Multiagent Systems

Boi Faltings

Laboratoire d'Intelligence Artificielle
boi.faltings@epfl.ch
http://moodle.epfl.ch/

Fall 2018

Multi-Agent Systems

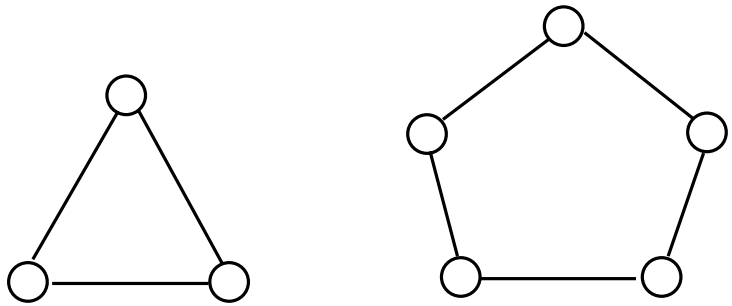
Multi-agent system (MAS) = system consisting of multiple interacting agents

Reasons for multi-agent systems:

- multiple agents required for solving a task
- secure against manipulation
- model real-life scenarios

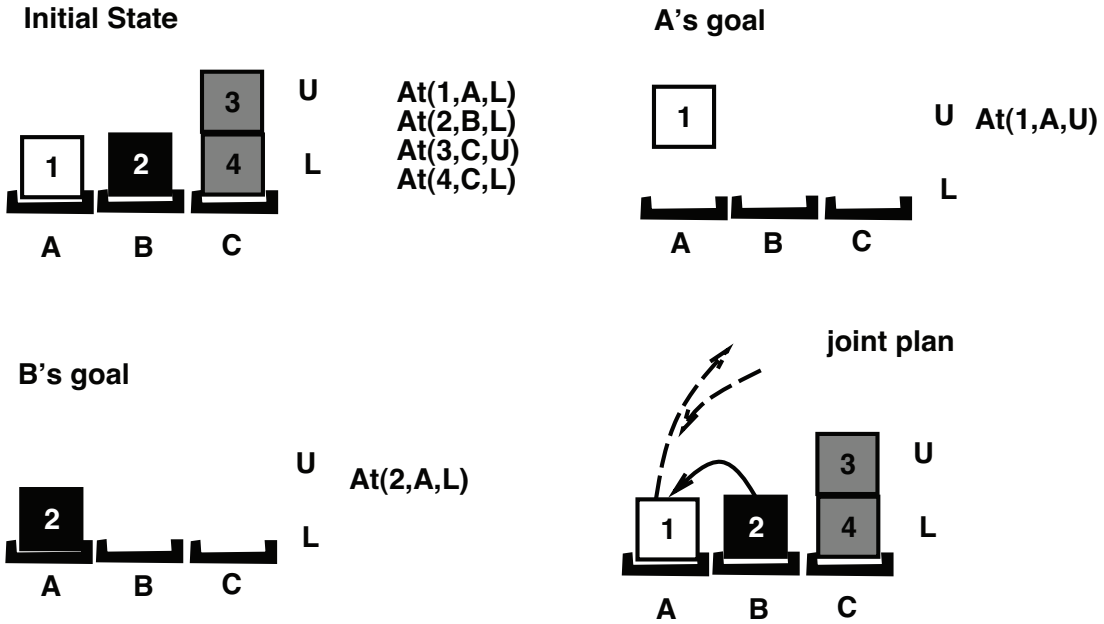
Need for multiple agents

Two robots can distinguish the graphs, but one cannot:



Gain from cooperation

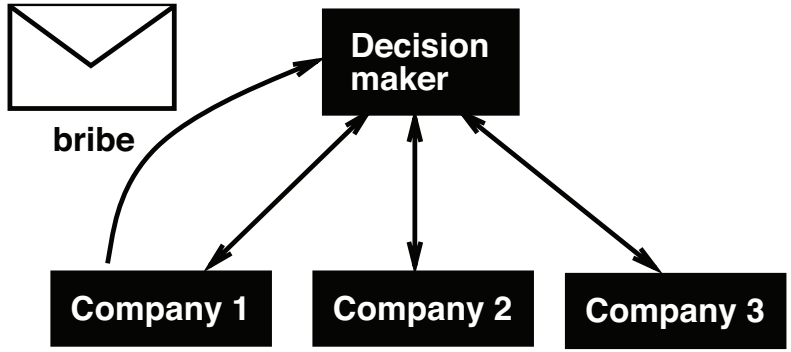
Two robots can achieve goal with less operations than a single one:



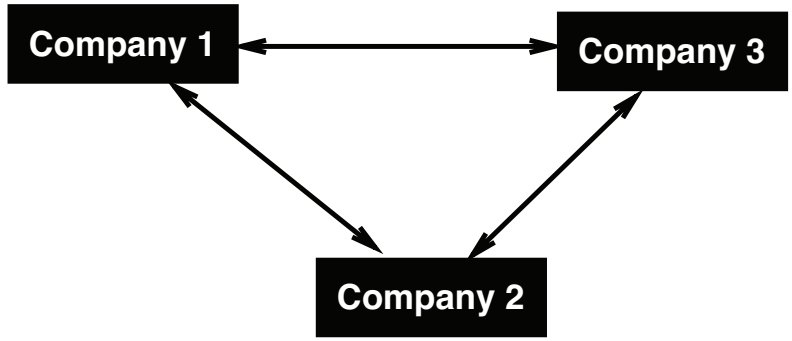
Similar situation: sharing information accesses.

