CPS 196.2
Introduction to Computational Economics

Instructor: Vincent Conitzer
(Assistant Professor of Computer Science & Economics)
conitzer@cs.duke.edu
TA: Mingyu Guo
(Ph.D. student in Computer Science)
mingyu@cs.duke.edu
Course web page: http://www.cs.duke.edu/courses/fall07/cps196.2/

What is Economics?

- "the social science that studies the production, distribution, and consumption of valuable goods and services" [Wikipedia, Jan. 07]
- Some key concepts:
  - Economic agents or players (individuals, households, firms, …)
  - Agents' current endowments of goods, money, skills, …
  - Possible outcomes ((re)allocations of resources, tasks, …)
  - Agents' preferences or utility functions over outcomes
  - Agents' beliefs (over other agents' utility functions, endowments, production possibilities, …)
  - Agents' possible decisions/actions
  - Mechanism that maps decisions/actions to outcomes

An economic picture

v( ) = 200

$ 800

v( ) = 100

v( ) = 400

v( ) = 200

v( ) = 400

$ 600

$ 200

After trade (a more efficient outcome)

v( ) = 200

$ 1100

… but how do we get here? Auctions? Exchanges? Unstructured trade?

v( ) = 100

v( ) = 400

v( ) = 200

v( ) = 400

$ 400

$ 100

Some distinctions in economics

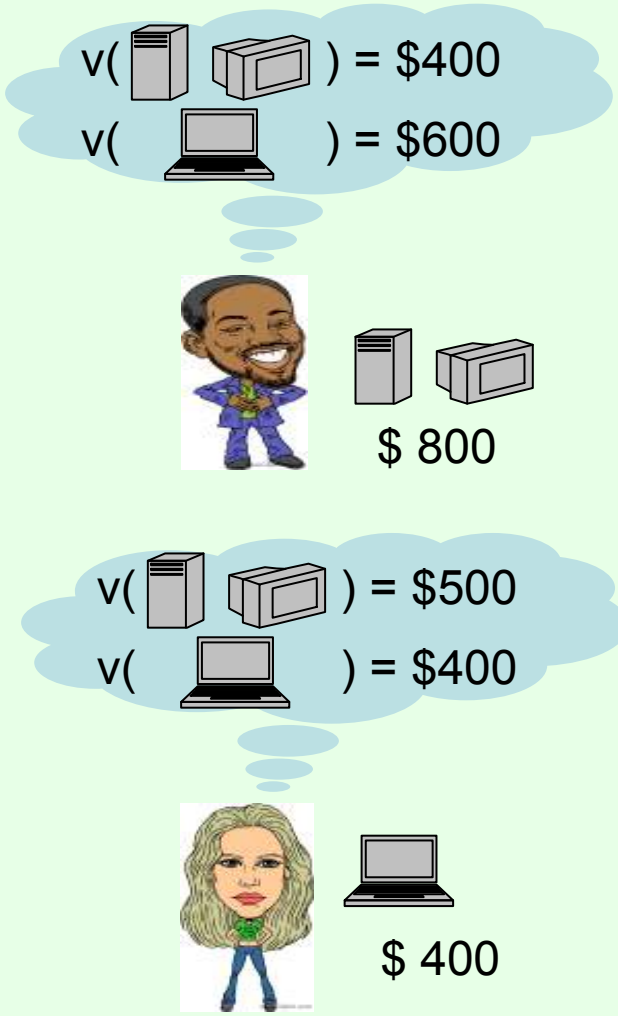
- Descriptive vs. normative economics
  - Descriptive:
    - seeks only to describe real-world economic phenomena
    - does not care if this is in any sense the "right" outcome
  - Normative:
    - studies how people "should" behave, what the "right" or "best" outcome is
- Microeconomics vs. macroeconomics
  - Microeconomics: analyzes decisions at the level of individual agents
    - deciding which goods to produce/consume, setting prices, …
    - "bottom-up" approach
  - Macroeconomics: analyzes "the sum" of economic activity
    - interest rates, inflation, growth, unemployment, government spending, taxation, …
    - "big picture"

What is Computer Science?

