CS4710: Artificial Intelligence
Multi-Agent Systems

Many systems require multiple intelligent agents that work together. Let's study a couple brief techniques involving these systems.

Topics

- What is a Multi-Agent System (MAS)?
- Examples of MAS
- Multi-agent CSP
- Contract-Net Protocol
- Negotiation

What is MAS?

Multiple-Agents

A **multi-agent system**, in general, is any system that involves multiple agents interacting autonomously.

Some properties:

The agents act **autonomously**, each with own info about the world and other agents

Outcome depends on the actions of all agents

Each agent can have its own utility

Multiple-Agents: Utility

Most interesting MAS problems force systems to balance the following:

Global Utility: Total utility shared by the entire system. Usually involves agents working together.

Local Utility: The utility of each individual agent.

** Sometimes you will have one or both*

Cooperative vs. Competitive MAS

Likewise, we have two extreme types of MAS:

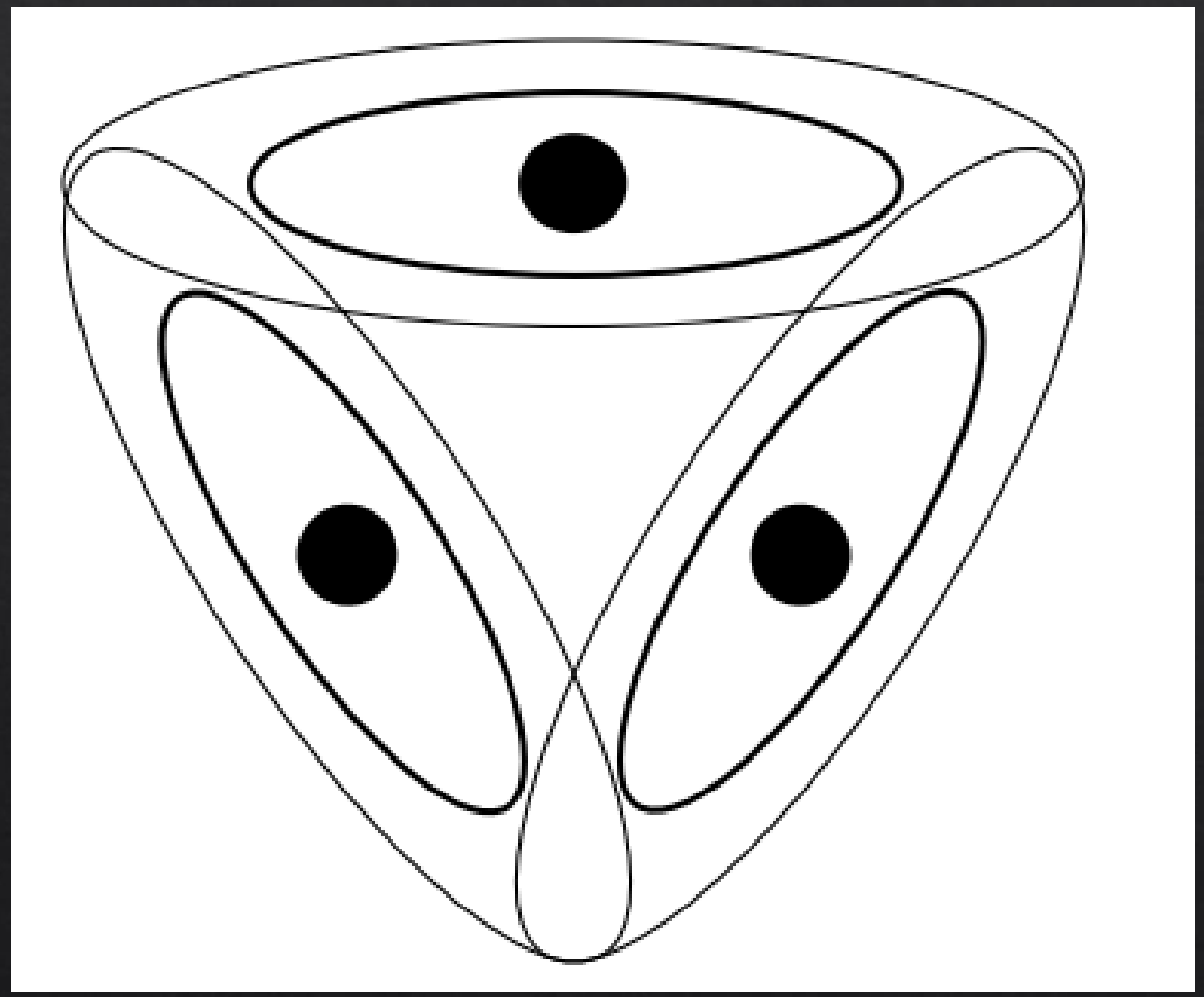
Fully Cooperative: Every agent shares the same utility function (this is what a global utility function is).

Fully Competitive: When one agent can only win when the other agent loses.

**Most practical systems fall somewhere in between these extremes*

Distributed CSP Problems

Constraint-Satisfaction Problems



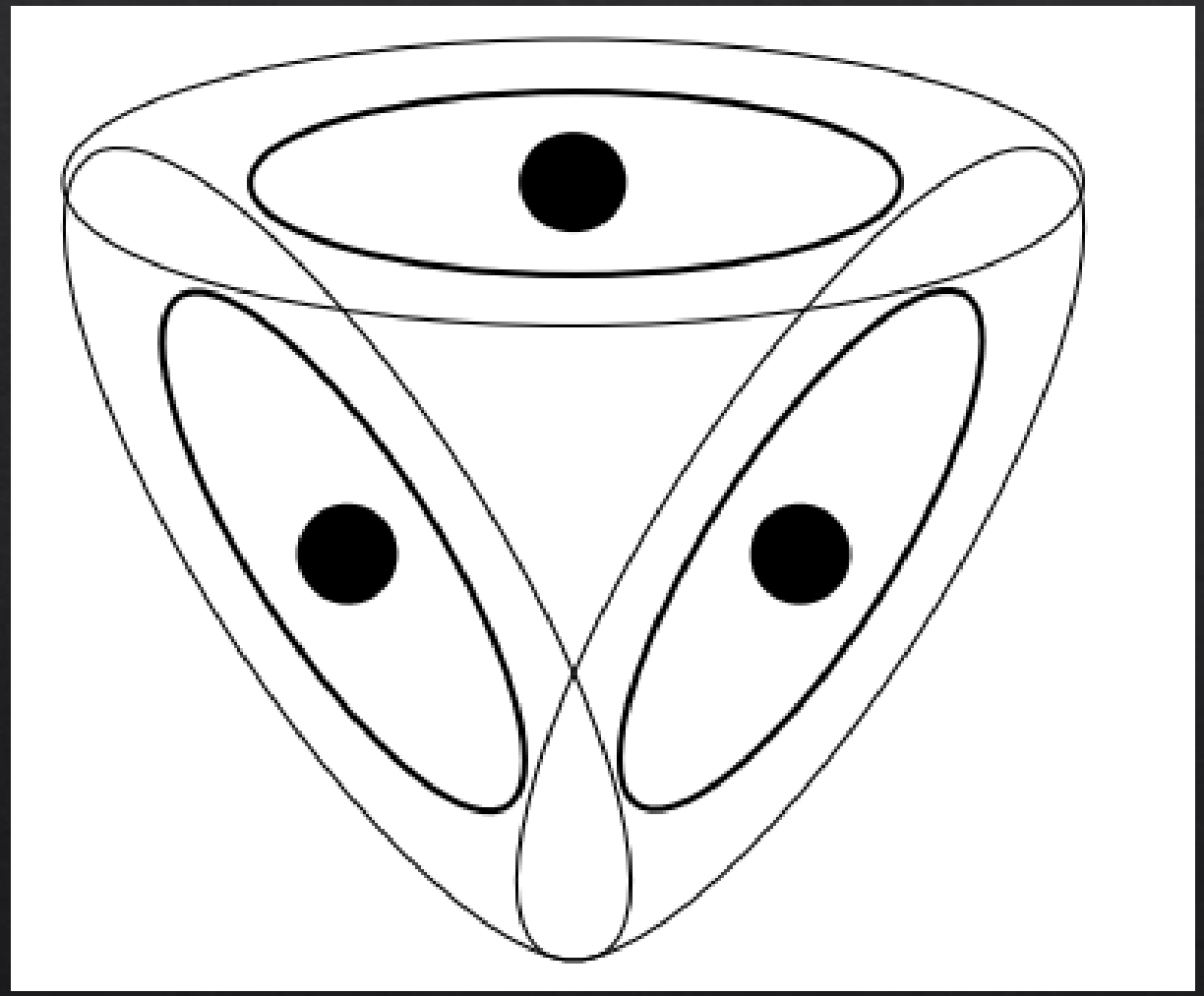
CSP problems contain:

1. A set of variables
2. A domain of values each variable can take
3. A set of constraints on which values variables can take simultaneously

Examples:

