Combinatorial Auctions

Yoav Shoham

required material on auctions, posted on web page, in addition this presentation and the one of April 25

Introduction to Multi-Agent Systems (draft)

Chapter 7: Mechanism Design
sections 7.3 and 7.4

by Y. Shoham (with T. Grenager)

(Only sections 7.3 and 7.4 are required; the rest are included just in case you're curious)

What are combinatorial auctions (CAs)

- Multiple goods are auctioned simultaneously
- Each bid may claim any combination of goods
- A typical combination: a bundle ("I bid $100 for the TV, VCR and couch")
- More complex combinations are possible

Motivation: complementarity and substitutability

- Complementary goods have a superadditive utility function:
  - $v(\{a,b\}) > v(\{a\}) + v(\{b\})$
  - In the extreme, $v(\{a,b\}) >> 0$ but $v(\{a\}) = v(\{b\}) = 0$
  - Example: different segments of a flight
- Substitutable goods have a subadditive utility function:
  - $v(\{a,b\}) < v(\{a\}) + v(\{b\})$
  - In the extreme, $v(\{a,b\}) = \max [V(\{a\}) , V(\{b\})]$
  - Examples: a United ticket and a Delta ticket

Overview of Lecture

- What *can* you bid: The expressive power of different bidding languages
- What *should* you bid: A taste for the game theory of CAs
- Computational complexity of CAs

Overview of Lecture

✓ What can you bid: The expressive power of different bidding languages

- What should you bid: A taste for the game theory of CAs
- Computational complexity of CAs

Bidding Language Requirements

A bid is a declaration of a valuation function; the bidding language must be:

- Expressive
  - Enough to represent all valuation functions
- Concise
- Natural
  - Easy for humans to understand
- Tractable
  - Easy for auctioneer algorithms to handle

Unstructured bidding is impractical

- Bidder sends his entire valuation function (over all possible allocations) to auctioneer.
  - Problem: Exponential size
- Bidder sends his valuation as a computer program
  - Problem: requires exponential access by any auctioneer algorithm

The alternative: structured bidding

- The basic building block: atomic bid (implicit AND)
  - Airports: {take-off right,landing right}
- More complex:

  Spectrum: {frequency-A} XOR {frequency-B}
  Network links: {a—b,b—c} XOR {a—d,d—c}
- Adding constraints:

  PC configuration: {disk size > 10G , speed >1M/sec}
  Equality constraints: {chair , sofa} – of matching colors
  Time constraints: {truck for 2 hours , forklift for 1 hour (later)}

What are the precise syntax and semantics?

Assumptions

- No externalities
- Free disposal
- Nothing-for-nothing

Simple case: identical goods
(aka “multiple units of a single good”)

- Additive valuations: $v_i(S) = c|S|$
- Single-item valuations: $v_i(S)=c$ for all $S \neq \{\}$
- General symmetric valuations:
  - j’th item is valued as p_j
  - $v_i(S) = \Sigma_{j=1}^{|S|} p_j$
- Downward-sloping valuations: $p_j >= p_{j+1}$

The general case (distinct goods)

Atomic (“AND”) bid:

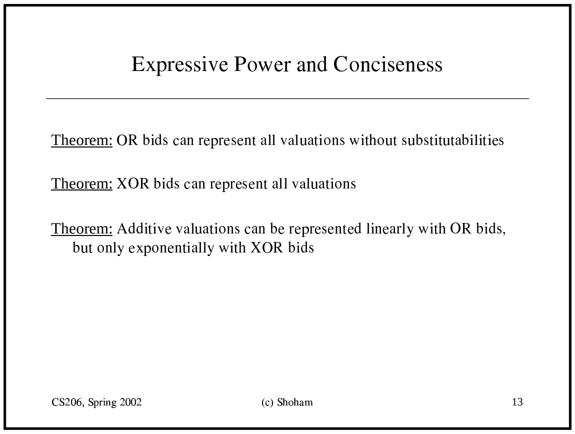
- ({left-sock, right-sock},10)
- Meaning: $v(T)=10$ if $S \subset T$; 0 otherwise

