Delousing the Market: From Basic Mechanisms to Intelligent Agents

Efficient token trading for optimal ~~social welfare~~ user experience

Casey Detrio http://cdetr.io

Market mechanisms are **Institutions** that govern economic interactions.

I will argue:

A. Continuous Auctions are **Evil**. Don't use them.
B. Instead, use Batch Auctions (with single-price clearing).

C. Continuous -> Batch is just a first evolutionary step from basic mechanisms to Intelligent Agents.

Four essential market mechanisms:

1. Issuer
   - creates/certifies tokens (e.g. bitcoin block reward)
2. Clearinghouse
   - validates token transfers (no double-spends)

Settlement & Clearing

3. Auctioneer

4. Market Maker

Four essential market mechanisms:

1. Issuer
2. Clearinghouse

Settlement & Clearing

3. Auctioneer
   - protocol/process for organizing trade (e.g. limit order book and matching engine)

4. Market Maker
   - intermediates between traders who arrive at different times (e.g. value-investor, HFT bot, LMSR)

Trading
&
Exchange

Four essential market mechanisms:

1. Issuer
2. Clearinghouse
3. Auctioneer
4. Market Maker

Evolution of crypto-currencies: from third-party agents to *on-chain mechanisms.*

- Bitcoin (1 & 2)
- Ripple/Mastercoin/EtherEx (3)
- Truthcoin/Augur/Gnosis (4)

The Four Mechanisms: on-chain vs. third-party

X = on-chain (decentralized) O = third-party agent (centralized)

	issuance	clearing house	auctioneer	market-maker
mtgox	O	O	O	O
coinbase	O	O	O	O
bitcoin	X	X	O	O
ripple	O	X	X	O
bitshares	X	X	X	O
truthcoin	X	X	O	X

Auctioneers: two basic types

1. Continuous-time auctions
   - **serial** processing of orders, one-by-one as they arrive
   - orders matched using **pay-as-bid** pricing (aka discriminatory-pricing)

2. Discrete-time auctions. aka call auctions or batch auctions
   - **batch** processing of orders, all-at-once at clearing time
   - orders matched using **single-price** clearing (aka uniform-price clearing)

“Let a thousand cryptocurrencies bloom.” -- vbuterin

Planning to exchange all these tokens with a pay-as-bid order matching mechanism?