1. Graph coloring
2. TSP
3. Knapsack problem

Constraint-Satisfaction Problems



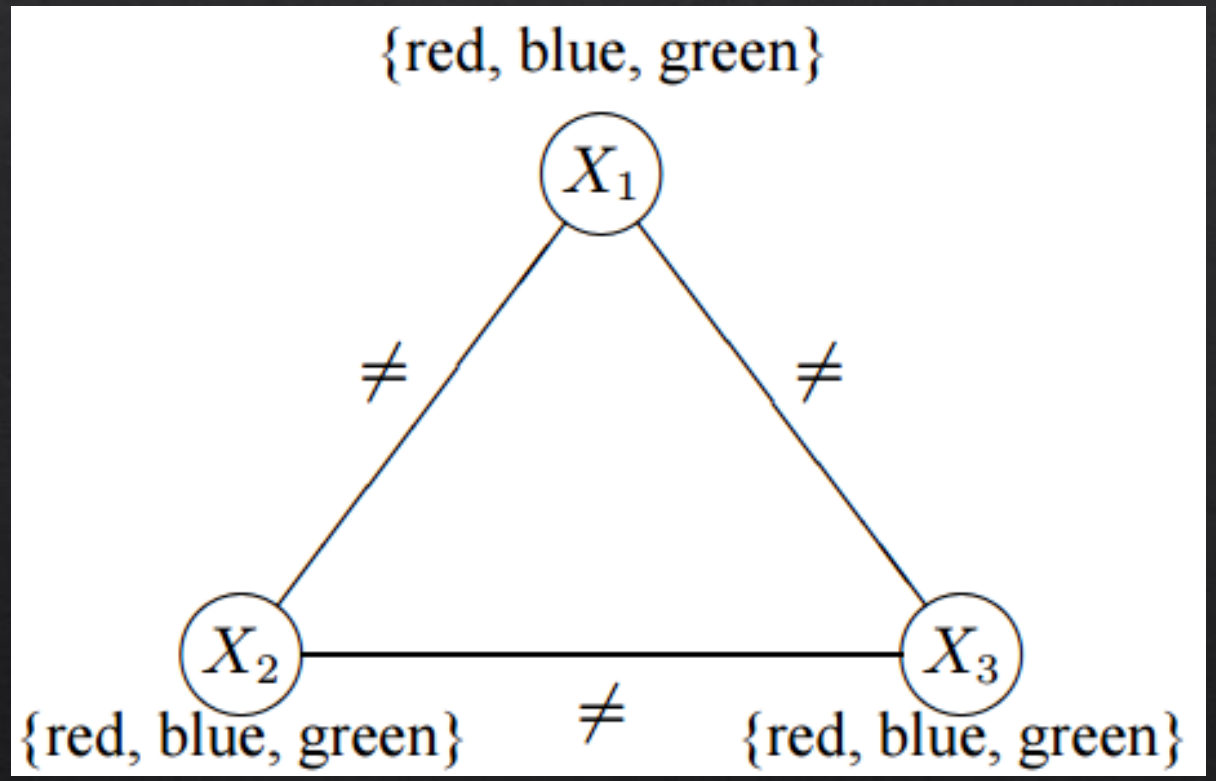
Example:

Consider three sensors (see image) can work on one of three frequencies. All work fine, but no two agents can work on the same frequency. How to select a valid assignment?

Things to remember:

1. The decision cannot be made centrally
2. Agents can communicate with one another
3. Communication is perfect
4. Messages arrive in order

Constraint-Satisfaction Problems

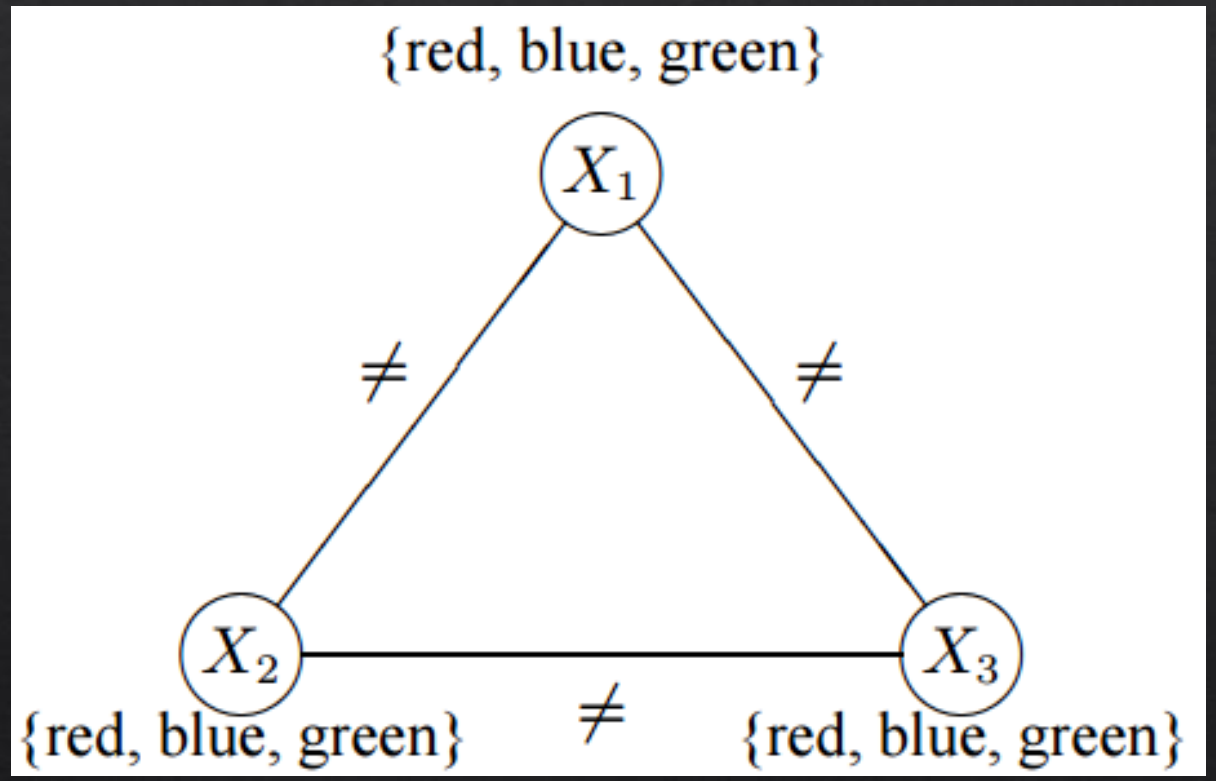


Example:

Consider three sensors (see image) can work on one of three frequencies. All work fine, but no two agents can work on the same frequency. How to select a valid assignment?

Really, this is equivalent to a graph coloring problem, so let's solve this problem instead.

A Filtering Algorithm

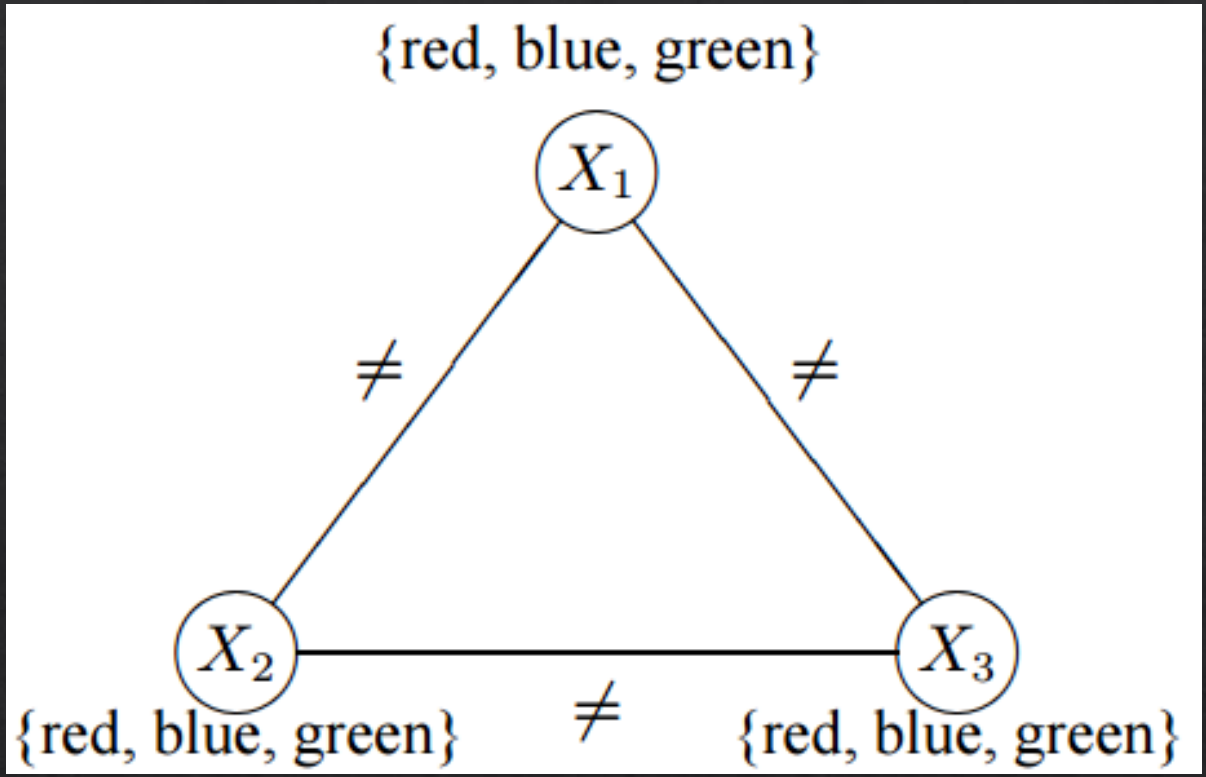


A simple (kind of dumb) algorithm to start with.

Idea:

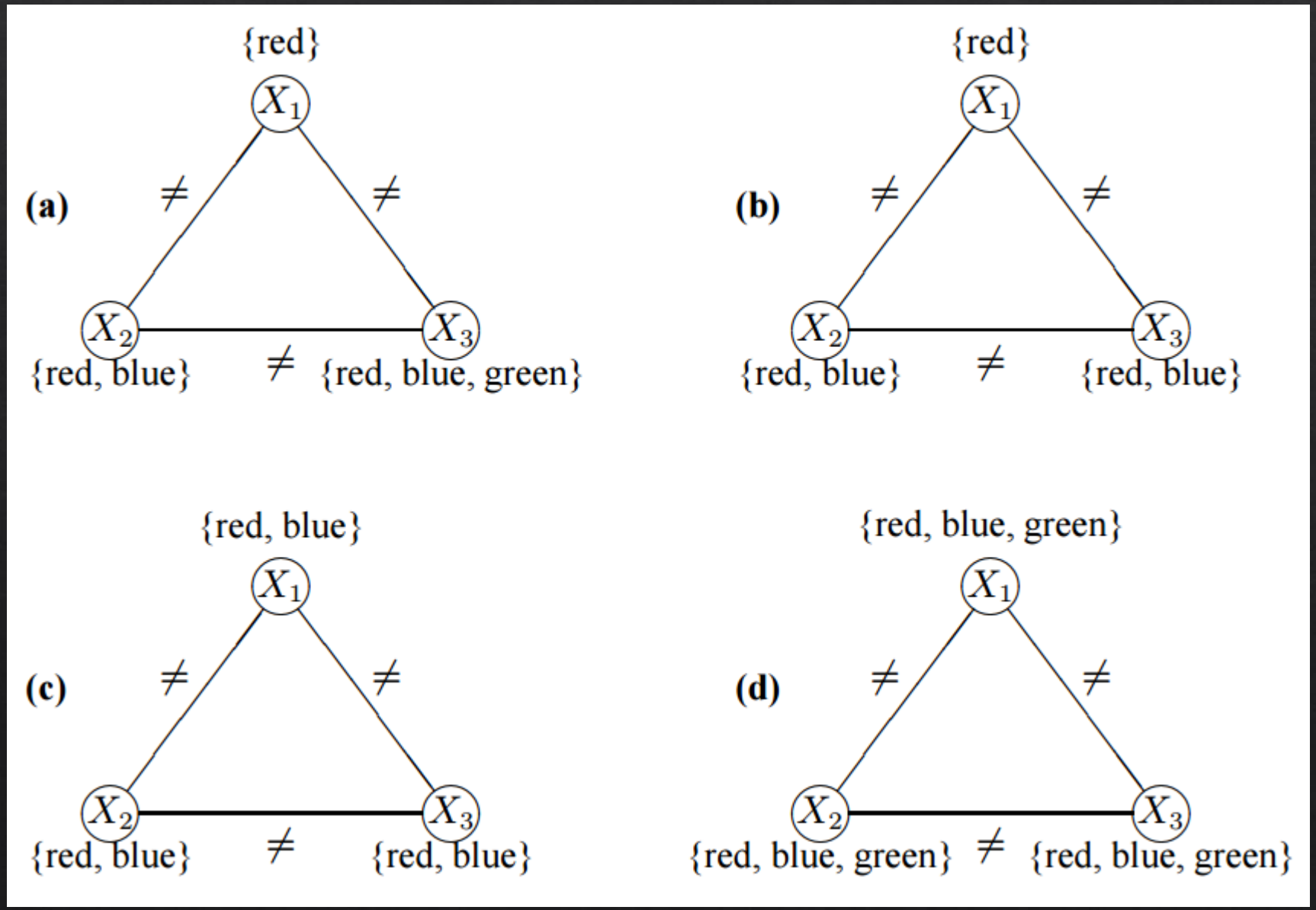
- have agents communicate their possible values to nearby agents.
- Have each agent attempt to rule out values from its own list when possible.
- If an agents list changes, update nearby agents of the change.

A Filtering Algorithm



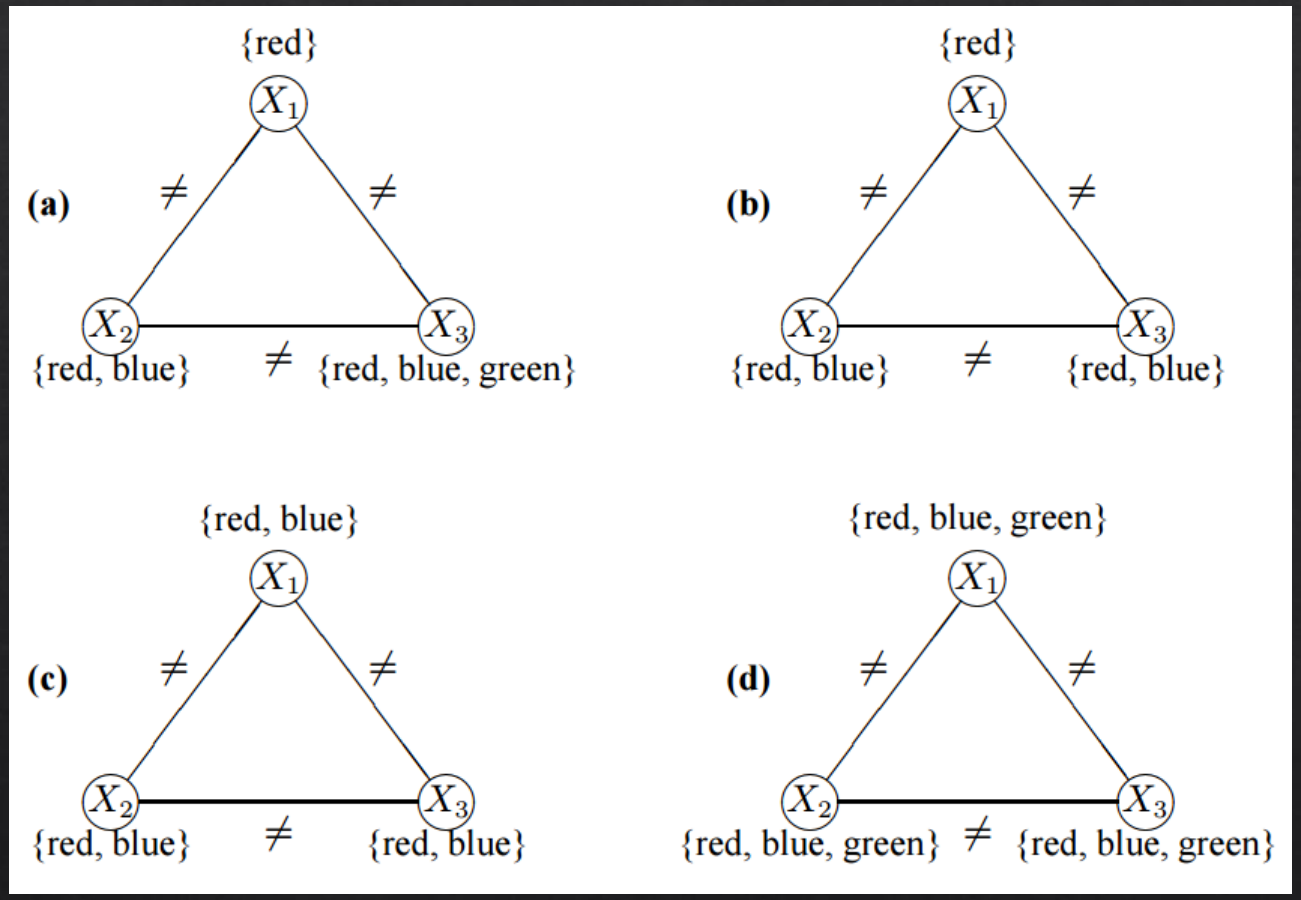
procedure Revise(x_i, x_j)
forall $v_i \in D_i$ **do**
  if *there is no value* $v_j \in D_j$ *such that* v_i *is consistent with* v_j **then**
    delete v_i from D_i

A Filtering Algorithm



What will happen on these?

A Filtering Algorithm



What will happen on these?

a) Works fine, solution found

b) Works, discovers no solution possible

c) Halts, doesn't know if there is a solution even though there is not

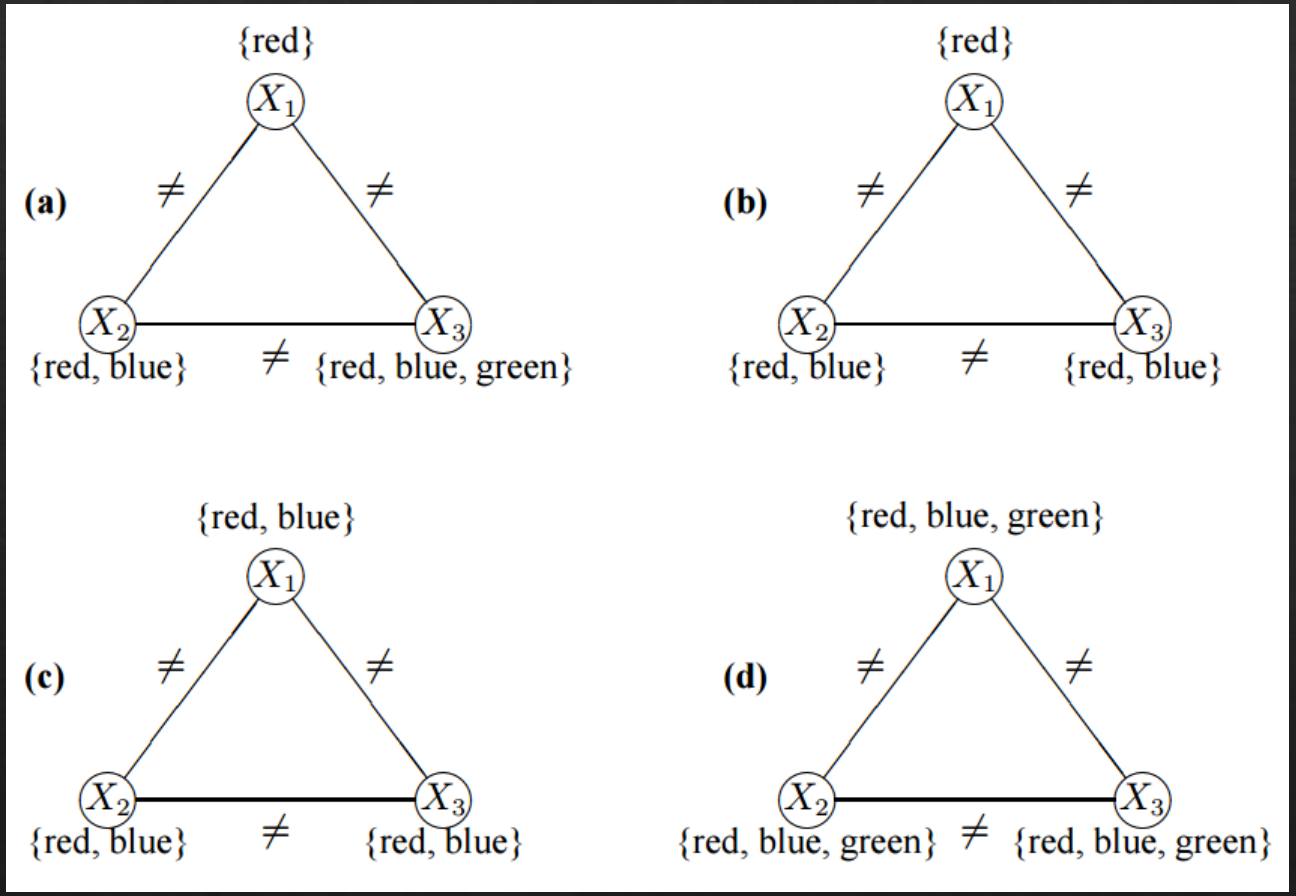
d) Halts, doesn't know if there is a solution even though there is

Algorithm is:

Sound (returns yes/no correctly)

Not Complete (doesn't always find an answer)

A Filtering Algorithm



This algorithm applies logic rules of the following form:

$$\frac{A_1 \quad \neg(A_1 \wedge A_2 \wedge \cdots \wedge A_n)}{\neg(A_2 \wedge \cdots \wedge A_n)}$$

Example:

$$\frac{x_1 = red \quad \neg(x_1 = red \wedge x_2 = red)}{\neg(x_2 = red)}$$

Hyper-resolution

$$\frac{\begin{array}{l} A_1 \vee A_2 \vee \cdots \vee A_m \\ \neg(A_1 \wedge A_{1,1} \wedge A_{1,2} \wedge \cdots) \\ \neg(A_2 \wedge A_{2,1} \wedge A_{2,2} \wedge \cdots) \\ \vdots \\ \neg(A_m \wedge A_{m,1} \wedge A_{m,2} \wedge \cdots) \end{array}}{\neg(A_{1,1} \wedge \cdots \wedge A_{2,1} \wedge \cdots \wedge A_{m,1} \wedge \cdots)}$$

Idea:

We can make this algorithm better by strengthening the inference rules that are being applied.