OR bid:

- ({TV,VCR},50) OR ({guitar},100) OR ({Xbox,TV},1000)
- Meaning: $(v_1 \text{ OR } v_2)(S) = \max_{R,T \subset S, R \cap T=\{\}} v_1(R) + v_2(T)$
- Note: v({TV,VCR,Xbox})=1000, not 1050

XOR bid:

- ({TV,VCR},50) XOR ({book},10) XOR ({TV,DVD},100)
- Meaning: $(v_1 \text{ XOR } v_2)(S) = \max(v_1(S),v_2(S))$

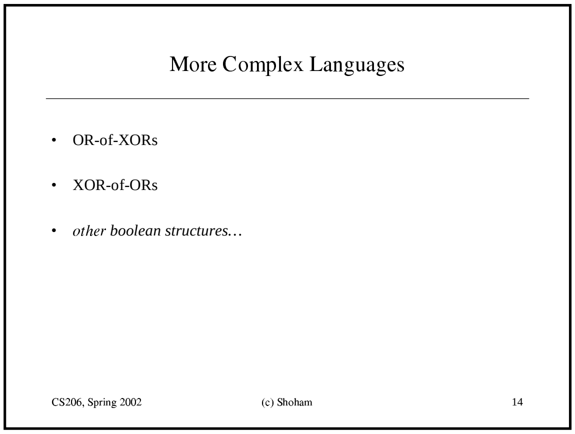


Expressive Power and Conciseness

Theorem: OR bids can represent all valuations without substitutabilities

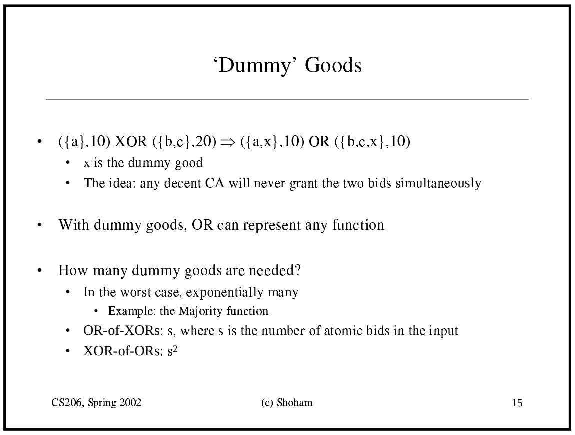
Theorem: XOR bids can represent all valuations

Theorem: Additive valuations can be represented linearly with OR bids, but only exponentially with XOR bids