Security against manipulation

Centralized decision mechanisms are vulnerable to manipulation:

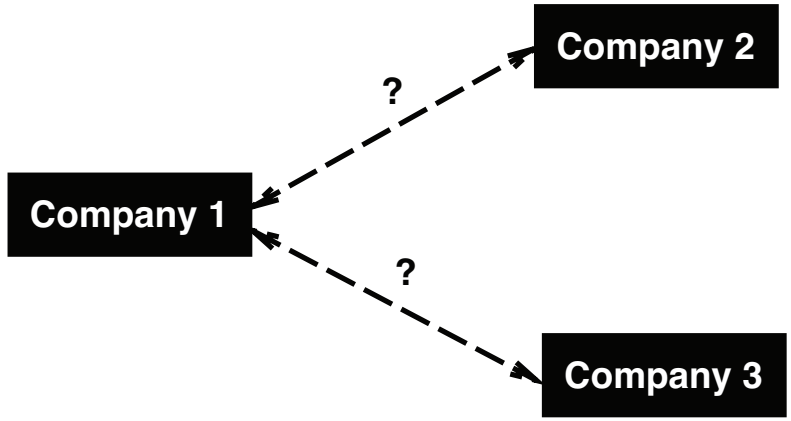


Impossible in distributed mechanisms:



Modeling real-life scenarios

Companies work together in varying configurations:



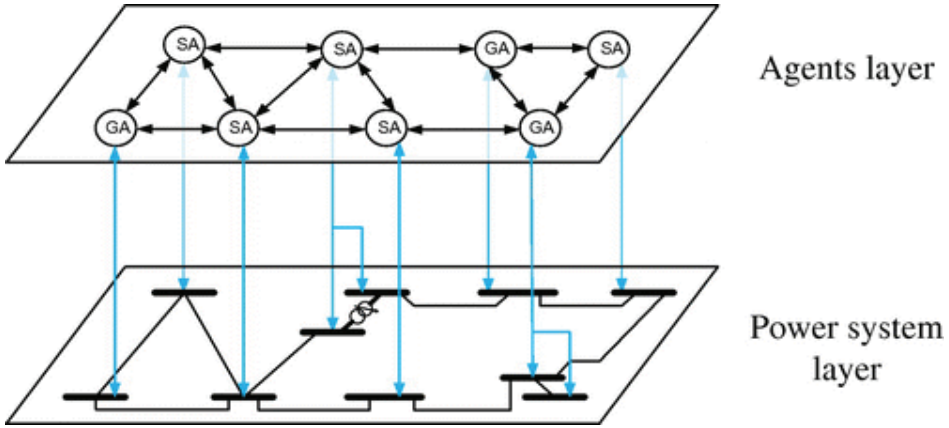
Software architecture should accomodate frequent changes in business relations and processes.

Multi-agent simulation

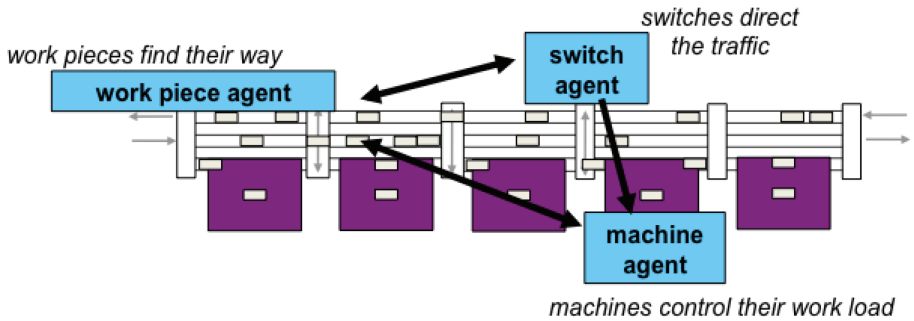
- Many real-life scenarios involve rational agents.
- Examples: economic systems, social sciences.
- Numerical simulation cannot predict their behavior.
- Multi-agent simultation models each agent and the emergent behavior.

Motivating Examples

Smart Grid: agents control generators and substations:



Factory floor: agents control flow of workpieces among machines:



Multi-agent Architectures

- Centralized, shared-memory: all agents run on a common platform and share the same data structures.
- Mediator: agents interact by well-defined messages through a central mediator.
- Distributed: agents interact through message exchange.
- Decentralized: agents exchange no messages, but might observe common signals (e.g. traffic lights, prices, movements).

Today: centralized/mediated architectures

Self-Interest

- Self-interested agents always maximize their own interest.
- To make them follow a joint plan, we need to create *incentives* to motivate them to act in the right way.
- Joint plan + incentives $\Rightarrow$ *mechanism*.

Address joint plan first, mechanism design later in this course.

Cooperative planning and coordination

- Delegation: central planner computes a plan for several agents $\Rightarrow$ similar to single-agent planning.
- Mediated: each agent makes its own partial plans; mediator coordinates them.
- Distributed: each agent makes its own plans and coordinates through message exchange.

Multiple reactive agents

- Multi-agent policy: single action $\Rightarrow$ one action for each agent.
- State is the combination of states for all agents.
- Gold is to optimize the sum of rewards for all agents.
- Potential for using factored model.

Optimizing sum of rewards

2-step procedure:

1. offline: each agent computes an approximation of the value function, taking into account all agent actions, using the approximation method for factored MDPs (find the closest factored value function using linear programming).
2. online: use the value function and the current state to compute the expected reward for each individual agent action, and choose the combination of actions that maximizes this sum.

