

Prototyping an Iterative Combinatorial Exchange

Airline Industry
Wye River Workshop,
June 23, 2004

David C. Parkes
Harvard University



Why Markets?

- Takeoff and landing slots are a constrained resource
- Markets can facilitate the efficient (re)allocation of slots:
 - administrative processes cannot do this
 - rationing cannot do this
 - multilateral negotiation cannot do this
- Markets can expose the true value of a slot and change strategic *investment* decisions:
 - new airport capacity
 - new technology, ...

Why an Exchange?

- Keep incumbents whole:
 - allocate initial property rights
 - don't force anyone to sell
 - not a new "taxation"
- Allow new-entrants to compete
- Extensible:
 - bring in additional resources (in-route capacity, gates, other airports, etc.)
 - bring in additional players (e.g. airports)

vs. one-sided markets & vouchers

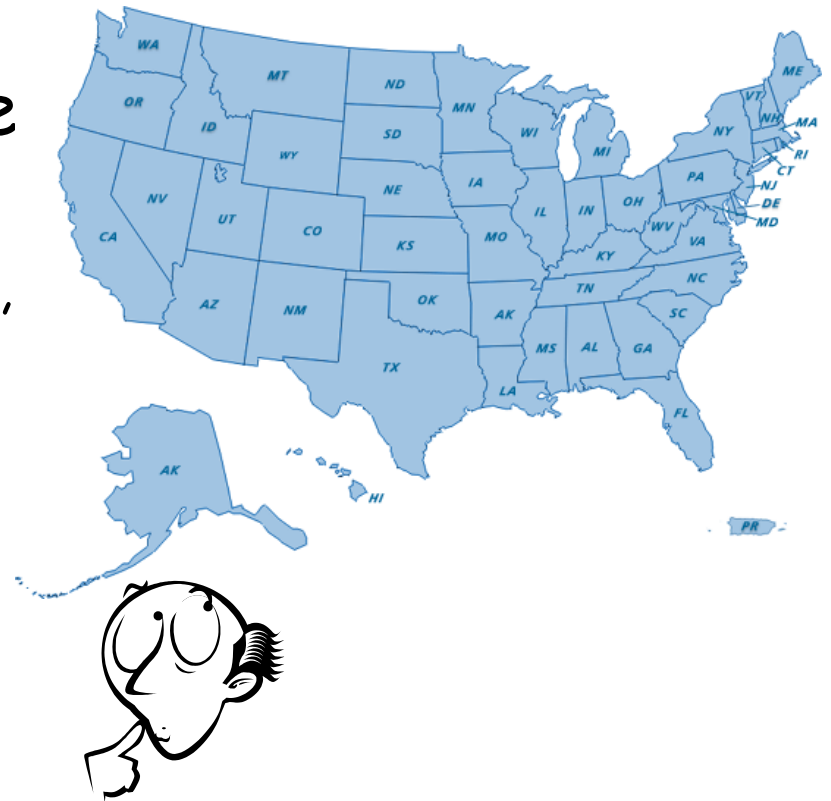
- more expressive, simpler

Why a Combinatorial Exchange?


- **Slots are complements:**
 - {9.00am landing & 10.00am takeoff} vs. {9.00am landing and 9.15am takeoff}
 - {9am takeoff@Logan, 10am landing@LGA} vs. {9am takeoff@Logan, 9.20am landing@LGA}
- **Slots are substitutes:**
 - {9am landing, 9.05am landing} vs {9am landing}
- **Business constraints:**
 - "need at least 5 landing slots during peak M-F time"
 - "need at least 2 landing slots between 10am and noon"
- **Alternative business plans:**
 - a) sell all slots, b) sell some slots, c) buy more slots

Why an Iterative Combinatorial Exchange?

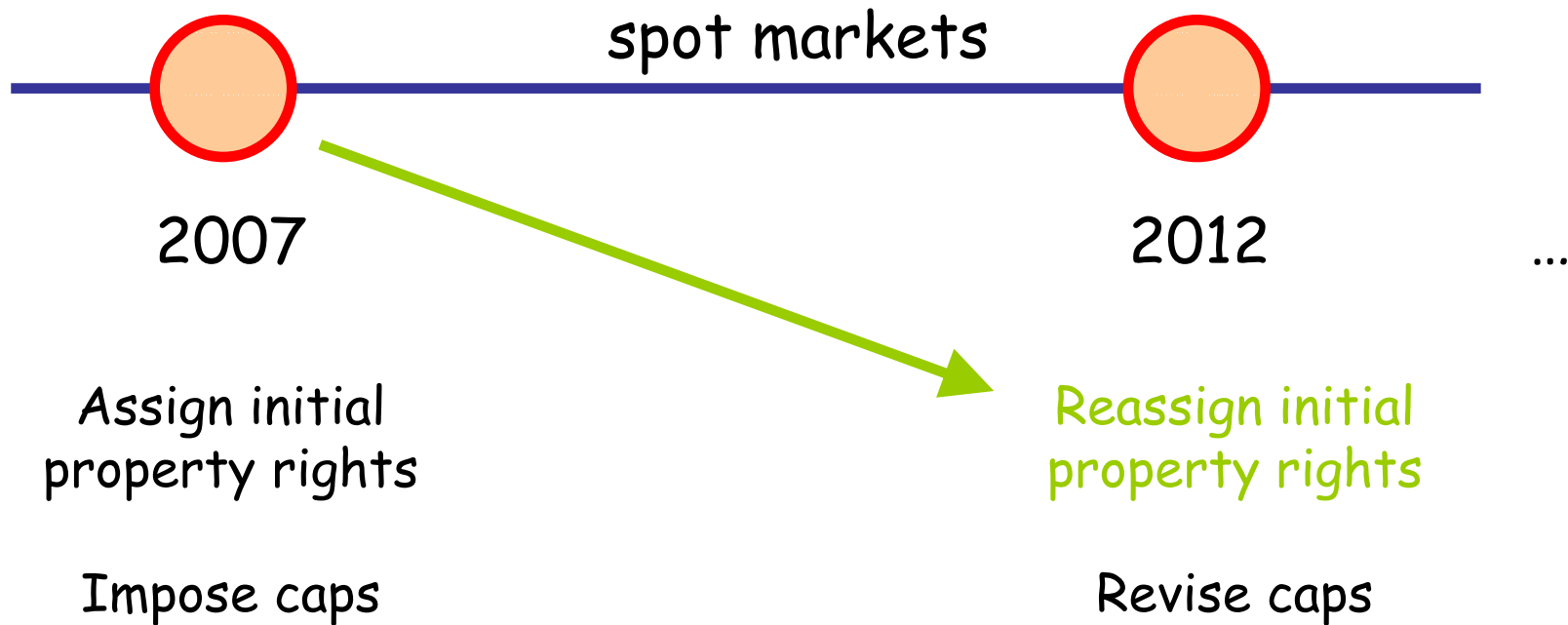
- Multiple rounds
 - allow participants to revise bids
- Important when good space is large and complex
 - 18 hrs, 4 blocks/hr, 10 slots/block, 2 runways, M-F, Sat, Sun: approx 4320 items/airport
 - ten's of airlines, each with hundred's of flights a day
 - multiple airports



How might this work?

- Fix goals (safety, efficiency, regional access, ...)
 - Define goods, & assign initial property rights
 - Host an exchange:
 - goods: landing (takeoff) slots for one plane
 - attributes: time of day, day of week, plane size, flexibility
 - Can impose additional constraints:
 - maximal market share
 - minimal level of competition
 - minimal level of regional service
-  policy tools

Long-term vs. Spot markets



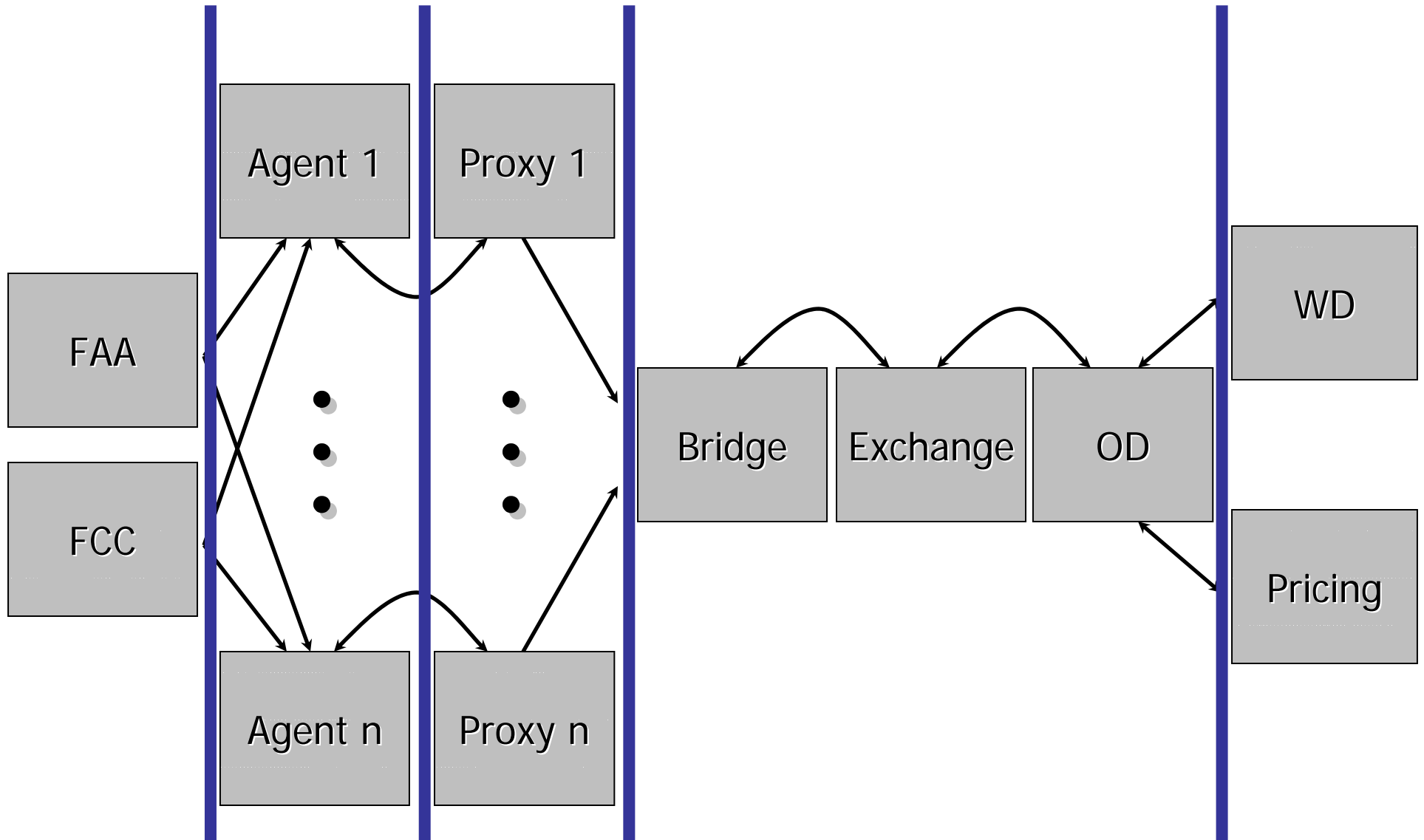
Prototyping an Exchange

- Summer, 2001
 - proposed clearing rules for a **one-shot** combinatorial exchange (Parkes, Kalagnanam, Eso, IJCAI'01)
 - October, 2001
 - presented exchange design to FCC-Wye river conference
 - Spring, 2003
 - experiments on incentive properties of "Threshold" rule
 - Summer, 2003
 - initial design for an **iterative** exchange
 - November, 2003
 - presented iterative design to FCC-Wye river conference
- Spring, 2004
 - **CS 286r**: Project class focused on "Iterative Combinatorial Exchanges"
 - www.eecs.harvard.edu/~parkes/cs286r
 - study FCC and FAA domain problems

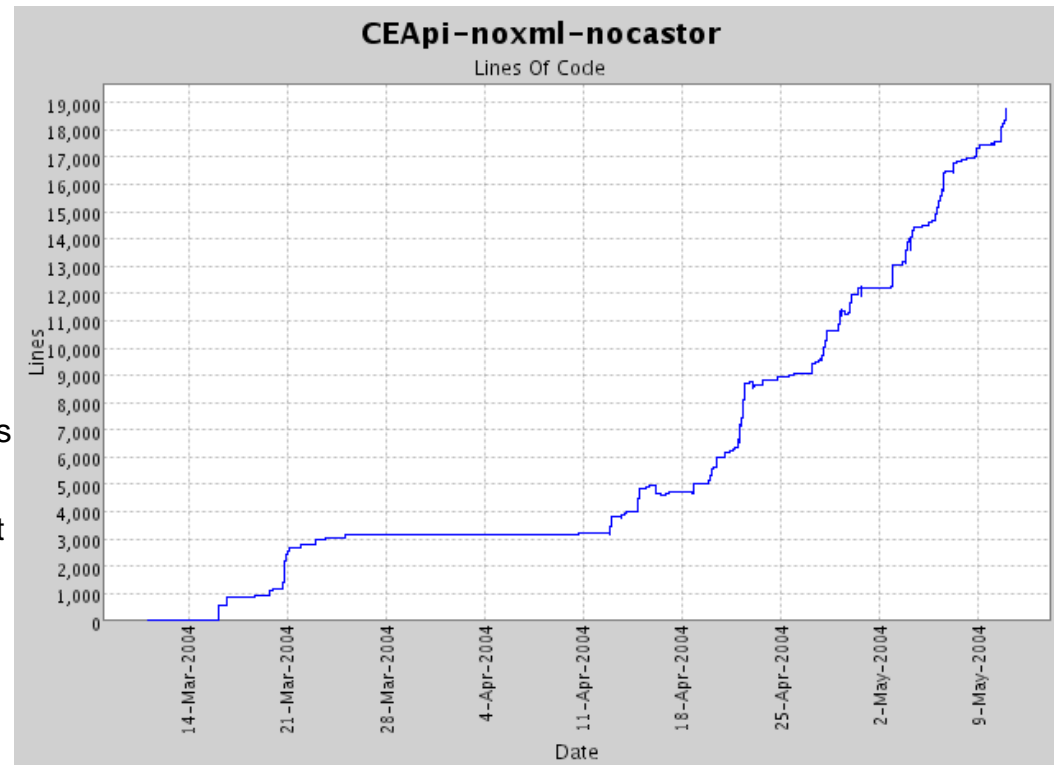
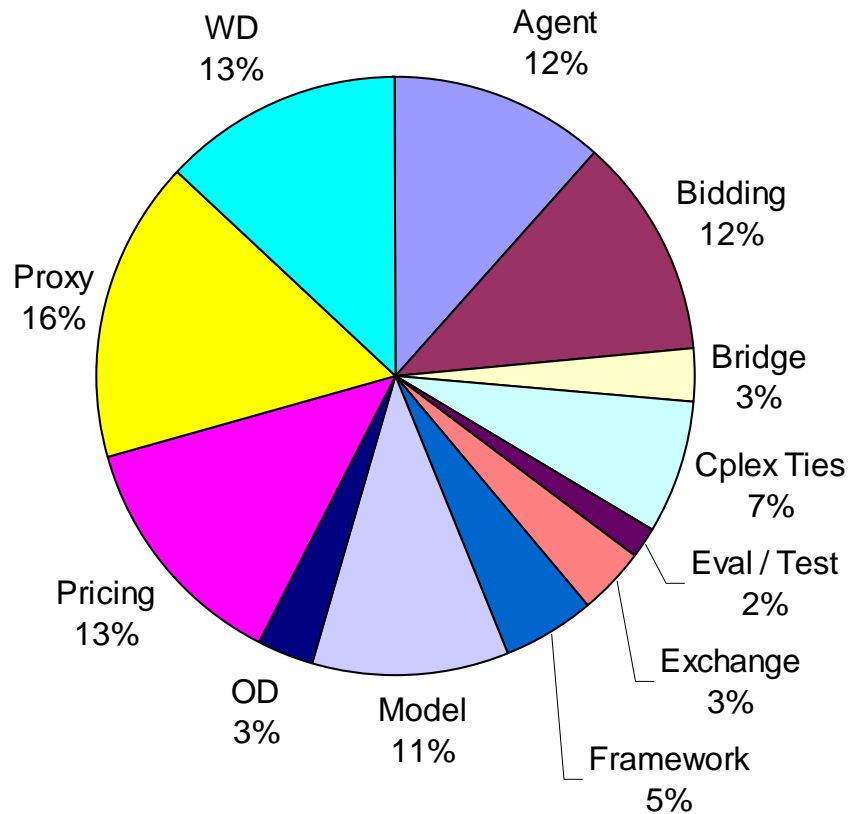
Exchange Design