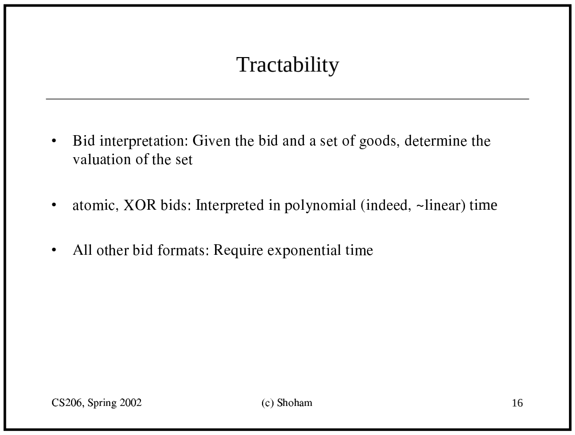
More Complex Languages

- OR-of-XORs
- XOR-of-ORs
- *other boolean structures…*



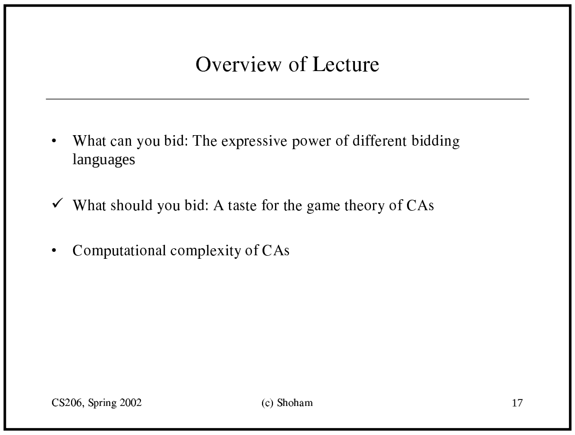
'Dummy' Goods

- ({a},10) XOR ({b,c},20) ⇒ ({a,x},10) OR ({b,c,x},10)
  - x is the dummy good
  - The idea: any decent CA will never grant the two bids simultaneously
- With dummy goods, OR can represent any function
- How many dummy goods are needed?
  - In the worst case, exponentially many
    - Example: the Majority function
  - OR-of-XORs: s, where s is the number of atomic bids in the input
  - XOR-of-ORs: s^2



Tractability

- Bid interpretation: Given the bid and a set of goods, determine the valuation of the set
- atomic, XOR bids: Interpreted in polynomial (indeed, ~linear) time
- All other bid formats: Require exponential time

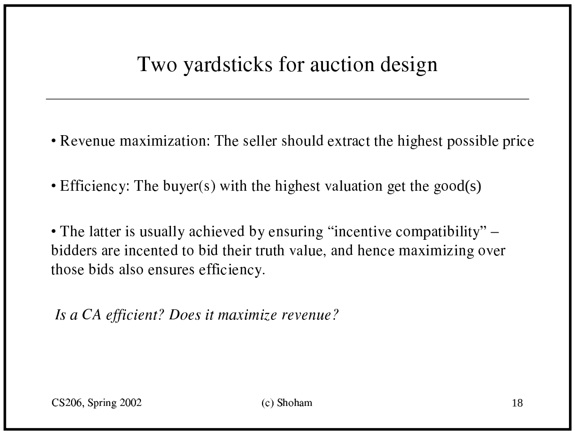


Overview of Lecture

- What can you bid: The expressive power of different bidding languages

✓ What should you bid: A taste for the game theory of CAs

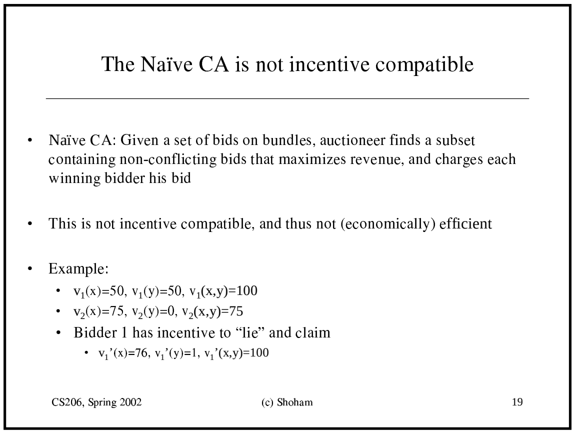
- Computational complexity of CAs



Two yardsticks for auction design

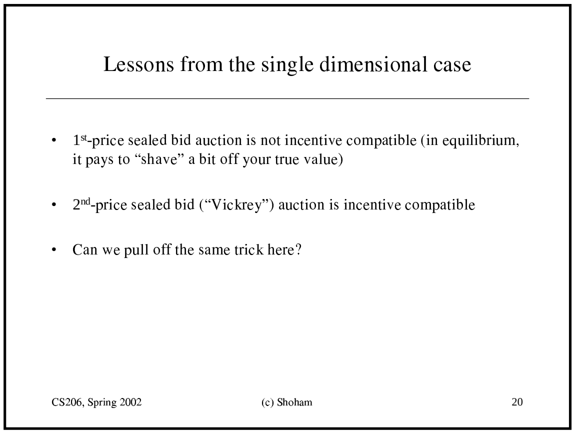
- Revenue maximization: The seller should extract the highest possible price
- Efficiency: The buyer(s) with the highest valuation get the good(s)
- The latter is usually achieved by ensuring "incentive compatibility" – bidders are incented to bid their truth value, and hence maximizing over those bids also ensures efficiency.