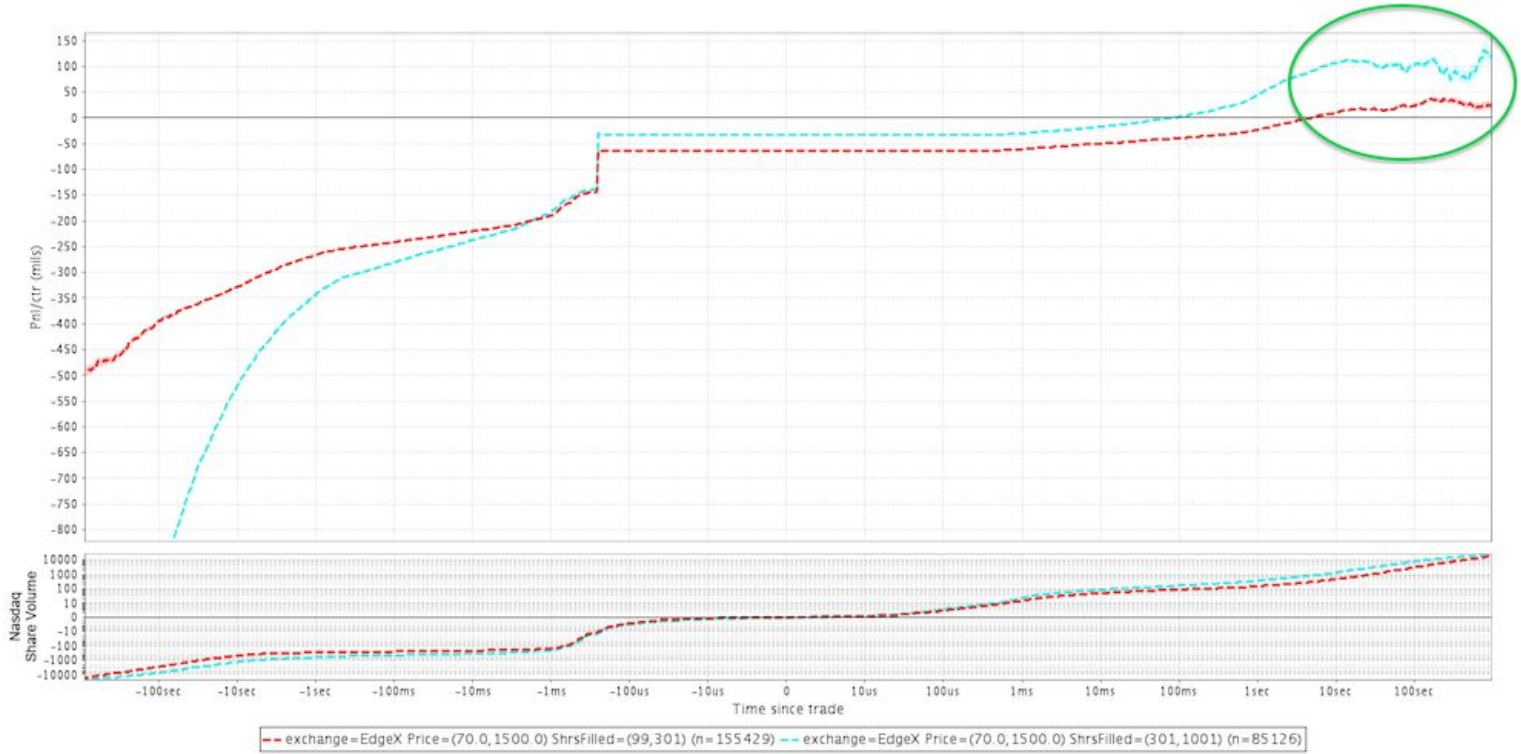
Why is pay-as-bid matching a problem?

Because (when matched with pay-as-bid pricing in continuous-time)

Limit Orders are FREE OPTIONS!

- Placing limit orders is not incentive-compatible. Introduces all kinds of games around strategic bidding, front-running, sniping, etc.
- Market orders aren’t much better (have to pay the spread, which turns trading into a negative expectation game).
- Death by a thousand cuts (nickel-and-dimed time after time).