- "the study of the theoretical foundations of information and computation and their implementation and application in computer systems" [Wikipedia, Jan. 07]
- A computational problem is given by a function f mapping inputs to outputs
  - For integer x, let f(x) = 0 if x is prime, 1 otherwise
  - For an initial allocation of resources x, let f(x) be the (re)allocation that maximizes the sum of utilities
- An algorithm is a fully specified procedure for computing f
  - E.g. sieve of Eratosthenes
  - A correct algorithm always returns the right answer
  - An efficient algorithm returns the answer fast
- Computer science is also concerned with building larger artifacts out of these building blocks (e.g. personal computers, the Internet, the Web, search engines, spreadsheets, artificial intelligence, …)

Resource allocation as a computational problem *(Part 1 of the course)*

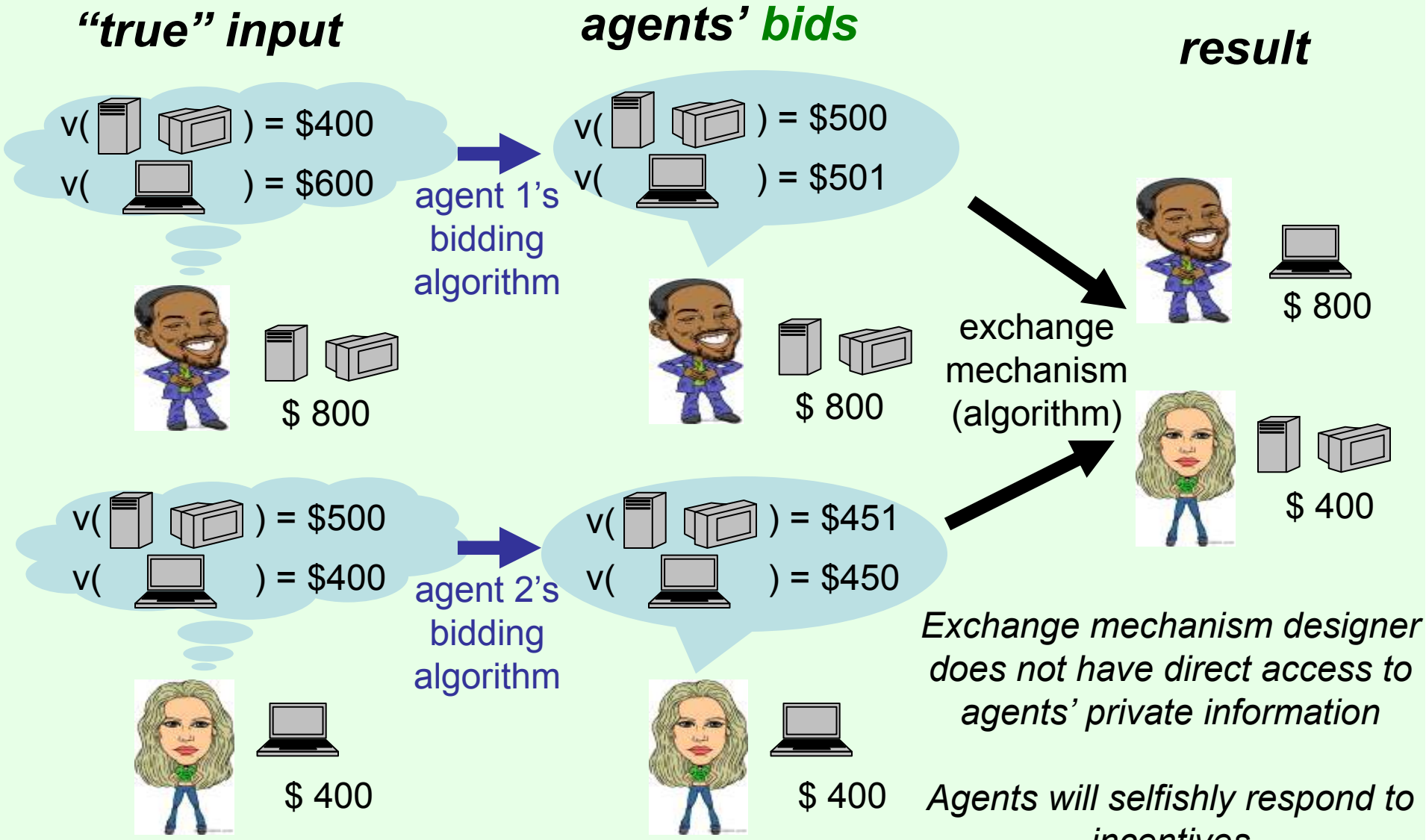
input



output

Here, gains from trade ($300) are divided evenly (not essential)