Is a CA efficient? Does it maximize revenue?



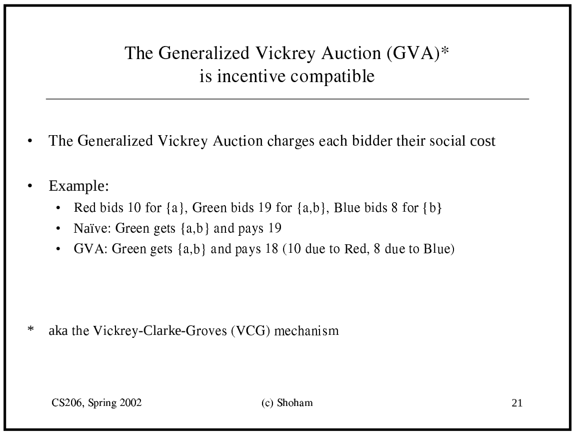
The Naïve CA is not incentive compatible

- Naïve CA: Given a set of bids on bundles, auctioneer finds a subset containing non-conflicting bids that maximizes revenue, and charges each winning bidder his bid
- This is not incentive compatible, and thus not (economically) efficient
- Example:
  - $v_1(x)=50$, $v_1(y)=50$, $v_1(x,y)=100$
  - $v_2(x)=75$, $v_2(y)=0$, $v_2(x,y)=75$
  - Bidder 1 has incentive to “lie” and claim
    - $v_1'(x)=76$, $v_1'(y)=1$, $v_1'(x,y)=100$



Lessons from the single dimensional case

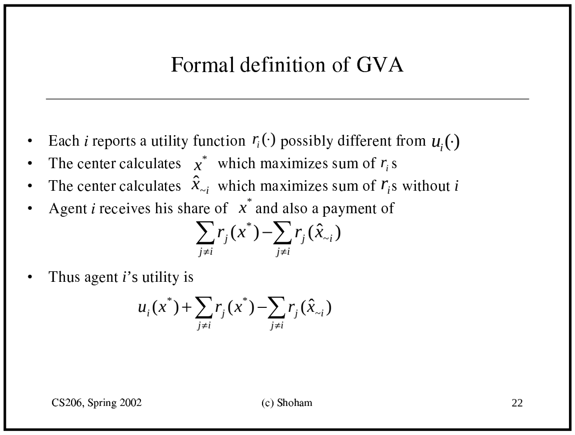
- 1st-price sealed bid auction is not incentive compatible (in equilibrium, it pays to “shave” a bit off your true value)
- 2nd-price sealed bid (“Vickrey”) auction is incentive compatible
- Can we pull off the same trick here?



The Generalized Vickrey Auction (GVA)* is incentive compatible

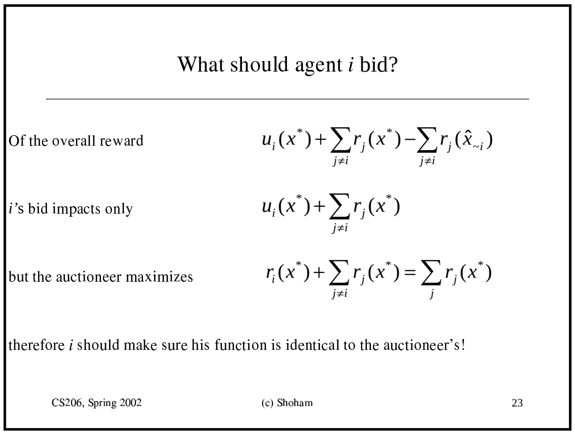
- The Generalized Vickrey Auction charges each bidder their social cost
- Example:
  - Red bids 10 for {a}, Green bids 19 for {a,b}, Blue bids 8 for {b}
  - Naïve: Green gets {a,b} and pays 19
  - GVA: Green gets {a,b} and pays 18 (10 due to Red, 8 due to Blue)