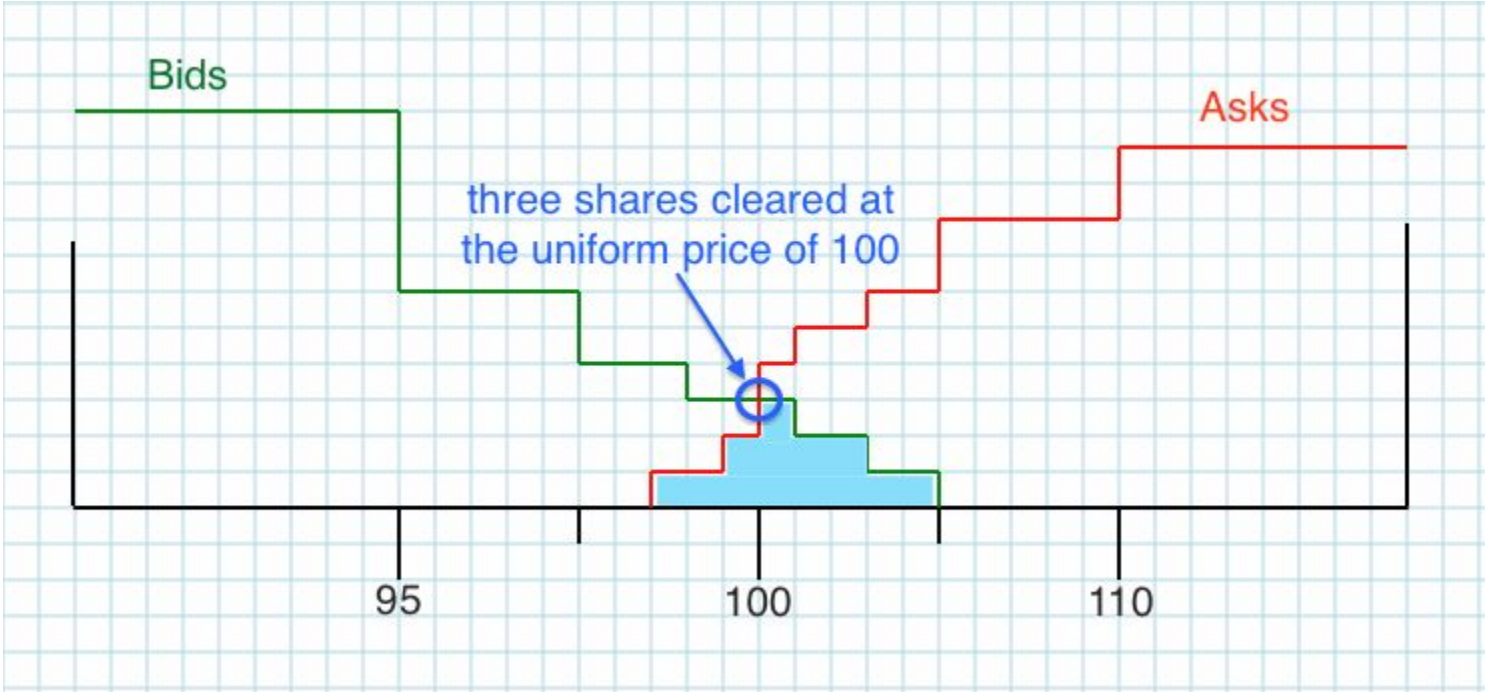
Identifying humans (stale limit orders, >5s old) is a profitable HFT strategy