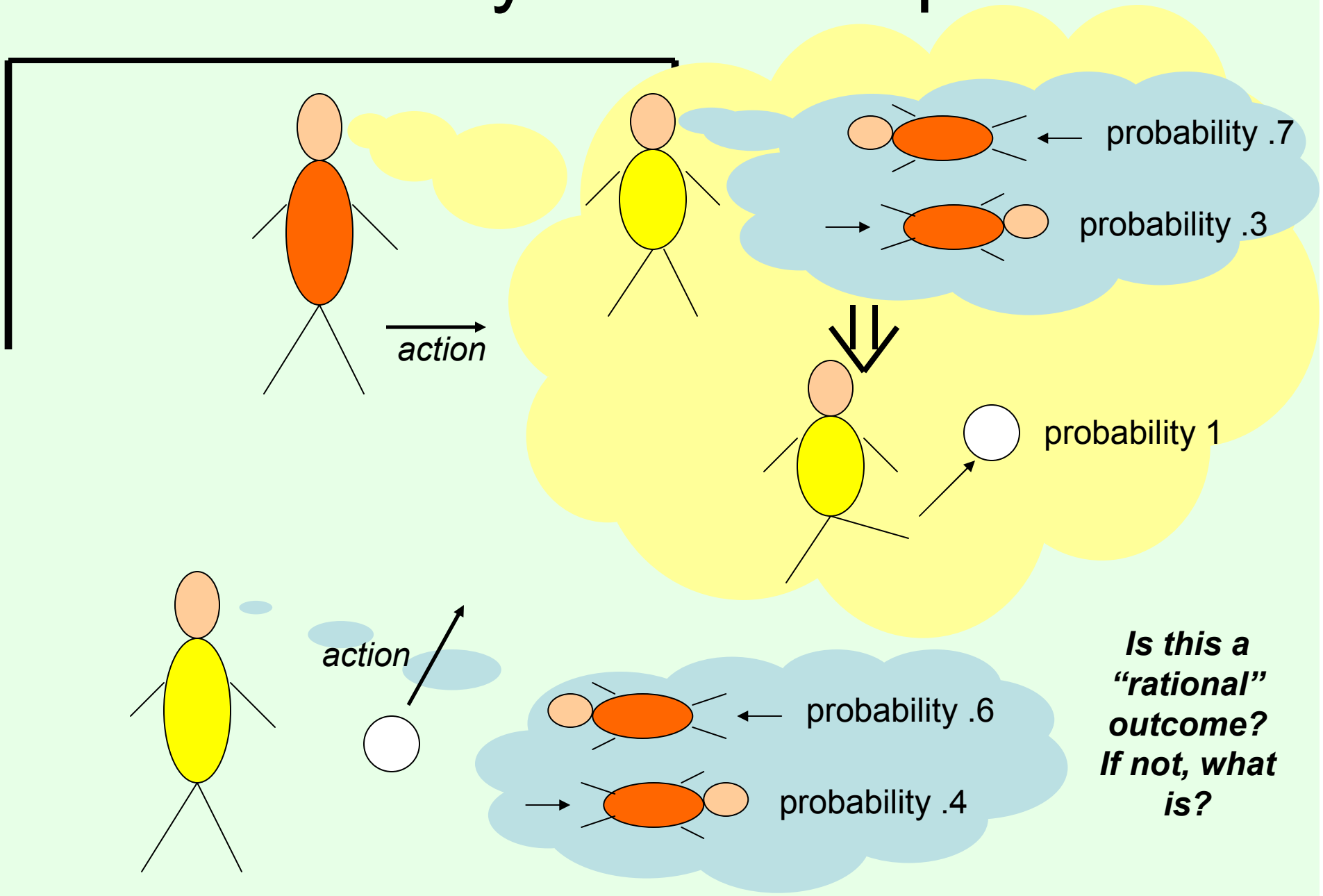
Economic mechanisms



Game theory
(Part 2 of the course)

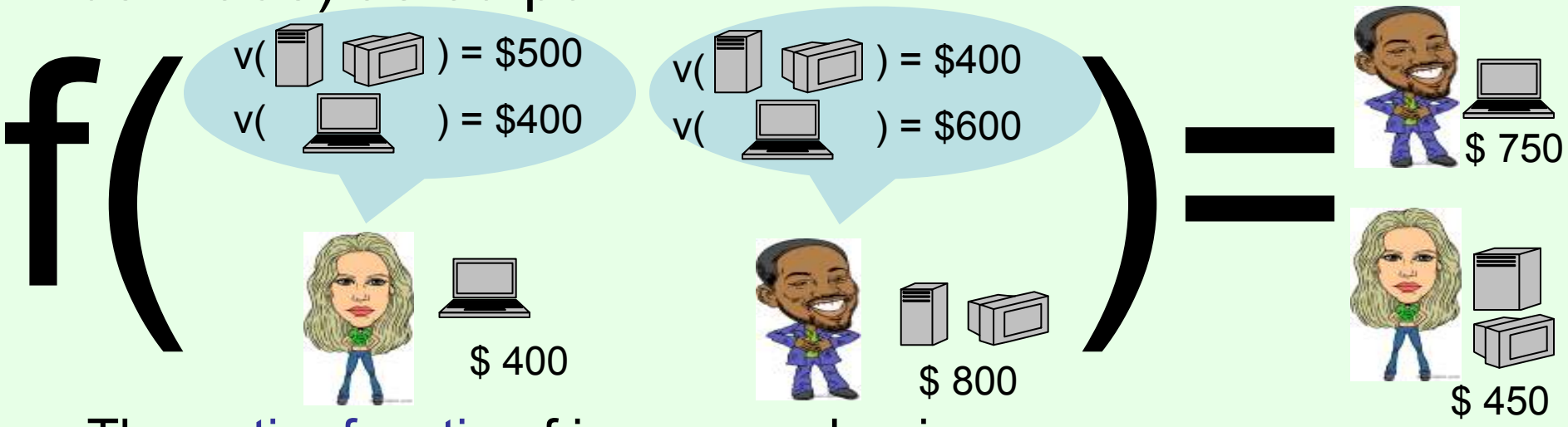
- Game theory studies settings where agents each have
  - different preferences (utility functions),
  - different actions that they can take
- Each agent's utility (potentially) depends on all agents' actions
  - What is optimal for one agent depends on what other agents do
    - Very circular!
- Game theory studies how agents can rationally form beliefs over what other agents will do, and (hence) how agents should act
  - Useful for acting as well as predicting behavior of others

Penalty kick example



Mechanism design
(Part 3 of the course)

- Mechanism = rules of auction, exchange, …
- A function that takes reported preferences (bids) as input, and produces outcome (allocation, payments to be made) as output



- The entire function f is one mechanism
- E.g. the mechanism from part 1: find allocation that maximizes (reported) utilities, distribute (reported) gains evenly
- Other mechanisms choose different allocations, payments

Mechanism design…

- Mechanism = game
- We can use game theory to predict what will happen under a mechanism
  - if agents act strategically
- When is a mechanism "good"?
  - Should it result in outcomes that are good for the reported preferences, or for the true preferences?
  - Should agents ever end up lying about their preferences (in the game-theoretic solution)?
  - Should it always generate the best allocation?
  - Should agents ever burn money?(!?)
- Can we solve for the optimal mechanism?

