Introduction to Multi-Agent Programming

2. Societies of Agents

Rational Agents, Contract nets, Blackboard systems, Selfish Agents

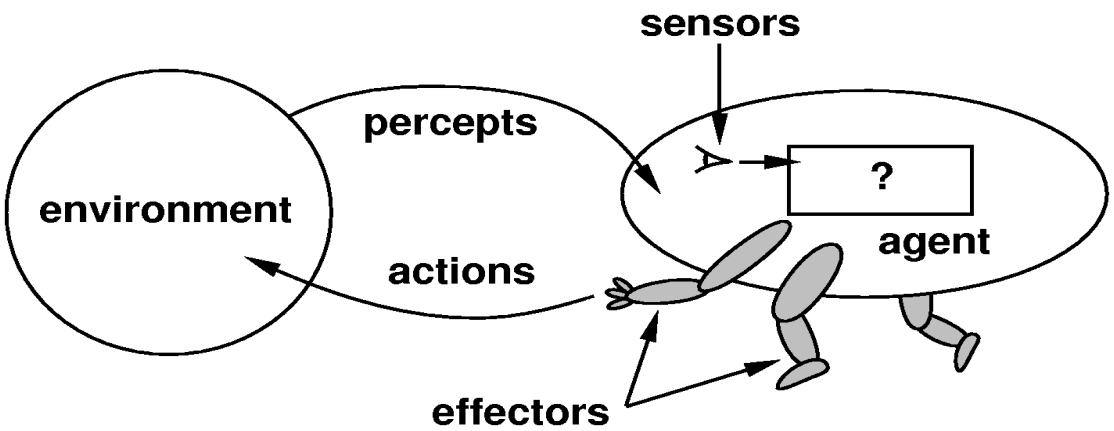
Alexander Kleiner, Bernhard Nebel

Contents

- Rational Agents
  - The structure of rational agents
  - Different classes of agents
  - Types of agent environments
- Societies of Agents
  - Coordination through interaction
    - Contract Nets
    - Blackboard Systems
  - Selfish Agents
    - Introduction to Game Theory

Rational Agents

- Perceive the environment through sensors
  (→ Percepts)
- Act upon the environment through actuators
  (→ Actions)
- Act rational with respect to a performance measure, e.g. time, energy, money, ...



Examples: Humans and animals, robots and software agents (softbots), temperature control, ABS, …

The Ideal Rational Agent

Rational behavior is dependent on

- Performance measures (goals)
- Percept sequences
- Knowledge of the environment
- Possible actions

Ideal rational agent: *For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.*

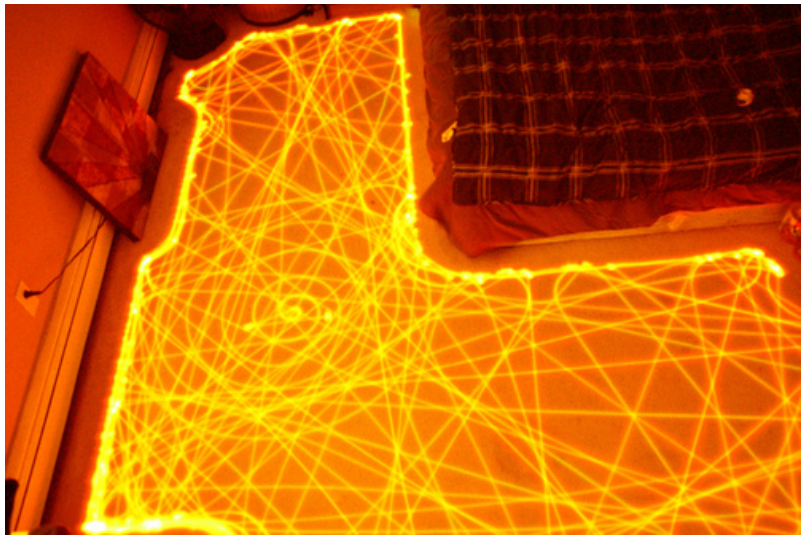
The ideal rational agent acts according to the function