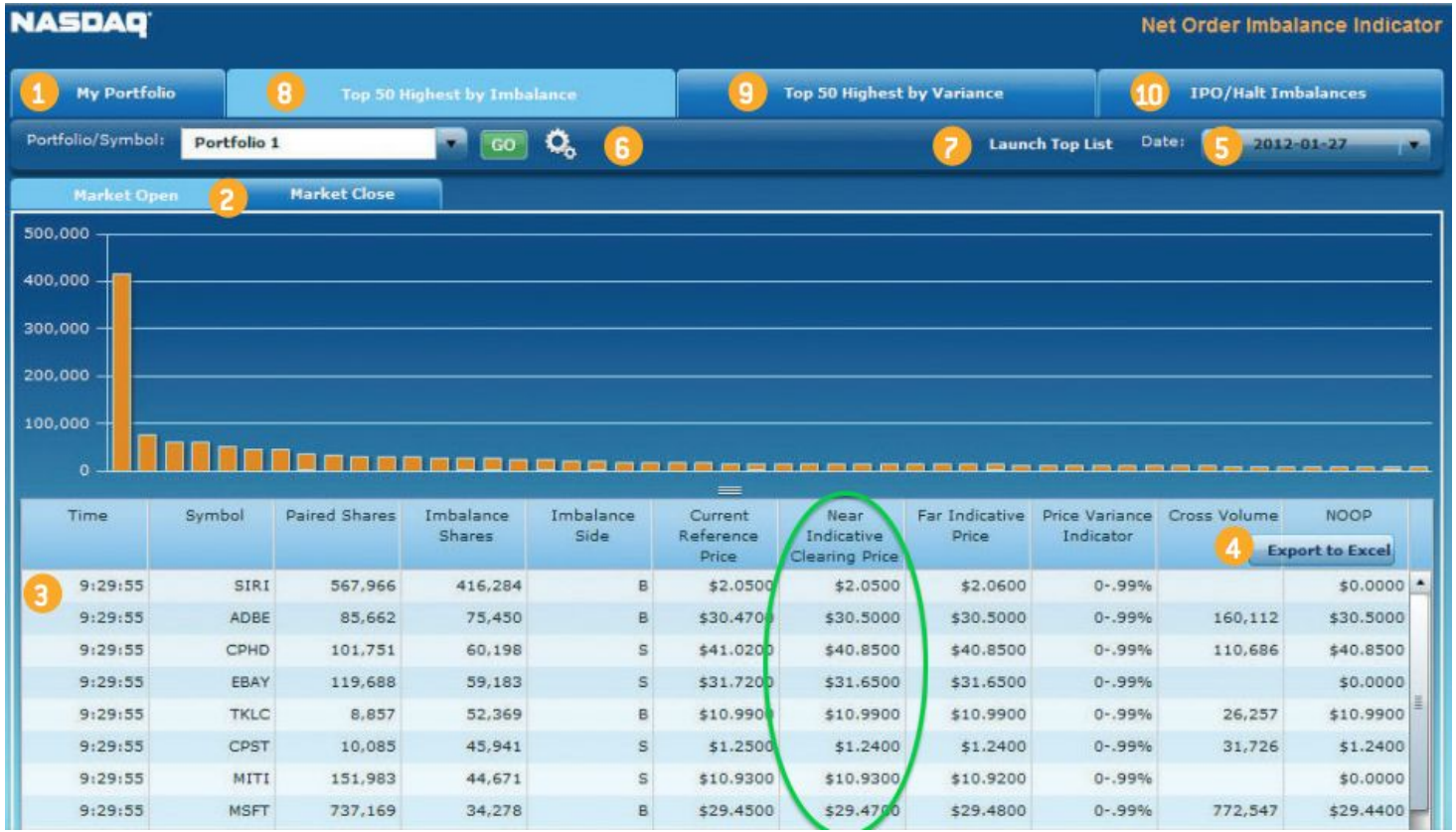
source: https://mechanicalmarkets.wordpress.com/2015/01/20/identifying-trader-type-pt-2/

A better mechanism: Single-price clearing!

A better mechanism: Single-price clearing!



- Used in opening and closing sessions on NYSE, Nasdaq, etc.



Source: http://www.nasdaqtrader.com/content/technicalsupport/userguides/tradingproducts/noii/noiiguide.pdf

Single-price clearing aka Uniform-price clearing (aka Call Auction or Batch Auction)

- Similar to a Vickrey Auction, or second-price auction

  “The most obvious generalization to multiple or divisible goods is to have all winning bidders pay the amount of the highest non-winning bid. This is known as a uniform price auction.” -- wiki/Vickrey_auction

- Besides NYSE/Nasdaq opening and closing sessions, also used in US Treasury auctions (since 1998), wholesale power markets, Google’s IPO (2004), ...
- Variants have been re-branded under different names over the years, e.g. Goldman Sachs / Deutsche Bank “Universal Dutch Auction” (2002), TruMid “trading swarms” (2015).

Goethe’s Second-Price Auction (1797)

“I am inclined to offer Mr. Vieweg from Berlin an epic poem, Hermann and Dorothea, which will have approximately 2000 hexameters…. Concerning the royalty we will proceed as follows: I will hand over to Mr. Counsel Böttiger a sealed note which contains my demand, and I wait for what Mr. Vieweg will suggest to offer for my work. If his offer is lower than my demand, then I take my note back, unopened, and the negotiation is broken. If, however, his offer is higher, then I will not ask for more than what is written in the note to be opened by Mr. Bottiger.”

