each agent is of *high* type with probability 0.2 and of *low* type with probability 0.8
  - Preferences of *high* type:
    - u(get the painting) = 100
    - u(other gets the painting) = 0
    - u(museum) = 40
    - u(get the pieces) = -9
    - u(other gets the pieces) = -10
  - Preferences of *low* type:
    - u(get the painting) = 2
    - u(other gets the painting) = 0
    - u(museum) = 1.5
    - u(get the pieces) = -9
    - u(other gets the pieces) = -10

Optimal *dominant-strategies* incentive compatible randomized mechanism for maximizing expected sum of utilities



How do we set up the optimization?

- Use linear programming
- Variables:
  - $p(o \mid \theta_1, \ldots, \theta_n)$ = probability that outcome o is chosen given types $\theta_1, \ldots, \theta_n$
  - (maybe) $\pi_i(\theta_1, \ldots, \theta_n)$ = i's payment given types $\theta_1, \ldots, \theta_n$
- Strategy-proofness constraints: for all $i, \theta_1, \ldots \theta_n, \theta_i'$:

  $\Sigma_o p(o \mid \theta_1, \ldots, \theta_n) u_i(\theta_i, o) + \pi_i(\theta_1, \ldots, \theta_n) \geq$

  $\Sigma_o p(o \mid \theta_1, \ldots, \theta_i', \ldots, \theta_n) u_i(\theta_i, o) + \pi_i(\theta_1, \ldots, \theta_i', \ldots, \theta_n)$
- Individual-rationality constraints: for all $i, \theta_1, \ldots \theta_n$:

  $\Sigma_o p(o \mid \theta_1, \ldots, \theta_n) u_i(\theta_i, o) + \pi_i(\theta_1, \ldots, \theta_n) \geq 0$
- Objective (e.g. sum of utilities)

  $\Sigma_{\theta_1, \ldots, \theta_n} p(\theta_1, \ldots, \theta_n) \Sigma_i (\Sigma_o p(o \mid \theta_1, \ldots, \theta_n) u_i(\theta_i, o) + \pi_i(\theta_1, \ldots, \theta_n))$
- Also works for BNE incentive compatibility, ex-interim individual rationality notions, other objectives, etc.
- For deterministic mechanisms, use mixed integer programming (probabilities in $\{0, 1\}$)
  - Typically designing the optimal deterministic mechanism is NP-hard

Computational complexity of automatically designing deterministic mechanisms

- Many different variants
  - Objective to maximize: Social welfare/revenue/designer's agenda for outcome
  - Payments allowed/not allowed
  - IR constraint: ex interim IR/ex post IR/no IR
  - IC constraint: Dominant strategies/Bayes-Nash equilibrium
- The above already gives 3 * 2 * 3 * 2 = 36 variants
- Approach: Prove hardness for the case of only 1 type-reporting agent
  - results imply hardness in more general settings

DSE & BNE incentive compatibility constraints coincide when there is only 1 (reporting) agent

Dominant strategies:

Reporting truthfully is optimal for *any* types the others report

	t_{21}	t_{22}
t_{11}	o_5	o_9
t_{12}	o_3	o_2

$u_1(t_{11},o_5) \geq u_1(t_{11},o_3)$

AND

$u_1(t_{11},o_9) \geq u_1(t_{11},o_2)$

Bayes-Nash equilibrium:

Reporting truthfully is optimal *in expectation* over the other agents' (true) types

	t_{21}	t_{22}
t_{11}	o_5	o_9
t_{12}	o_3	o_2

$P(t_{21})u_1(t_{11},o_5) +$
$P(t_{22})u_1(t_{11},o_9) \geq$
$P(t_{21})u_1(t_{11},o_3) +$
$P(t_{22})u_1(t_{11},o_2)$

With only 1 reporting agent, the constraints are the same

	t_{21}
t_{11}	o_5
t_{11}	o_3

$u_1(t_{11},o_5) \geq u_1(t_{11},o_3)$

is equivalent to

$P(t_{21})u_1(t_{11},o_5) \geq P(t_{21})u_1(t_{11},o_3)$

Ex post and *ex interim* individual rationality constraints coincide when there is only 1 (reporting) agent

Ex post:

Participating never hurts (for *any* types of the other agents)

	t_{21}	t_{22}
t_{11}	o_5	o_9
t_{12}	o_3	o_2

$u_1(t_{11},o_5) \geq 0$
AND
$u_1(t_{11},o_9) \geq 0$

Ex interim:

Participating does not hurt *in expectation* over the other agents' (true) types

	t_{21}	t_{22}
t_{11}	o_5	o_9
t_{12}	o_3	o_2

$P(t_{21})u_1(t_{11},o_5) +$
$P(t_{22})u_1(t_{11},o_9) \geq 0$

With only 1 reporting agent, the constraints are the same

	t_{21}
t_{11}	o_5
t_{11}	o_3

$u_1(t_{11},o_5) \geq 0$
is equivalent to
$P(t_{21})u_1(t_{11},o_5) \geq 0$

How hard is designing an optimal *deterministic* mechanism?

NP-complete (even with 1 reporting agent):	**Solvable in polynomial time (for any constant number of agents):**
1. Maximizing social welfare (no payments) 2. Designer's own utility over outcomes (no payments) 3. General (linear) objective that doesn't regard payments 4. Expected revenue	1. Maximizing social welfare (not regarding the payments) (VCG)

1 and 3 hold even with no IR constraints

AMD can create optimal (expected-revenue maximizing) combinatorial auctions

- Instance 1
  - 2 items, 2 bidders, 4 types each (LL, LH, HL, HH)
  - H=utility 2 for that item, L=utility 1
  - But: utility 6 for getting both items if type HH (complementarity)
  - Uniform prior over types
  - Optimal *ex-interim* IR, BNE mechanism (0 = item is burned):
  - Payment rule not shown
  - Expected revenue: 3.94 (VCG: 2.69)
- Instance 2
  - 2 items, 3 bidders
  - Complementarity and substitutability
  - Took 5.9 seconds
  - Uses randomization

	LL	LH	HL	HH
LL	0,0	0,2	2,0	2,2
LH	0,1	1,2	2,1	2,2
HL	1,0	1,2	2,1	2,2
HH	1,1	1,1	1,1	1,1

Optimal mechanisms for a public good

- AMD can design optimal mechanisms for public goods, taking money burning into account as a loss
- Bridge building instance
  - Agent 1: High type (prob .6) values bridge at 10. Low: values at 1
  - Agent 2: High type (prob .4) values bridge at 11. Low: values at 2
  - Bridge costs 6 to build
- Optimal mechanism (*ex-post* IR, BNE):

Outcome rule

	Low	High
Low	Don't build	Build
High	Build	Build

Payment rule

	Low	High
Low	0, 0	0, 6
High	4, 2	.67, 5.33

- There is no general mechanism that achieves budget balance, *ex-post* efficiency, and *ex-post* IR [Myerson-Satterthwaite 83]
- However, for this instance, AMD found such a mechanism

Combinatorial public goods problems

- AMD for interrelated public goods
- Example: building a bridge and/or a boat
  - 2 agents each uniform from types: {None, Bridge, Boat, Either}
    - Type indicates which of the two would be useful to the agent
    - If something is built that is useful to you, you get 2, otherwise 0
  - Boat costs 1 to build, bridge 3
- Optimal mechanism (*ex-post* IR, dominant strategies):

Outcome rule
(P(none), P(boat), P(bridge), P(both))

	None	Boat	Bridge	Either
None	(1,0,0,0)	(0,1,0,0)	(1,0,0,0)	(0,1,0,0)
Boat	(.5,.5,0,0)	(0,1,0,0)	(0,.5,0,.5)	(0,1,0,0)
Bridge	(1,0,0,0)	(0,1,0,0)	(0,0,1,0)	(0,0,1,0)
Either	(.5,.5,0,0)	(0,1,0,0)	(0,0,1,0)	(0,1,0,0)

- Again, no money burning, but outcome not always efficient
  - E.g., sometimes nothing is built while boat should have been

Additional & future directions

- Scalability is a major concern
  - Can sometimes create more concise LP formulations
    - Sometimes, some constraints are implied by others
  - In restricted domains faster algorithms sometimes exist
    - Can sometimes make use of partial characterizations of the optimal mechanism
- Automatically generated mechanisms can be complex/hard to understand
  - Can we make automatically designed mechanisms more intuitive?
- Using AMD to create conjectures about general mechanisms