Mechanism Design for Multiagent Systems

Vincent Conitzer
Assistant Professor of Computer Science and Economics
Duke University
conitzer@cs.duke.edu

Introduction

- Often, decisions must be taken based on the preferences of multiple, self-interested agents
  - Allocations of resources/tasks
  - Joint plans
  - …
- Would like to make decisions that are "good" with respect to the agents' preferences
- But, agents may lie about their preferences if this is to their benefit
- Mechanism design = creating rules for choosing the outcome that get good results nevertheless

Part I: “Classical” mechanism design

- *Preference aggregation settings*
- *Mechanisms*
- *Solution concepts*
- *Revelation principle*
- *Vickrey-Clarke-Groves mechanisms*
- *Impossibility results*

Preference aggregation settings

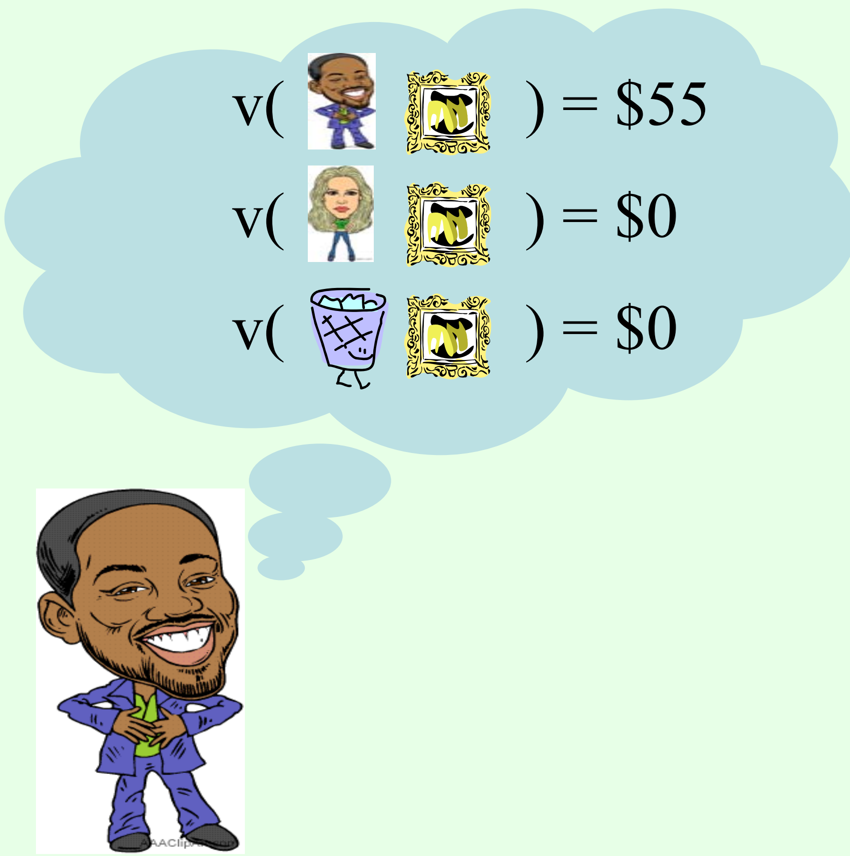
- Multiple agents…
  - humans, computer programs, institutions, …
- … must decide on one of multiple outcomes…
  - joint plan, allocation of tasks, allocation of resources, president, …
- … based on agents' preferences over the outcomes
  - Each agent knows only its own preferences
  - "Preferences" can be an ordering $\geq_i$ over the outcomes, or a real-valued utility function u_i
  - Often preferences are drawn from a commonly known distribution

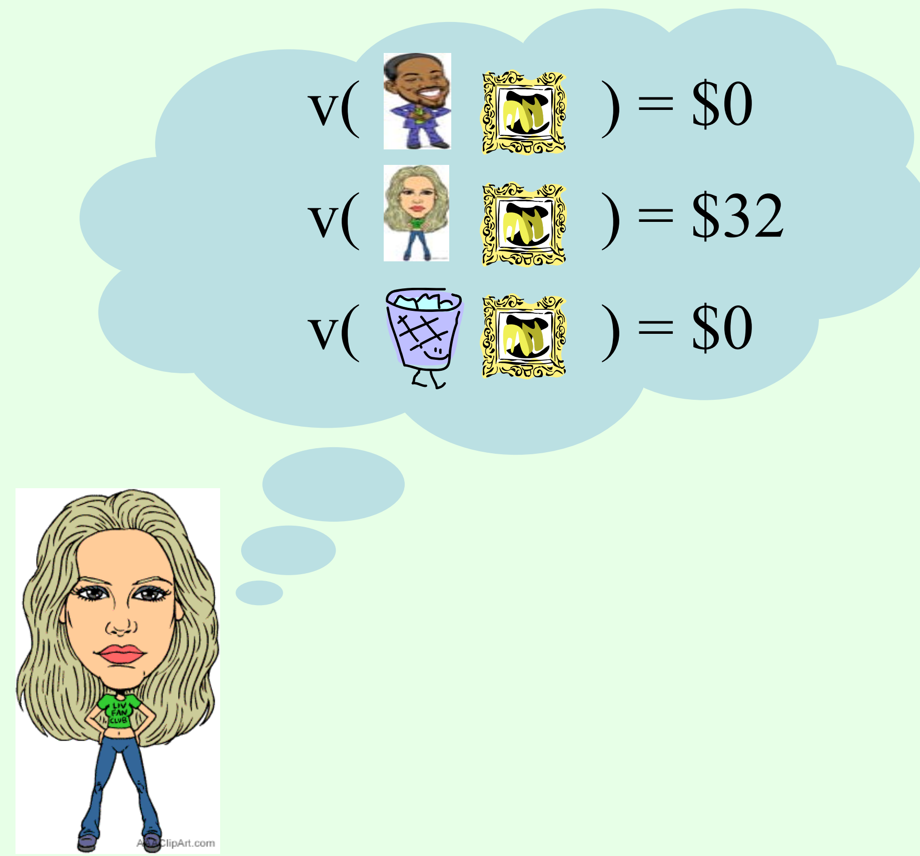
Elections

Outcome space = {  ,  ,  }

Resource allocation 

Outcome space = {  }



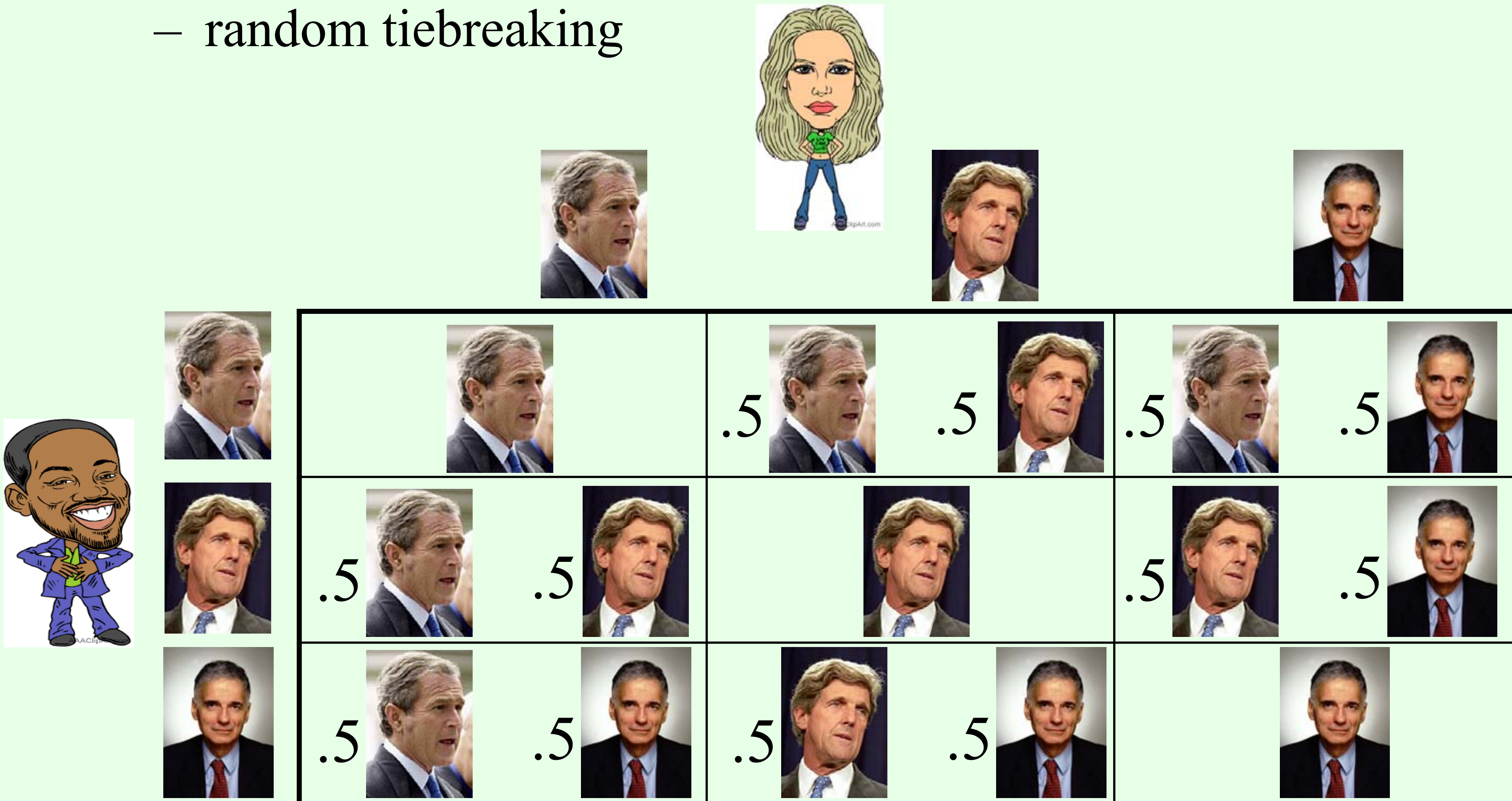


So, what is a mechanism?