- Bidding Language
 - expressive, compact
- Winner determination
 - scalable
- Feedback
 - prices
- Activity rules, termination
 - drive progress
- Distribution
 - final payments

Four Components



Code Development



Eclipse development environment, Java, CVS support

Interfaces between components, Design for threading and distributed processing

CPLEX RMI servers, sitting behind a load balancer

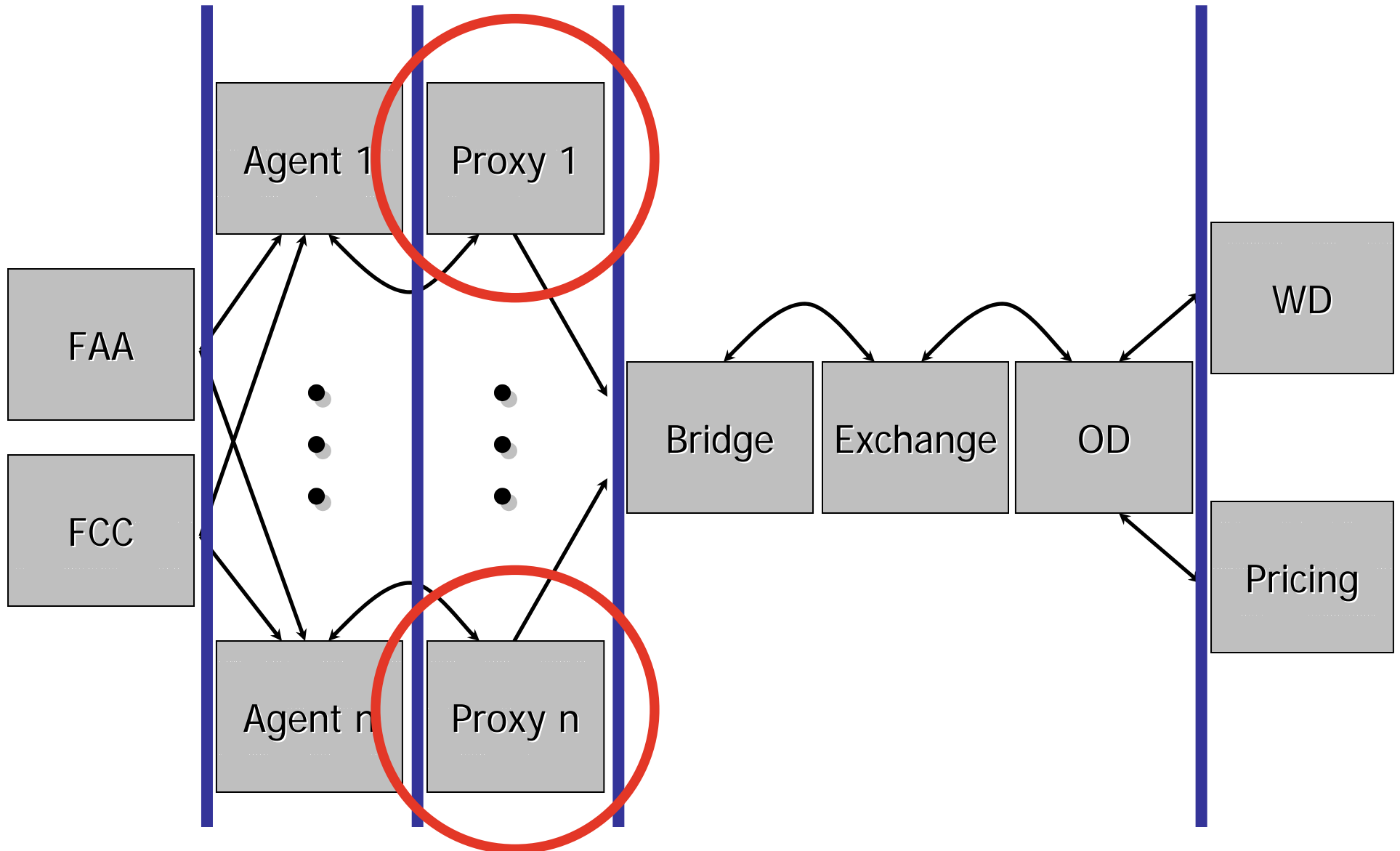
XML language for component specifications, and simulation infrastructure

Run on two, four-processor, Blade machines

Class mantra

"No enumeration of goods..."

First Component: Bidding Language



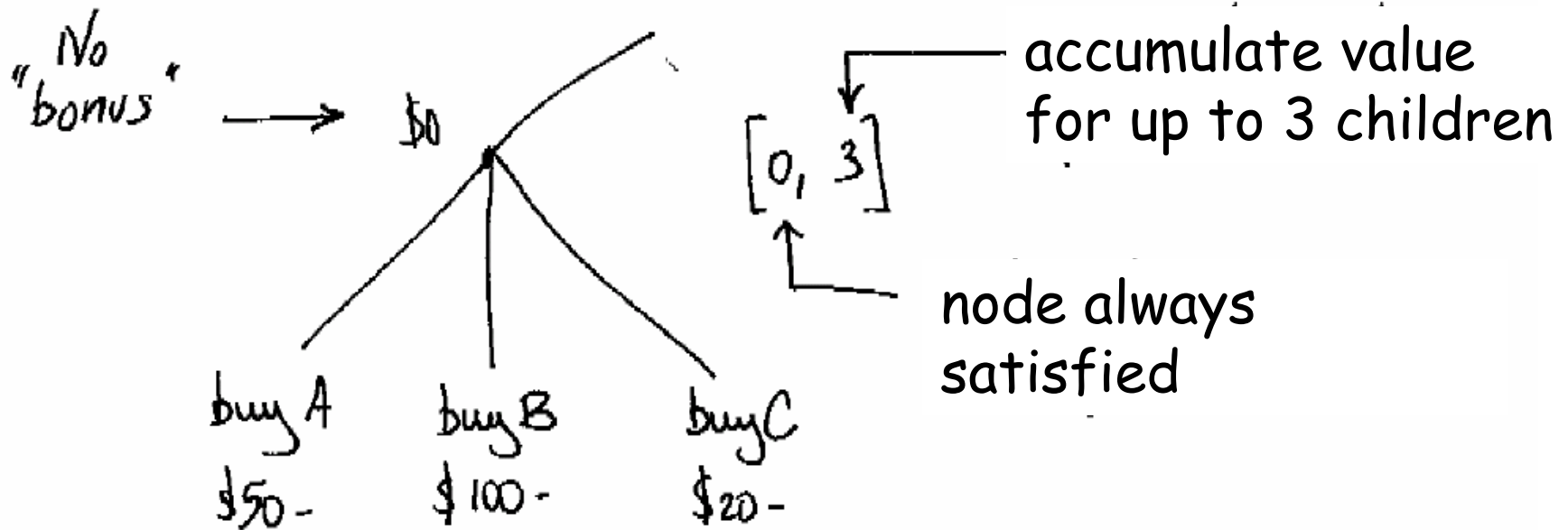
Bidding in the Exchange

(related to Boutilier's L_{GB} language)

- Compact and Expressive bidding language
 - logical structure ("one of", "all of", "some of", ...)
 - goods at leaves ("buy A", "sell B")
- Buyer:
 - define value for acquiring new slots
- Seller:
 - define value (negative) for no longer holding slots
- Mixed buyer/sellers
 - define value (+ve, -ve) for a "bundled" trade

E.g. Buy any number of slots.

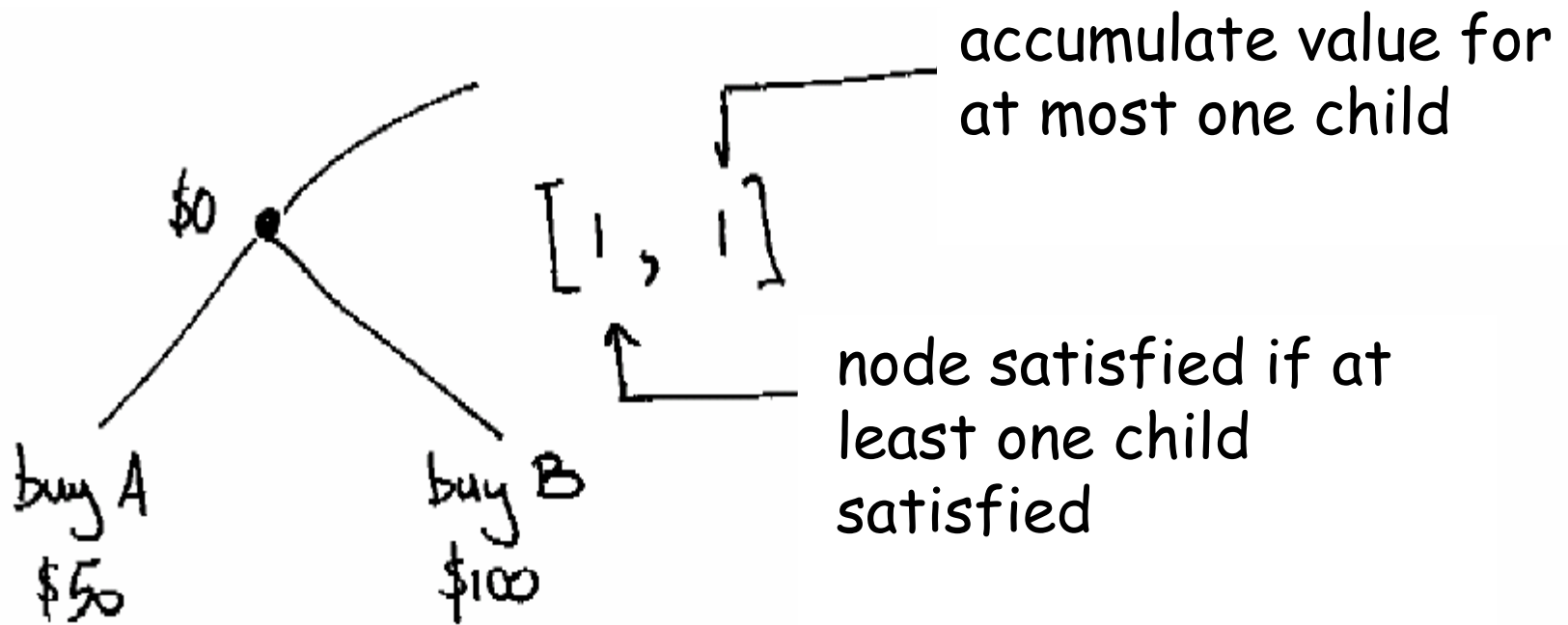
OR==(0,K)



"buy A" leaf satisfied if item A allocated to that node.

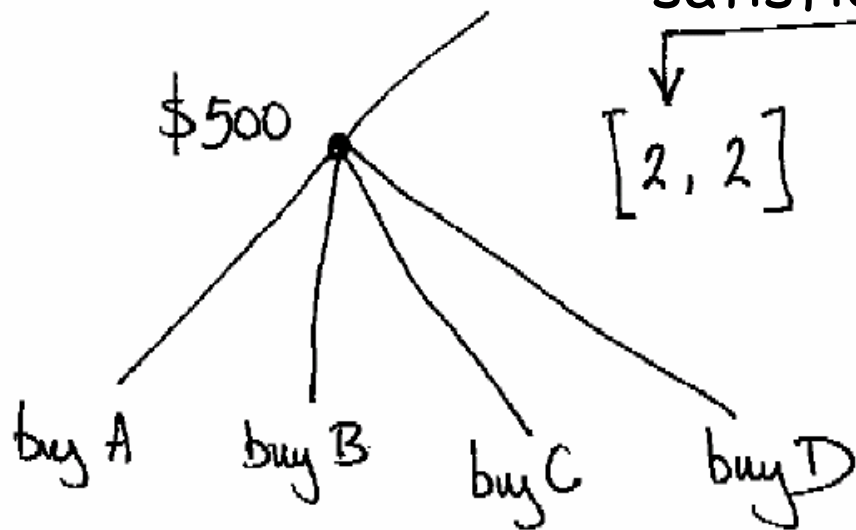
E.g. Buy at most one slot

$$\text{XOR} == (1, 1)$$

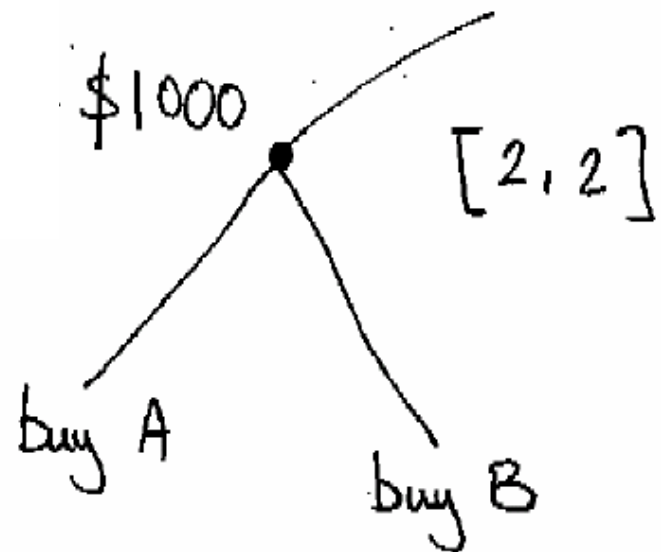


Buy any two slots, Buy all slots...

satisfied if any
two children are
satisfied



$[2, 2]$

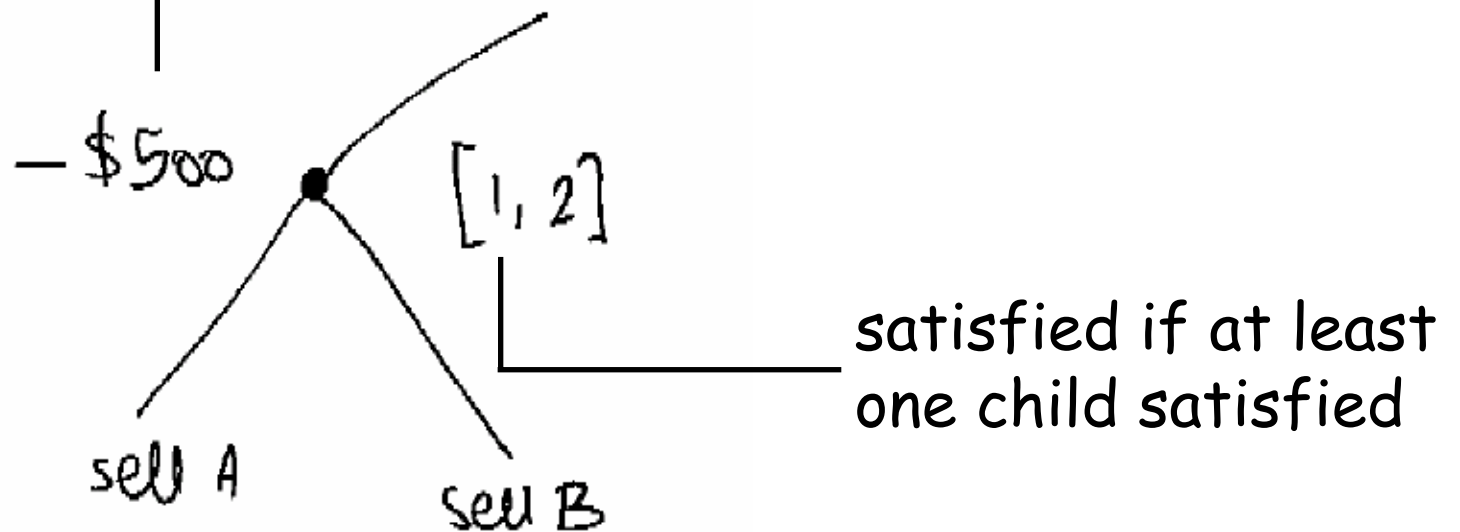


$[2, 2]$

AND== (K, K)