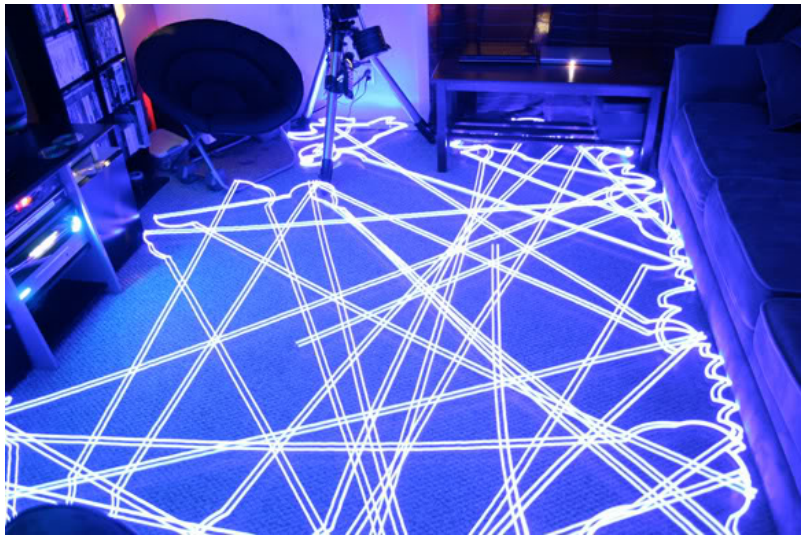
Percept Sequence x World Knowledge → Action

Examples of Rational Agents

Agent Type	Performance Measure	Environment	Actuators	Sensors
soccer robot	goal ratio	soccer field with other players	wheels (motors), kick-device	color camera, wheel odometry, laser range finder
rescue robot	victims found	rescue arena	wheels or tracks, pan-tilt unit	Color + thermo camera, CO_2 + audio sensor
vacuum cleaner	Cleaning completeness	household	wheels, suck device	camera, ultra-sonic, bumpers
taxi driver	speed, safety, ...	street map	steering wheel, accelerator, brake, horn	cameras, speedometer, GPS, ...
Refinery controller	maximize purity, yield safety	refinery, operators	valves, pumps, heaters, displays	temperature, pressure, chemical sensors

Example Roomba Cleaning Robot





Structure of Rational Agents (1)

Realization of the ideal mapping through an

- *Agent program*, maps from percept histories to actions f: $P^* \rightarrow A$, executed on an

- *Architecture* which also provides and interface to the environment (percepts, actions)

→ Agent = Architecture + Program

Structure of Rational Agents (2)

Example Skeleton

```
function Skeleton-Agent(percept) returns action
  static: memory, the agent's memory of the world

  memory ← Update-Memory(memory, percept)
  action ← Choose-Best-Action(memory)
  memory ← Update-Memory(memory, action)
  return action
```

Note:

- Memory capacity can be zero
- Performance measure is not part of the agent

The Simplest Design: Table-Driven Agents

```
function TABLE-DRIVEN-AGENT(percept) returns action
  static: percept sequence, initially empty; a table indexed by
          percept sequences, initially fully specified

  append percept to the end of percept sequence
  action ← LOOKUP(percepts, table)
return action
```