- A mechanism prescribes:
  - actions that the agents can take (based on their preferences)
  - a mapping that takes all agents' actions as input, and outputs the chosen outcome
    - the "rules of the game"
    - can also output a probability distribution over outcomes
- Direct revelation mechanisms are mechanisms in which action set = set of possible preferences

Example: plurality voting

- Every agent votes for one alternative
- Alternative with most votes wins
  - random tiebreaking



Some other well-known voting mechanisms

- In all of these rules, each voter ranks all m candidates (direct revelation mechanisms)
- Other scoring mechanisms
  - Borda: candidate gets m-1 points for being ranked first, m-2 for being ranked second, …
  - Veto: candidate gets 0 points for being ranked last, 1 otherwise
- Pairwise election between two candidates: see which candidate is ranked above the other more often
  - Copeland: candidate with most pairwise victories wins
  - Maximin: compare candidates by their worst pairwise elections
  - Slater: choose overall ranking disagreeing with as few pairwise elections as possible
- Other
  - Single Transferable Vote (STV): candidate with fewest votes drops out, those votes transfer to next remaining candidate in ranking, repeat
  - Kemeny: choose overall ranking that minimizes the number of disagreements with some vote on some pair of candidates

The “matching pennies” mechanism 

- Winner of “matching pennies” gets to choose outcome

Mechanisms with payments

- In some settings (e.g. auctions), it is possible to make payments to/collect payments from the agents
- Quasilinear utility functions: $u_i(o, \pi_i) = v_i(o) + \pi_i$
- We can use this to modify agents' incentives

A few different 1-item auction mechanisms

- English auction:
  - Each bid must be higher than previous bid
  - Last bidder wins, pays last bid
- Japanese auction:
  - Price rises, bidders drop out when price is too high
  - Last bidder wins at price of last dropout
- Dutch auction:
  - Price drops until someone takes the item at that price
- Sealed-bid auctions (direct revelation mechanisms):
  - Each bidder submits a bid in an envelope
  - Auctioneer opens the envelopes, highest bid wins
    - First-price sealed-bid auction: winner pays own bid
    - Second-price sealed bid (or Vickrey) auction: winner pays second highest bid

What can we expect to happen?

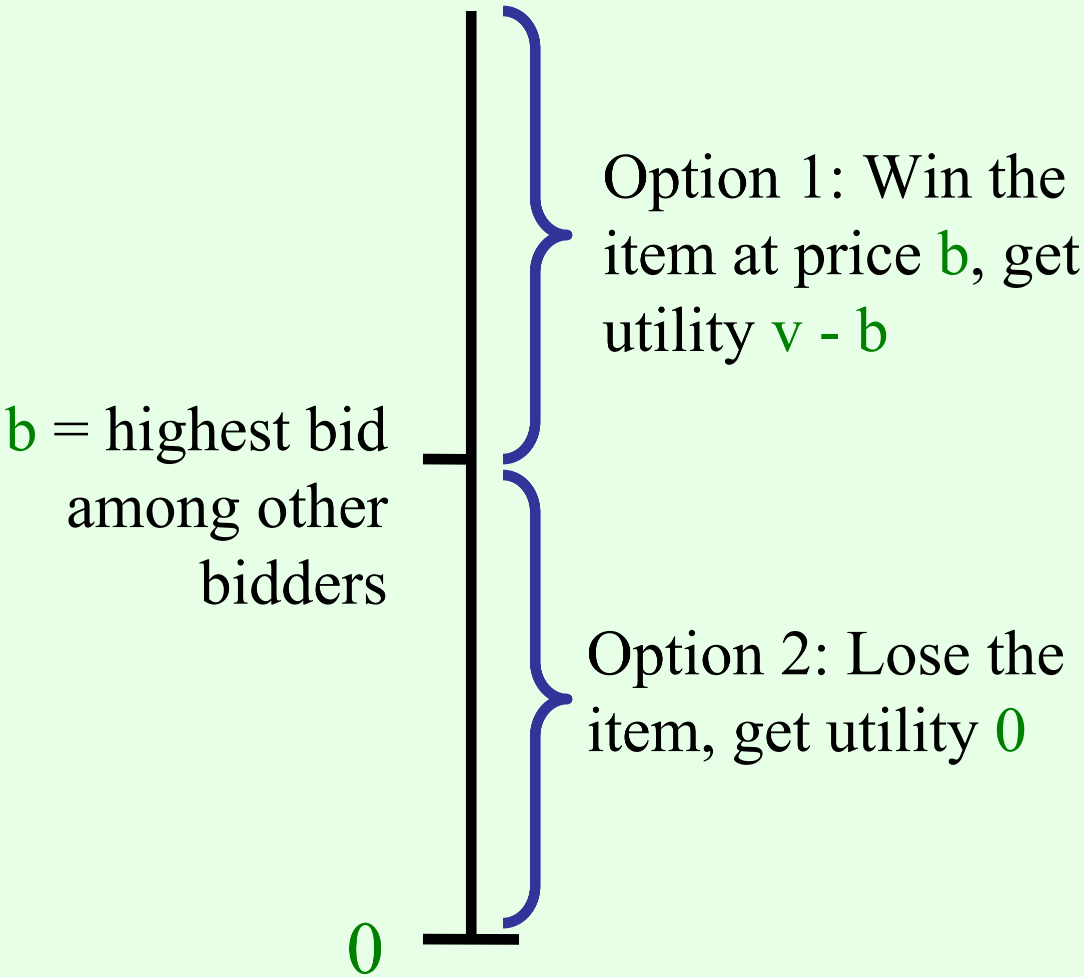
- In direct revelation mechanisms, will (selfish) agents tell the truth about their preferences?
  - Voter may not want to "waste" vote on poorly performing candidate (e.g. Nader)
  - In first-price sealed-bid auction, winner would like to bid only ε above the second highest bid
- In other mechanisms, things get even more complicated…

A little bit of game theory

- Θ_i = set of all of agent i's possible preferences ("types")
  - Notation: $u_i(\theta_i, o)$ is i's utility for o when i has type θ_i
- A strategy s_i is a mapping from types to actions
  - $s_i: \Theta_i \rightarrow A_i$
  - For direct revelation mechanism, $s_i: \Theta_i \rightarrow \Theta_i$
  - More generally, can map to distributions, $s_i: \Theta_i \rightarrow \Delta(A_i)$
- A strategy s_i is a dominant strategy if for every type θ_i, *no matter what the other agents do*, $s_i(\theta_i)$ maximizes i's utility
- A direct revelation mechanism is strategy-proof (or dominant-strategies incentive compatible) if telling the truth ($s_i(\theta_i) = \theta_i$) is a dominant strategy for all players
- (Another, weaker concept: Bayes-Nash equilibrium)

The Vickrey auction is strategy-proof!