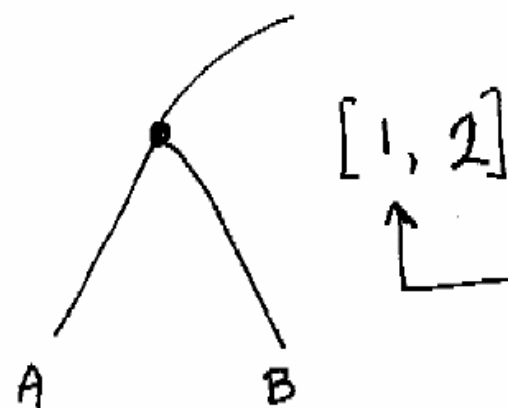
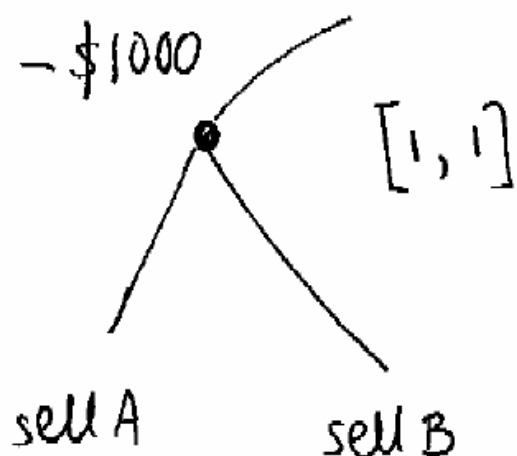
E.g. Sell all slots

loss in value for
selling one or both
of these slots



"sell A" leaf satisfied if item A **not** allocated to agent.

E.g. Sell at most one slot...

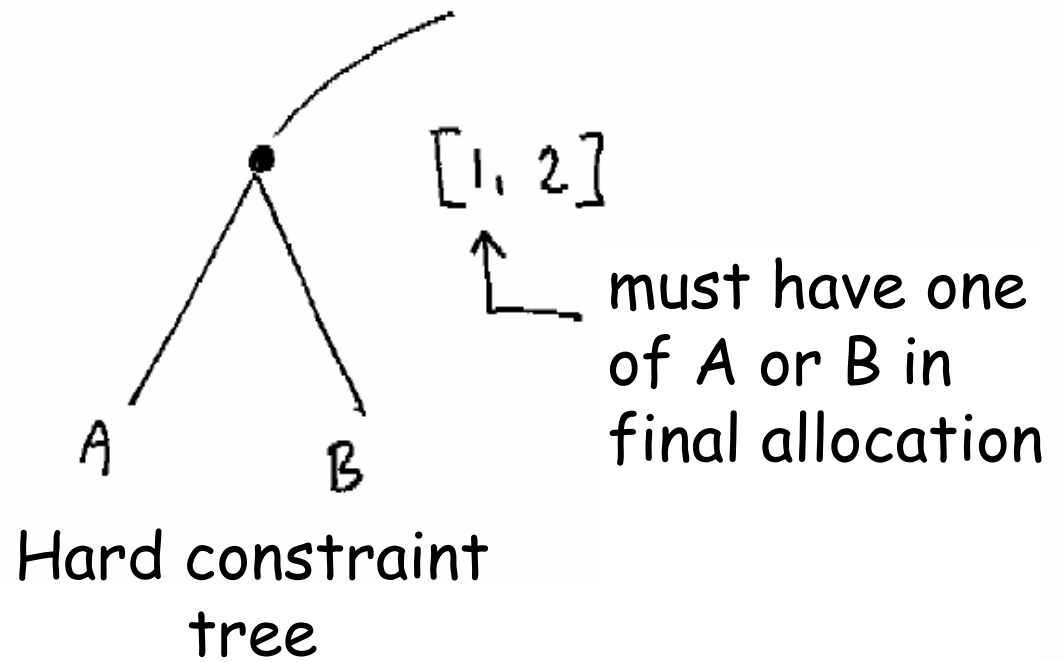
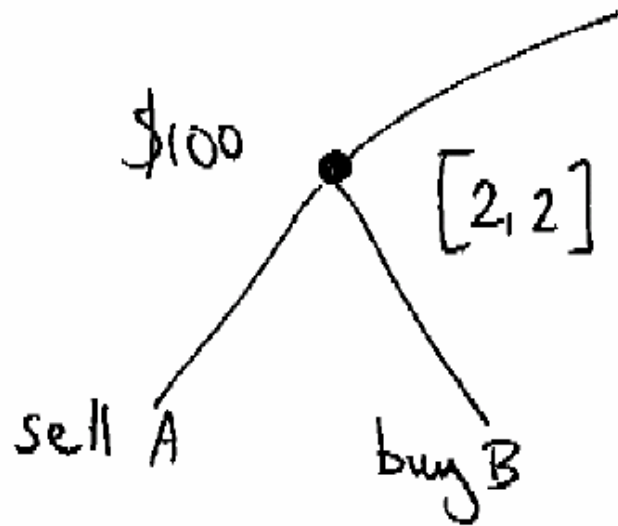


must have one of
A or B in final
allocation

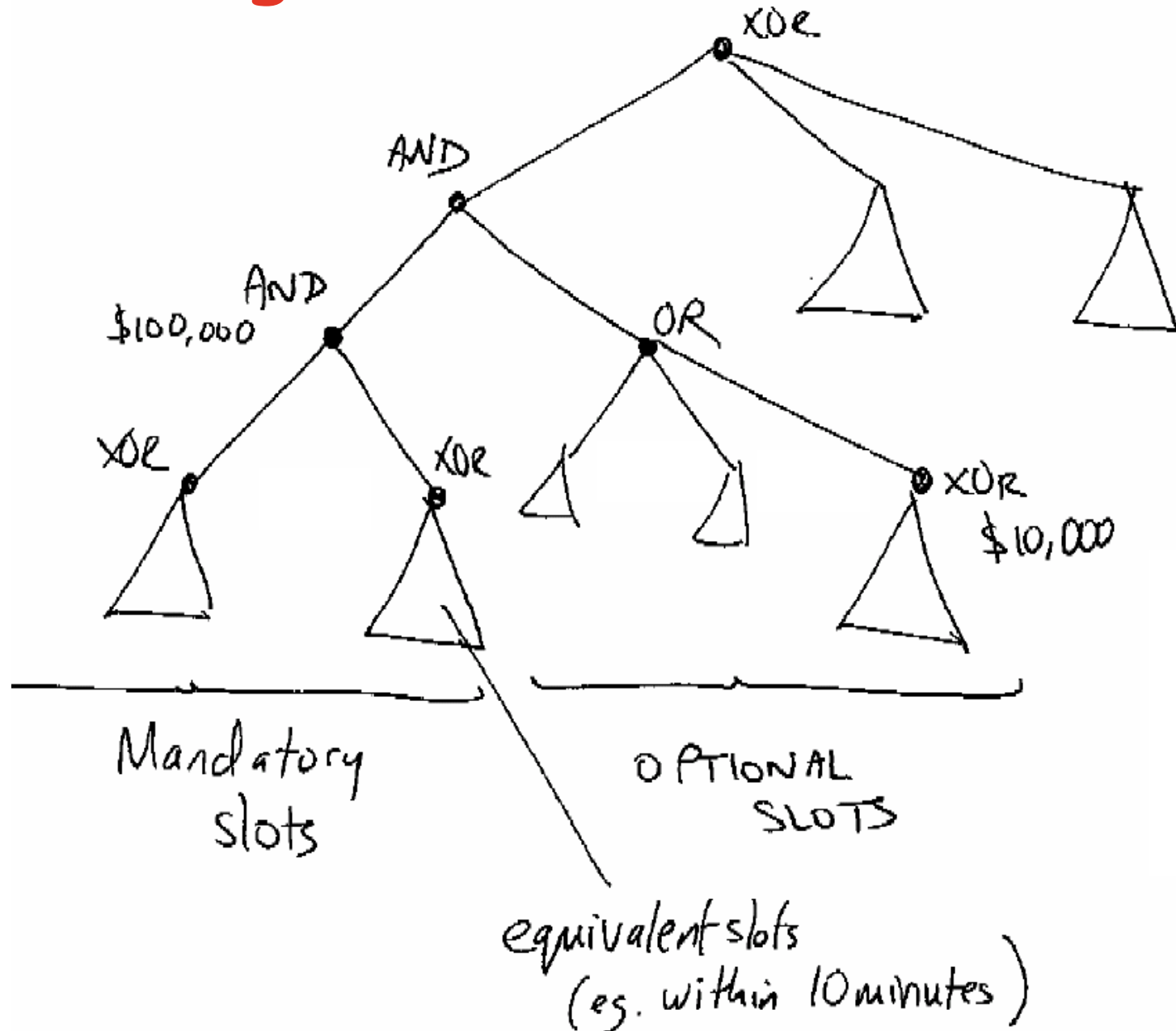
Hard constraint
tree

A seller can supplement valuation tree with
hard constraints (satisfied by initial allocation)

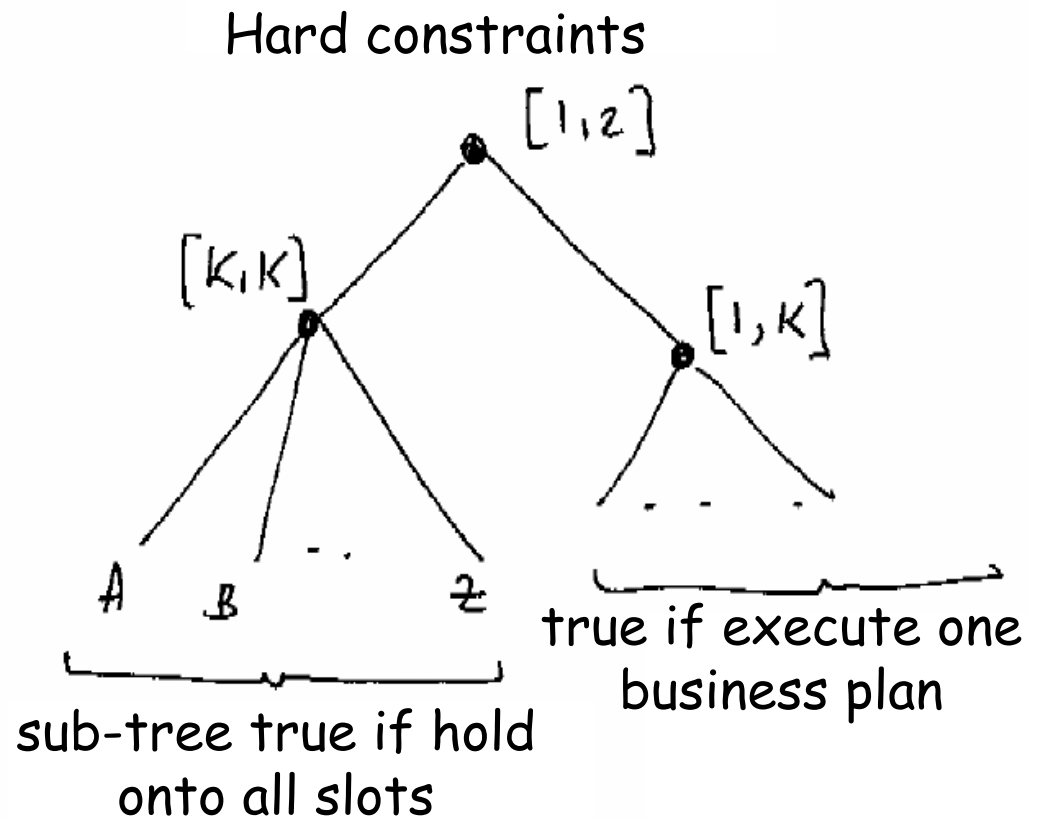
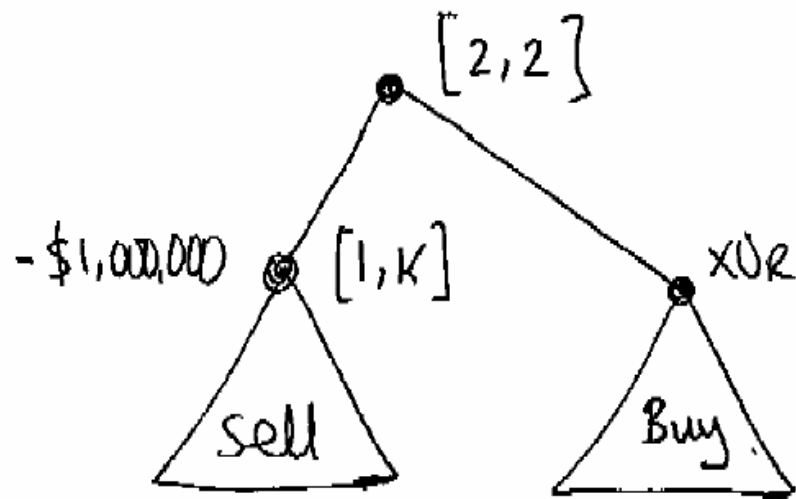
E.g. Swap A for B



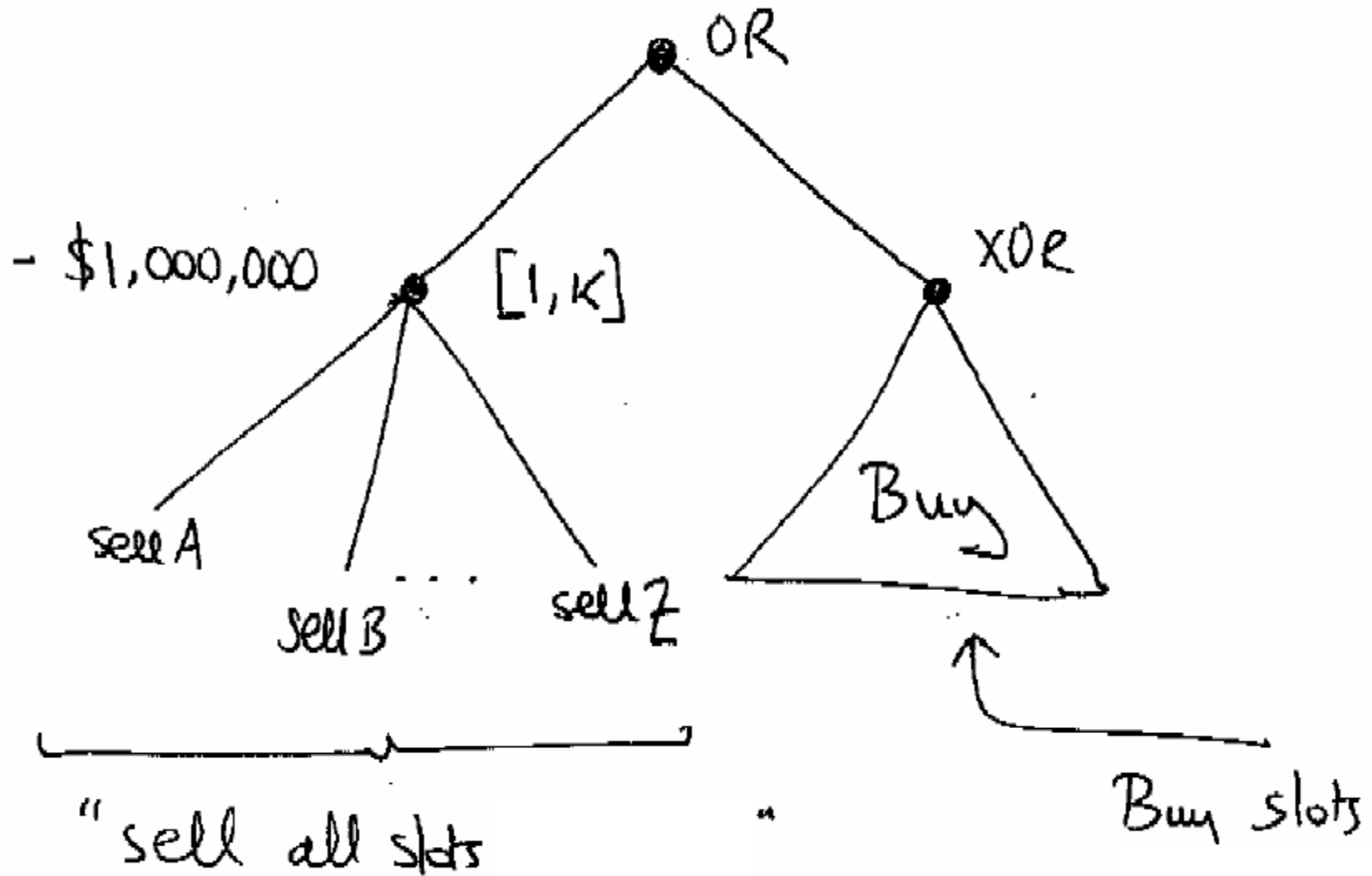
E.g. More Elaborate Plans...