To the left, we have a generalization of the rule from the previous slide.

Can handle the case where another agent can take on multiple values.

Hyper-resolution

$$
\begin{array}{l}
A_1 \vee A_2 \vee \cdots \vee A_m \\
\neg(A_1 \wedge A_{1,1} \wedge A_{1,2} \wedge \cdots) \\
\neg(A_2 \wedge A_{2,1} \wedge A_{2,2} \wedge \cdots) \\
\vdots \\
\neg(A_m \wedge A_{m,1} \wedge A_{m,2} \wedge \cdots) \\
\hline
\neg(A_{1,1} \wedge \cdots \wedge A_{2,1} \wedge \cdots \wedge A_{m,1} \wedge \cdots)
\end{array}
$$

Top row is the values another agent told us it can take.

Next n rows is the invalid combinations (constraints) that multiple agents can take

Last row is the conclusion, a new constraint that must be satisfied for there to be a solution.

Hyper-resolution

$$
\begin{array}{l}
A_1 \vee A_2 \vee \cdots \vee A_m \\
\neg(A_1 \wedge A_{1,1} \wedge A_{1,2} \wedge \cdots) \\
\neg(A_2 \wedge A_{2,1} \wedge A_{2,2} \wedge \cdots) \\
\vdots \\
\neg(A_m \wedge A_{m,1} \wedge A_{m,2} \wedge \cdots) \\
\hline
\neg(A_{1,1} \wedge \cdots \wedge A_{2,1} \wedge \cdots \wedge A_{m,1} \wedge \cdots)
\end{array}
$$

Each Ai,j in the formula represents what we call a NoGood (a silly name, I know)

A NoGood, is a propositional representation of a set of values that cannot be taken on in a valid solution

e.g., !(x1 = red ^ x2 = red)

Usually stored as tuples (can be higher dimension)

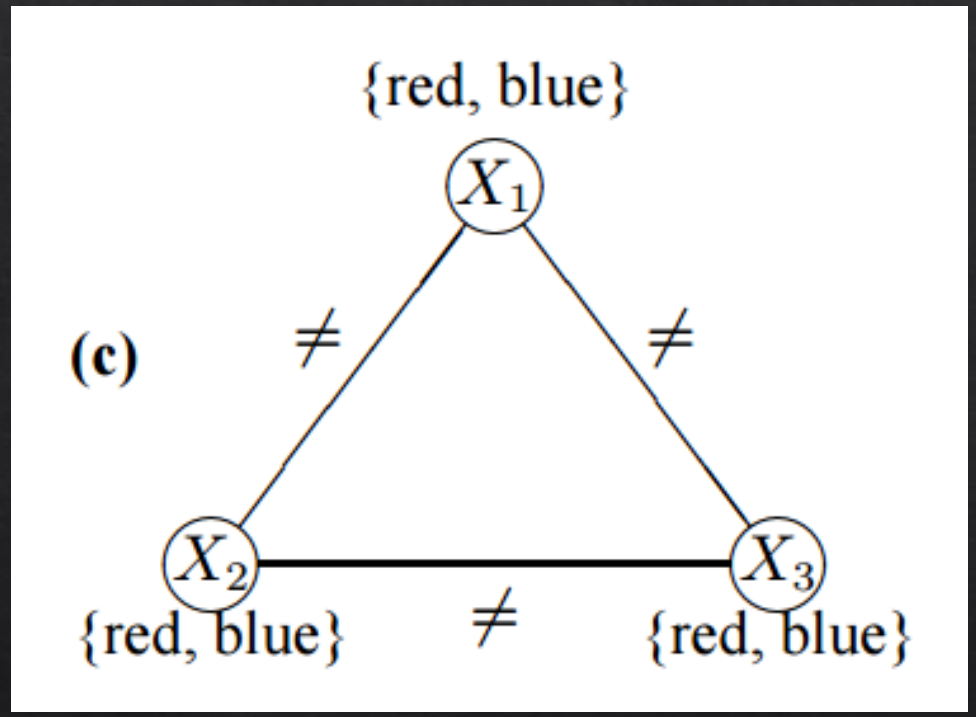
e.g., {x1=red, x2=red}

Hyper-resolution

So, now we have a complete and sound algorithm, let's see an example though.

procedure ReviseHR(NG_i, NG_j^*)
repeat
  $NG_i \leftarrow NG_i \bigcup NG_j^*$
  let NG_i^* denote the set of new Nogoods that i can derive from NG_i and his domain using hyper-resolution
  if NG_i^* *is nonempty* **then**
    $NG_i \leftarrow NG_i \bigcup NG_i^*$
    send the Nogoods NG_i^* to all neighbors of i
    if $\{\} \in NG_i^*$ **then**
      stop
until *there is no change in* i*'s set of Nogoods* NG_i

Hyper-resolution: Example



X1: {X1=red, X2=red}

{X1=red, X3=red}

{X1=blue, X2=blue}

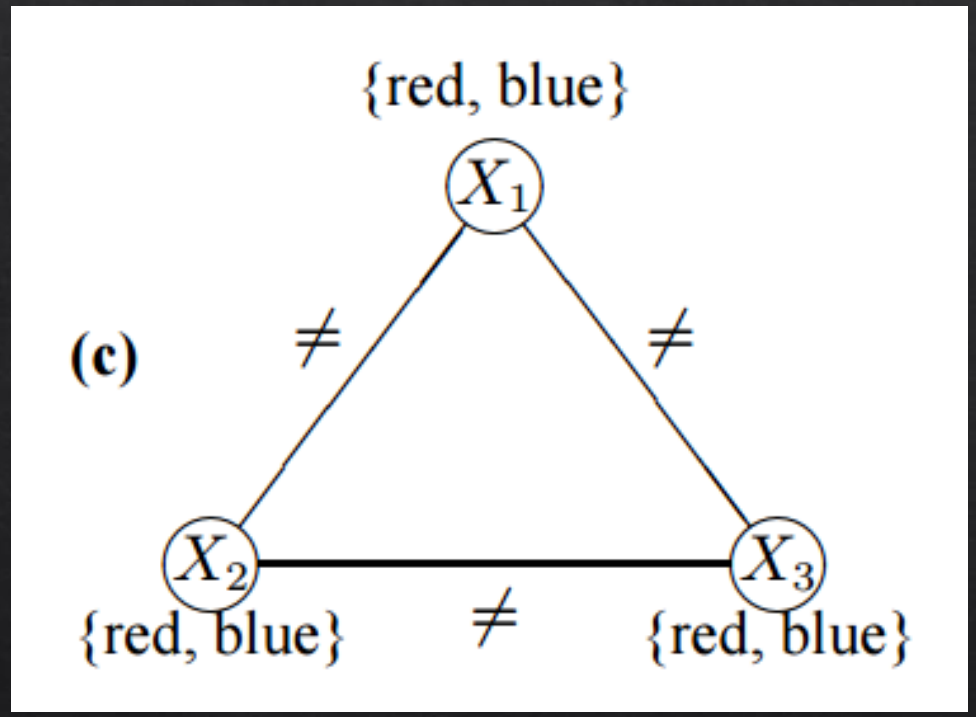
{X1=blue, X3=blue}

{X1=red OR X1=blue}

X1 knows these at the beginning because these are the constraints involving X1

No communication needed to know this.

Hyper-resolution: Example



Using Hyper-resolution, we can reason that:

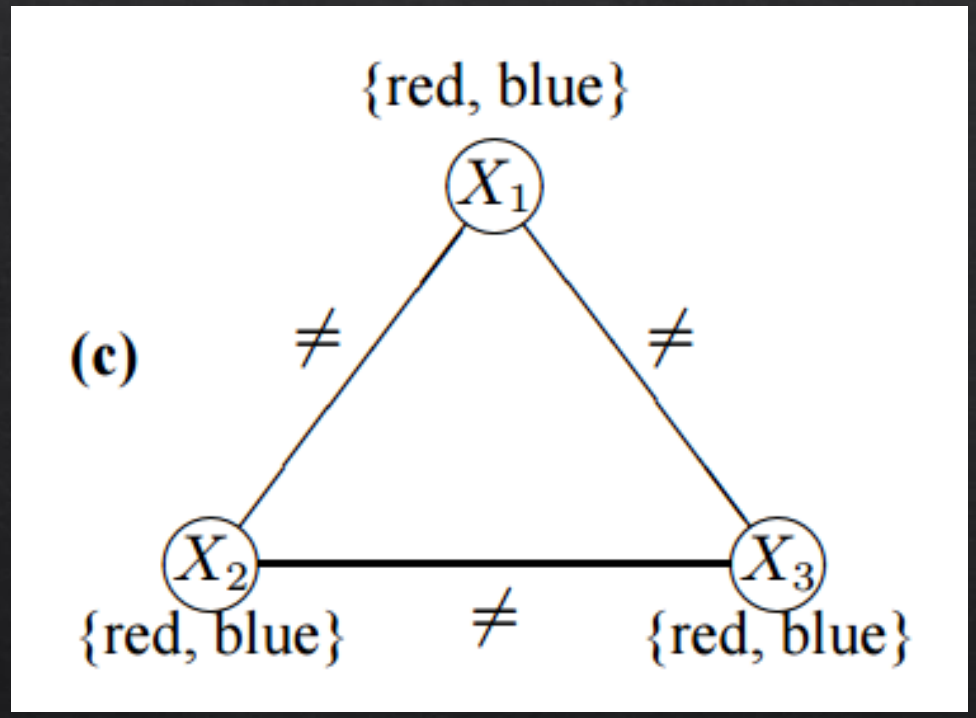
$$x_1 = red \vee x_1 = blue$$
$$\neg(x_1 = red \wedge x_2 = red)$$
$$\neg(x_1 = blue \wedge x_3 = blue)$$

$$\neg(x_2 = red \wedge x_3 = blue)$$

So, X1 has learned that it cannot be the case that X2 is red and X3 is blue (otherwise there will be no solution)

X1 needs to 1) generate more rules of this form and 2) send this new rules to its neighbors

Hyper-resolution: Example



X2 receives these rules (from previous slide) from X1 and can now reason:

$$x_2 = red \vee x_2 = blue$$
$$\neg(x_2 = red \wedge x_3 = blue)$$
$$\neg(x_2 = blue \wedge x_3 = blue)$$

$$\neg(x_3 = blue)$$

Woah! Now X2 can tell X3 to eliminate blue from its set of choices. Cool!