Source: Moldovanu, Benny, and Manfred Tietzel. "Goethe's Second-Price Auction."*Journal of Political Economy* 106, no. 4 (1998): 854-859.

http://scienceblogs.com/purepedantry/2006/12/06/goethe-was-a-game-theorist/

https://en.wikipedia.org/wiki/Becker-DeGroot-Marschak_method
> an incentive-compatible procedure used in experimental economics to measure willingness to pay (WTP) ... From the subject's perspective, the method is equivalent to a Vickrey auction… An early attempt at a BDM-type method was by Johann Wolfgang von Goethe.[6] In 1797...

Isn’t it better to match orders super-fast? NO!

Continuous-time markets (pay-as-bid pricing by definition!):

- More efficient in *time-space,* but **less efficient in *volume-space***
- Arbitrage-rents inherent to the mechanism (limit orders are free options!)

Discrete-time markets (uniform-price clearing by definition!):

- Respect the reality of computers (discrete clocks, not infinitely fast)
- Defeat HFT arb-rents => more liquidity, better prices, improved welfare.
- A better fit for blockchains (block times as batch periods)

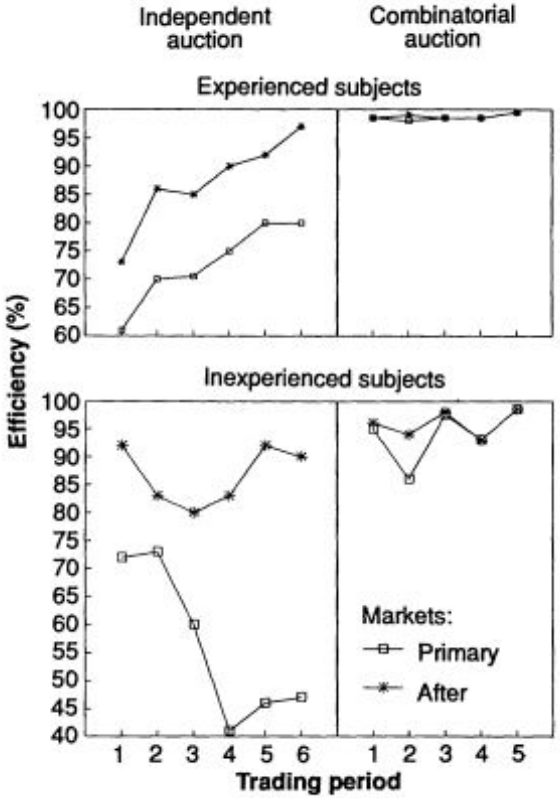
Reference: Budish, Eric B., Peter Cramton, and John J. Shim. "The high-frequency trading arms race: Frequent batch auctions as a market design response." *Fama-Miller Working Paper* (2013): 14-03.

Spectrum of mechanisms: [basic <---> advanced]

- Posted-price (i.e. sticker-price)
- Continuous Double Auction (CDA)
  - a simple sorting process! (serial execution - match orders as they arrive)
- Batch Auction
  - sort and calculate uniform-clearing price (simple batch optimization)
- Automated Market Maker (e.g. LMSR)
  - use a cost function to maintain inventory (simple online optimization)
- Combinatorial Auction
  - optimally match orders for bundles of different items (complex batch optimization)
- Combinatorial Auction + Integrated Market Maker
  - match bundles (batch optimization) + maintain inventory (online optimization)

Combinatorial Auctions vs. Independent Auctions

Fig. 3. Plot of period to period average efficiency of four experiments in each cell of a 2 by 2 design. Efficiency is measured as the percentage of potentially realizable surplus achieved by all subjects. Regardless of subject experience, the computer-assisted combinatorial auction generally achieved such high efficiency in the primary market that very small gains from exchange were realizable in an aftermarket. Coordination difficulties and speculative bidding require the secondary market to carry a much heavier burden when the primary market uses independent auctions for resource items.