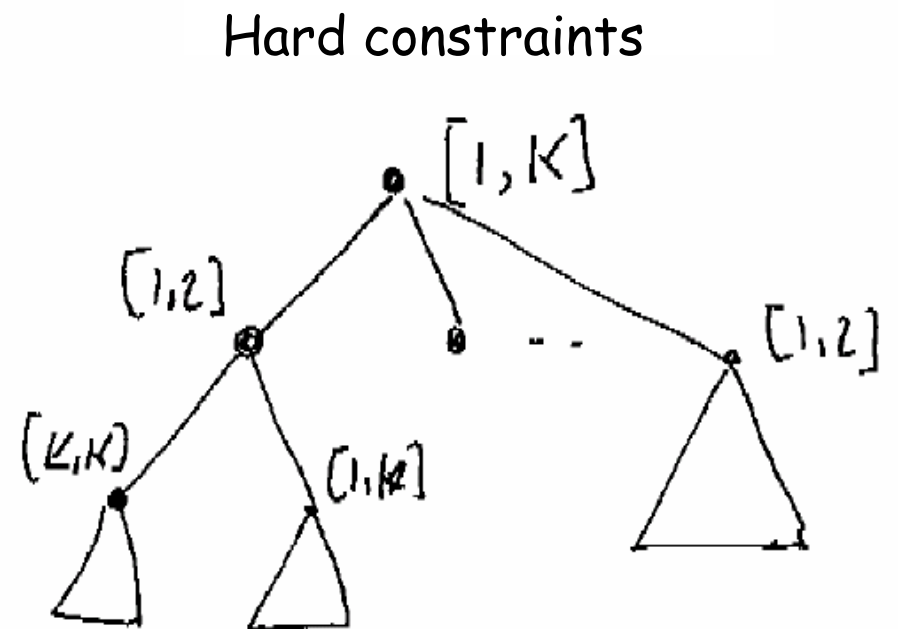
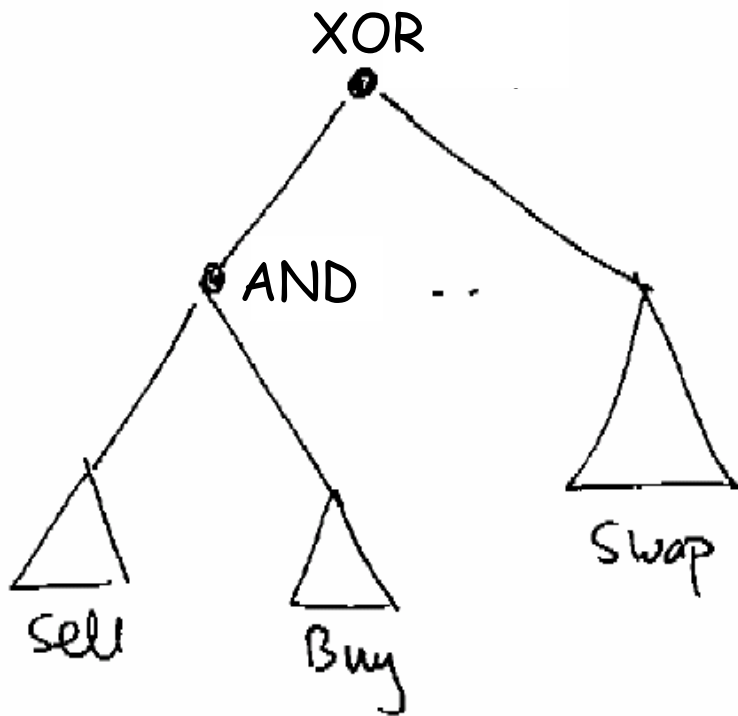
E.g. "Swap Peak Slots for Off-peak"



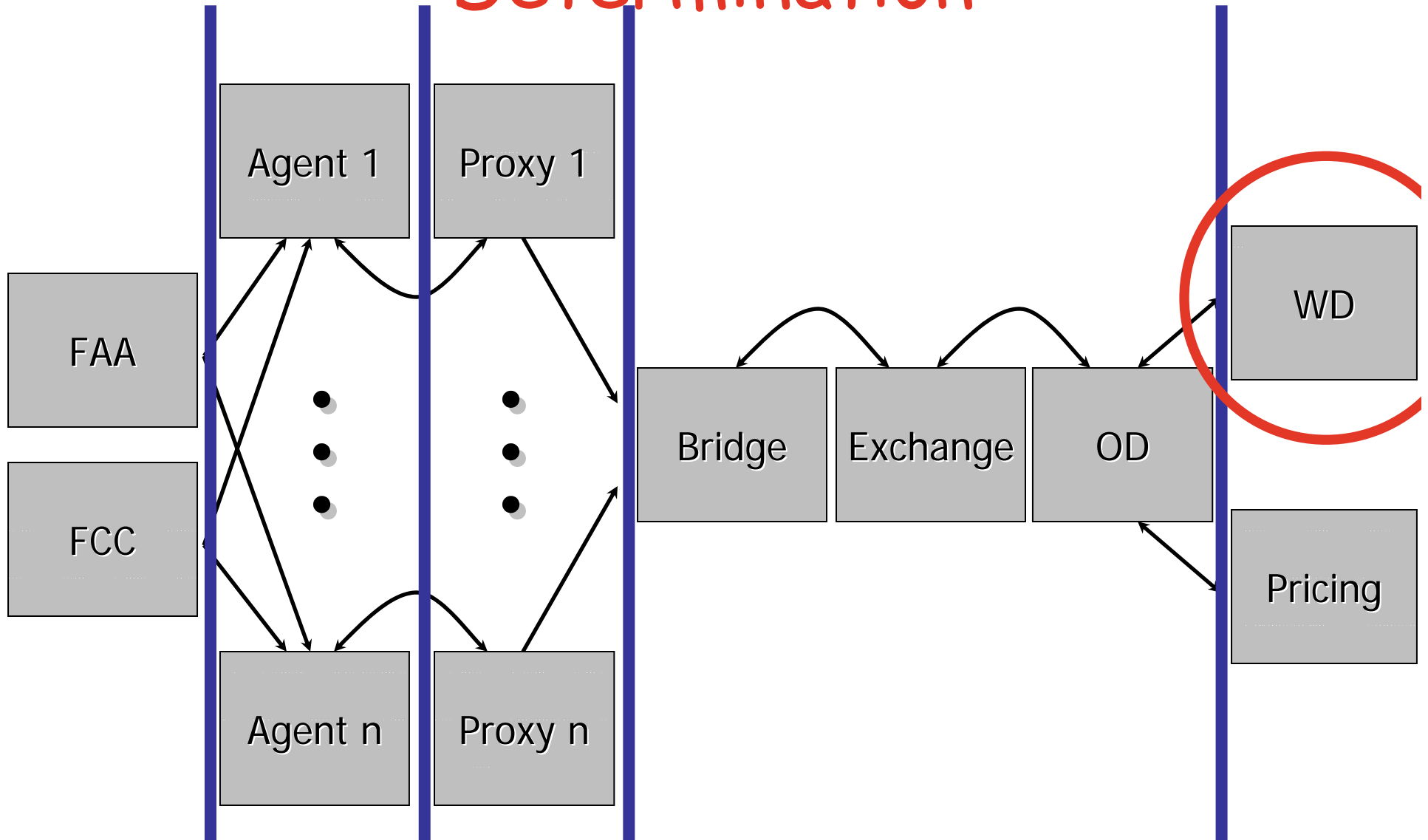
E.g. Sell for sure, Try to Buy back



E.g. Multiple Business Plans...

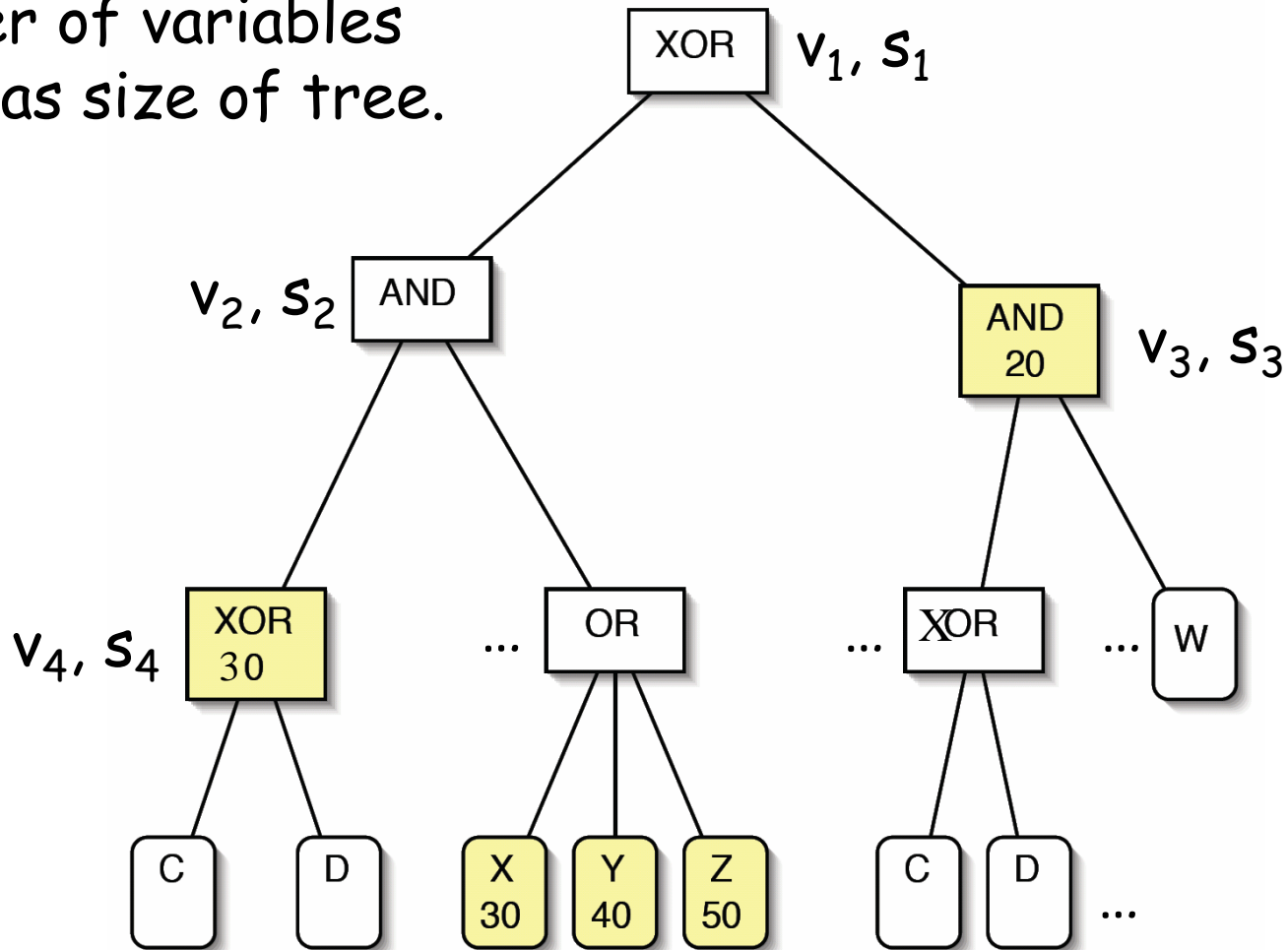


Second Component: Winner Determination

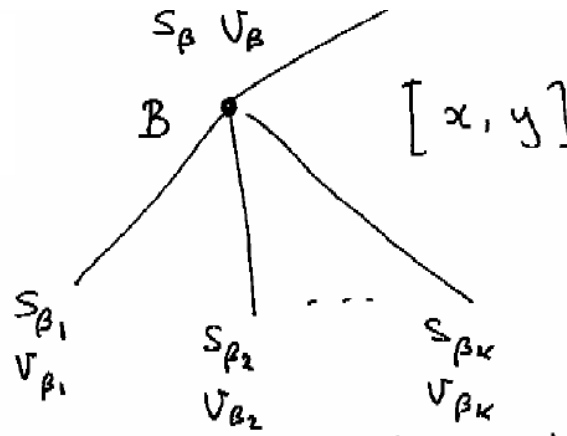


Winner-Determination

Formulate as a MIP.
Number of variables
scales as size of tree.



Internal Node Constraints



if at least x children satisfied, then

$$s_\beta = 1$$

$$v_\beta = B + \max_{\substack{K \subseteq \text{Sat}_\beta \\ |K| \leq y}} \sum_{j \in K} v_{\beta_j} + \text{Pen}_\beta$$

else, $s_\beta = 0, v_\beta = \text{Pen}_\beta$

Pen_β is total value across -ve valued children

Sat_β is set of satisfied children

$$x \leq s_\beta \cdot \sum_i s_{\beta_i}$$

$$v_{\beta_i} \cdot M \cdot t_{\beta_i}, \delta_i$$

$$\sum_i t_{\beta_i} \cdot y$$

General MIP Formulation

$$\max \sum_{i \in N} v_i$$

s.t. internal node constraints

hard constraints

$$\sum_{\beta \in \text{Sell}(A)} s_{\beta} \cdot 1 - x_i(A)$$

$$\sum_{\beta \in \text{Buy}(B)} s_{\beta} \cdot x_i(B)$$

$$\sum_{i \in N} x_i(A) \cdot \text{supply}(A)$$

...

assign tokens
to leaf nodes

supply +
demand
balance

buy(A): have good, one true. don't have good, all false
sell(A): have good, all false. don't have good, one true.

$x(\phi)$ variables define allocation.