X3 updates its X3=blue OR X3 = red rule to just X3=red

Pros and Cons

FILTERING ALGORITHM

Pros:

Fast, efficient!

Sound (if answer found it is correct)

Cons:

Not complete

HYPER-RESOLUTION

Pros:

Complete and Sound

Easy to Implement

Cons:

Can be intractable

Contract-Net Protocol

CNP

- Basic Idea:
  - Several independent agents exist in a system
  - Want a protocol for communicating and distributing tasks throughout the system
  - Needs to be standardized so agent communication is understood by all parties

Requirements

- Utility:
  - System must have concept of global & local utility
  - Agents must be aware of this utility
- Sub-tasks:
  - System must support independent tasks requirements which can be broken down into sub-tasks
- Help
  - System must support agents asking for and receiving help (e.g., please do this task for me)

Contract Net Protocol (CNP)

- Multiple agents in a network, can communicate with one another.
- Each node has tasks (or even receives new tasks direct from environment)
- Node can either work on the task or contract that task out to another agent
  - Other agent needs to agree to do the task
- Optional:
  - Agent needs to pay the contractor to do the work, etc.

Task Distribution

- Five Steps:
  - Recognition
  - Announcement
  - Bidding
  - Awarding
  - Expediting

Recognition

- An agent realizes it has a task ‘X’ that it needs help on.
- This task can come direct from environment…
- Or…can be a task that was just contracted to this agent.
- ‘X’ could be a sub-task of a much larger task
  - Need to cook thanksgiving dinner
    - Sub-task: need to make potatoes, but I don’t know how
    - Only contract out this one part

Announcement

- Agent sends out a call for bids.
- Let's other agents know what task it needs accomplishing
  - Along with other parameters (payments, utility that will be gained, urgency, etc.)
- Usually has a time-limit
  - Task must be done by 't' time

Bidding

- Other agents that are interested in the contract send in a bid to the original agent
- Usually contains pertinent info:
  - When that agent can finish by
  - Perhaps a lower price they are willing to work for
  - How much local utility they will lose in the process

Awarding

- Original agent awards the contract to one bidder. Let's that bidder know.
  - Usually there is some penalty if the contractor ends up not fulfilling the promise.
- Original agent now free to wait for contractor to do the work.

Expediting

- New agent may choose to sub-contract the task to yet another agent. This is fine, the process simply begins again.

Example: Smart-Home

- Suppose the Laundry agent has the task:
  - *Do_Laundry*
- This tasks has several sub-tasks
  - *Collect_Laundry*
  - *Wash_Cycle*
  - *Dry_Cycle*
  - *Alert_Done*
- Laundry machine can’t do the first one, needs help!

Announcement

- Laundry let's other agents know about this task.
- Important Meta-Information?
  - What do you think?

Bidding

- Other agents, capable of performing the task, bid on it.
- What info might they send to the laundry machine agent?
- What else might influence the decision to bid?

Awarding

- Laundry chooses an agent to award the task to.

Expediting

- Robot might choose to sub-contract the task. Why might the robot do this?
- Thoughts?

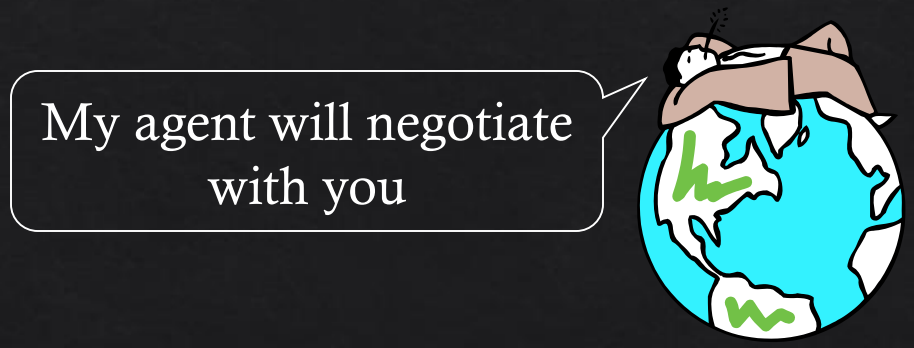
Is This "Intelligence"

- You decide, but…think about my thanksgiving at my house:
  - Agents: Me and my family
  - Local Utility: Everyone has something they prefer doing
    - My dad wants to watch football
    - My mom wants to not cook and chat
    - My mother-in-law wants to eat
  - Global Utility:
    - Everyone wants a big delicious meal, for no fights to occur etc.
  - Contracts:
    - Hey Mom, I need help with the potatoes
    - Mom chooses to give up local utility to increase global utility

Negotiation

Automated Negotiation in Multi-agent Systems (MAS)

- Conflict of interest
- Cooperative behavior in competitive situation
- Applications: distributed problem solving, resource allocation, e-commerce



Classification of Negotiation

Protocols:

- auctions
- bargaining //HW 4 uses this one
- voting, etc.

Negotiated Items:

- single attribute (e.g., price)
- multiple attribute (e.g., price and quality)
- orderings (e.g., top choice, second choice, etc.) //HW 4 uses this

Auctions

Very efficient, but:

- Scheduled in advanced
- Non-negotiable
- Only for price
- Controlled by auctioneer

Alternative: Bargaining!!

Axiomatic vs. Strategic Bargaining

- **Axiomatic bargaining**
  - Bargainers provide information (proposals, facts, and other arguments)
  - Arbitrator sets axioms
  - Arbitrator decides outcomes (guaranteed)
  - E.g. Egalitarian bargaining solution, Nash bargaining solution, etc.

Axiomatic vs. Strategic Bargaining

- **Strategic bargaining**
  - Set a protocol, both bargainers agreed on it
  - Start bargaining (Bargainers offer proposals)
  - Bargainers decide final outcomes (not guaranteed)
  - We will be coding up this one (fyi)
  - E.g. alternating-offer bargaining

Alternating-offer Bargaining

Scenario 1:

Buyer: How much?

Seller: $1000.

Buyer: $500?

Seller: $800.

Buyer: $600?

Seller: $700!

Buyer: OK, $700.

Alternating-offer Bargaining

Scenario 2:

Buyer: How much?

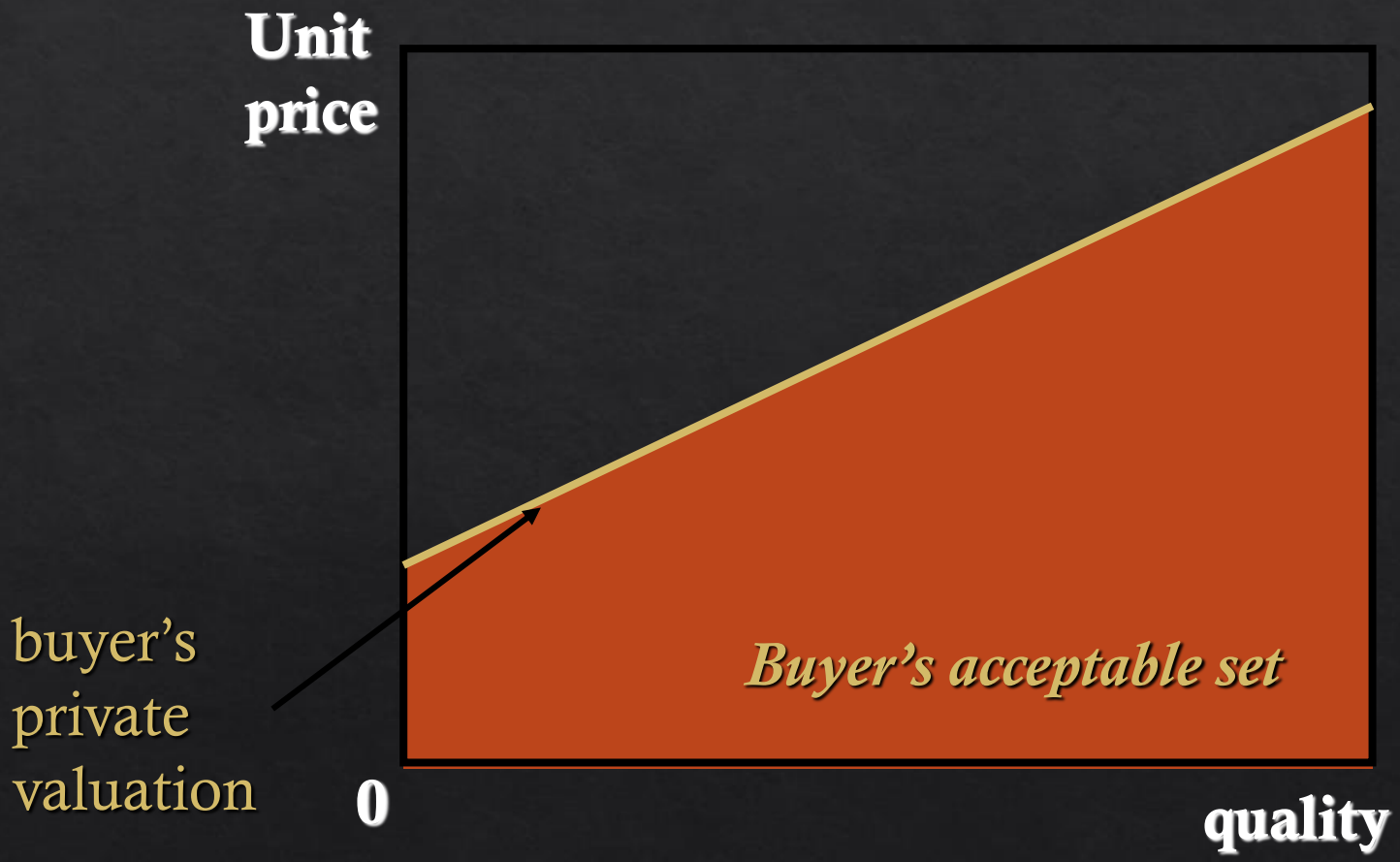
Seller: $1000.