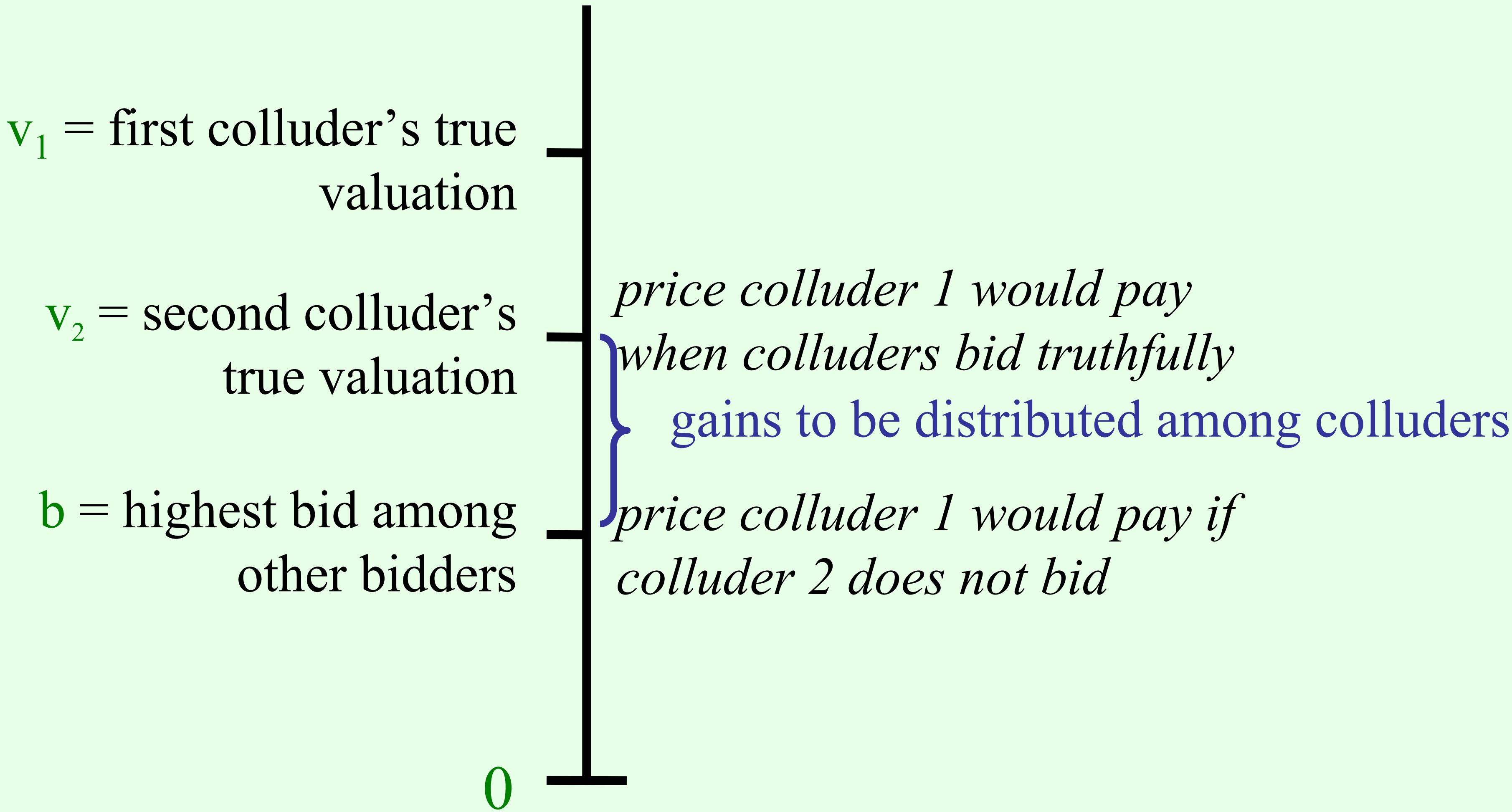
- What should a bidder with value v bid?



Would like to win if and only if v - b > 0 – but bidding truthfully accomplishes this!

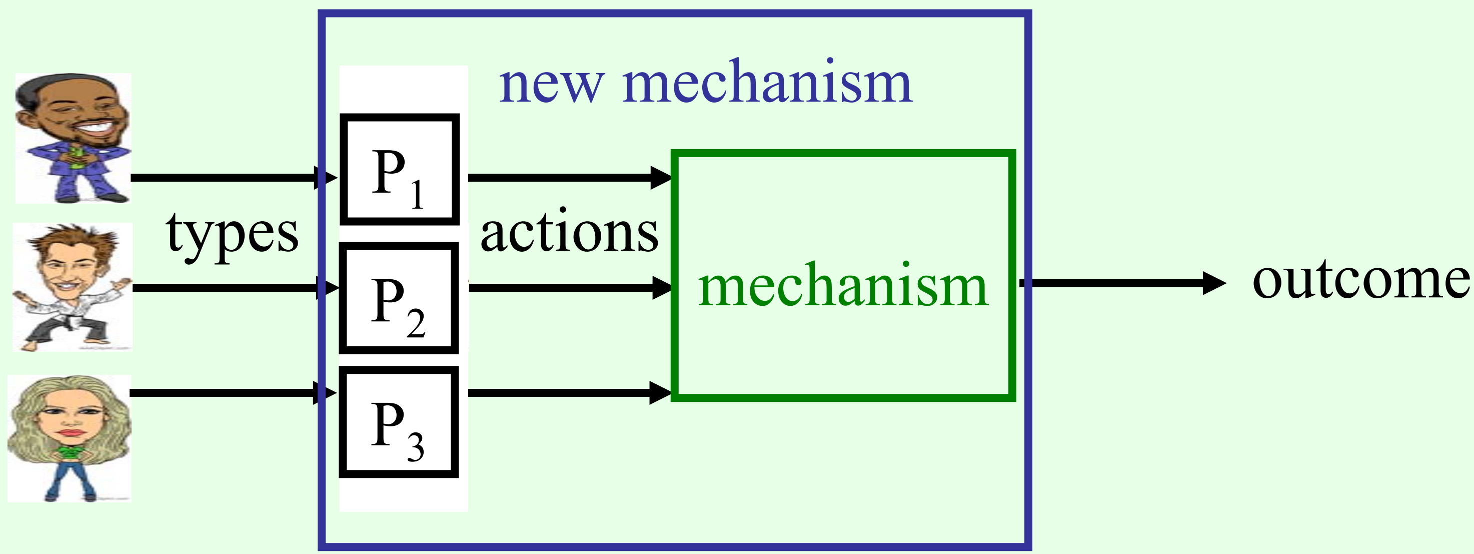
Collusion in the Vickrey auction

- Example: two colluding bidders



The revelation principle

- For any (complex, strange) mechanism that produces certain outcomes under strategic behavior…
- … there exists an incentive compatible direct revelation mechanism that produces the same outcomes!
  - “strategic behavior” = some solution concept (e.g. dominant strategies)



The Clarke mechanism [Clarke 71]

- Generalization of the Vickrey auction to arbitrary preference aggregation settings
- Agents reveal types directly
  - θ_i' is the type that i reports, θ_i is the actual type
- Clarke mechanism chooses some outcome o that maximizes $\Sigma_i\, u_i(\theta_i', o)$
- To determine the payment that agent j must make:
  - Choose o' that maximizes $\Sigma_{i\neq j}\, u_i(\theta_i', o')$
  - Make j pay $\Sigma_{i\neq j}\, (u_i(\theta_i', o') - u_i(\theta_i', o))$

- Clarke mechanism is:
  - individually rational: no agent pays more than the outcome is worth to that agent
  - (weak) budget balanced: agents pay a nonnegative amount

Why is the Clarke mechanism strategy-proof?

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

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

The Clarke mechanism is not perfect

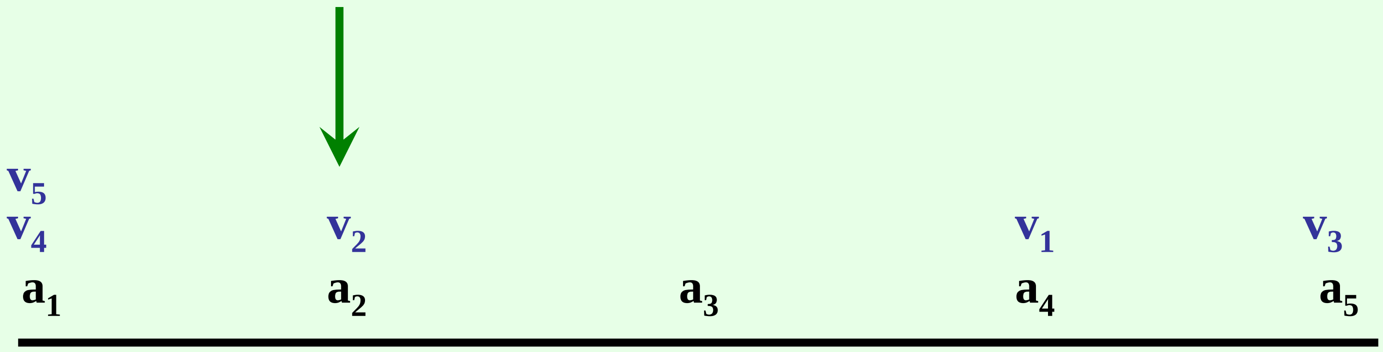
- Requires payments + quasilinear utility functions
- In general money needs to flow away from the system
- Vulnerable to collusion, false-name manipulation
- Maximizes sum of agents’ utilities, but sometimes we are not interested in this
  - E.g. want to maximize revenue