* aka the Vickrey-Clarke-Groves (VCG) mechanism



Formal definition of GVA

- Each *i* reports a utility function $r_i(\cdot)$ possibly different from $u_i(\cdot)$
- The center calculates x^* which maximizes sum of r_i s
- The center calculates $\hat{x}_{\sim i}$ which maximizes sum of r_is without *i*
- Agent *i* receives his share of x^* and also a payment of
$$\sum_{j\neq i} r_j(x^*) - \sum_{j\neq i} r_j(\hat{x}_{\sim i})$$
- Thus agent *i*’s utility is
$$u_i(x^*) + \sum_{j\neq i} r_j(x^*) - \sum_{j\neq i} r_j(\hat{x}_{\sim i})$$



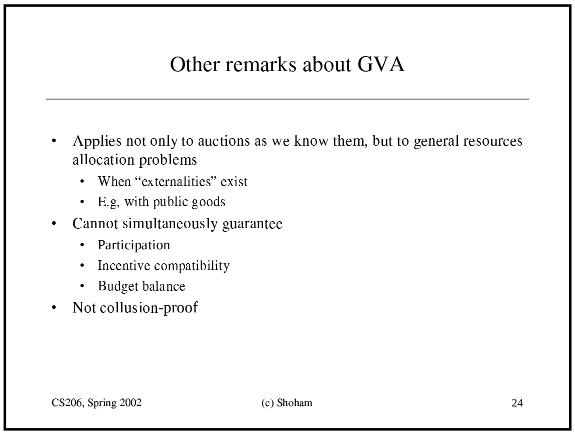
What should agent *i* bid?

Of the overall reward $$u_i(x^*) + \sum_{j\neq i} r_j(x^*) - \sum_{j\neq i} r_j(\hat{x}_{\sim i})$$

i’s bid impacts only $$u_i(x^*) + \sum_{j\neq i} r_j(x^*)$$

but the auctioneer maximizes $$r_i(x^*) + \sum_{j\neq i} r_j(x^*) = \sum_{j} r_j(x^*)$$

therefore *i* should make sure his function is identical to the auctioneer’s!



Other remarks about GVA

- Applies not only to auctions as we know them, but to general resources allocation problems
  - When “externalities” exist
  - E.g, with public goods
- Cannot simultaneously guarantee
  - Participation
  - Incentive compatibility
  - Budget balance
- Not collusion-proof

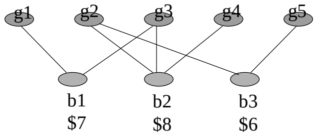
Overview of Lecture

- What can you bid: The expressive power of different bidding languages
- What should you bid: A taste for the game theory of CAs

✓ Computational complexity of CAs

The optimization problem of CAs

- “Given a set of bids on bundles, find a subset containing non-conflicting bids that maximizes revenue”
- Performed once by the naïve method, n+1(?) times by GVA
- Requires exponential time in the number of goods and bids (assuming they are polynomially related)



What’s known about the problem?

- Known as the Set Packing Problem (SPP)
- It is NP-complete, meaning that effectively the only algorithms guaranteed to find the optimal solution will run exponentially long in the worst case
- Furthermore, you cannot even uniformly approximate the optimal solution (there isn’t an algorithm that can guarantee that you always reach within a fixed fraction of it, no matter how small the fraction, although you can get within $1/\sqrt{k}$ of it, where K is the number of goods)
- Nonetheless, progress has been made recently on algorithms optimized for this problem…

Approaches to taming the computational complexity of CAs

- Finding tractable special cases
- LP-relaxation of the IP problem
- Applying complete heuristic methods
- Applying incomplete heuristic methods
- How to test these algorithms? The need for a test suite
- Learning where the hard problems lie

SPP as an Integer Program

- *n* items -- indexed by *i* (some may be dummy)
- *m* atomic bids: *(Sj,pj)* (maybe multiple ones from same bidder)
- Goal: optimize social efficiency

$$\begin{aligned}
&\textit{Maximize} \quad \sum_{j=1}^{m} x_j p_j \\
&\textit{Subject} \quad \textit{to}: \\
&\sum_{i \in S_j} x_j \le 1 \quad \forall i \\
&x_j \in \{0,1\} \quad \forall j
\end{aligned}$$