How are we going to solve these problems? *(Part 0)*

- This is not a programming course

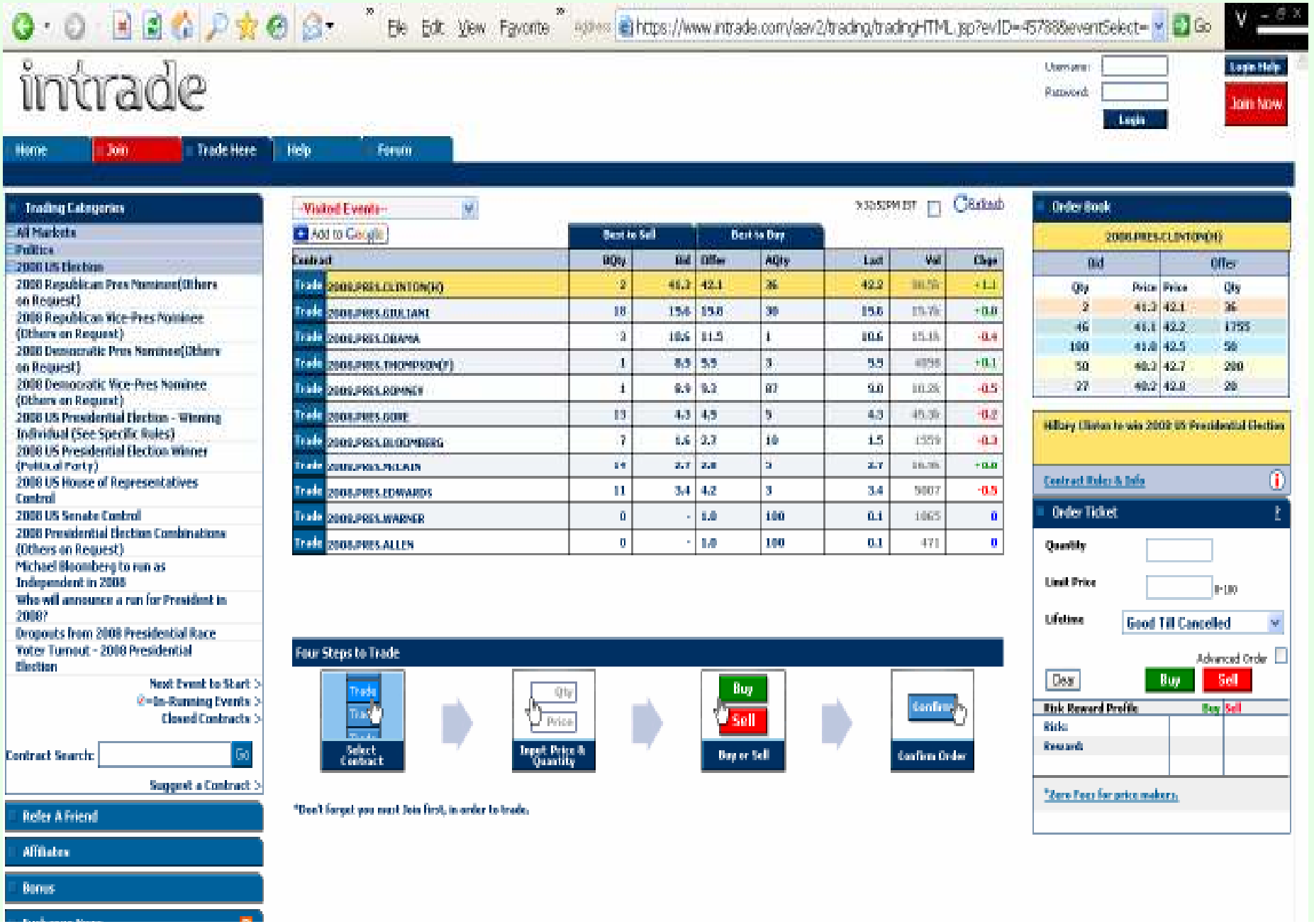
- Will use optimization software
  - GNU Linear Programming Kit (GLPK)
  - Linear programming, mixed integer linear programming