Impossibility results without payments

- Can we do without payments (voting mechanisms)?
- Gibbard-Satterthwaite [Gibbard 73, Satterthwaite 75] impossibility result: with three or more alternatives and unrestricted preferences, no voting mechanism exists that is
  - deterministic
  - strategy-proof
  - onto (every alternative can win)
  - non-dictatorial (more than one voter can affect the outcome)
- Generalization [Gibbard 77]: a randomized voting rule is strategy-proof if and only if it is a randomization over unilateral and duple rules
  - unilateral = at most one voter affects the outcome
  - duple = at most two alternatives have a possibility of winning

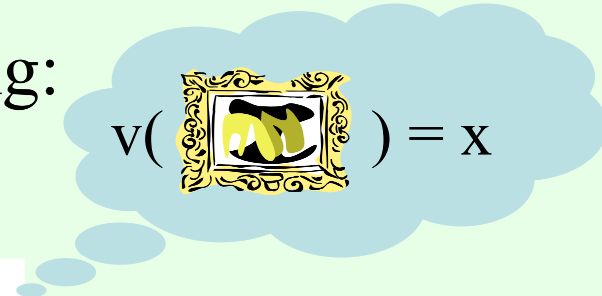
Single-peaked preferences [Black 48]

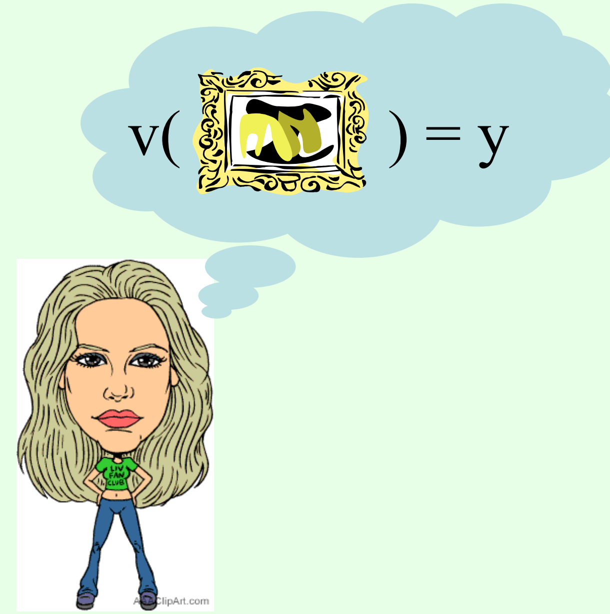
- Suppose alternatives are ordered on a line
- Every voter prefers alternatives that are closer to her most preferred alternative
- Let every voter report only her most preferred alternative (“peak”)
- Choose the median voter’s peak as the winner
- Strategy-proof!



Impossibility result with payments

- Simple setting:





- We would like a mechanism that:
  - is efficient (trade iff y > x)
  - is budget-balanced (seller receives what buyer pays)
  - is strategy-proof (or even weaker form of incentive compatible)
  - is individually rational (even just in expectation)
- This is impossible! [Myerson & Satterthwaite 83]

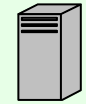
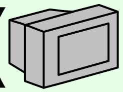
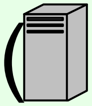
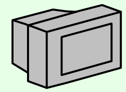
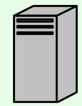
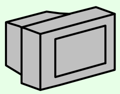
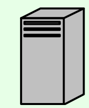
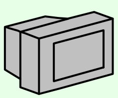
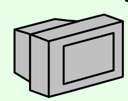
Part II: Enter the computer scientist

- *Computational hardness of executing classical mechanisms*
- *New kinds of manipulation*
- *Computationally efficient approximation mechanisms*
- *Automatically designing mechanisms using optimization software*
- *Designing mechanisms for computationally bounded agents*
- *Communication constraints*

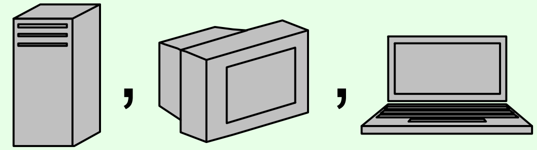
How do we compute the outcomes of mechanisms?

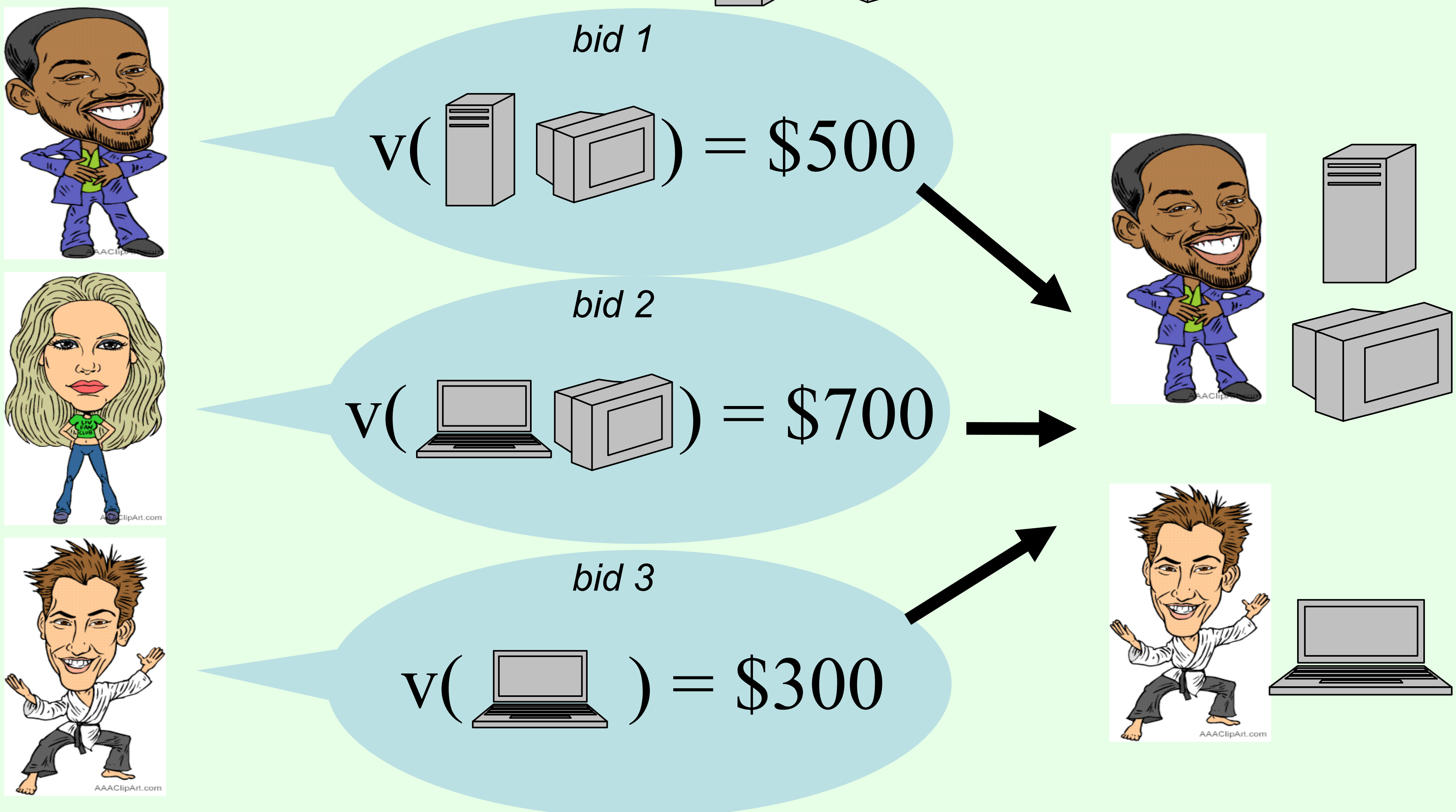
- Some voting mechanisms are NP-hard to execute (including Kemeny and Slater) [Bartholdi et al. 89, Dwork et al. 01, Ailon et al. 05, Alon 05]
  - In practice can still solve instances with fairly large numbers of alternatives [Davenport & Kalagnanam AAAI04, Conitzer et al. AAAI06, Conitzer AAAI06]
- What about Clarke mechanism? Depends on setting

Inefficiency of sequential auctions

- Suppose your valuation function is v() = \$200, v() = \$100, v( ) = \$500 (complementarity)
- Now suppose that there are two (say, Vickrey) auctions, the first one for  and the second one for 
- What should you bid in the first auction (for )?
- If you bid \$200, you may lose to a bidder who bids \$250, only to find out that you could have won  for \$200
- If you bid anything higher, you may pay more than \$200, only to find out that  sells for \$1000
- Sequential (and parallel) auctions are inefficient

Combinatorial auctions

Simultaneously for sale: 



used in truckload transportation, industrial procurement, radio spectrum allocation, …

The winner determination problem (WDP)