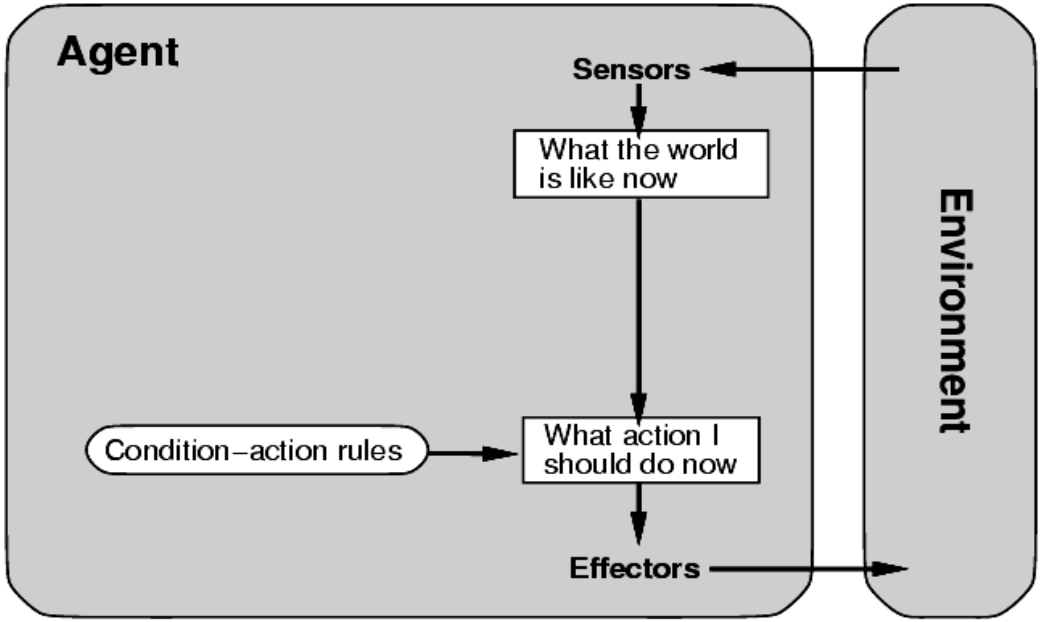
Problems:

- The table can become very large
- and it usually takes a very long time for the designer to specify it (or to learn it)
- … practically impossible

A Simple Reflex Agent

- Uses extracted condition-action rules
- Rule matching
- Percepts have to be interpreted
- Example fire fighters domain:

```
If (tank_is_empty) then
   return_to_refuge
```

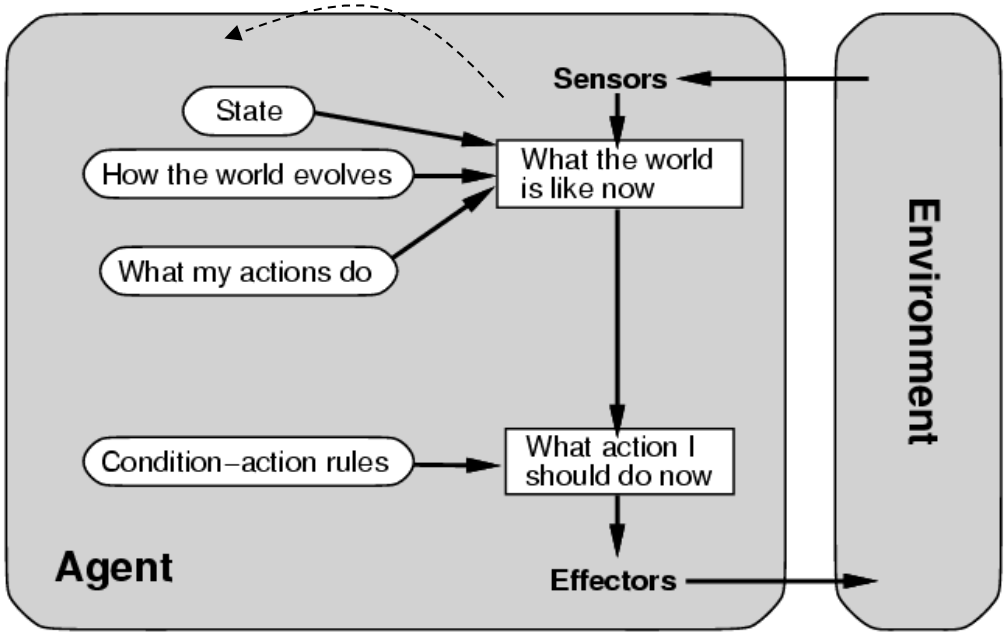


function *SIMPLE-REFLEX-AGENT*(*percept*) **returns** action
  static: *rules*, a set of condition-action rules

  state ← INTERPRET-INPUT(*percept*)
  rule ← RULE-MATCH(*state, rules*)
  action ← RULE-ACTION[*rule*]
  return *action*