Approximating the value function

- For each agent, factor value function into basis functions $b_i(U), U \subset X$, such that $V(X) = \sum w_i b_i$ (Koller and Parr, 1999)
- Combine all basis functions from all agents (additive)
- Approximate value recurrence:

$$V_\pi(S) = R(S, \pi(S)) + \gamma \sum_{S' \in \mathcal{S}} T(S, \pi(S), S') V_\pi(S')$$

  using linear program to minimize error.
- Linear program has one constraint for each state and action combination.

Selecting the optimal action

- Agent chooses action $a \in \{a_1, ..., a_k\}$.
- Each basis function of the value function depends on one or several action choices.

$\Rightarrow$ constraint optimization problem: choose combination of actions that maximizes the reward.

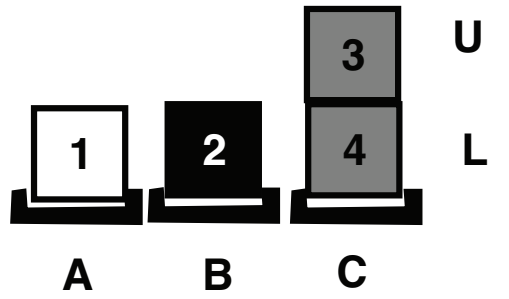
- Solved using centralized or distributed solver.

Multiple deliberative agents

- Assumption: each deliberative agent has its own goals and plans
- Conflicts: plans require the same resource
- Synergies: plans achieve similar goals
- Need to integrate plans into a coherent whole

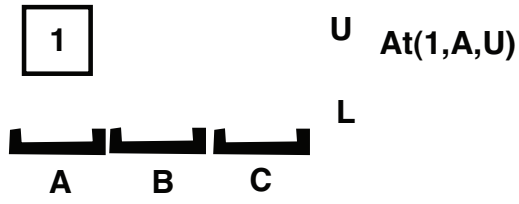
Example

Blocks world initial state:

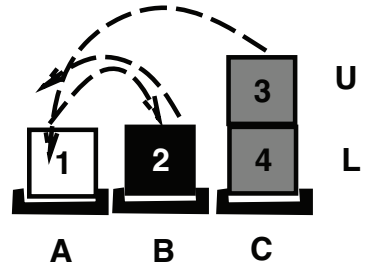


At(1,A,L)
At(2,B,L)
At(3,C,U)
At(4,C,L)

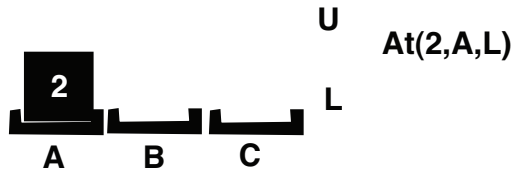
Agent A's goal and plan:



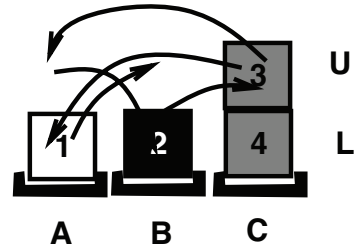
At(1,A,U)



Agent B's goal and plan:



At(2,A,L)



Blackboard systems

Idea: similar to board in a meeting room.
Contents of the blackboard:

- current goals/reward structure
- current state
- each agent's plans

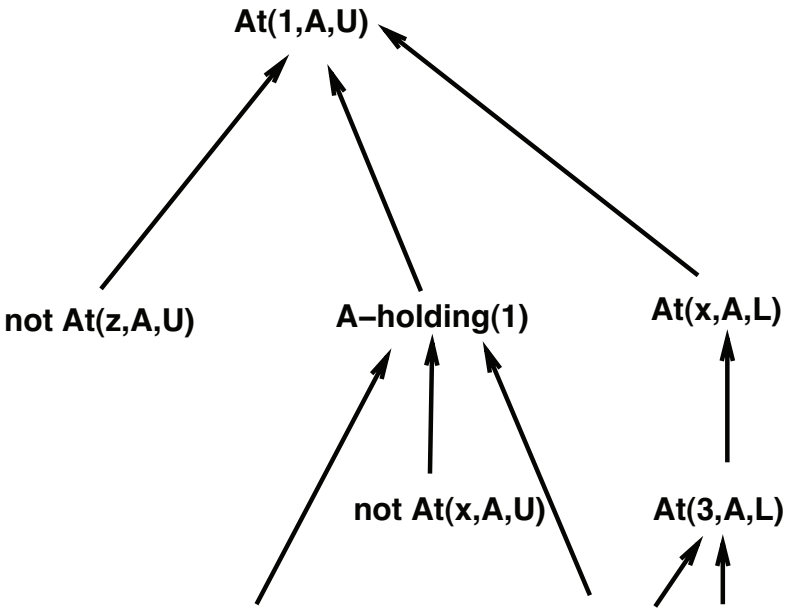
$\Rightarrow$ mediator can explicitly detect conflicts and synergies

Partial-global-planning (PGP)

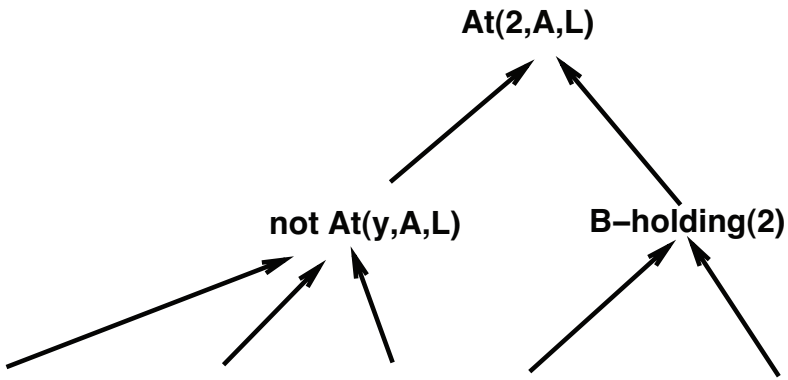
Basic structure = goal-tree

Each agent inserts its *partial plans*

Agent A:



Agent B:



Agents can discover joint goals:

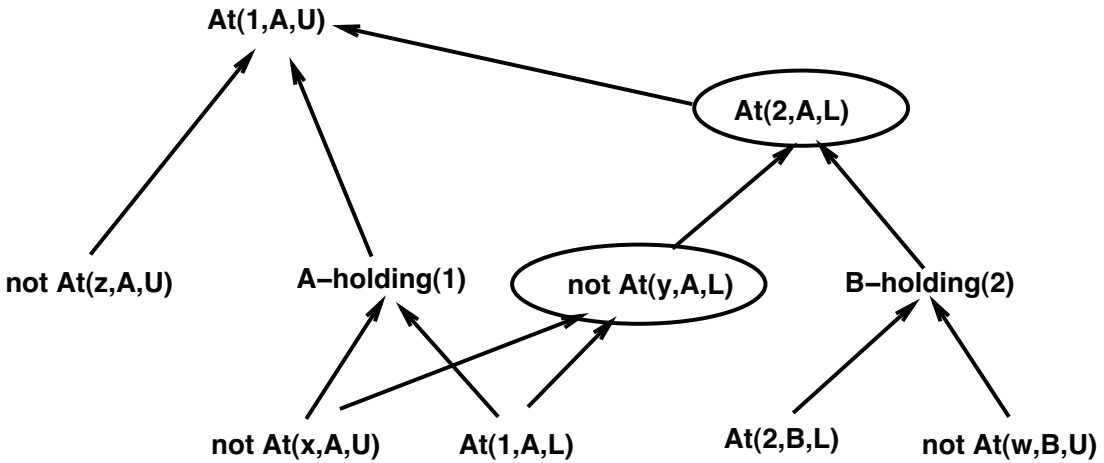
`At(x,A,L)` (agent A) matches `At(2,A,L)` (agent B)

PGP (2)

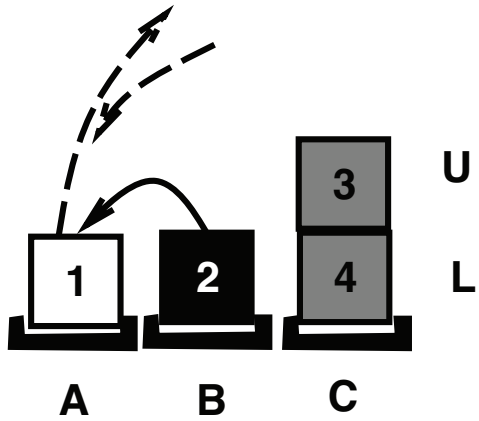
Joint goals:

- Agent A achieves `not At(1,A,L)` $\Leftrightarrow$ `not At(y,A,L)`
- Agent B achieves `At(2,A,L)` $\Leftrightarrow$ `At(x,A,L)`

$\Rightarrow$ simplified, combined plan:



Resulting operations



PGP (3)

Actions of partial plans are scheduled
$\Rightarrow$ further optimization:

- reorder actions to reduce total time
- exchange tasks between agents to optimize resource usage

Scheduling a multi-agent plan

Formulate as constraint optimization problem
Variables: for each action a_i:

- $agent(a_i)$ = agent that will carry out a_i
- $start(a_i)$ = start time of action a_i

Constraints:

- resource: $(agent(a_i) = agent(a_j)) \Rightarrow$
  $start(a_i) \notin [start(a_j)..start(a_j) + dur(a_j)]$
- precedence: $a_i \prec a_j \Rightarrow start(a_i) < start(a_j)$

Problems with blackboard systems

- Central database = central point of failure
- Very complex when number of agents is large, every agent has to plan for everyone else
- No concurrency, only one agent can modify at a time

Publish-subscribe systems

- Idea: identify potential conflicts and create explicit objects for them
- Example: placing blocks in lower/upper position of stack A
- When an agent's plan involves the resource, all others are notified $\Rightarrow$ detection of conflicts/synergies

$\Rightarrow$ peer-to-peer negotiations for optimal joint plan

Differences with blackboard systems

Eliminates many of the weaknesses, but only incremental changes (hillclimbing)
Systematic formalism: distributed constraint satisfaction

- variables = goals
- values = actions to achieve goals + agents that execute them
- constraints express common resources/preconditions

Heterogeneous data structures

- Agents are written by different people, at different times

$\Rightarrow$ no common model or data structures

- Agreement through shared *ontologies*: descriptions of concepts and their connections