- Choose a subset A (the accepted bids) of the bids B,
- to maximize $\Sigma_{b \text{ in } A} v_b$,
- under the constraint that every item occurs at most once in A
  - This is assuming free disposal, i.e. not everything needs to be allocated

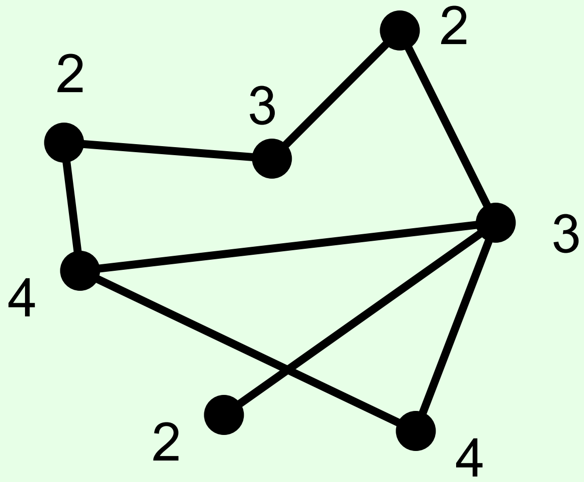
WDP example

- Items A, B, C, D, E
- Bids:
- ({A, C, D}, 7)
- ({B, E}, 7)
- ({C}, 3)
- ({A, B, C, E}, 9)
- ({D}, 4)
- ({A, B, C}, 5)
- ({B, D}, 5)

An integer program formulation

- x_b equals 1 if bid b is accepted, 0 if it is not
  - maximize $\Sigma_b v_b x_b$
  - subject to
    - for each item j, $\Sigma_{b:\, j \text{ in } b}\, x_b \leq 1$
- If each x_b can take any value in [0, 1], we say that bids can be partially accepted
- In this case, this is a linear program that can be solved in polynomial time
- This requires that
  - each item can be divided into fractions
  - if a bidder gets a fraction f of each of the items in his bundle, then this is worth the same fraction f of his value v_b for the bundle

Weighted independent set

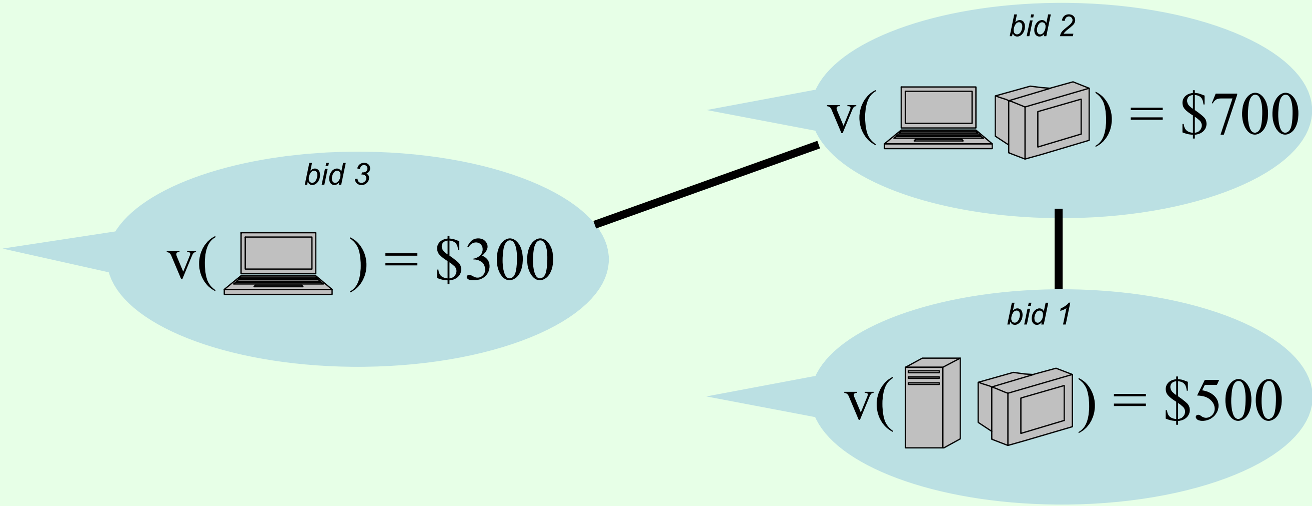


- Choose subset of the vertices with maximum total weight,
- Constraint: no two vertices can have an edge between them

- NP-hard (generalizes regular independent set)

The winner determination problem as a weighted independent set problem

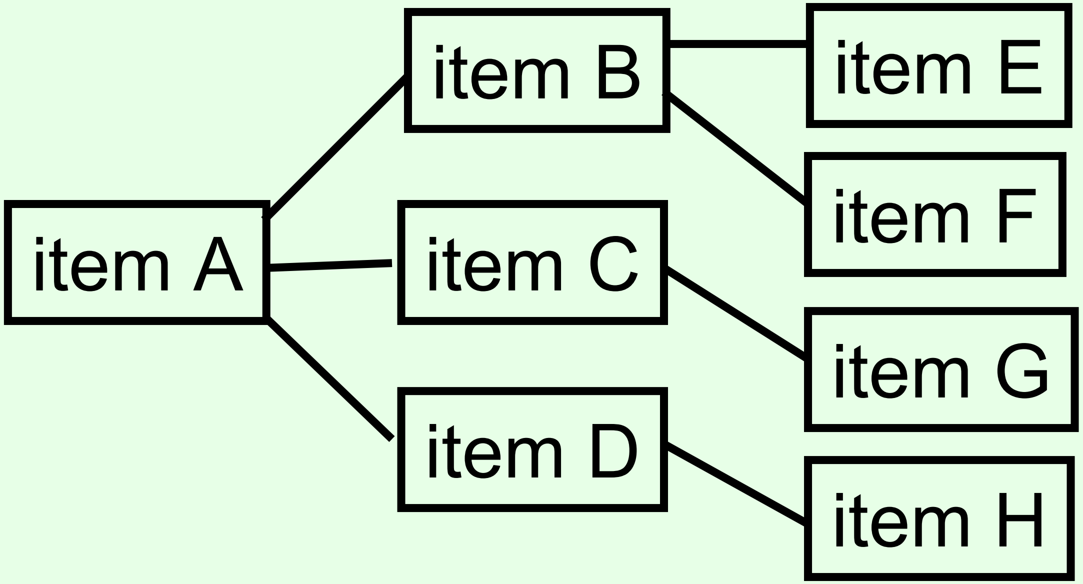
- Each bid is a vertex
- Draw an edge between two vertices if they share an item



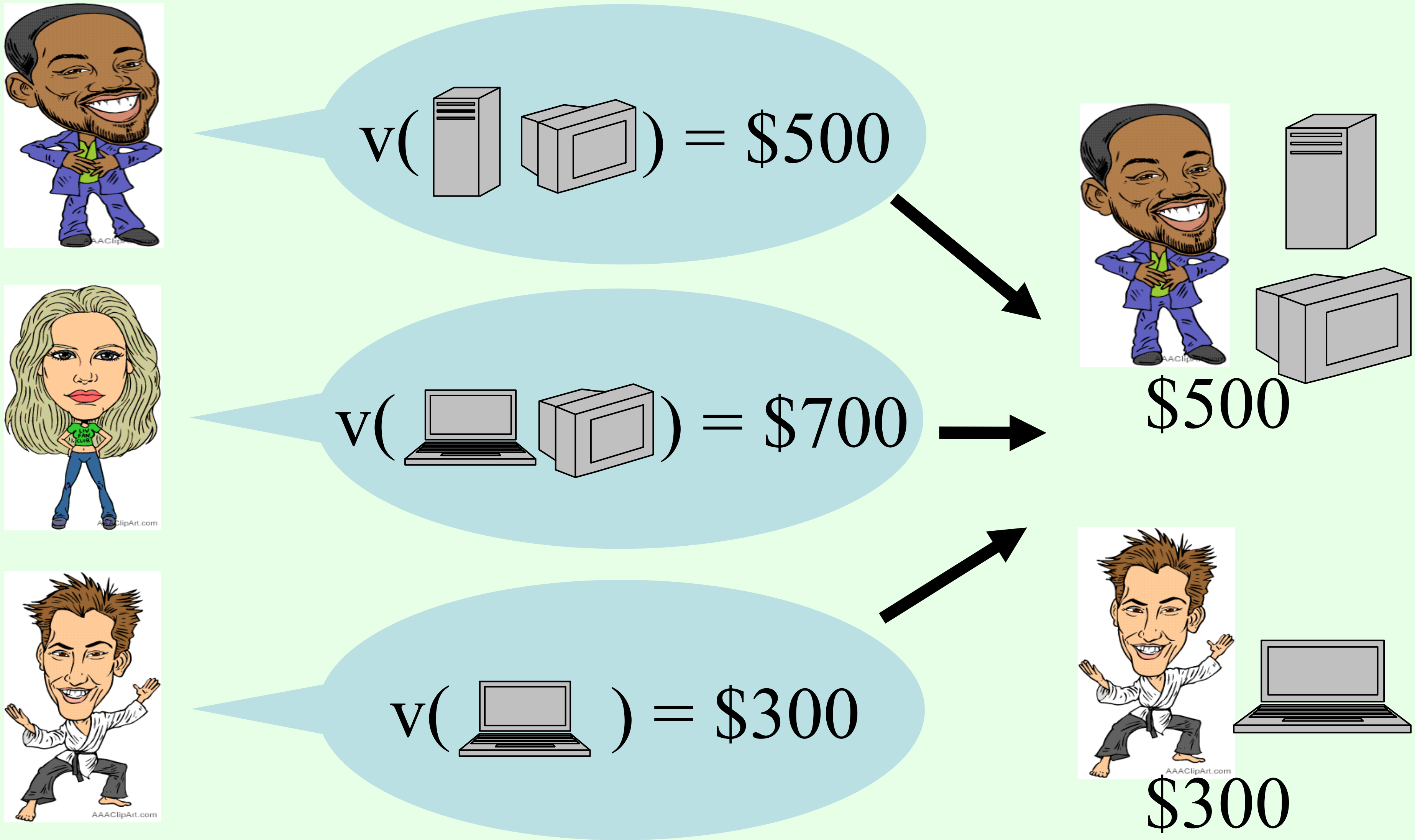
- Optimal allocation = maximum weight independent set
- Can model any weighted independent set instance as a CA winner determination problem (1 item per edge (or clique))
- Weighted independent set is NP-hard, even to solve approximately [Håstad 96] - hence, so is WDP
  - [Sandholm 02] noted that this inapproximability applies to the WDP