Model-based Reflex Agents

- Updating of internal state representing the history of percepts
- Prediction of effects of actions given the state
- Example fire fighters domain:
  - Update size of fire in a district
  - Predict amount of water needed to extinguish district

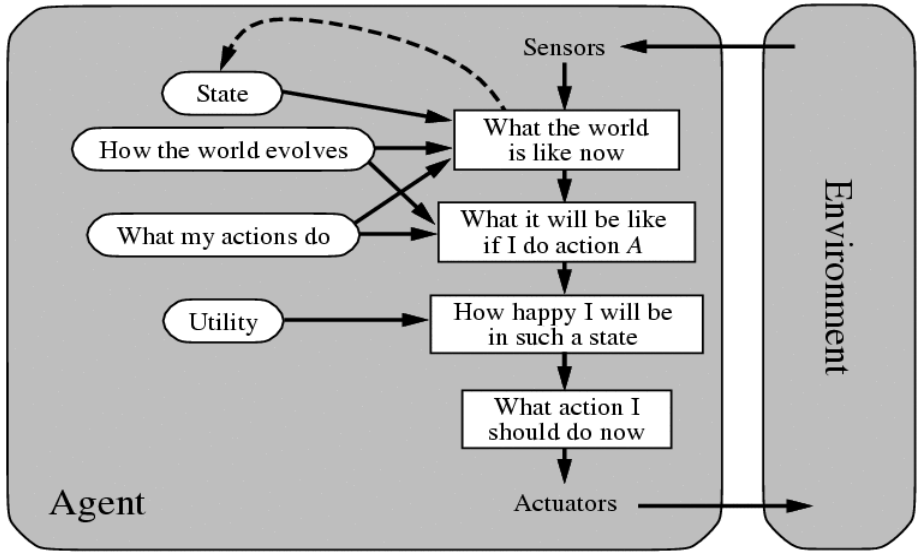


function *REFLEX-AGENT-WITH-STATE*(*percept*) **returns** action
  static: *rules*, a set of condition-action rules
        state, a description of the current world

  state ← UPDATE-STATE(*state*, *percept*)
  rule ← RULE-MATCH(*state, rules*)
  action ← RULE-ACTION[*rule*]
  state ← UPDATE-STATE(*state*, *action*)
  return *action*

Utility-Goal-Based Agents (1)

- Explicit goal representation
- Selection of goal with highest expected utility
- Actions are generated by planning to reach goal state

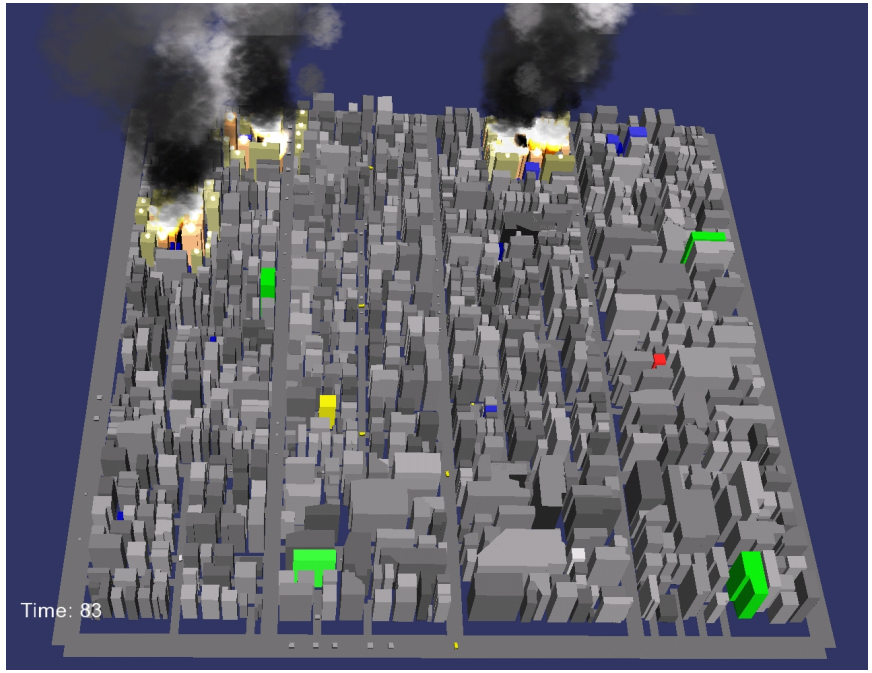



```
function UTILITY-BASED-AGENT(percept) returns action
static: rules, state, goal
state ← UPDATE-STATE(state, percept)
goal ← FORMULATE-GOAL(state, perf-measure)
search-space ← FORMULATE-PROBLEM (state, goal)
plan ← SEARCH(search-space , goal)
while (plan not empty) do
    action ← RECOMMENDATION(plan, state)
    plan ← REMAINDER(plan, state)
    output action
End
```