Ontologies

- "Ground level" of agent communication: shared vocabulary
- Can be source of controversy: e.g. `due-date`:
  - shipper: date merchandise shipped
  - receiver: date merchandise received
- Simplest ontology:

  list of terms with agreed meaning

`shipper-ontology` $\neq$ `receiver-ontology`

Adding knowledge

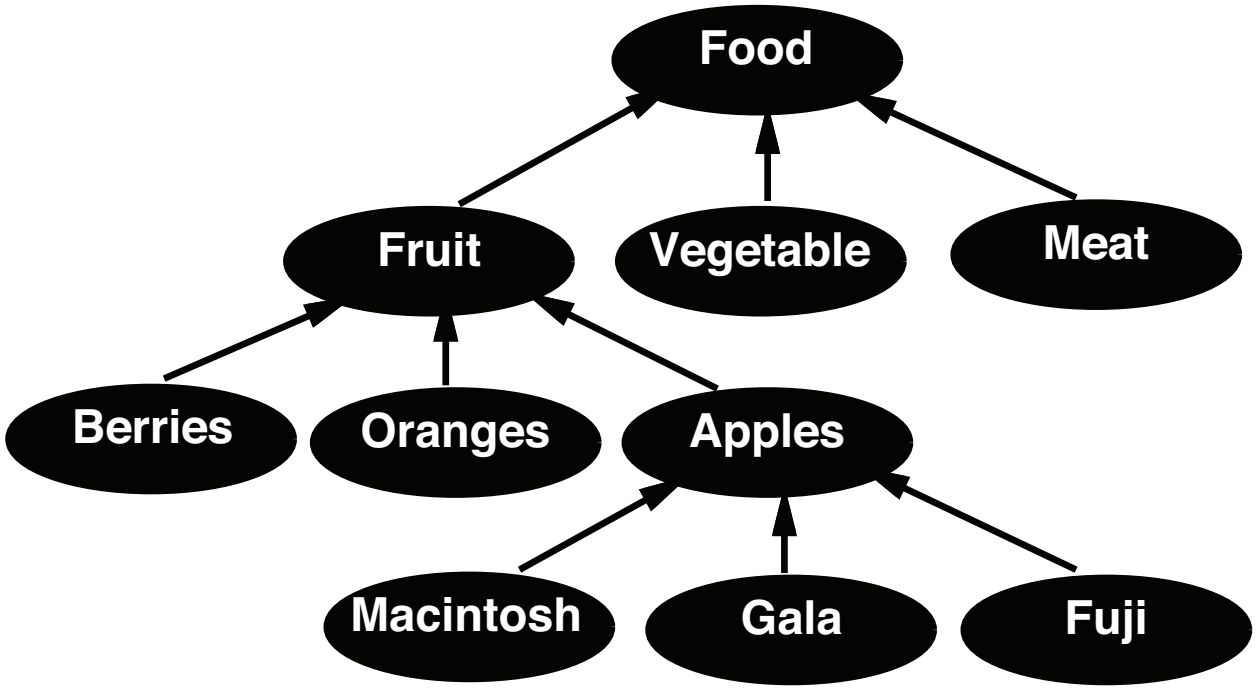
Terms are related by subclass relations:

`apple` *is-a* `fruit` *is-a* `food`

Sharing subclass relations is useful for communication, e.g.:

- do you sell `food`?:
  `apple` and `fruit` also match
- do you sell `apples`?:
  maybe other `fruit` could also be suggested

Class hierarchies



Relation = subclass

Properties

- Individuals in a class have certain *properties*
- Properties are attached to classes and defined by names and domains
- Each individual inherits properties from its class and all superclasses
- Example:
  `food` has-property `weight` domain `number`
  `fruit` has-property `ripe` domain `true/false`

Additional knowledge

- Relations between instances: `part-of`
- Restrictions on properties: a person has only one father
- (Common-sense knowledge: mother has age ≥ 14))

$\Rightarrow$ could be used for inferring `is-a` relations from background information (description logics)

- This is the main purpose of the *semantic web*

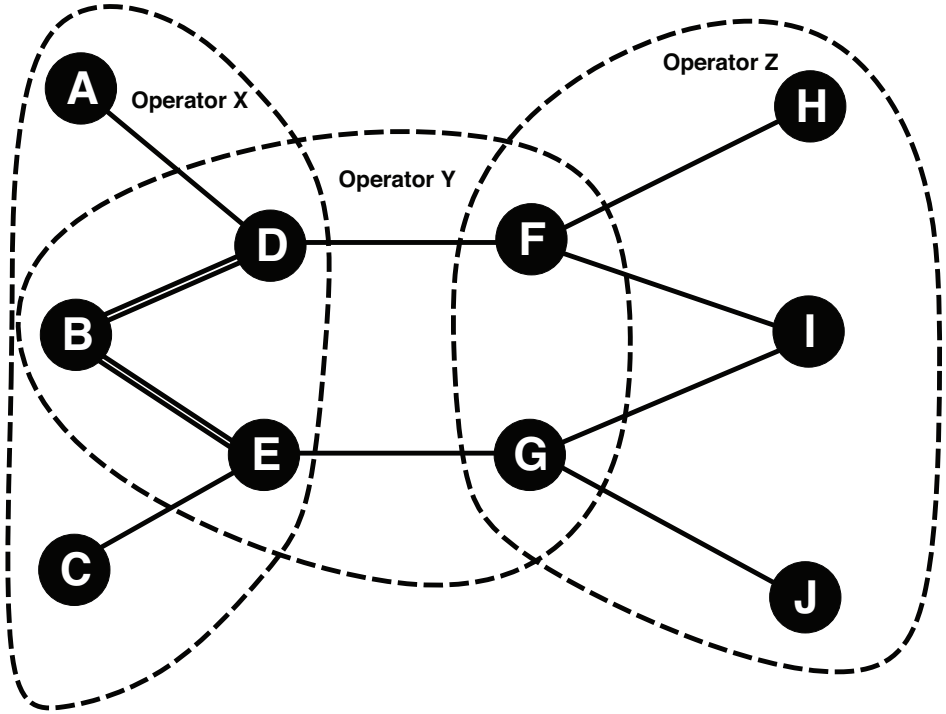
Standards and tools for Ontologies

- *OWL* (`http://www.w3.org/TR/owl2-overview/`):
  Ontology Web Language, a standard of the www consortium
- `http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/`
  Reasoning tools for *OWL*
- *Protege* (`http://protege.stanford.edu/`):
  tools for writing ontologies, especially used in medical informatics
- *LOOM*
  `http://www.isi.edu/isd/LOOM/PowerLoom/index.html`
  tool for constructing knowledge bases with description logics

Task assignment

- Agents have to carry out a set of tasks.
- Each has a certain payoff.
- Planner has generated a hierarchical structure where tasks are broken into subtasks that can be handled by individual agents.
- Tasks and agent capabilities modelled in ontologies $\Rightarrow$ allows to identify possible matches between tasks and agents.