Polynomial-time solvable special cases

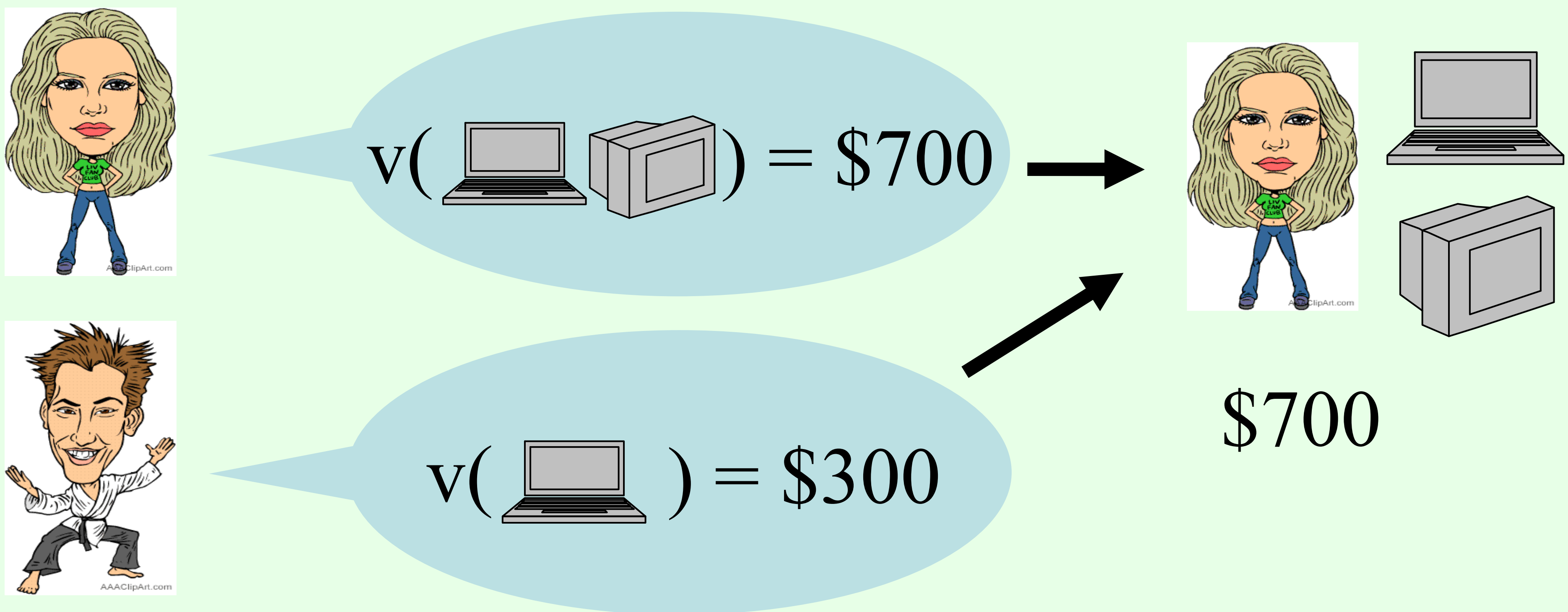
- Every bid is on a bundle of size at most two items [Rothkopf et al. 98]
  - ~maximum weighted matching
  - With 3 items per bid, NP-hard again (3-COVER)
- Items are organized on a tree & each bid is on a connected set of items [Sandholm & Suri 03]
  - More generally, graph of bounded treewidth [Conitzer et al. AAAI04]
  - Even further generalization given by [Gottlob & Greco EC07]



Clarke mechanism in CA
(aka. Generalized Vickrey Auction, GVA)