Utility-Based Agents (2)

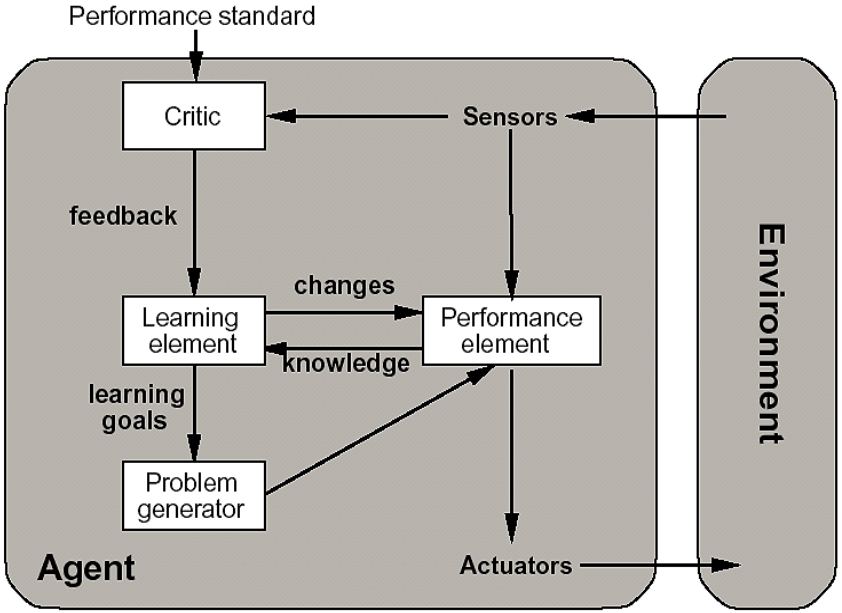
Example: fire fighters domain

- Each burning district instantiates a goal
  - UPDATE_STATE updates the fire parameters of each district
- Prediction of actions: time needed to extinguish each district
- Utility function: Civilians saved from fire
  - FORMULATE_GOAL selects district with highest expected outcome
- Planning to goals
  - SEARCH finds a path to a fire district (e.g. by BFS, A*, ...), and buildings to extinguish



Learning Agents (1)

- Any agent can be transformed into a learning agent
- Learning element: responsible for making improvements
- Performance element: has to select external actions
- Critic: determines the performance of the agent
- Problem generator: suggests informative actions (exploration)