Example: Telecom service providers



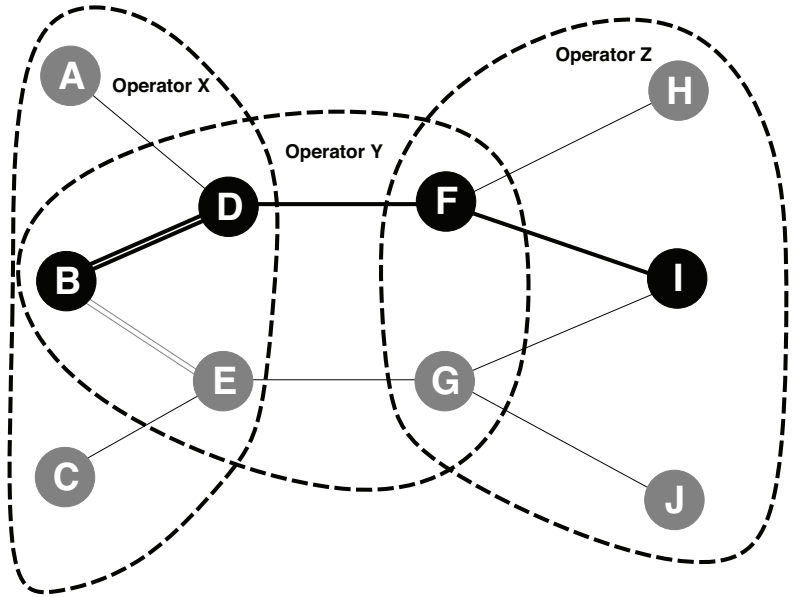
Communication from A to H must be carried by several operators: X, Y and Z

Contract Nets

Cooperation requires *contracts*
Contract net protocol:

- *managers* divide tasks
- *contractors* bid
- manager makes contract for lowest bid
- no negotiation of bids

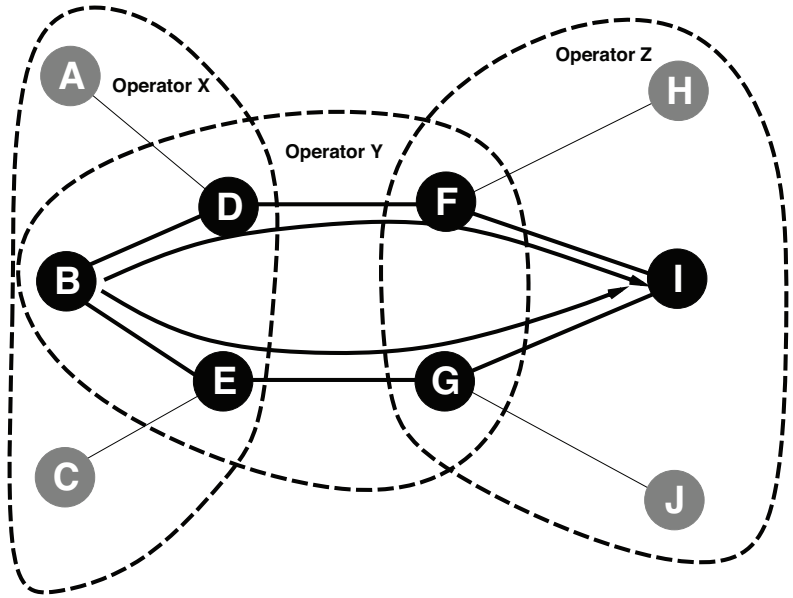
Example (contract nets)



Communication from B to I broken up into:

- B $\rightarrow$ D: X bids Fr. 3, Y bids Fr. 5: select X
- D $\rightarrow$ F: Y only bids Fr. 4: select Y
- F $\rightarrow$ I: Z only bids Fr. 6: select Z

Disjunctions



Two routes from B to I:

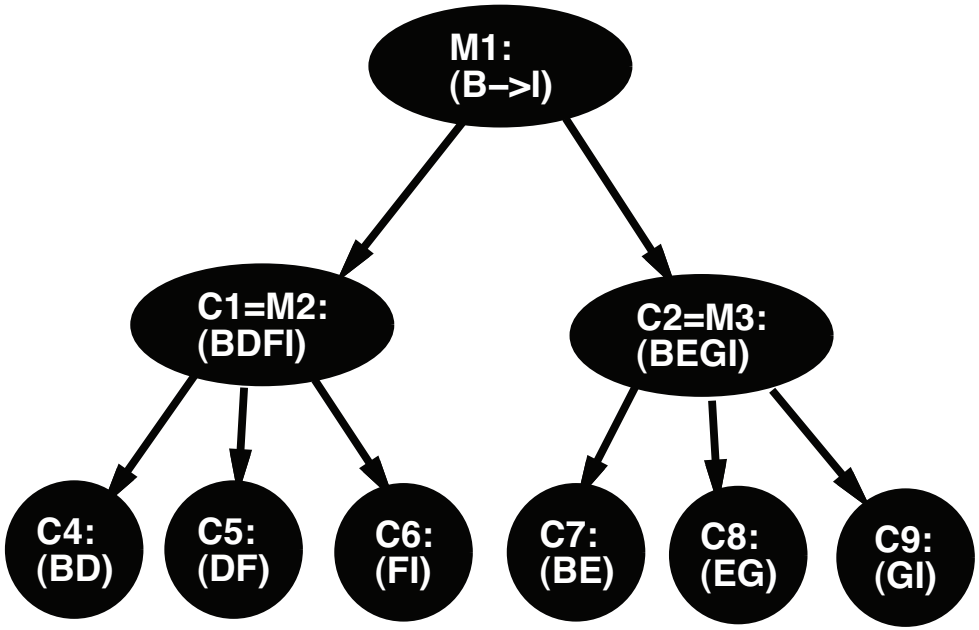
1. B $\rightarrow$ D $\rightarrow$ F $\rightarrow$ I
2. B $\rightarrow$ E $\rightarrow$ G $\rightarrow$ I