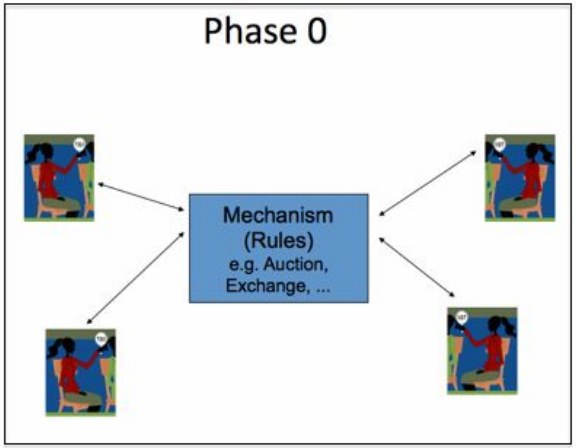
Source: Kiven A. McCabe, Stephen J. Rassenti, Vernon L. Smith. (1991). Smart Computer-Assisted Markets. Science, Volume 254, Issue 5031, 534-538

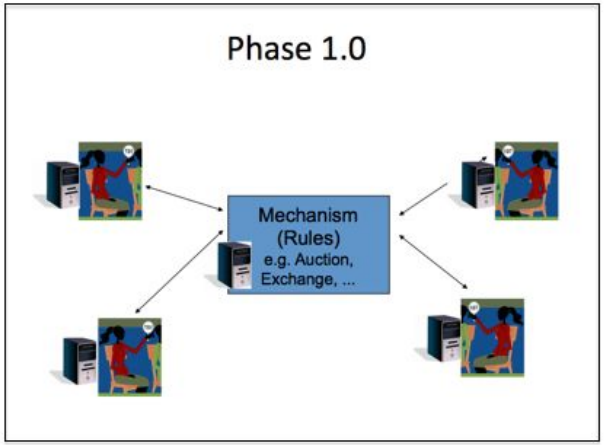
Mirowski's Markomata Hierarchv

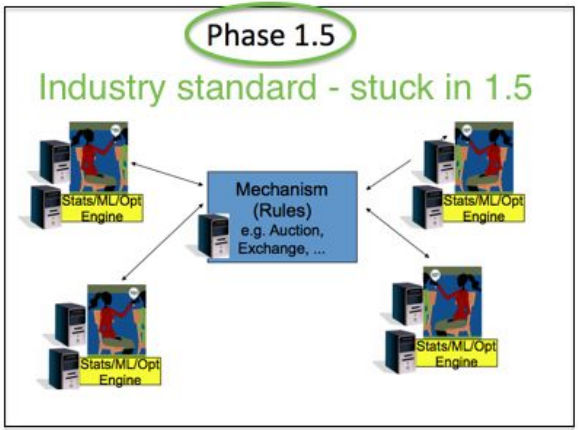
Table 1: Markomata Hierarchy of order execution

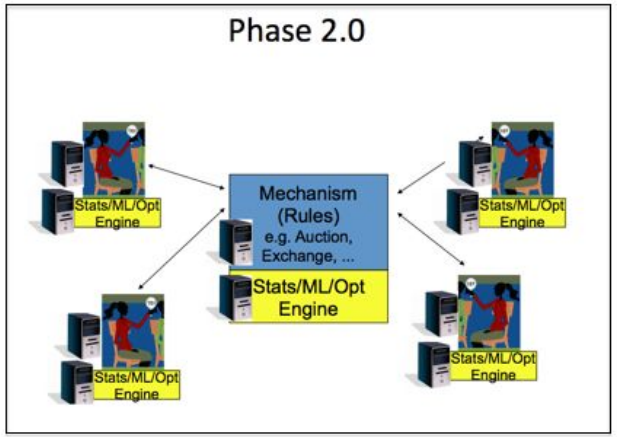
Automaton type	Recognizes lang.	Memory	Markomata
Finite щ	Regular	None	Posted-price
Pushdown	Context-free	Pushdown stack	Sealed bid
Linear bounded	Context sensitive	Finite tape	Double auction
Turing Machine	Recursively enumerable	Infinite tape	None