Uses of LP, MIP in this course

	Linear programming	**Mixed integer linear programming**
Part 1 (expressive marketplaces)	Winner determination in auctions, exchanges, … with partially acceptable bids	Winner determination in auctions, exchanges, … without partially acceptable bids
Part 2 (game theory)	Dominated strategies Minimax strategies Correlated equilibrium	Nash equilibrium
Part 3 (mechanism design)	Designing optimal mechanisms that use randomization	Designing optimal mechanisms that do not use randomization

Other settings/applications

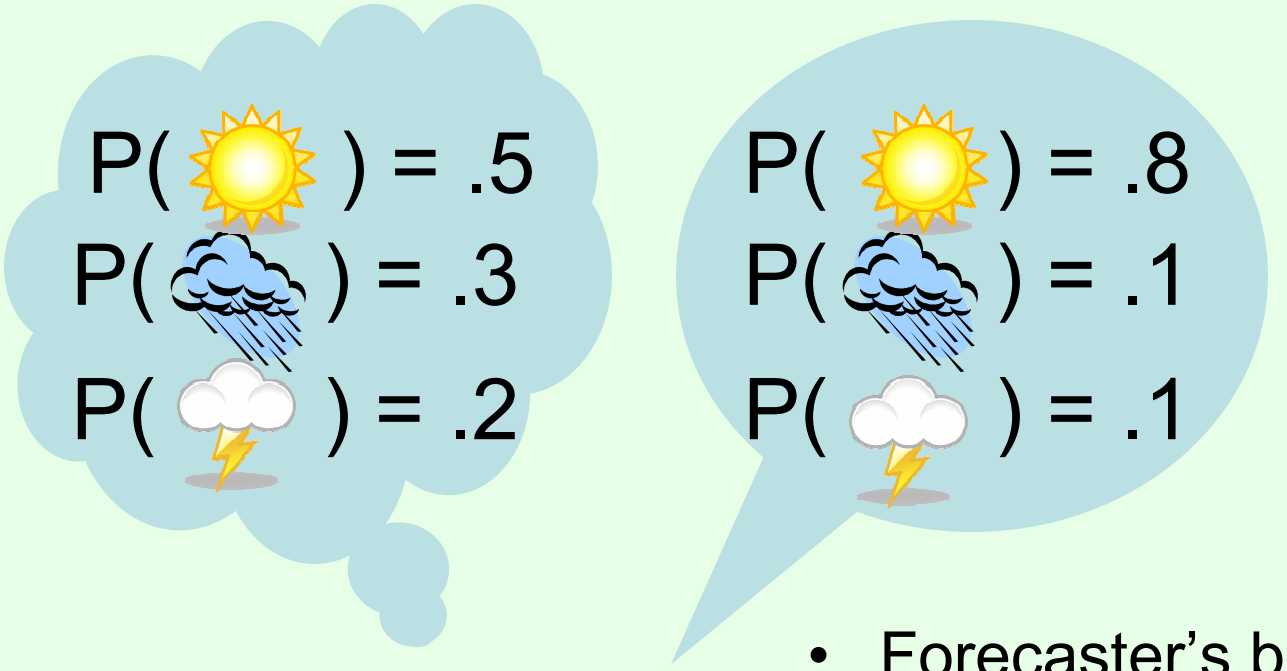
Prediction markets



Financial securities (in Part 1)