Manager opens bid for two routes

Contracts become managers at the next level

$\Rightarrow$ contract net

Routing determines two sub-managers (M2,M3):



Problems with contract nets

First come, first served

$\Rightarrow$ impossible to resolve conflicts.

Example:

$$communication\ B \rightarrow D \rightarrow F \rightarrow I$$

may block

$$communication\ A \rightarrow D \rightarrow F \rightarrow H$$

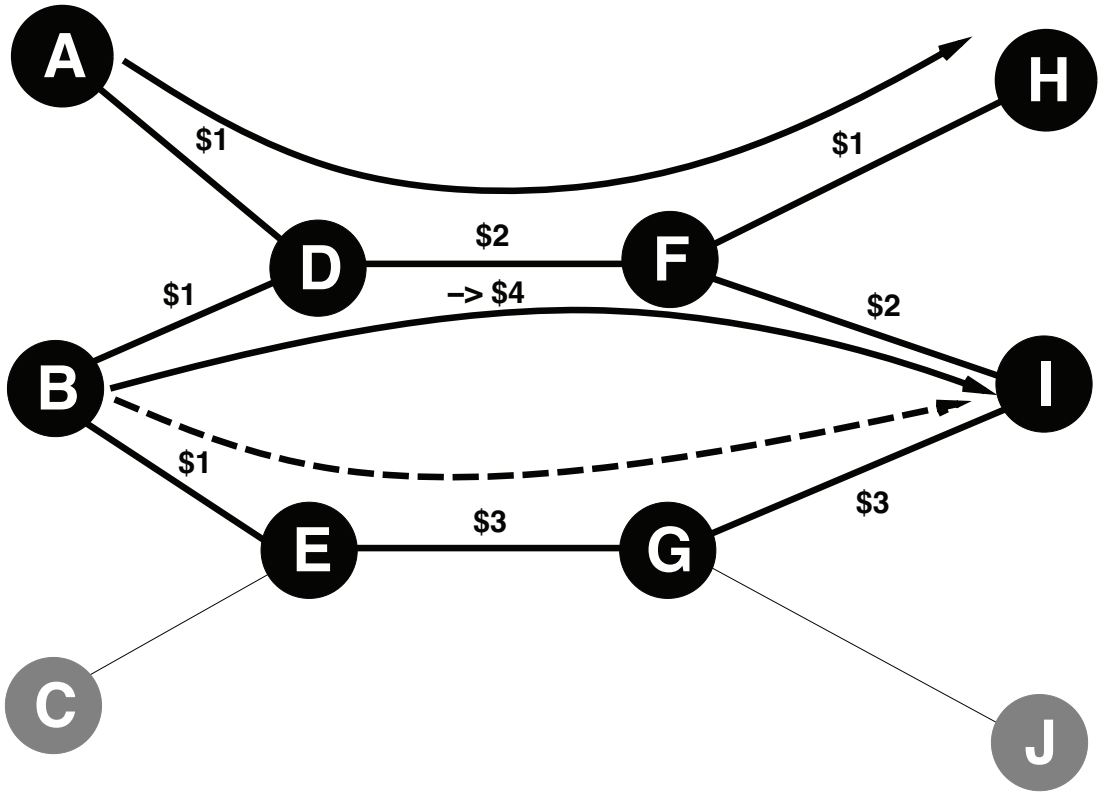
$\Rightarrow$ design bidding behaviors such that conflicts are avoided!

Market-based contract nets

Contract net protocol leaves open how prices are chosen
$\Rightarrow$ incremental bidding protcol:

- managers set the prices
- managers increase prices slowly when they cannot obtain contracts for all subtasks, as long as their output is taken at the resulting price
- bidding stops when no more changes occur
- if all tasks are taken, the result is a valid assignment