Buyer: $10.

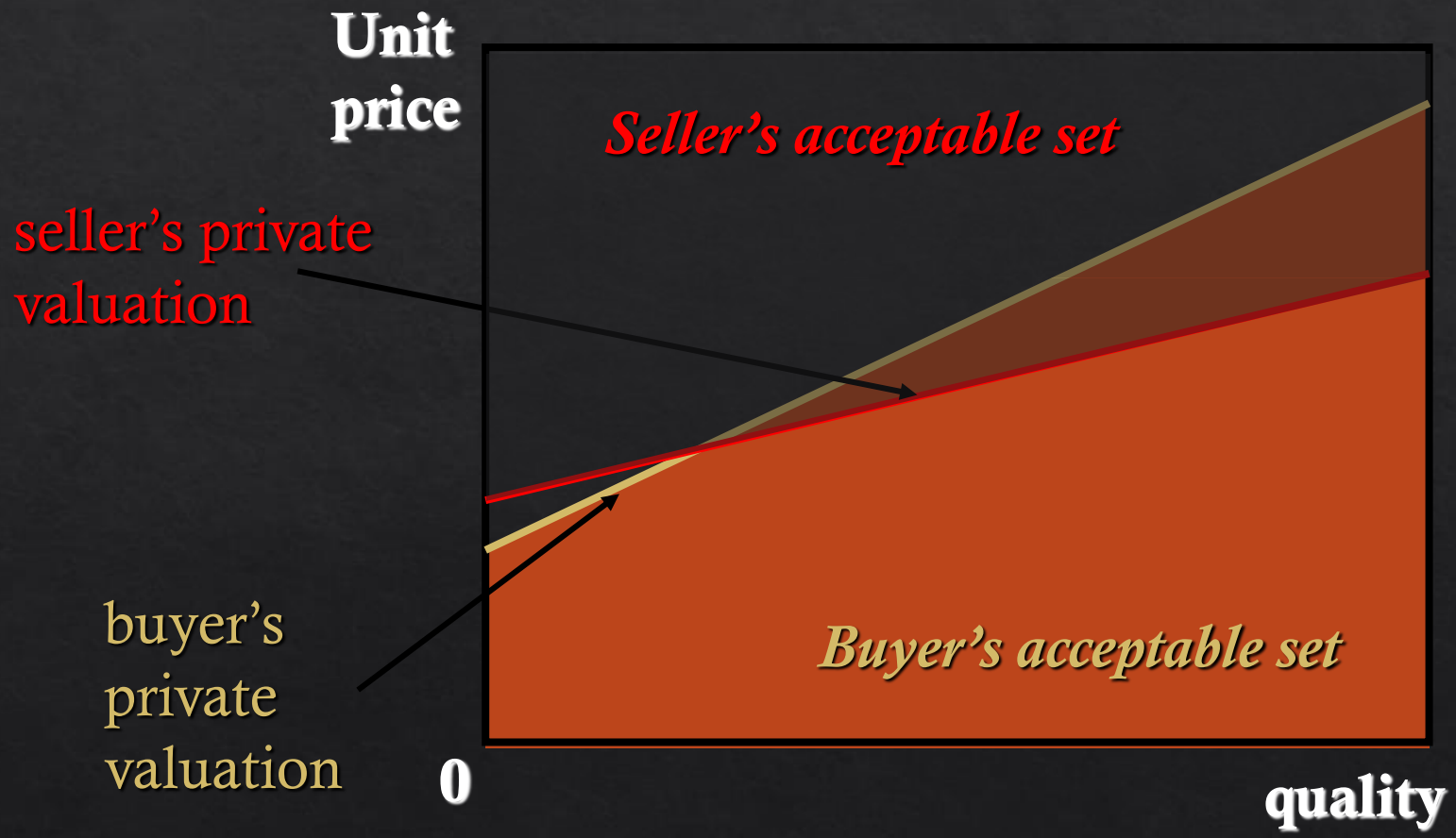
Seller hangs up the phone.

Uh Oh!!!!!

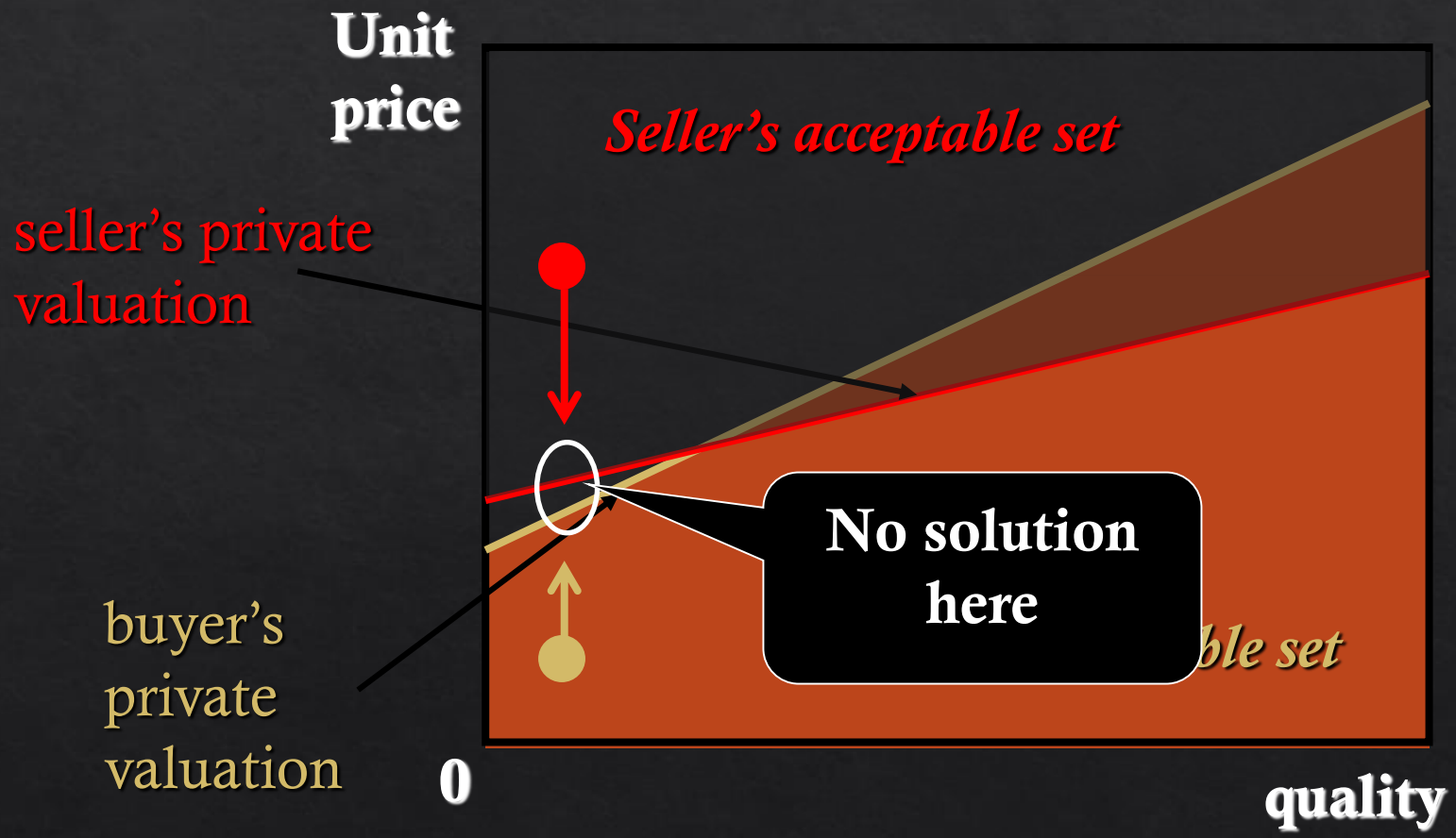
Alternating-offer Bargaining Space: an Example