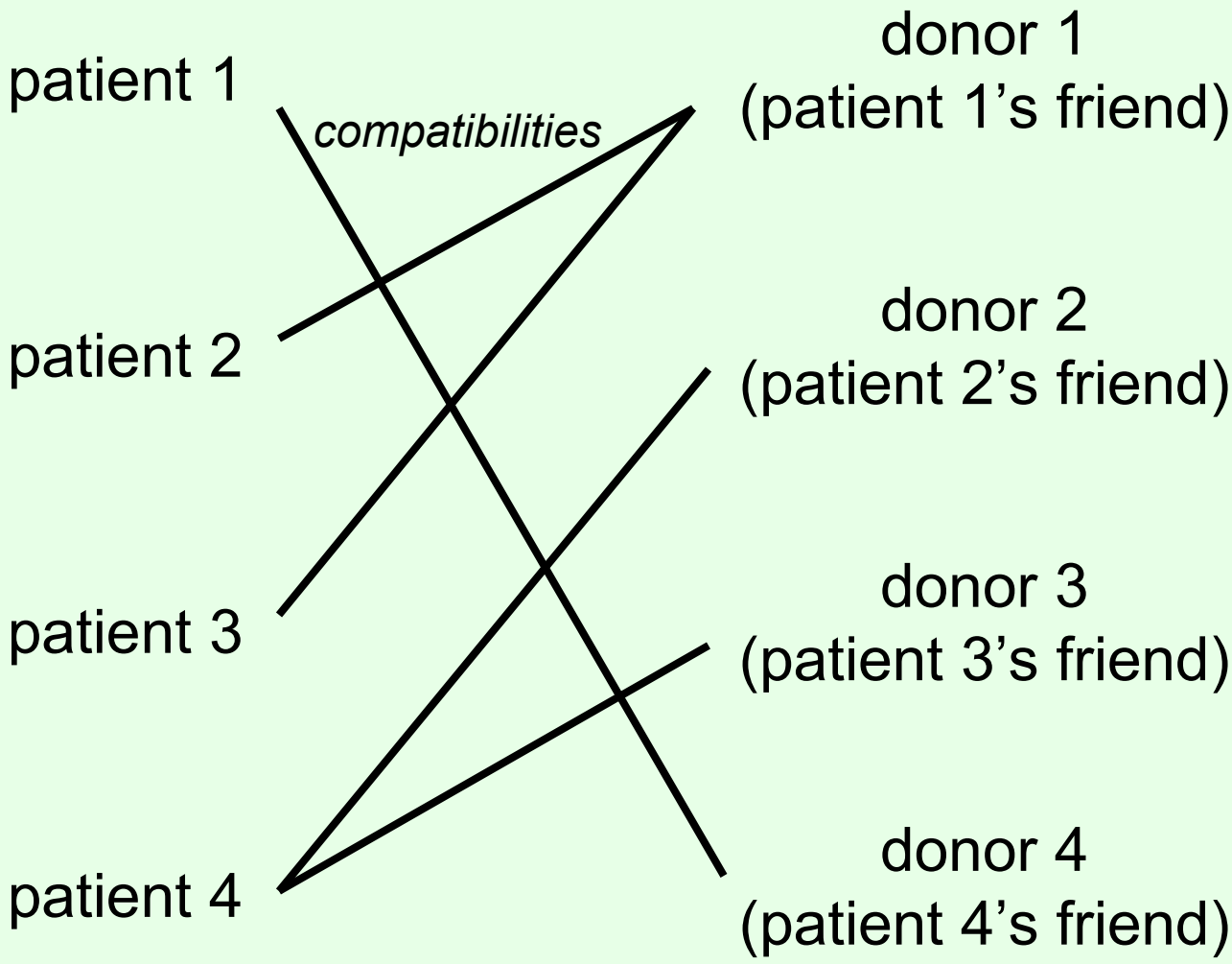
- Tomorrow there must be one of 
- Agent 1 offers \$5 for a security that pays off \$10 if  or 
- Agent 2 offers \$8 for a security that pays off \$10 if  or 
- Agent 3 offers \$6 for a security that pays off \$10 if 
- Can we accept some of these at offers at no risk?