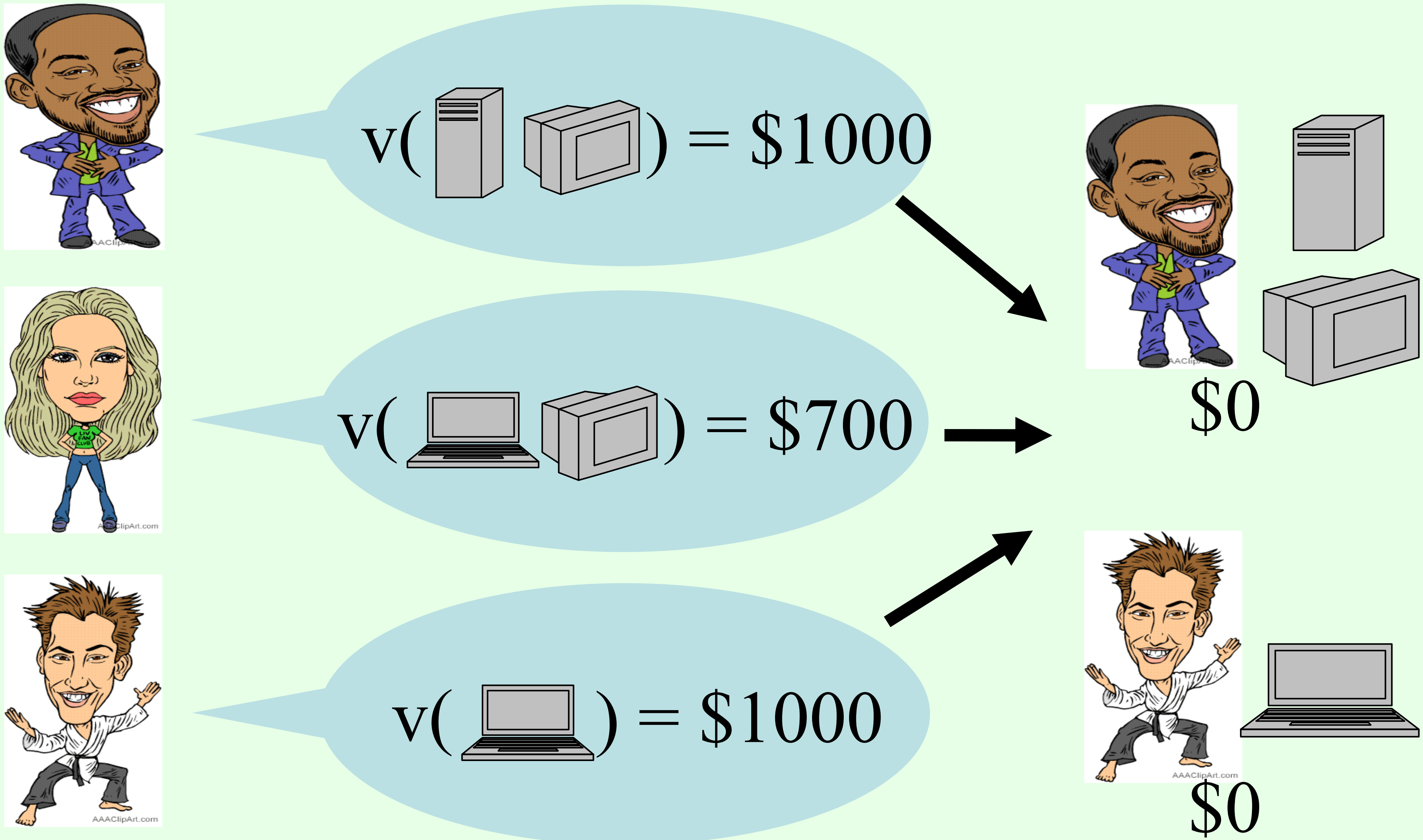


Clarke mechanism in CA…



pays $700 - $300 = $400

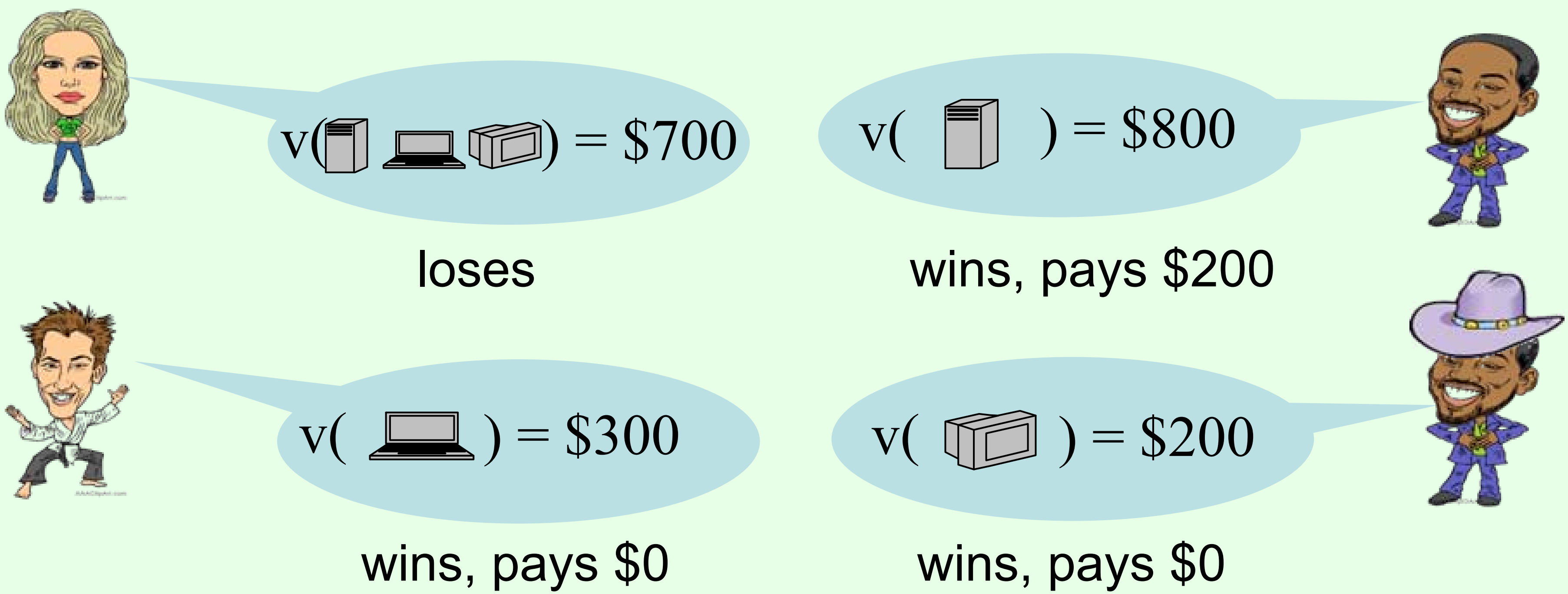
Collusion under GVA



E.g. [Ausubel and Milgrom 06]; general characterization in [Conitzer & Sandholm AAMAS06]

False-name bidding

[Yokoo et al. AIJ2001, GEB2003]



A mechanism is **false-name-proof** if bidders never have an incentive to use multiple identifiers

No mechanism that allocates items efficiently is false-name-proof
[Yokoo et al. GEB2003]

Characterization of false-name-proof voting rules

- **Theorem** [Conitzer 07]
- Any (neutral, anonymous, IR) false-name-proof voting rule f can be described by a single number k_f in [0,1]
- With probability k_f, the rule chooses an alternative uniformly at random
- With probability 1- k_f, the rule draws two alternatives uniformly at random;
  - If all votes rank the same alternative higher among the two, that alternative is chosen
  - Otherwise, a coin is flipped to decide between the two alternatives

Alternative approaches to false-name-proofness

- Assume there is a cost to using a false name [Wagman & Conitzer AAMAS08]
- Verify some of the agents' identities after the fact [Conitzer TARK07]

Strategy-proof mechanisms that solve the WDP approximately

- Running Clarke mechanism using approximation algorithms for WDP is generally not strategy-proof
- Assume bidders are single-minded (only want a single bundle)
- A greedy strategy-proof mechanism [Lehmann, O'Callaghan, Shoham JACM 03]:

1. Sort bids by (value/bundle size)
2. Accept greedily starting from top

- √ {a}, 11 — 1*(18/2) = 9
- √ {b, c}, 20 — 2*(7/1) = 14
- ✗ {a, d}, 18
- ✗ {a, c}, 16
- ✗ {c}, 7
- √ {d}, 6 — 0

3. Winning bid pays bundle size times (value/bundle size) of first bid forced out by the winning bid

Worst-case approximation ratio = (#items)

Can get a better approximation ratio, √(#items), by sorting by value/√(bundle size)

Clarke mechanism with same approximation algorithm does not work

✓ {a}, 11
✓ {b, c}, 20
✗ {a, d}, 18
✗ {a, c}, 16
✗ {c}, 7
✓ {d}, 6

Total value to bidders other than the {a} bidder: 26

✓ {b, c}, 20
✓ {a, d}, 18
✗ {a, c}, 16
✗ {c}, 7
✗ {d}, 6

Total value: 38

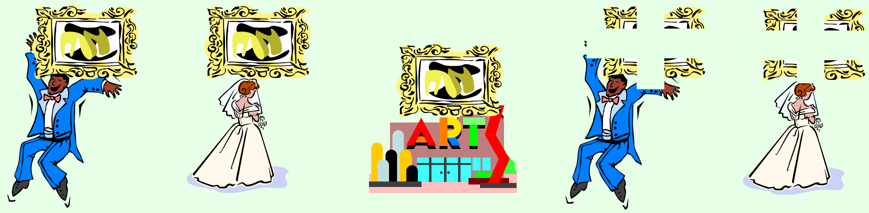
{a} bidder should pay 38 - 26 = 12, more than her valuation!

Designing mechanisms automatically

- Mechanisms such as Clarke are very general…
- … but will instantiate to something specific for specific settings
  - This is what we care about
- Different approach: solve mechanism design problem automatically for setting at hand, as an optimization problem [Conitzer & Sandholm UAI02]