Linear Programming Relaxation of the IP

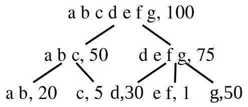
$$\begin{aligned}
&\textit{Maximize} \quad \sum_{j=1}^{m} x_j p_j & & \textit{Maximize} \quad \sum_{j=1}^{m} x_j p_j \\
&\textit{Subject} \quad \textit{to}: & \Rightarrow \quad & \textit{Subject} \quad \textit{to}: \\
&\sum_{i \in S_j} x_j \le 1 \quad \forall i & & \sum_{i \in S_j} x_j \le 1 \quad \forall i \\
&x_j \in \{0,1\} \quad \forall j & & x_j \ge 0 \quad \forall j
\end{aligned}$$

- Good news: LP is easy
- Bad news: Will produce “fractional” allocations: *xj* specifies what fraction of bid *j* is obtained.
- Pretty good news: If we’re lucky, the solution will be integer anyway

When do we get lucky?

- Tree structured bundles; e.g., wine cases
- Contiguous single-dimensional goods; e.g., time intervals
- A general condition: Total Unimodular (TU) matrices
- Bundles of size at most 2

Tree-Structured Bundles



- Example: Wine cases
- Direct algorithm: bottom-up maximization
  - Compute the maximum between the value of the parent of leaves and the sum of their children
  - Continue this way up to the root.
- Complexity: O(n)

More General: Contiguous 1-Dimensional Goods

- Example: blocks of time, contiguous lots
- Direct algorithm: recursive procedure
  - Consider the 1-dimensional good abcdefghijkl
  - Wlog assume you have bids for all intervals: a, ab, abc, bc, bc, bcd, cd, abcd, *etc*.
  - Now compute recursively the optimal partition of all prefixes
  - Inductive step:
    - Assume you've found the maximal revenue for abcdef
    - g will either be a singleton, a pair fg, a triple efg, etc.; by induction, in each case you know how to maximize the revenue for the initial prefix
- Complexity: O(n^2)

Generalizing Both: Totally Unimodular (TU) Matrices

- Problem in matrix form:

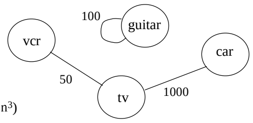
$$\text{good} \longrightarrow \begin{pmatrix} 1100 \\ 0111 \\ 1001 \\ 0011 \\ 1101 \\ 0010 \end{pmatrix} * \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \leq \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$\uparrow \text{bid}$$

- M is TU iff the determinant of each of its square submatrices is 1, 0 or –1
- In this case the solution to the LP is integer
- Complexity: ~O(n^3)
- Observation: Still holds when you allow multiple units of each good, but still allow each bidder at most one unit of each good

A separate easy case: maximum of two goods per bundle

- Example: ({TV,VCR},50) OR ({guitar},100) OR ({car,TV},1000)
- Algorithm: Maximal weighted matching in undirected graphs



- Complexity: O(n^3)

State of the art regarding the general case

- Recent years have seen an explosion of specialized search algorithms for CAs
- Complete methods guarantee optimal results, but not quick convergence. On test cases the algorithms scale to xx-xxx goods and xxxxxx+ bids.
- Incomplete, greedy-search methods sometimes perform an order of magnitude faster
- CPLEX 7.0 pretty much as good as it gets …
- A major challenge: testing the algorithms
  - A universal test suite (CATS)
  - Using machine learning to find the hard instances

required material on auctions, posted on web page, in addition this presentation and the one of April 25

Introduction to Multi-Agent Systems (draft)

Chapter 7: Mechanism Design
sections 7.3 and 7.4

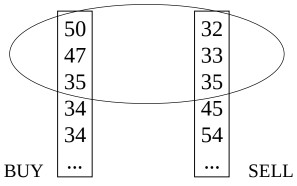
by Y. Shoham (with T. Grenager)