Leaves

- satisfied if "A-token"
buy A assigned to this leaf
- satisfied if "not A-token"
sell A assigned to this leaf

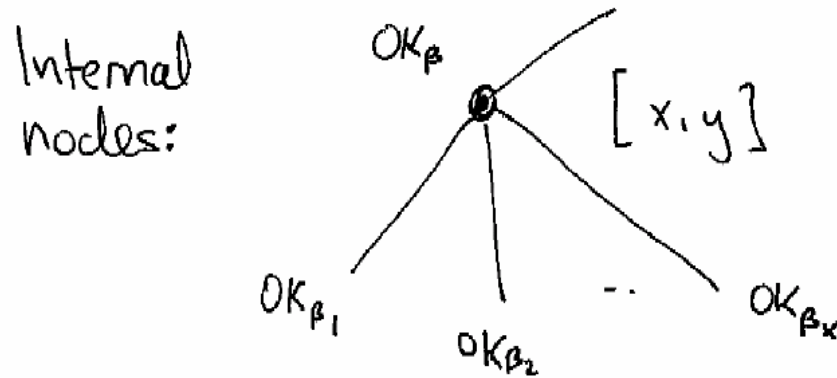
every node:

$$v_{\beta} = B_{\beta} \phi s_{\beta} + \sum_i v_{\beta i}$$

$$v_{\beta i} = M \phi s_{\beta i}, \delta i$$

Hard Constraints Tree

Feasible , Root node satisfied



$OK_{\beta} = \text{True}$, only when between x and y children are True.

●
A

satisfied if good A is allocated to agent, and assigned to this leaf in the constraint tree

Simple Example

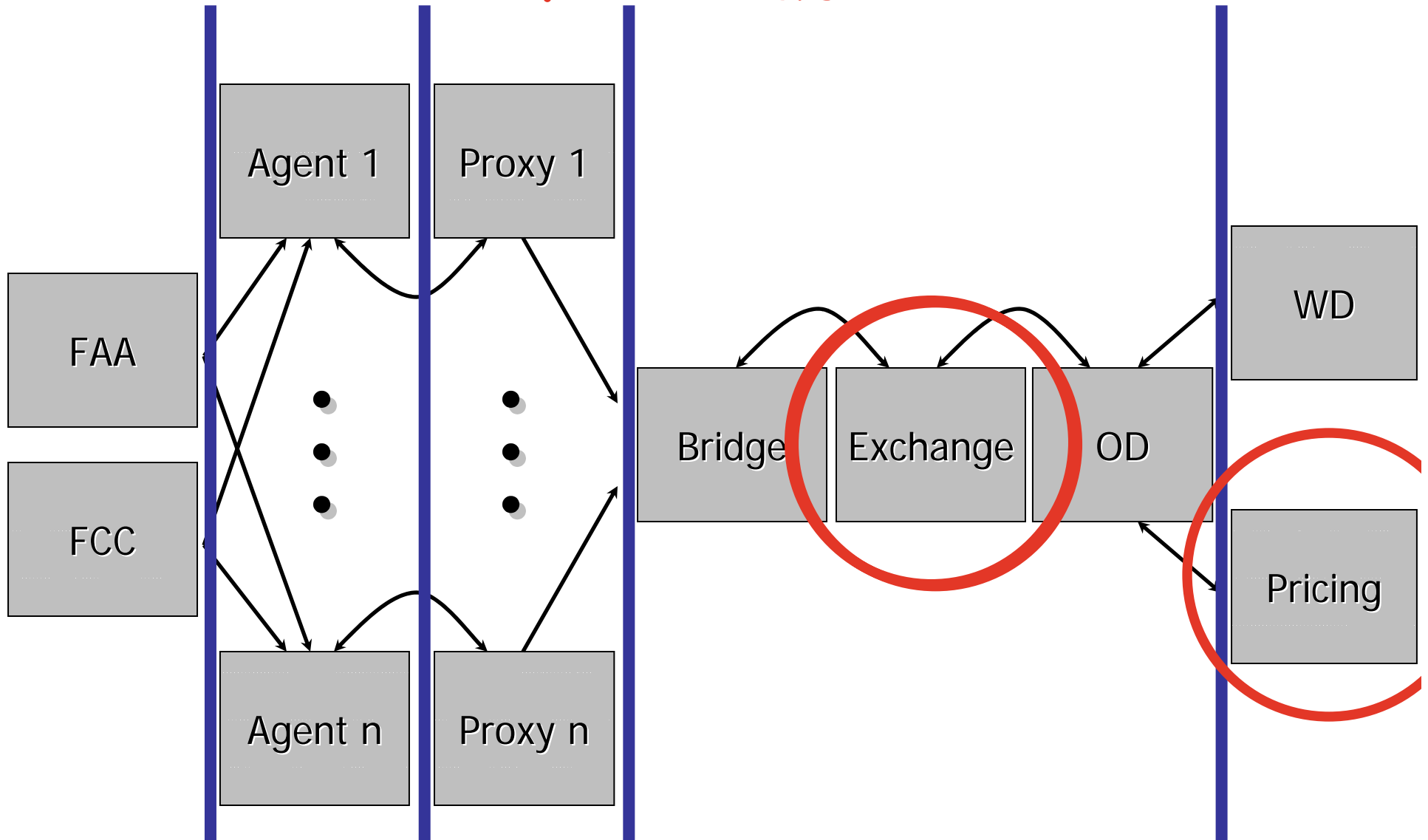


Final allocation: Agent 3 sells B to agent 1.

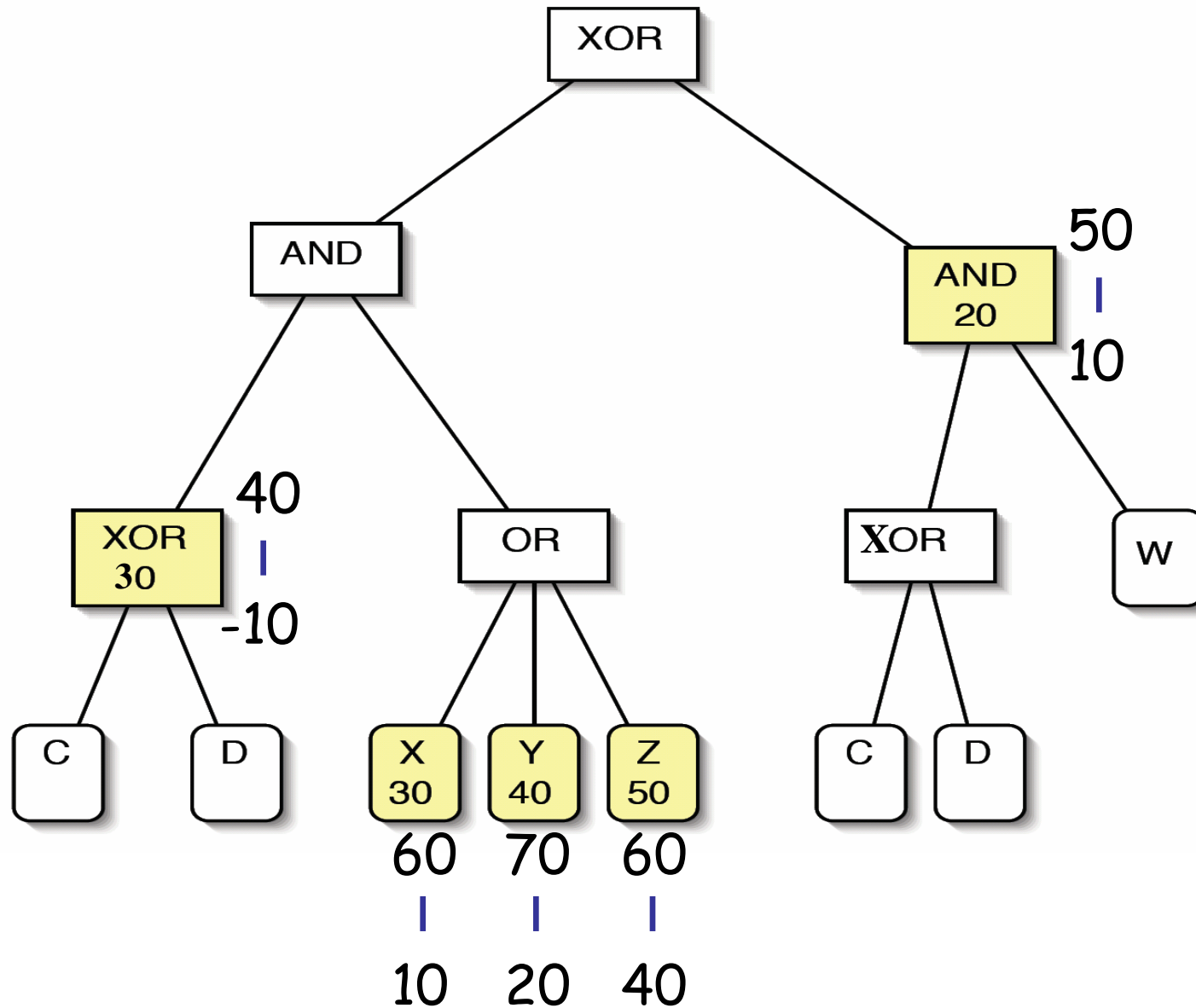
Surplus: \$50

"Threshold" payments: Agent 1 pays \$40 to agent 3.