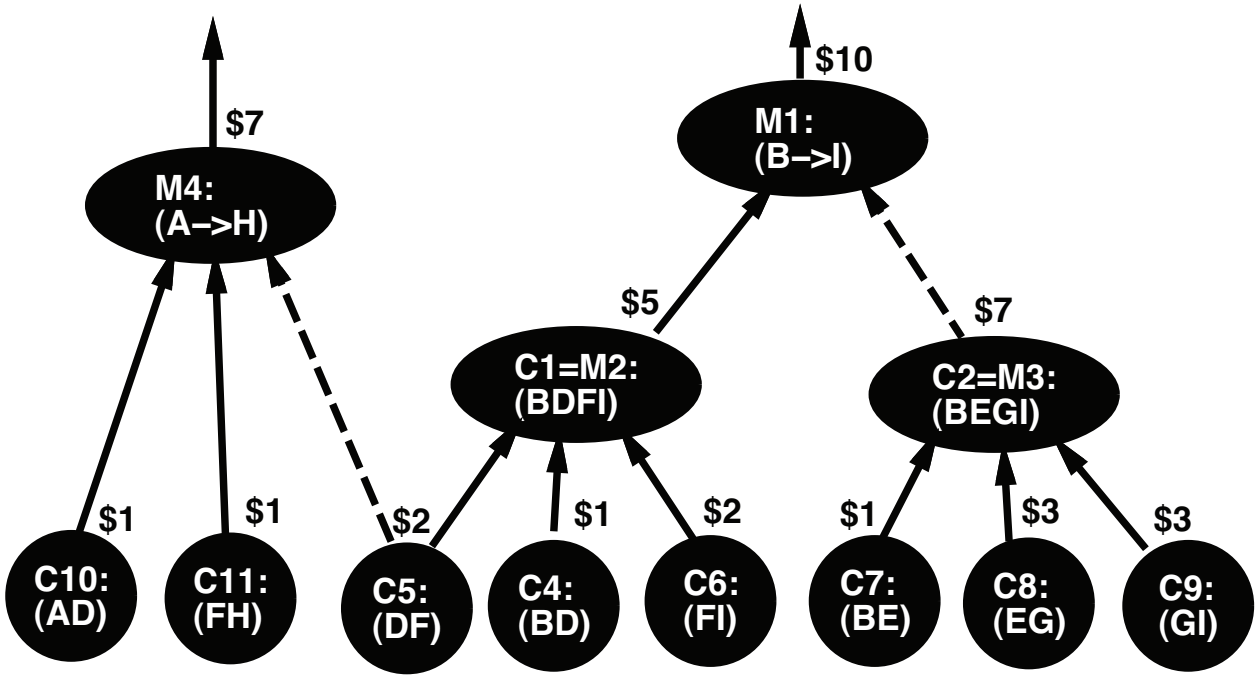
Example (market-based contract net)



Price of link D $\rightarrow$ F will increase from \$2 to $>$ \$4

Example (2)

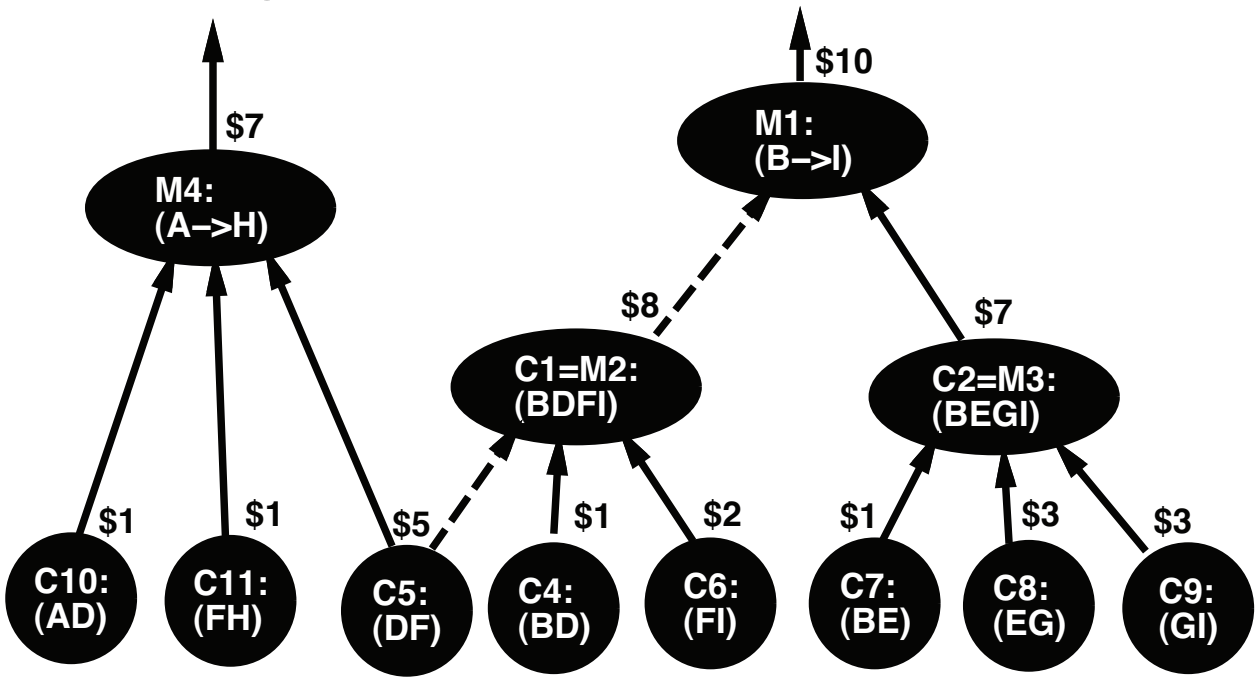
Tentative contract net:



M4 cannot obtain link D $\rightarrow$ F

Example (3)

Incremental bidding: $\$2 \rightarrow \$3 \rightarrow \$4 \rightarrow \$5$



Influence of increments

If increment = \$0.05:

need 41 rounds for change, but
final price is only \$4.05

If increment = \$2.00:

need to increase to \$6 ⇒
price for M4 will be \$8 ⇒
not acceptable!

Summary

- Some tasks require multiple agents
- Some systems are best modelled as multiple agents
- cooperative planning: centralized or through mediators
- communication among heterogenous agents through ontologies
- task assignment through contract nets