(Only sections 7.3 and 7.4 are required; the rest are included just in case you're curious)

Some remaining issues on auctions

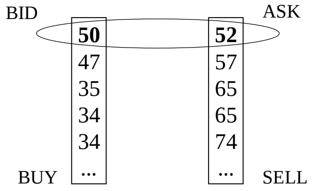
- Two-sided markets
- Beyond zoology

Double markets: m'th and m+1st prices

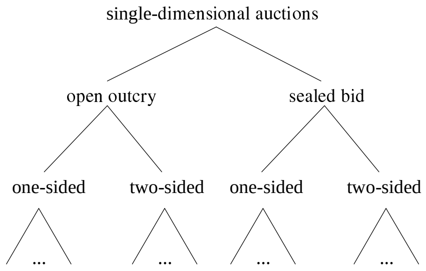


Double markets: the "Bid-Ask" Spread

- Terminology comes from familiar CDA, as in financial exchanges.



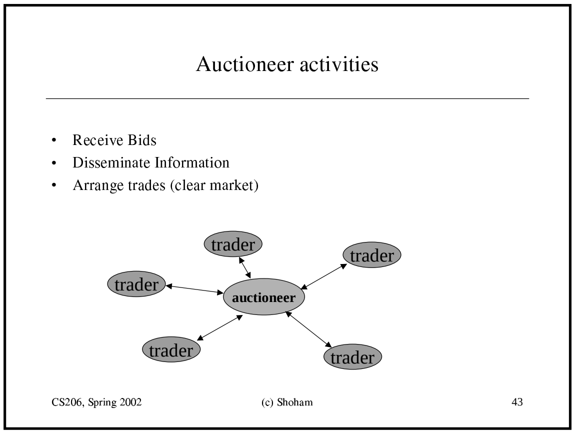
What we've done so far: zoological categorization



A deeper look at what auctions really are

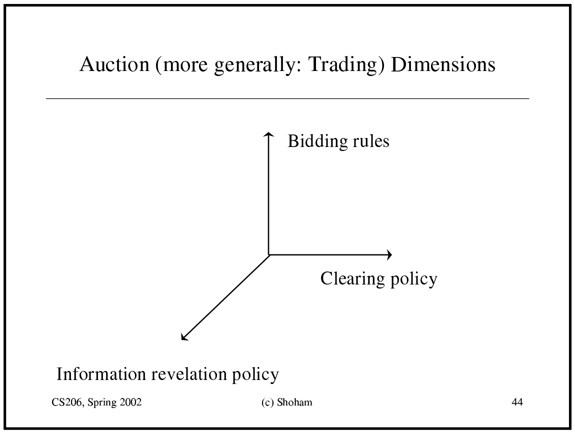
Definition: An auction is any negotiation mechanism that is:

- Mediated
- Well-specified (runs according to explicit rules)
- Market-based (determines an exchange in terms of standard currency)

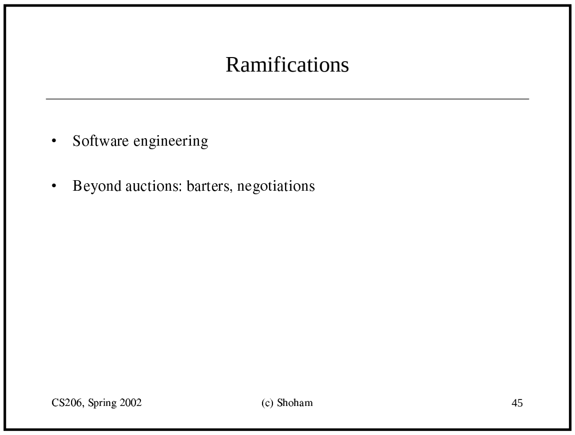


Auctioneer activities

- Receive Bids
- Disseminate Information
- Arrange trades (clear market)



Auction (more generally: Trading) Dimensions



Ramifications

- Software engineering
- Beyond auctions: barters, negotiations