Third Component: Feedback

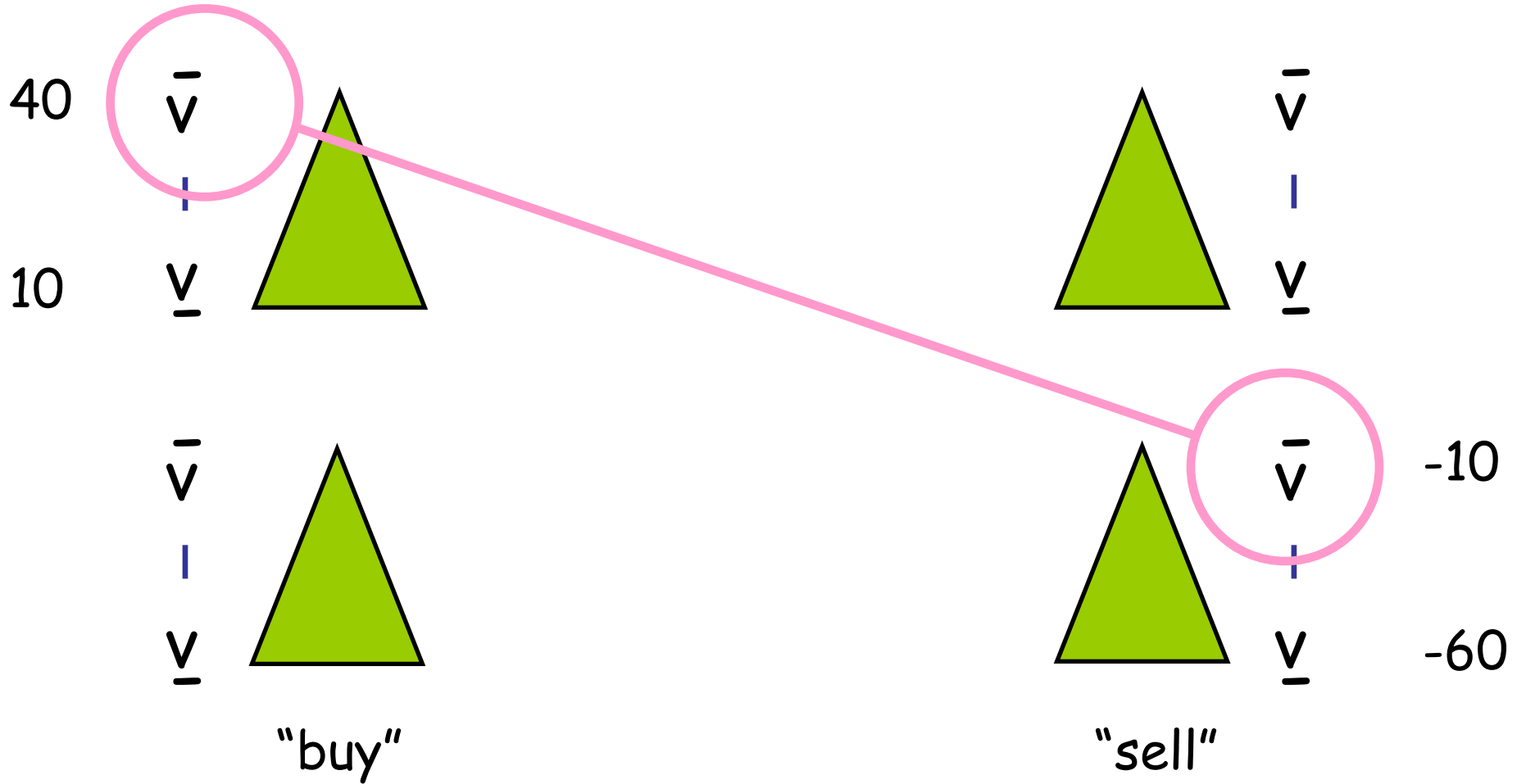


Incremental Bidding



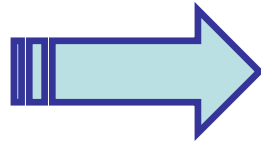
Exchange Phases

Phase I
Optimistic clearing

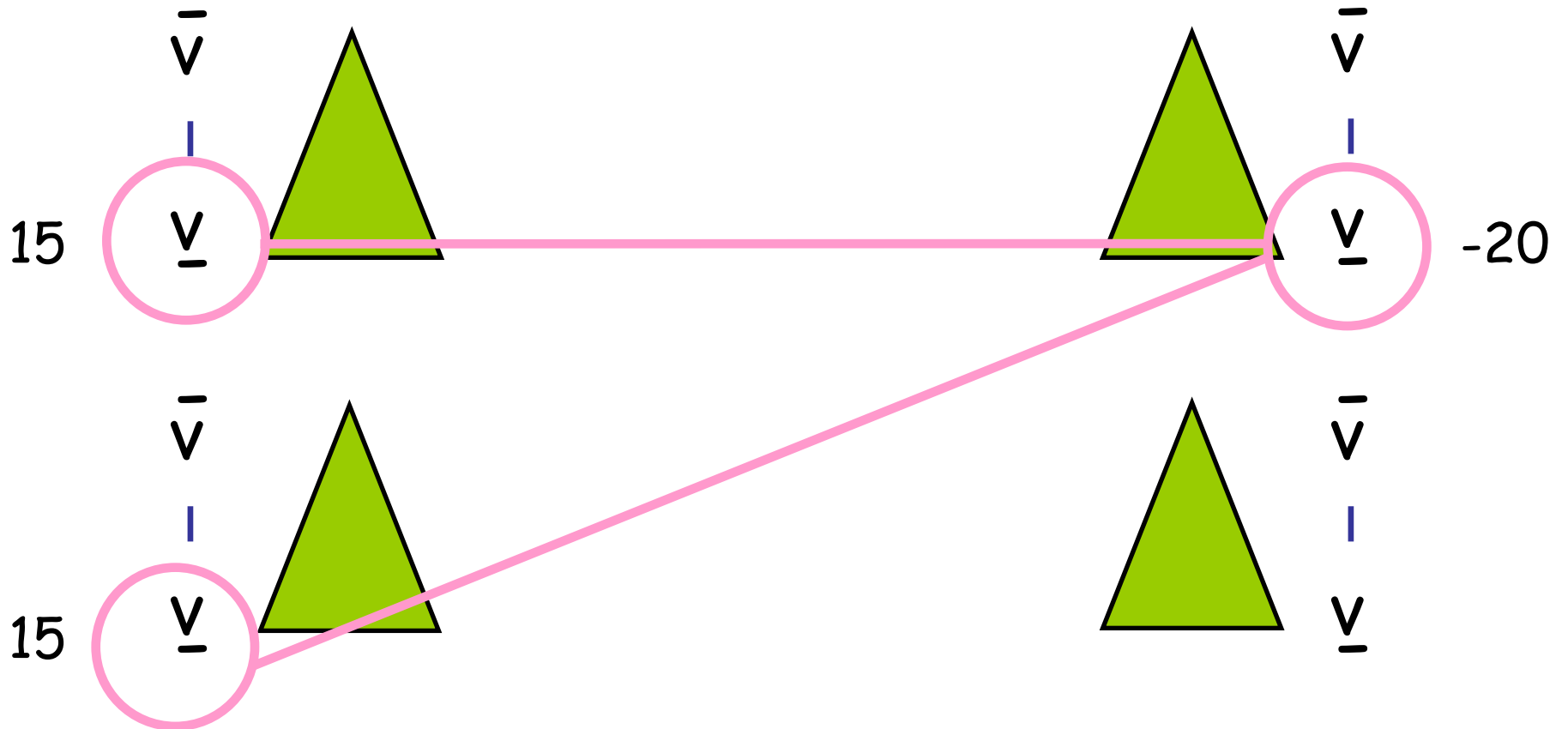


Exchange Phases

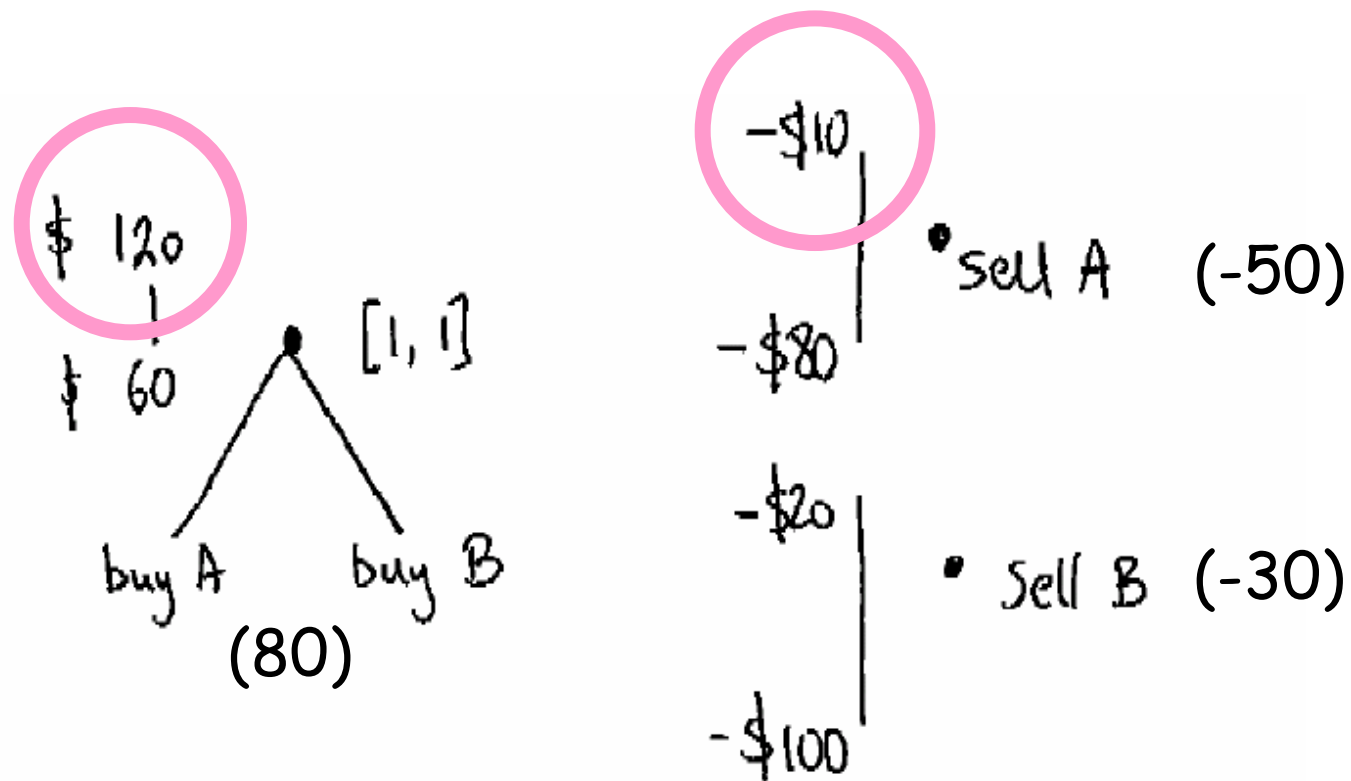
Phase I
Optimistic clearing



Phase II
Pessimistic clearing



Example: Round 1

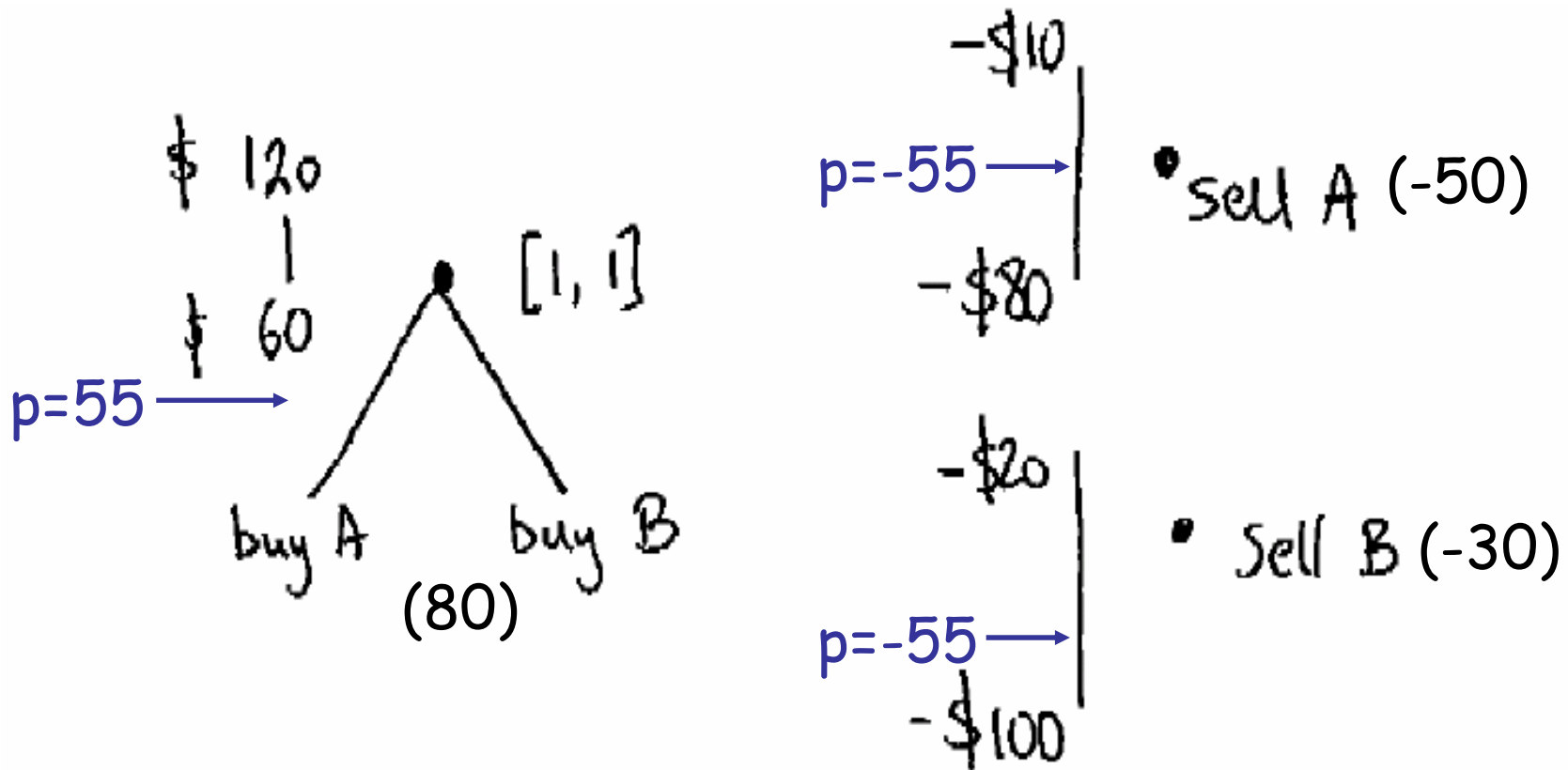


10. $p_A = 120$, $p_A \cdot p_B$ (winner optimistic values)

$p_B = 100$ (loser pessimistic value)

) $p_A = p_B = 55$

Bid Refinement: Round 1

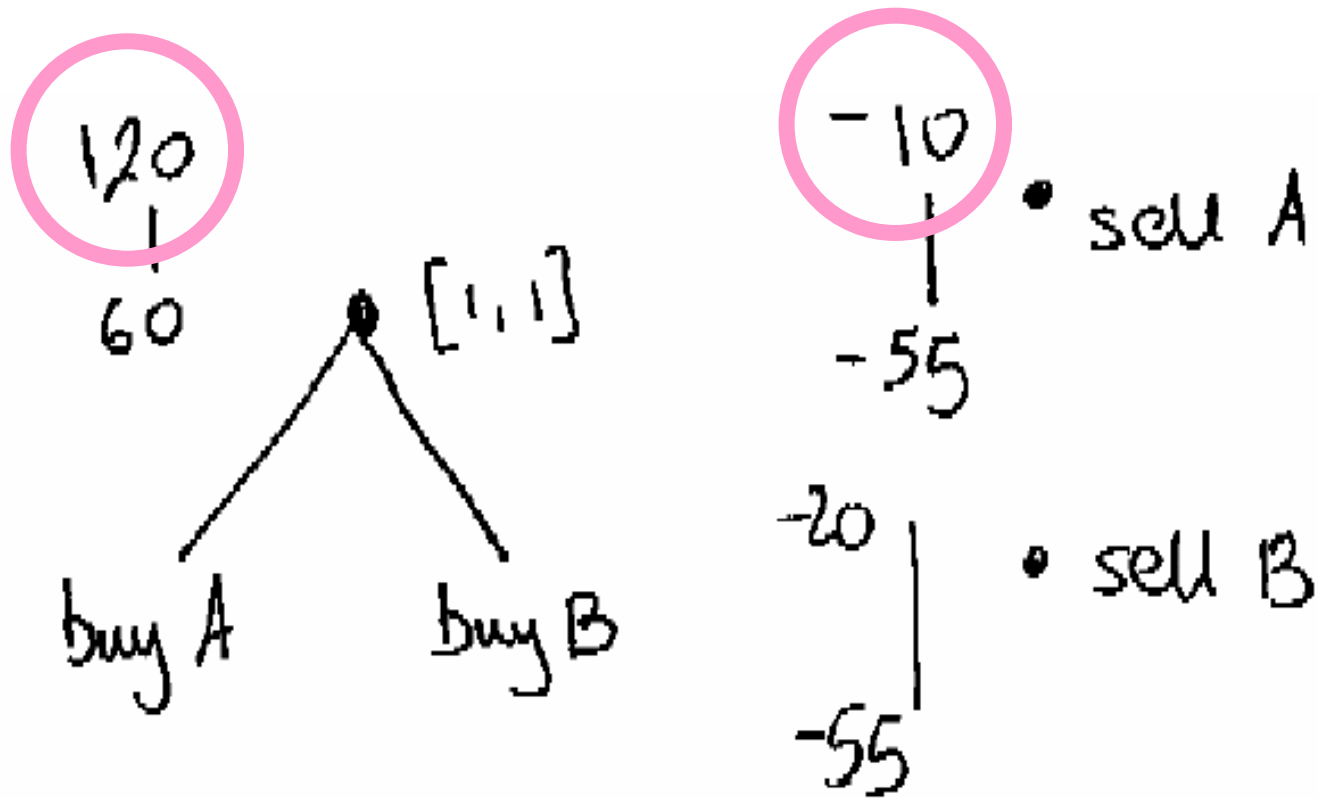


Activity rules:

Winners: ask winners to refine u.b.'s to meet price

Losers: ask losers to refine l.b.'s to meet price

Round 2

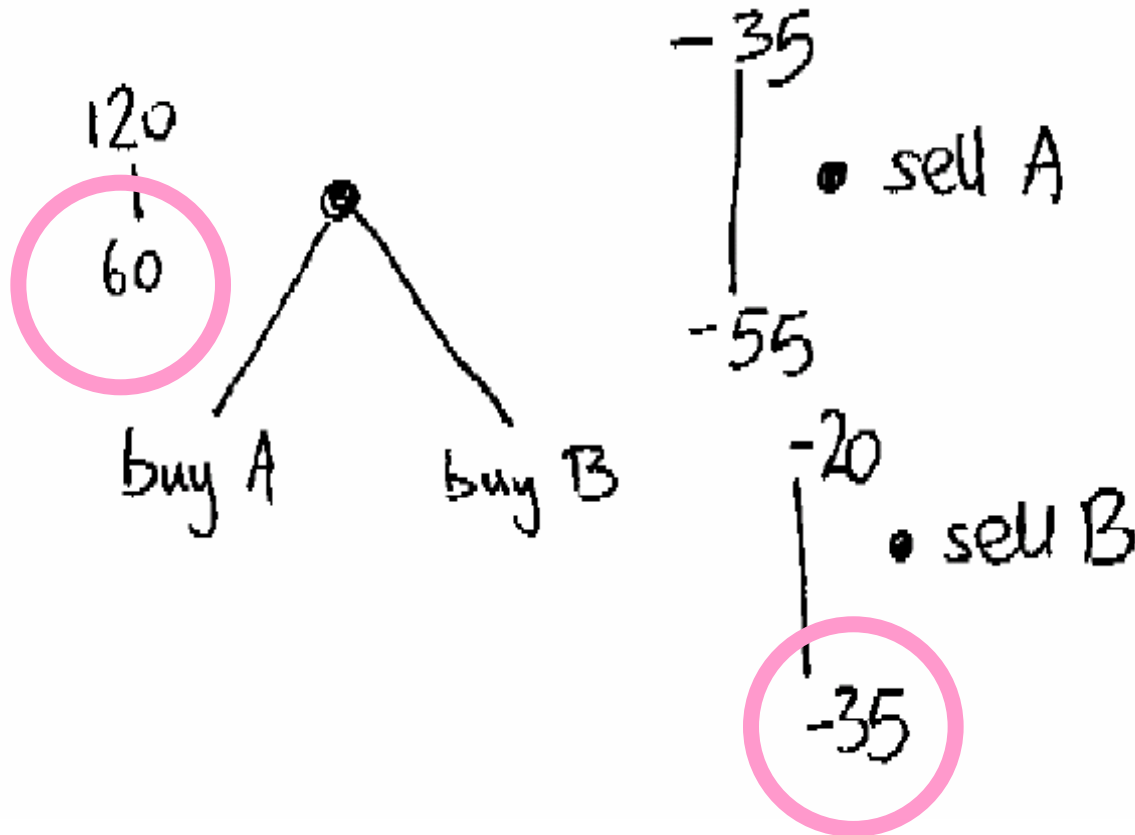


10. $p_A = 120$, $p_A \cdot p_B$ (optimistic values)

$p_B = 55$ (pessimistic value)