"the posted price format possesses no memory capacity and therefore qualifies as a finite automaton, a sealed bid auction requires the comparison of a submitted bid to an ordered array of previously entered bids stored in a memory, and therefore qualifies as one of a number of k-headed pushdown automata (Mirowski, 2002, p.571). Sealed bid order execution requires an ordering of submitted bids, which can be captured by a first-in first-out memory stack: hence the 'pushdown'. The standard double auction requires even more prodigious memory capacity, given that sequences of bids and asks stored in different identifiable memory locations must be retrieved and compared, and therefore should exhibit the computational capacity of (at least) a linear bounded automaton." (Philip Mirowski. (2004). Markets Come To Bits: Evolution, Computation and Markomata in Economic Science)



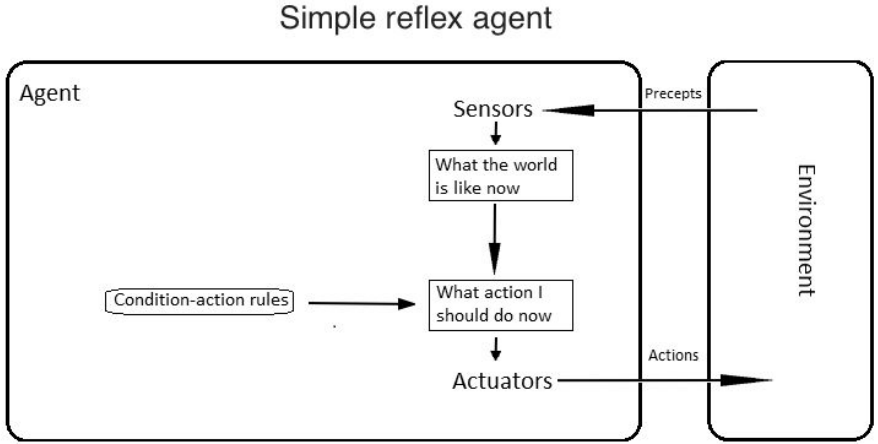


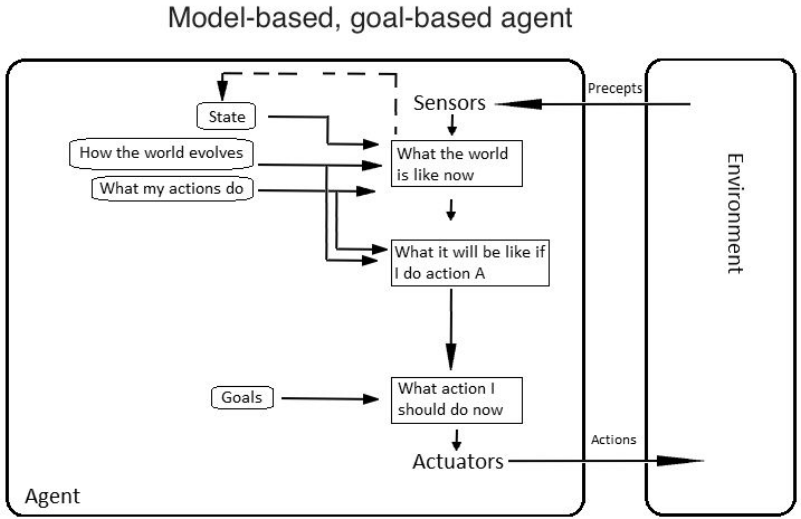




Pennock's Phases [source: David Pennock. (2012). https://www.youtube.com/watch?v=d-zxxvHpWTg]

Intelligent Agents





Russell & Norvig classification. Source: https://en.wikipedia.org/wiki/Intelligent_agent