How to incentivize a weather forecaster (in Part 3)



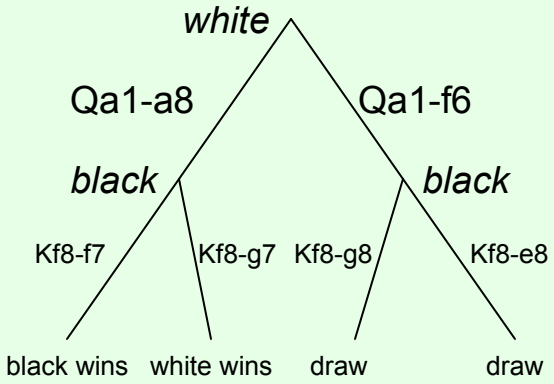
- Forecaster's bonus can depend on
  - Prediction
  - Actual weather on predicted day
- Reporting true beliefs should maximize expected bonus

Kidney exchange (Part 1)



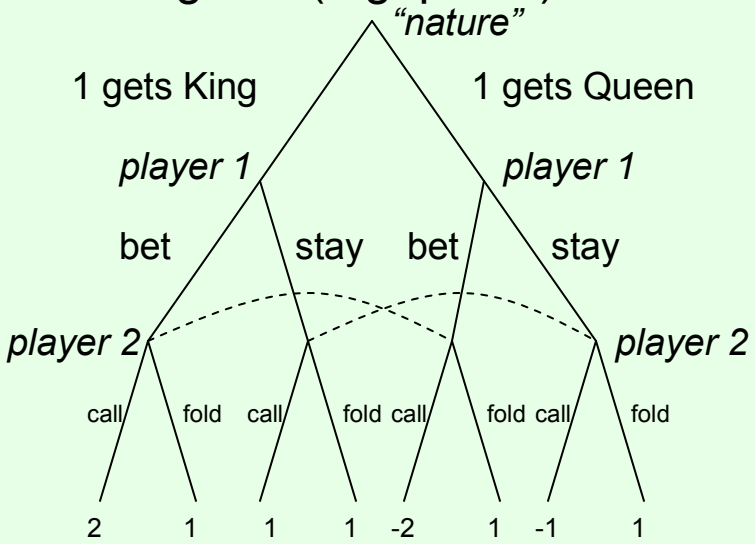
Game playing & AI (in Part 2)

perfect information games: no uncertainty about the state of the game (e.g. tic-tac-toe, chess, Go)



- Optimal play: value of each node = value of optimal child for current player (backward induction, minimax)
- For chess and Go, tree is too large
  - Use other techniques (heuristics, limited-depth search, alpha-beta, …)
- Top computer programs (arguably) better than humans in chess, not yet in Go

imperfect information games: uncertainty about the state of the game (e.g. poker)



- Player 2 cannot distinguish nodes connected by dotted lines
  - Backward induction fails; need more sophisticated game-theoretic techniques for optimal play
- Small poker variants can be solved optimally
- Humans still better than top computer programs at full-scale poker
- Top computer (heads-up) poker players are based on techniques for game theory

Why should economists care about computer science?

- Finding efficient allocations of resources is a (typically hard) computational problem
  - Sometimes beyond current computational techniques
  - If so, unlikely that any market mechanism will produce the efficient allocation (even without incentives issues)
  - Market mechanisms must be designed with computational limitations in mind
  - New algorithms allow new market mechanisms

Why should economists care about computer science…

- Agents also face difficult computational problems in participating in the market
  - Especially acting in a game-theoretically optimal way is often computationally hard
  - Game-theoretic predictions will not come true if they cannot be computed
    - Sometimes bad (e.g. want agents to find right bundle to trade)
    - Sometimes good (e.g. do not want agents to manipulate system)

Why should computer scientists care about economics?

- Economics provides high-value computational problems
- Interesting technical twist: no direct access to true input, must incentivize agents to reveal true input
- Conversely: Computer systems are increasingly used by multiple parties with different preferences (e.g. Internet)
- Economic techniques must be used to
  - predict what will happen in such systems,
  - design the systems so that they will work well
- Game theory is relevant for artificial intelligence
  - E.g. computer poker