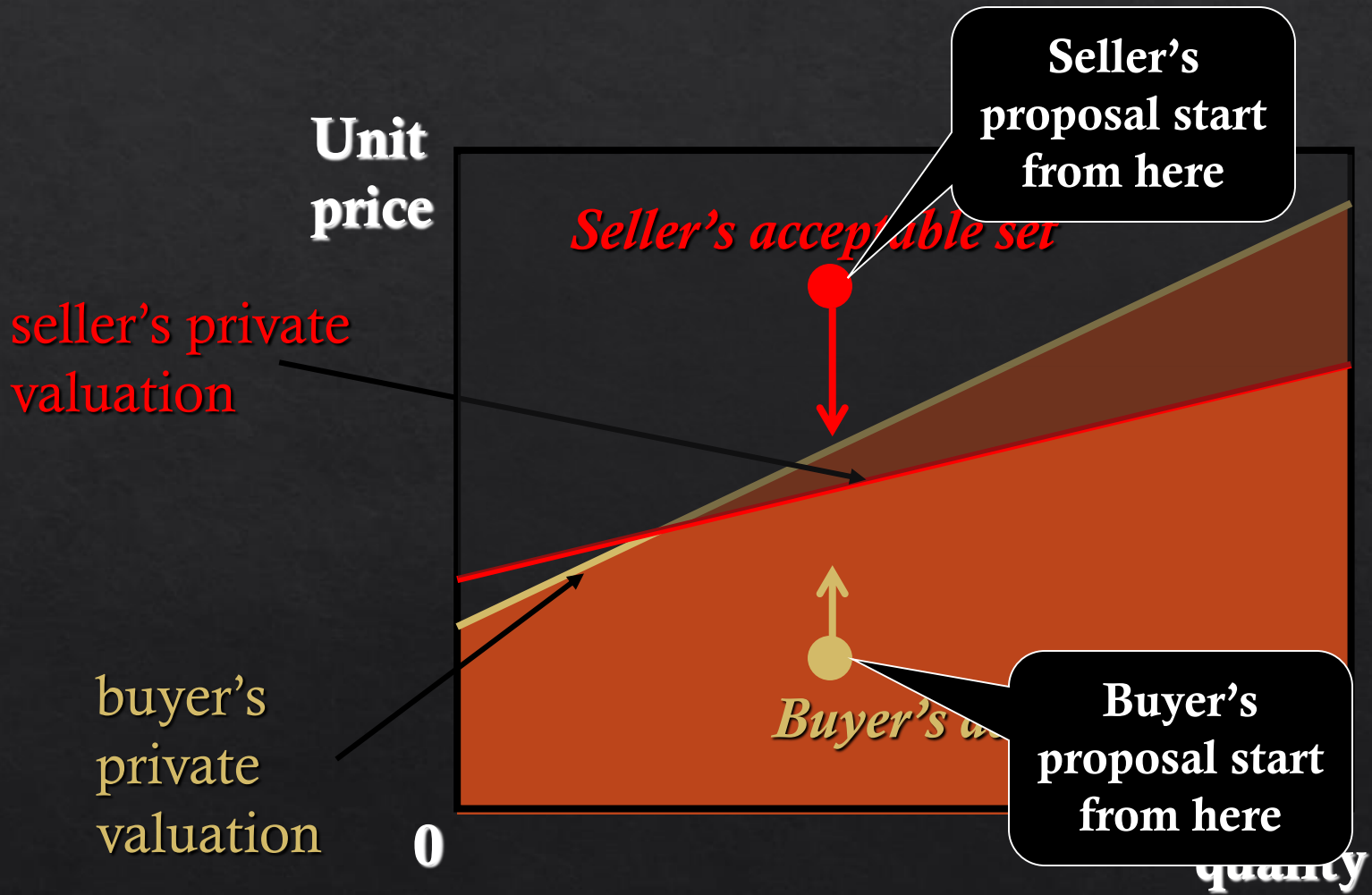
Alternating-offer Bargaining Space: an Example



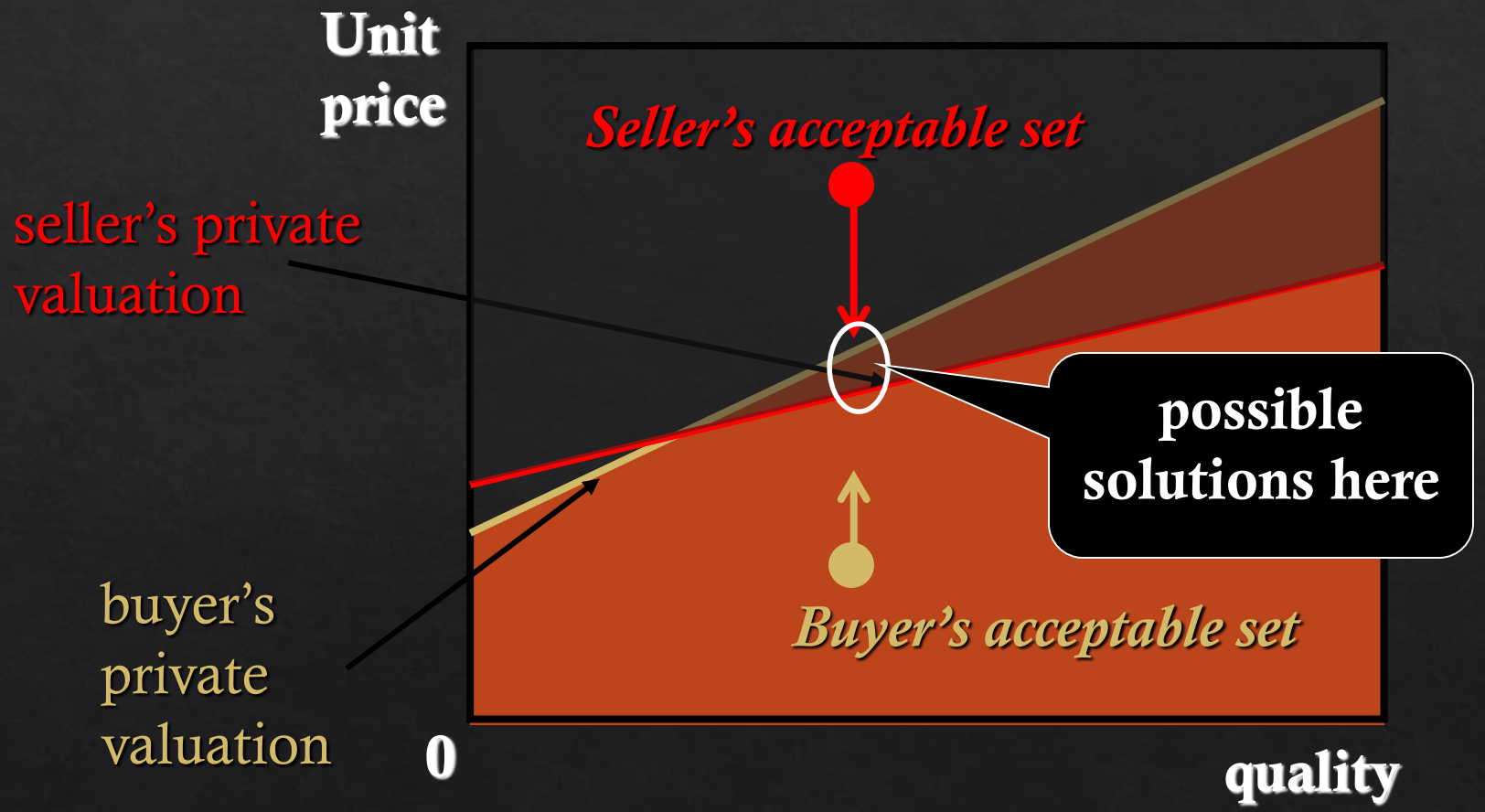
Alternating-offer Bargaining Space: an Example



Alternating-offer Bargaining Space: an Example



Alternating-offer Bargaining Space: an Example



Alternating-offer Bargaining

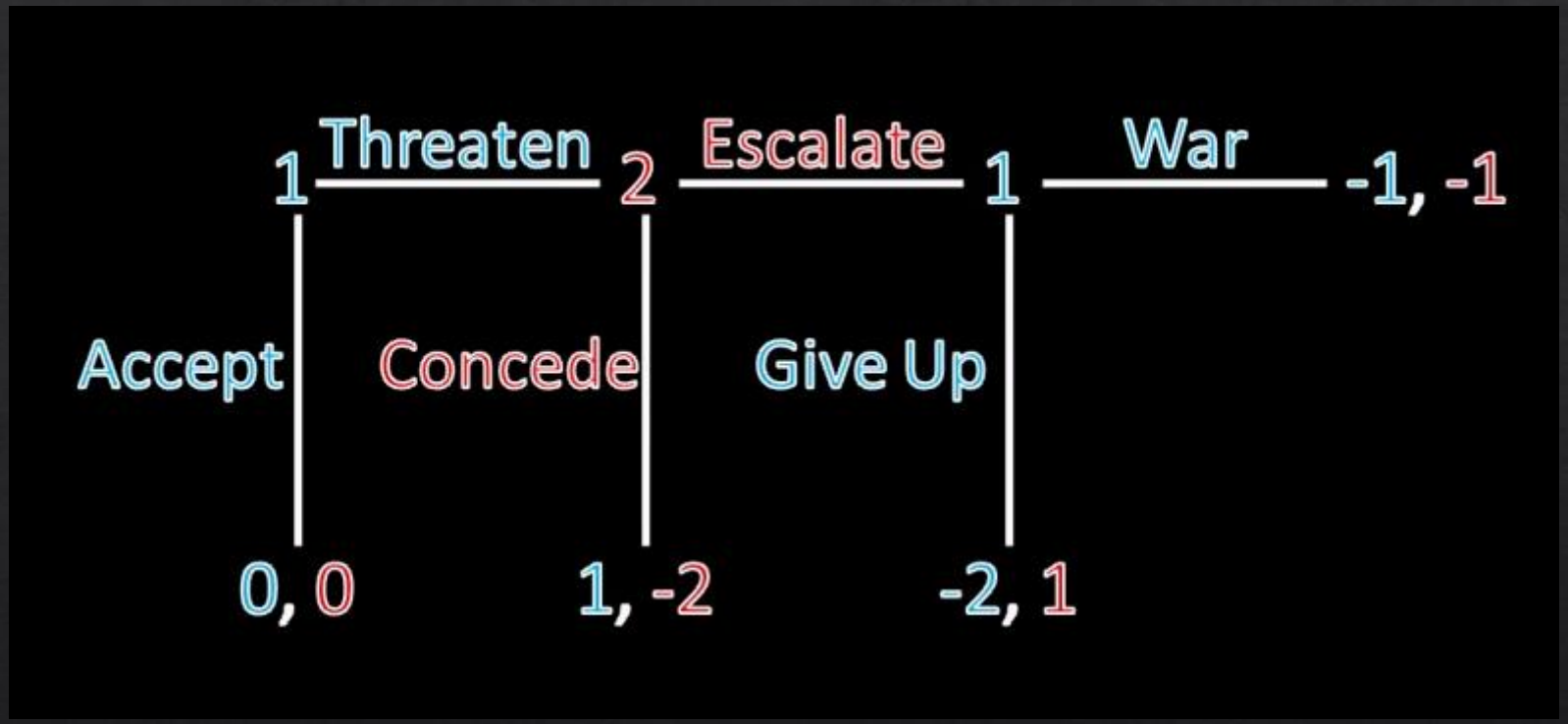
- Bargaining problem: $\langle X, D, \geq_1, \geq_2 \rangle$
  - **X: feasible set**
  - **D: disagreement**
  - **$\geq$: preference order of bargainers 1 and 2.**
- Goal: achieve $x \in X$
- Solving method: backward induction (Game-theoretic approach)
- Assumptions:
  - Perfect rationality
  - Perfect foresight