) $p_A = p_B = 35$

Round 3

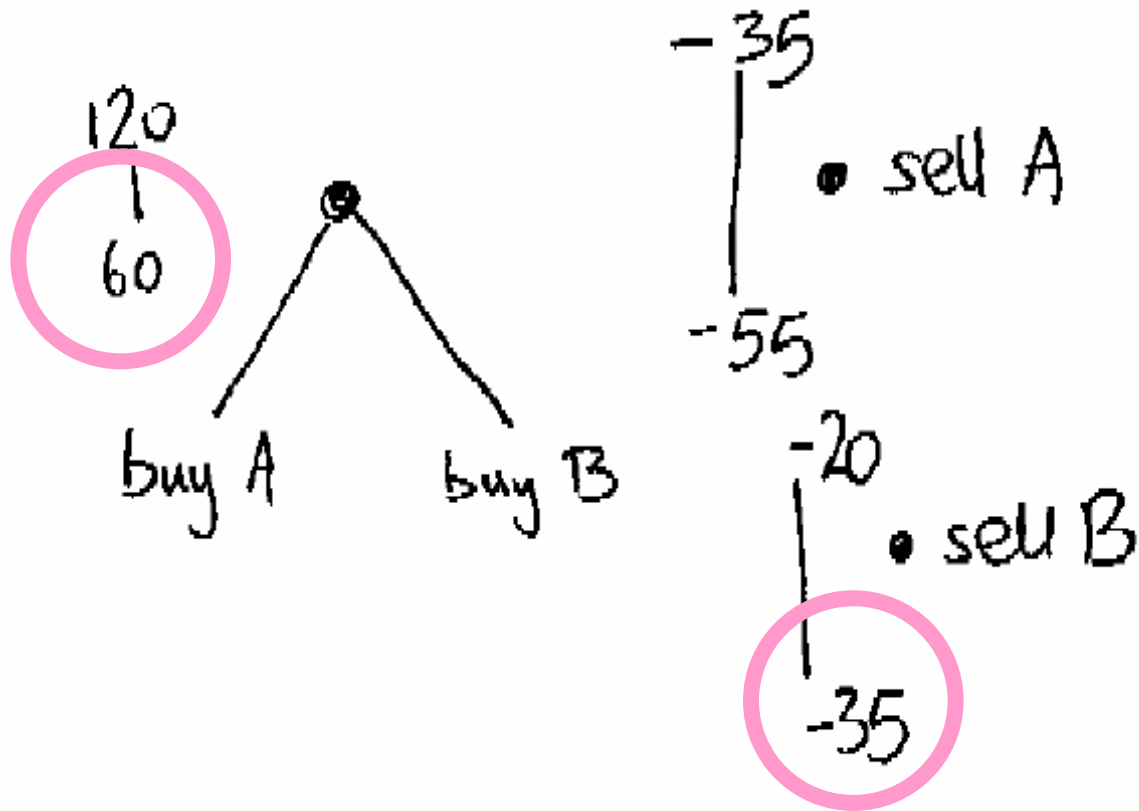


35. $p_B = 60$, $p_B = p_A$ (pessimistic values)

$p_A = 35$ (optimistic value)

) $p_A = p_B = 35$

Last & Final Round



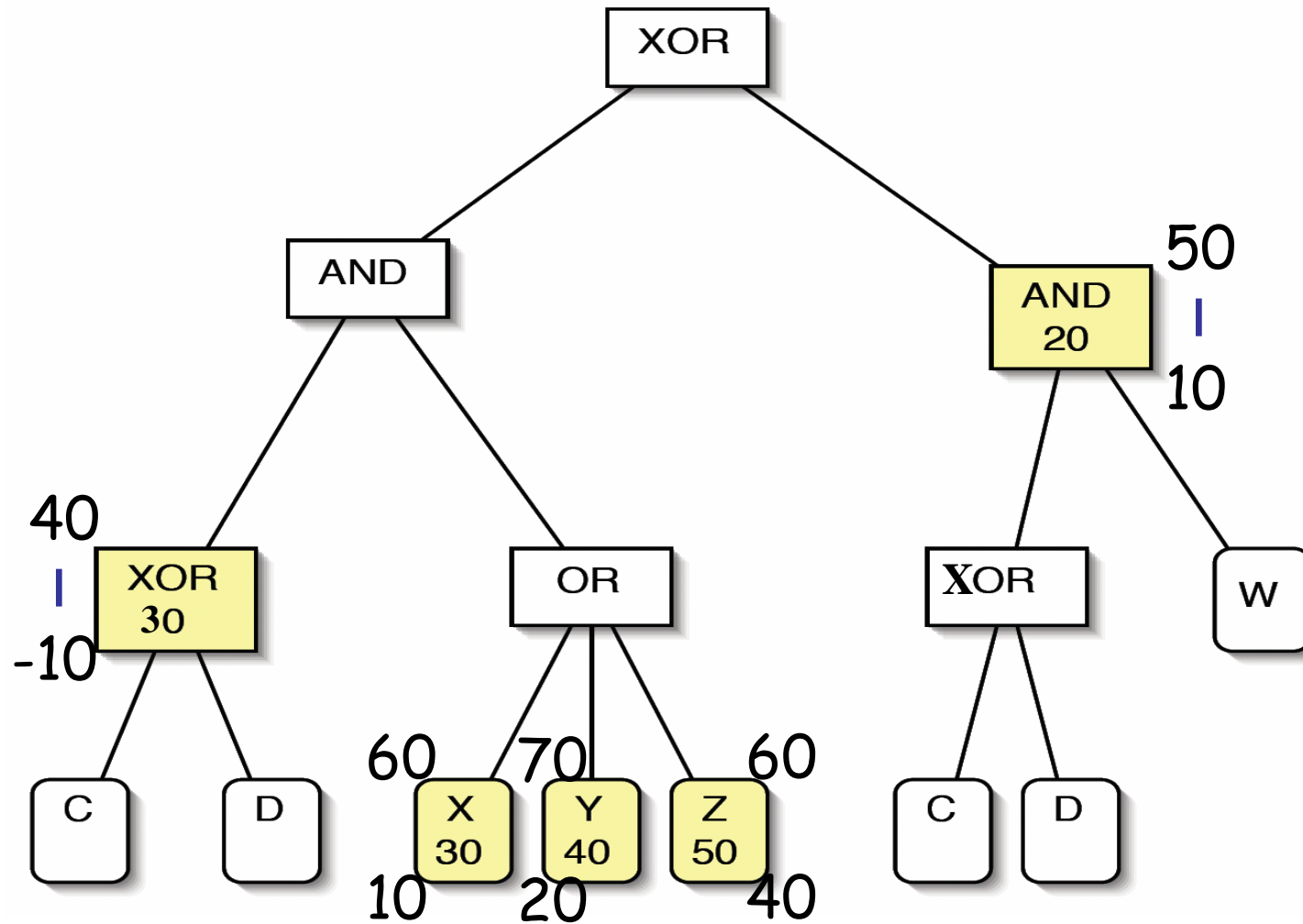
Final allocation: Agent 3 sells B to agent 1.

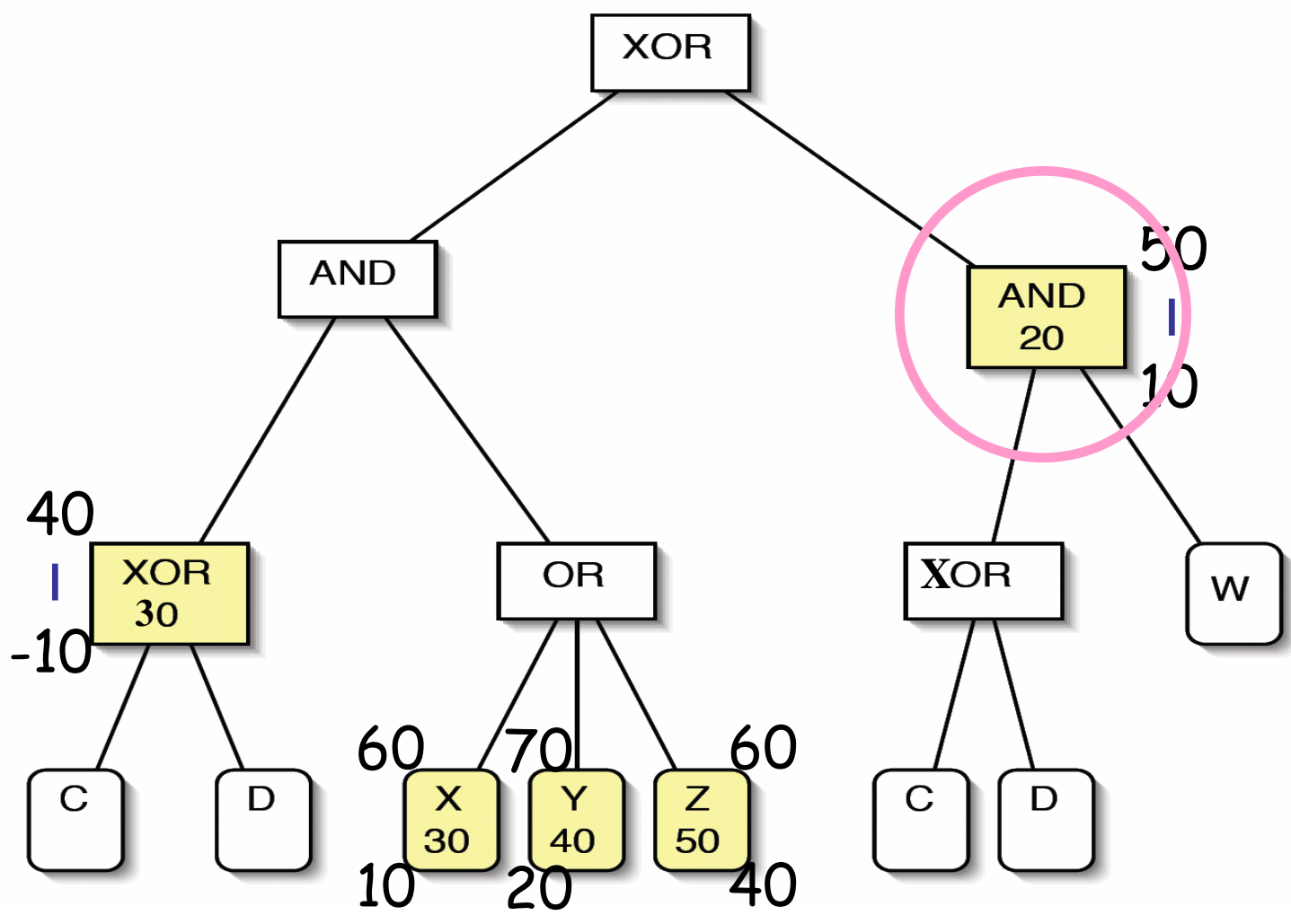
"Threshold" payments: Agent 1 pays \$40 to agent 3.