Helpful mechanisms and Proxy Agents

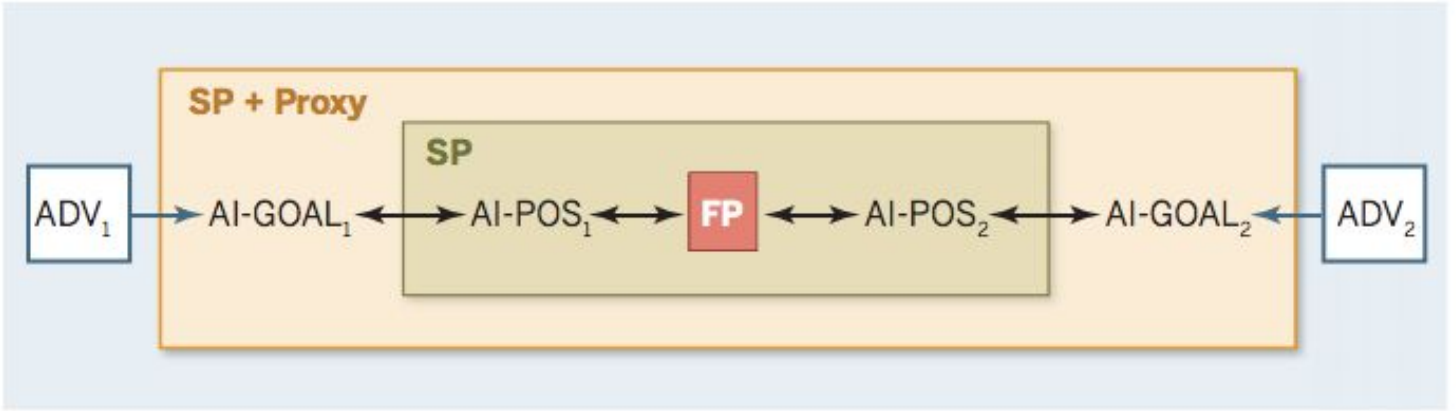


Fig. 4. Two generations of sponsored search mechanisms. Early designs were first price (FP), and advertisers (ADV) used AIs (AI-POS) to maintain a position on the list of search results at the lowest possible price. Second-price (SP) auction mechanisms were introduced, designed to replace the combination of FP and AI-POS. Advertisers adopted new AIs (AI-GOAL) to achieve higher-level goals such as to maximize profit or to maximize the number of clicks. The second price auction was extended to include proxy agents (SP+Proxy), designed to replace the combination of SP and AI-GOAL.

From Parkes, David C., and Michael P. Wellman. "Economic reasoning and artificial intelligence." *Science* 349.6245 (2015): 267-272. Reprinted with permission from AAAS.

Mock Solidity code: efficiently-verify the solution to a convex optimization problem (match bids in a batch combinatorial auction)

```
contract VerifyConvexOpt {

    address submitter;
    uint8 rows = 3;
    uint8 cols = 7;
    int16[7][3] bidMatrix;
    int16[3] subColVector;
    int16[7] bVector;
    int16[7] rowVector;
    int16 submissionResult;
    int16 optimalSubmission;

    // Current state of the bounty.
    address public winningSubmitter;
    uint public optimalSubmission;

    function VerifySubmission(int16[3] subColVector) {
        uint8 row_i = 0;
        uint8 col_j = 0;
        while (row_i < rows) {
            col_j = 0;
            rowVector[row_i] = 0;
            while(col_j < subColVector.length) {
                rowVector[row_i] = rowVector[row_i] + bidMatrix[row_i][col_j]* subColVector[col_j];
                col_j++;
            }
            row_i++;
        }

       submissionResult = multiplyVecs(rowVector, bVector);

        if (submissionResult <= optimalSubmission)
            // If the minimization is not higher, send the
            throw;

        winningSubmitter = msg.sender;
        optimalSubmission = msg.value;
        NewWinningSubmission(msg.sender, msg.value);
    }

```