Backward Induction

- A simple game theory algorithm
- Idea: Compare the benefit of accepting the opponents most recent offer with a guess of what will happen if you reject
- Kind of like minimax in that we take the best path down a decision tree

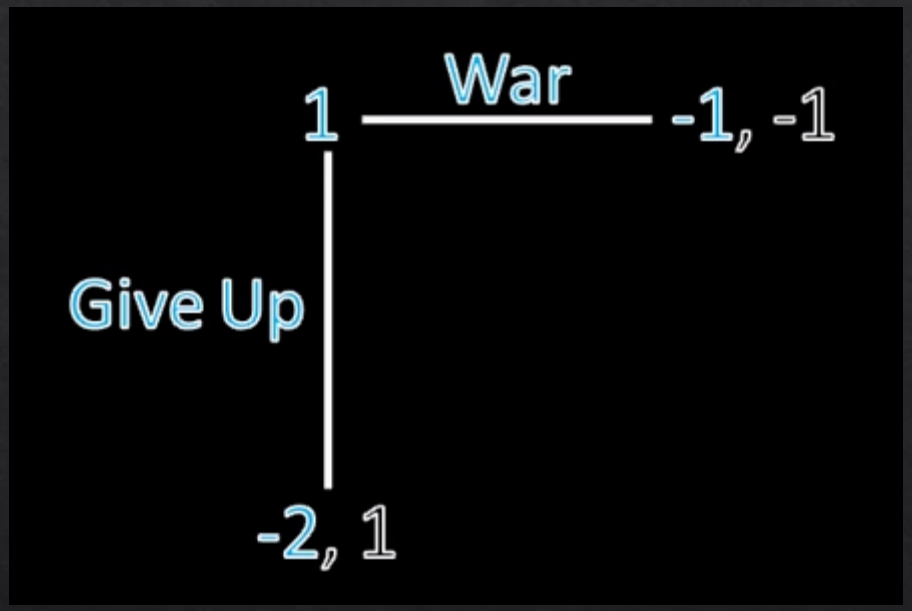
- Goal is to calculate expected reward and compare to accept reward, choose the better option!

Backward Induction: Simple Example



- From: https://www.youtube.com/watch?t=214&v=pyLKkN5HpDY

Backward Induction: Simple Example

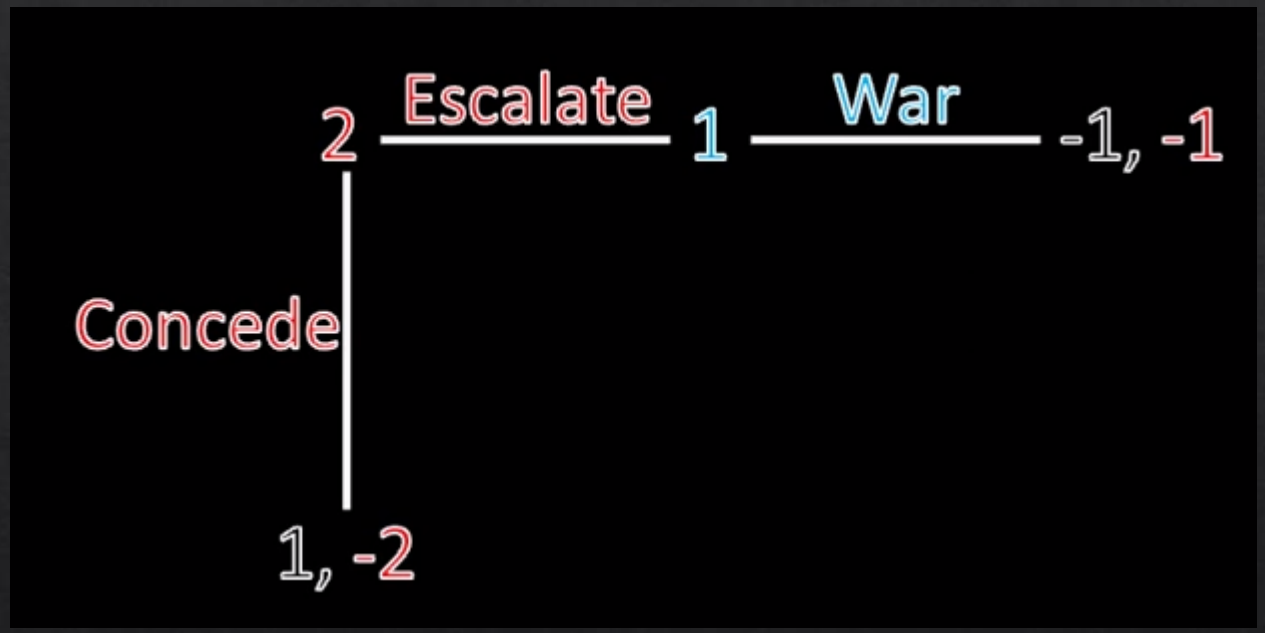


If we get to this state, blue player will always choose war!

Sounds like minimax still ☺

- From: https://www.youtube.com/watch?t=214&v=pyLKkN5HpDY

Backward Induction: Simple Example

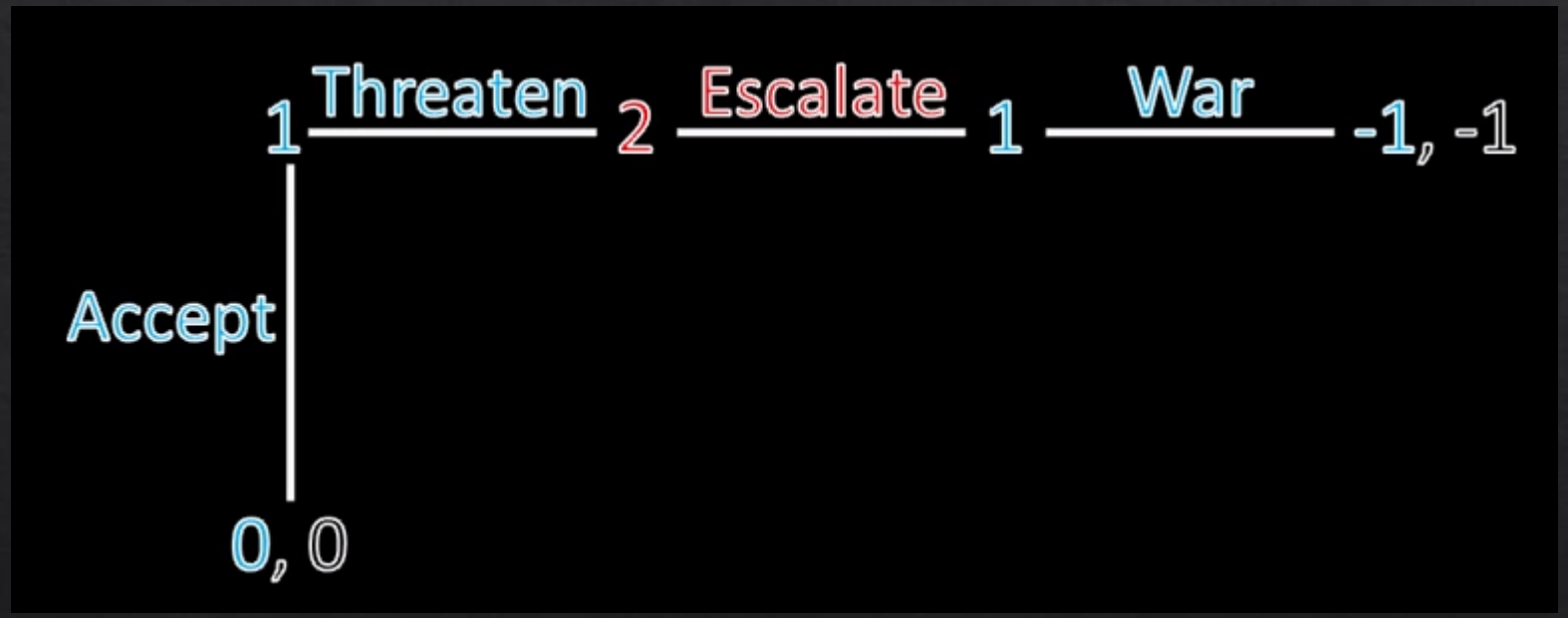


What will red player choose?

Escalate! Why?

- From: https://www.youtube.com/watch?t=214&v=pyLKkN5HpDY

Backward Induction: Simple Example



So, what will happen?

- From: https://www.youtube.com/watch?t=214&v=pyLKkN5HpDY

Backward Induction

- Backward induction assumes perfect information. So, decision tree can be perfectly calculated.

- Best strategy then is to estimate what your opponent will do with probability functions and make the best decision you can.

- Goal is to calculate expected reward and compare to accept reward, choose the better option!

Other Features of Alternating-offer Protocol

- Argumentation (persuade opponent's belief)
- Strategic delay
- Free revision (non-monotonic)
- Range offer (instead of one point offer)

Strategic delay & Argumentation

Strategic delay:

Buyer: How much?

Seller: $1000.

Buyer: …

Seller: $800.

Buyer: hmmm….

Seller: $700!

Buyer: …

Seller: $500!

Buyer: $300.

Seller: OK, $300!

Argumentation:

Buyer: How much?

Seller: $1000.

Buyer: $500?

Seller: My price is lower than others'.

Buyer: $700?

Seller: $1000 is very cheap.

Buyer: $800, OK?

Seller: OK, $800.

Free revision & Range offer

Free revision:

Buyer: How much?

Seller: $1000.

Buyer: $500?

Seller: 800.

Buyer: $600?

…….(Seller got a call)

Seller: $2000.

Buyer: What?

Seller: The market price increases now.

Range offer:

Buyer: How much?

Seller: $1000.

Buyer: I can’t afford more than $500.

Seller: $499.

Buyer: $400, OK?

Seller: OK, $400.