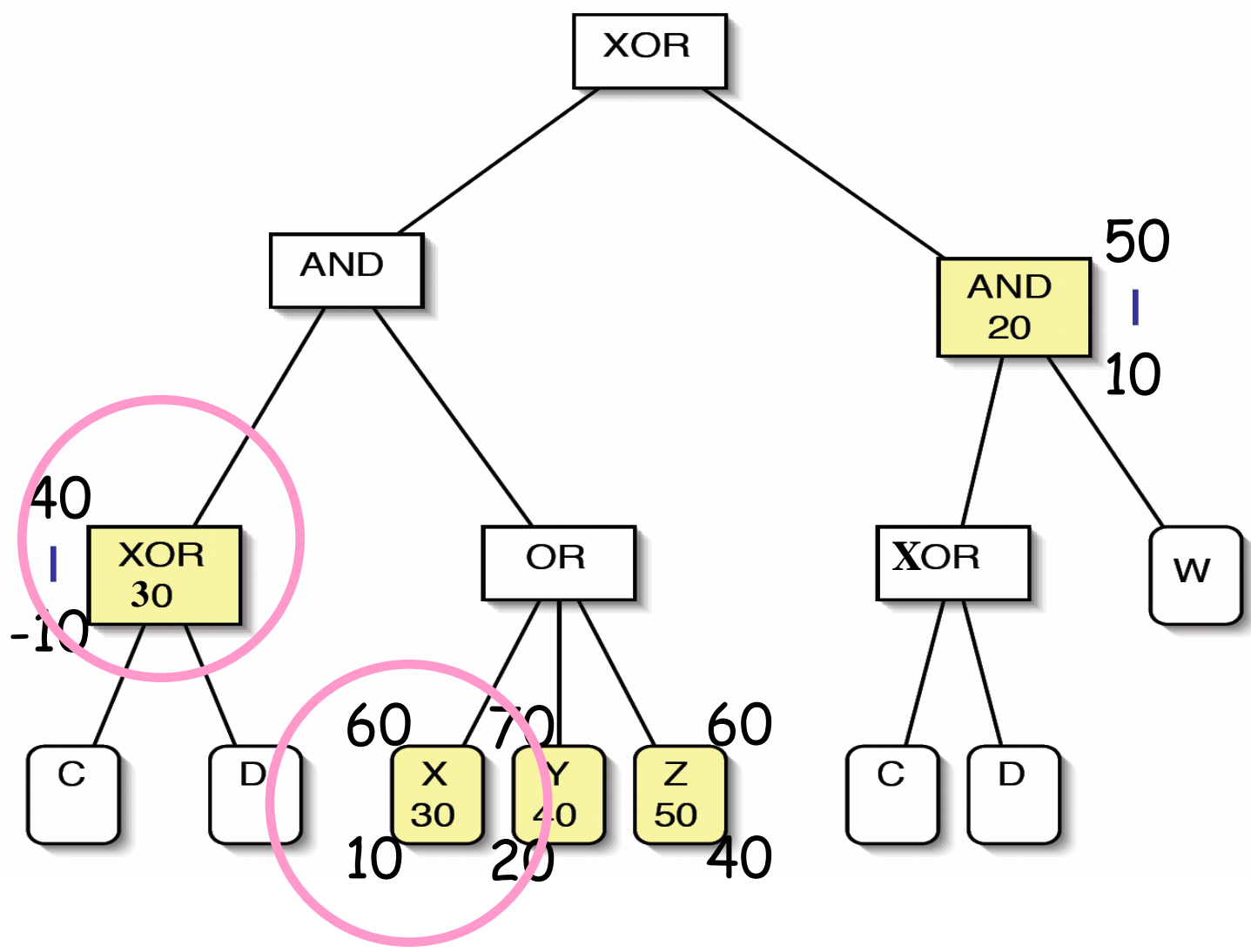
Prices and Activity Rules

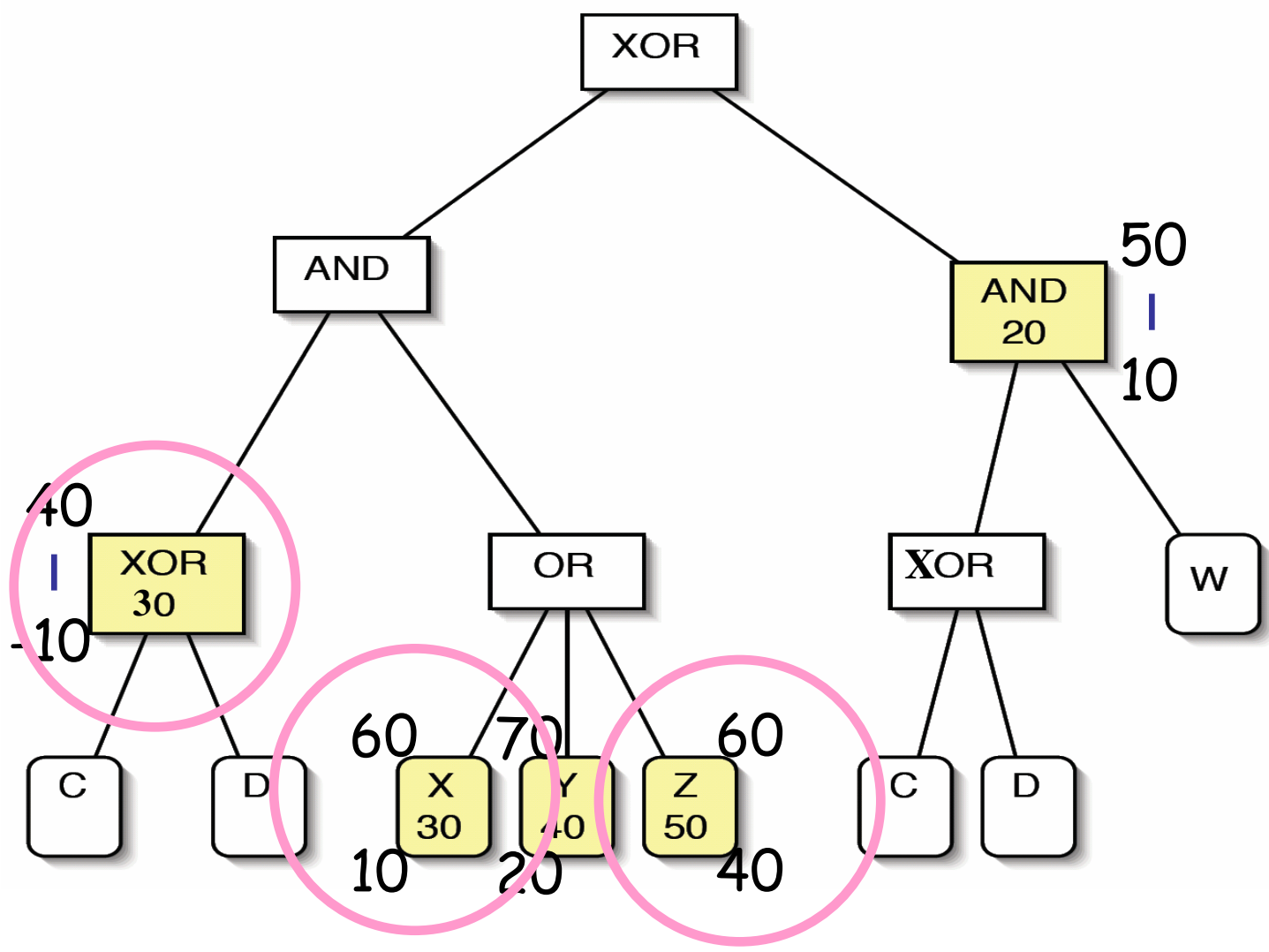
- **Phase 1: optimistic outcome**
 - drive price feedback in early rounds
 - use winner u.b's and loser l.b's to set prices
- **Phase 2: pessimistic outcome**
 - drive price feedback in later rounds
 - use winner l.b's and loser u.b's to set prices
- **Activity rules:**
 - winners must lower u.b's to meet activity
 - losers must increase l.b's to meet activity

Activity Rules work on Tree

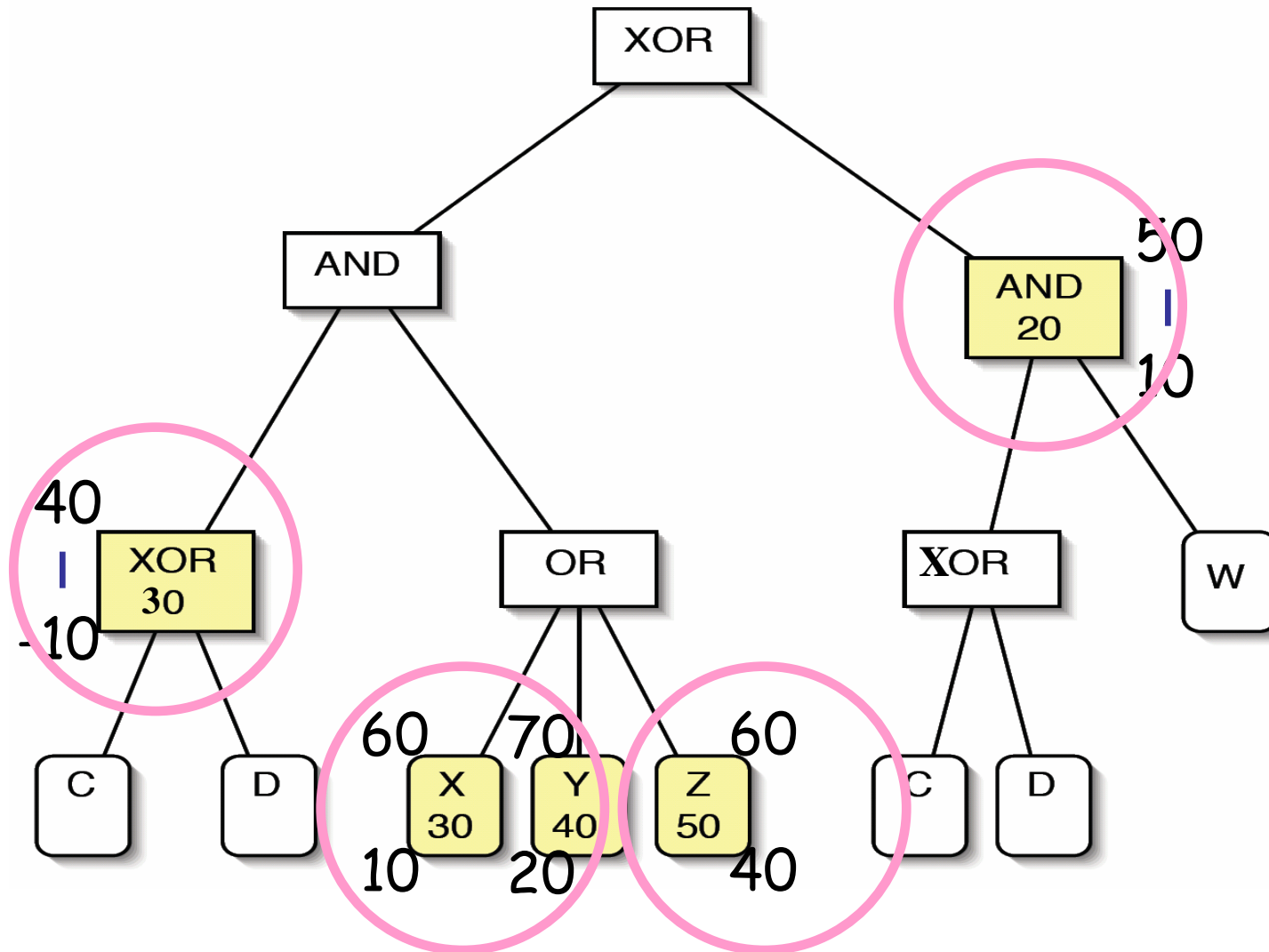








Default Action



Approximate Linear Prices

- Given WD outcome λ^* , compute prices to solve:

$$\min \delta + \Delta(p, p^{t-1})$$

$$\text{s.t. } v_i(\lambda^*_i) - p \leq \lambda^*_i, \quad v_i(\lambda'_i) - p \leq \lambda'_i, \quad \forall \lambda'_i \in M, \forall i$$

where $\lambda_{ij} = 1$ if i buys j , $= -1$ if i sells j

- $\Delta(p, p^{t-1})$ is a price smoothing term (Hoffman et al.)

Solve with *column generation* to avoid enumeration.

Linear prices: Column Generation

- Consider restricted master problem:

$$\min \delta + \Delta(p, p^{t-1}) \quad (\text{MP})$$

$$\text{s.t. } v_i(\lambda^*_i) - p \phi \lambda^*_i, v_i(\lambda') - p \phi \lambda', \quad \forall \lambda' \in C, \forall i$$

where $C \subseteq M$ is a subset of all possible bundles.

- Get a feasible, but perhaps suboptimal solution
- Solve restricted problem:

$$\max [v_{\beta i} - p \phi \lambda_i] - [v^{\alpha}_i - p \phi \lambda^*_i + \delta] \quad (\text{RP})$$

$$\text{s.t. (constraints describing } i\text{'s valuation tree)}$$

- If $(\text{RP} > 0)$ then add new bundle to (MP), and resolve.

Dynamic Feedback to Participants

- Current provisional allocation and payments
- Prices to guide bid refinement
- Can also provide "smart quoting"
 - how should I improve my bid to be a winner?
- Participants don't see:
 - other bids
 - allocation of other participants
 - ...

Final Outcome

- Move to last and final round
- Give participants last chance to refine valuations
- Clear exchange to maximize reported value
- Allocate surplus
 - no-one pays more than bid,
 - no-one receives less than ask
 - distribute surplus to mitigate bargaining and improve efficiency

Summary: Key Features

- Compact and expressive bidding language
- Staged proxy design w/ linear price feedback between stages
 - prices to guide value refinement
 - activity rules to drive progress
- Final “proxy round”
 - expressive bids
 - final clearing, final payments

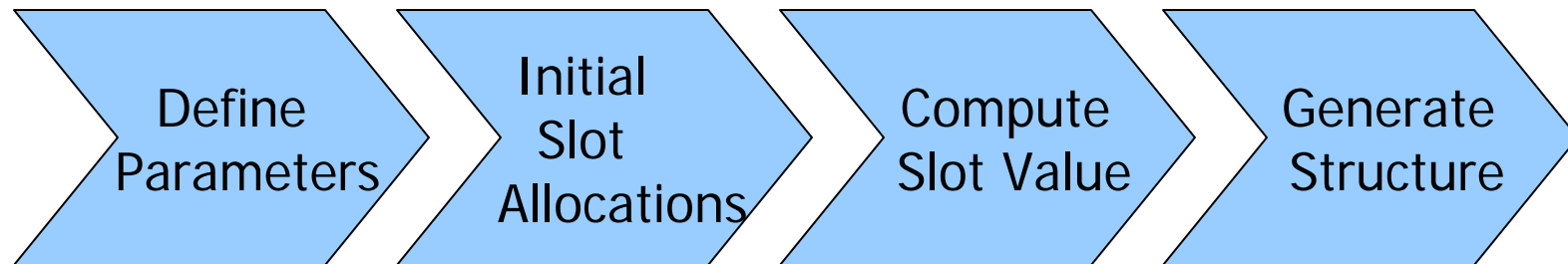
Simulation and Testing

- Model FAA problem domain
 - problem generator
- Simulate bidding strategies
 - truthful & straightforward
 - ...
- Test Exchange
 - economic and computational properties

FAA: Domain Modeling

	LDC2004 "Donohue"	LPS2000 "CATS"	CS286r
Value of slots	<ul style="list-style-type: none"> • Proportional to size of aircraft 	<ul style="list-style-type: none"> • Random utility level 	<ul style="list-style-type: none"> • aircraft size • miles flown • unit cost, revenue • airline type • peak/non-peak
Deviation from optimal slot	<ul style="list-style-type: none"> • Current schedule preferred • same value 	<ul style="list-style-type: none"> • Current schedule preferred • value scaled 	<ul style="list-style-type: none"> • Current schedule preferred • value scaled
Expressiveness	OR	Atomic bids	XOR-(AND,OR)-XOR
Source of schedule data	Real data (from ATL)	Randomly generated	Randomly generated (can mimic real schedule)

Stages in Domain Modeling



- airport
- airlines

- schedule
- flight details

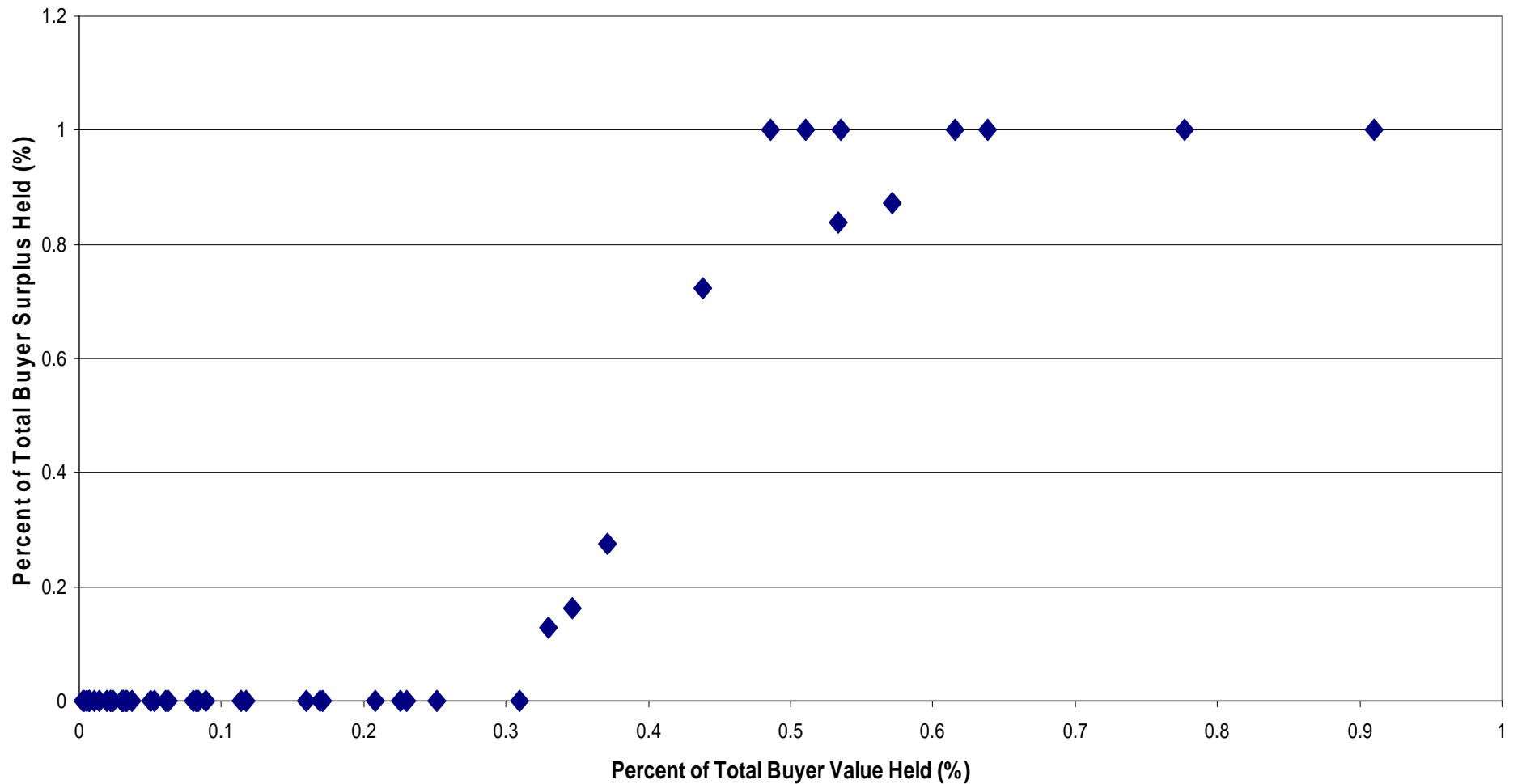
- base value
- adjust for peak times

- adjacent slots
- complementarities
- “must-get” slots

Results in a complete specification of desired slots and valuation for each airline at an airport

Econ Analysis

Surplus versus Value for Buyers in One-Shot, Truthful WD-only Mechanism
(over 10 Runs parameterized to 6,1,1,1)



Continued Work

- Experiments, to study:
 - speed of convergence
 - informativeness of linear prices
 - scalability
 - opportunities for strategic behavior
 - economic impact of exchange
 - policy tools (e.g. assignment of incumbent rights)
- Appeal for help:
 - guide this process!
 - policy goals for design
 - models of participants

Summary

- Combinatorial market technology is real
 - used every day for complex procurement problems
- Expressive languages simplify:
 - allow participants to "say what want"
- Proxied & iterative exchange:
 - expressive bidding language, constraints for sellers
 - linear prices to guide bidding
 - bidding through refinement of value
 - final sealed-bid round

Acknowledgements

- Students in CS 286r, especially
 - S.Lahaie, R.Cavallo, N.Elprin, A.Juda, A.Kirsch, A.Kulesza, B.Lubin, L.Michael, E.Ou, J-F.Raymond, J.Shneidman, B.Szekely, H.Sultan, A.Sumiyama, K.Venkatram.

Threshold Rule