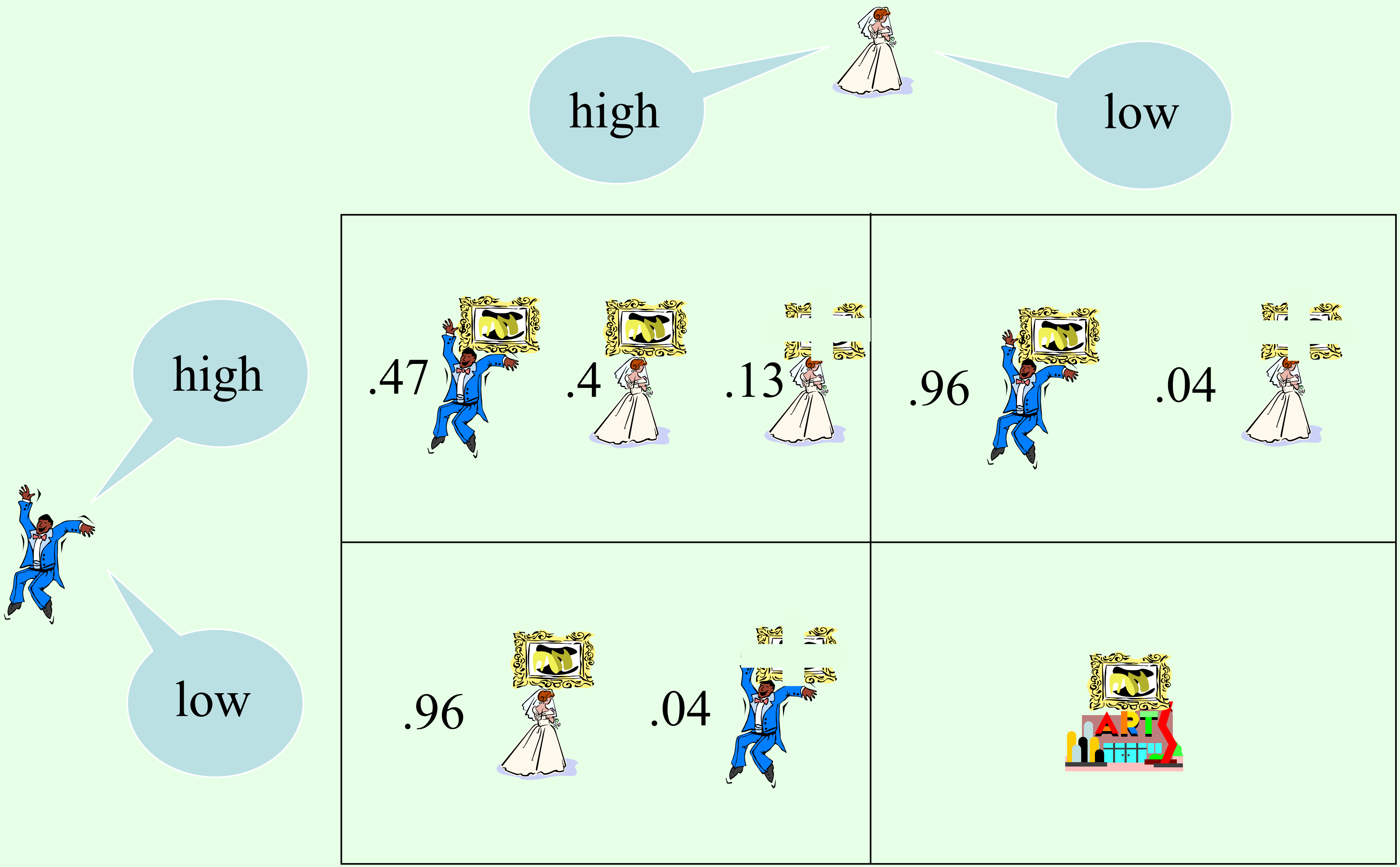
Small example: divorce arbitration

- 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 other incentive compatibility/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 limitations on the agents

- Will agents always be able to figure out what action is best for them?
- Revelation principle assumes this
  - Effectively, does the manipulation for them!
- **Theorem** [Conitzer & Sandholm 04]. There are settings where:
  - Executing the optimal (utility-maximizing) incentive compatible mechanism is NP-complete
  - There exists a non-incentive compatible mechanism, where
    - The center only carries out polynomial computation
    - Finding a beneficial insincere revelation is NP-complete for the agents
    - If the agents manage to find the beneficial insincere revelation, the new mechanism is just as good as the optimal truthful one
    - Otherwise, the new mechanism is strictly better

Hardness of manipulation of voting mechanisms

- Computing the strategically optimal vote ("manipulating") given others' votes is NP-hard for certain voting mechanisms (including STV) [Bartholdi et al. 89, Bartholdi & Orlin 91]
- Well-known voting mechanisms can be modified to make manipulation NP-hard, #P-hard, or even PSPACE-hard [Conitzer & Sandholm IJCAI03, Elkind & Lipmaa ISAAC05]
- Ideally, we would like manipulation to be usually hard, not worst-case hard
  - Several impossibility results [Procaccia & Rosenschein AAMAS06, Conitzer & Sandholm AAAI06, Friedgut et al. 07]

Preference elicitation

- Sometimes, having each agent communicate all preferences at once is impractical
- E.g. in a combinatorial auction, a bidder can have a different valuation for every bundle ($2^{\#items}-1$ values)
- Preference elicitation:
  - sequentially ask agents simple queries about their preferences,
  - until we know enough to determine the outcome

Preference elicitation (CA)

"30"

"v({A})?"

"v({A,B,C}) < 70?"

"yes"

"40"

"v({B, C})?"

center/auctioneer/ organizer/…

"What would you buy if the price for A is 30, the price for B is 20, the price for C is 20?"

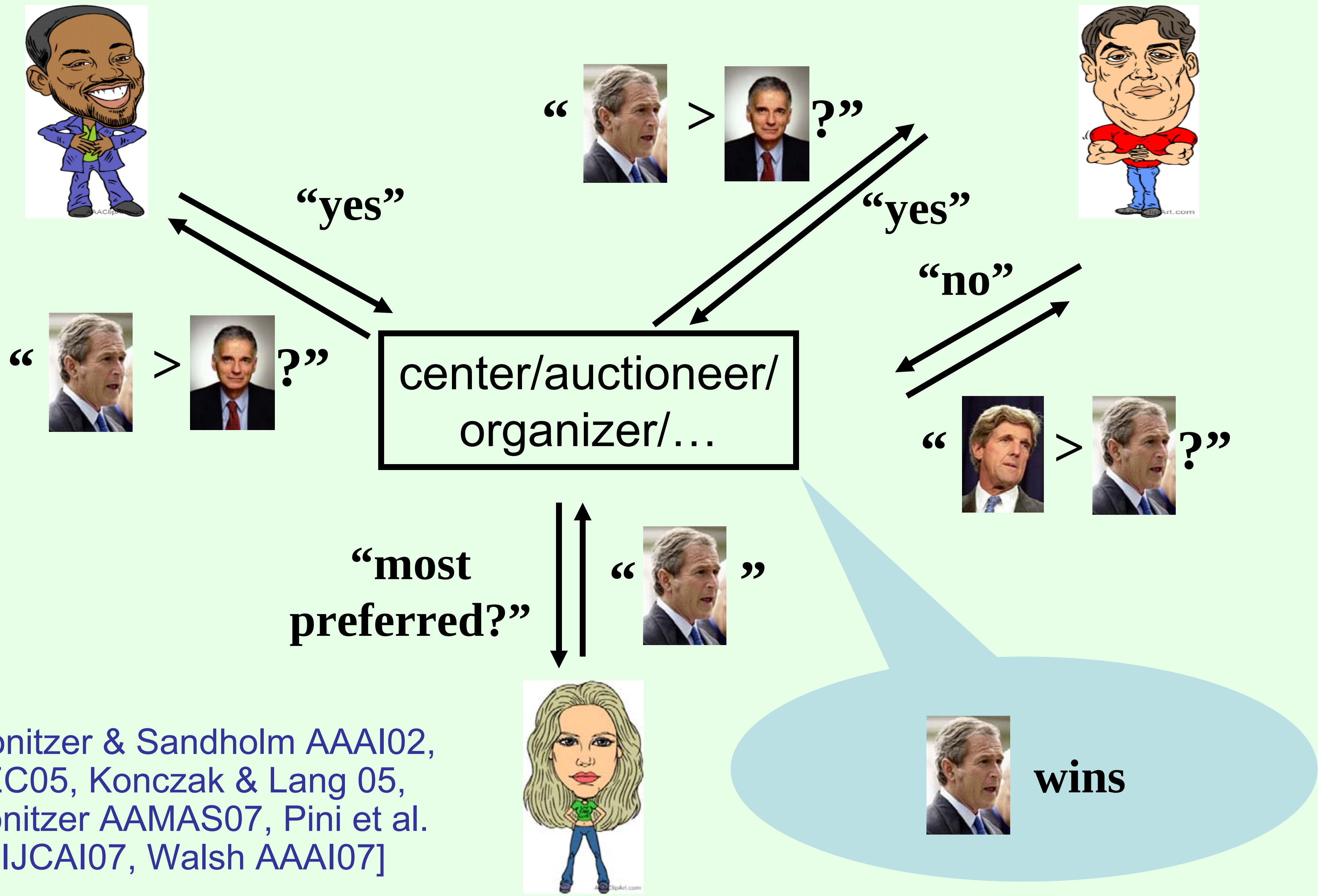
"nothing"

gets {A}, pays 30

gets {B,C}, pays 40

[Parkes, Ausubel & Milgrom, Wurman & Wellman, Blumrosen & Nisan, Conen & Sandholm, Hudson & Sandholm, Nisan & Segal, Lahaie & Parkes, Santi et al, …]

Preference elicitation (voting)



[Conitzer & Sandholm AAAI02, EC05, Konczak & Lang 05, Conitzer AAMAS07, Pini et al. IJCAI07, Walsh AAAI07]

Benefits of preference elicitation

- Less communication needed
- Agents do not always need to determine all of their preferences
  - Only where their preferences matter

Other topics

- Online mechanism design: agents arrive and depart over time [Lavi & Nisan 00, Friedman & Parkes 03, Parkes & Singh 03, Hajiaghayi et al. 04, 05, Parkes & Duong 07]
- Distributed implementation of mechanisms [Parkes & Shneidman 04, Petcu et al. 06]

Some future directions

- General principles for how to get incentive compatibility without solving to optimality
- Are there other ways of addressing false-name manipulation?
- Can we scale automated mechanism design to larger instances?
  - One approach: use domain structure (e.g. auctions [Likhodedov & Sandholm, Guo & Conitzer])
- Is there a systematic way of exploiting agents' computational boundedness?
  - One approach: have an explicit model of computational costs [Larson & Sandholm]

Thank you for your attention!