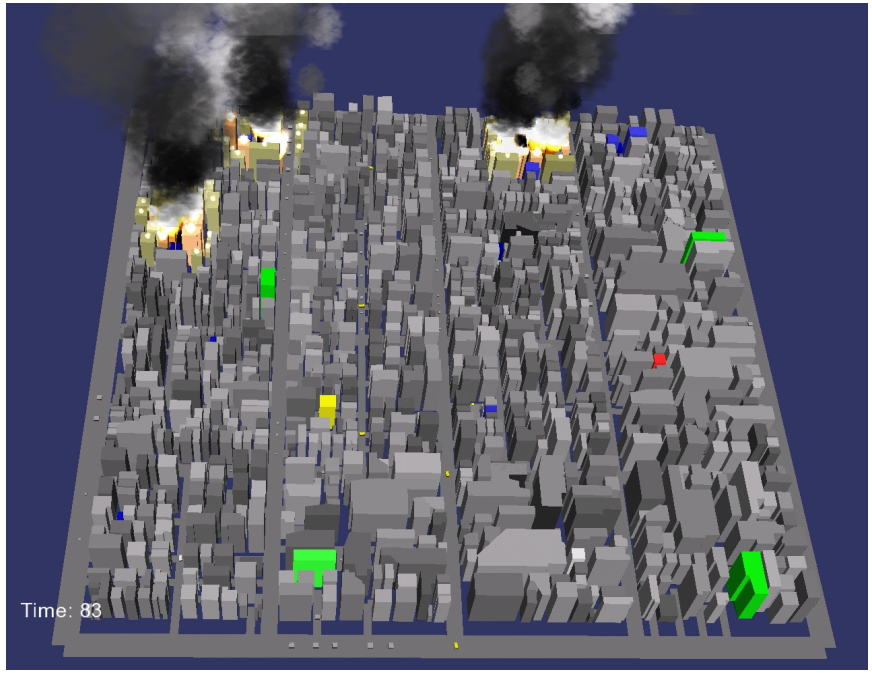
function *LEARNING-REFLEX-AGENT*(*percept, reward*) **returns** action
  static: *rules*, a set of condition-action rules

  state ← INTERPRET-INPUT(*percept*)
  rule ← RULE-MATCH(*state, rules*)
  action ← RULE-OR-EXPLORATIVE-ACTION[*rule*]
  rules ← RULES-CRITIC-UPDATE(reward, *rules*)
  return *action*

Learning Agents (2)

Example: fire fighters domain

- Partially or fully blocked roads cause more travel time
- Blockage of roads is unknown at start-up time
- Agents learn the travel time needed for each road segment during execution
- Planer prefers fast roads, i.e. those without or little blockage



The Environment of Rational Agents

- accessible vs. inaccessible (fully observable vs. partially observable)

  Are the relevant aspects of the environment accessible to the sensors?
- deterministic vs. stochastic

  Is the next state of the environment completely determined by the current state and the selected action? If only actions of other agents are nondeterministic, the environment is called strategic.
- episodic vs. sequential

  Can the quality of an action be evaluated within an episode (perception + action), or are future developments decisive for the evaluation of quality?
- static vs. dynamic

  Can the environment change while the agent is deliberating? If the environment does not change but if the agent's performance score changes as time passes by the environment is denoted as semi-dynamic.
- discrete vs. continuous

  Is the environment discrete (chess positions) or continuous (robot positions)?
- single agent vs. multi-agent

  Which entities have to be regarded as agents? There are competitive and cooperative scenarios.

Examples of Environments

Task	Observable	Deterministic	Episodic	Static	Discrete	Agents
Crossword puzzle	fully	deterministic	sequential	static	discrete	single
Chess with a clock	fully	strategic	sequential	semi	discrete	multi
poker	partially	stochastic	sequential	static	discrete	multi
backgammon	fully	stochastic	sequential	static	discrete	multi
taxi driving	partially	stochastic	sequential	dynamic	continuous	multi
medical diagnosis	partially	stochastic	sequential	dynamic	continuous	single
image analysis	fully	deterministic	episodic	semi	continuous	single
part-picking robot	partially	stochastic	episodic	dynamic	continuous	single
refinery controller	partially	stochastic	sequential	dynamic	continuous	single
Interactive English tutor	partially	stochastic	sequential	dynamic	discrete	multi

Whether an environment has certain property also depends on the conception of the designer.

Summary

- An agent is something that perceives and acts. It consists of an architecture and an agent program.
- An ideal rational agent always takes the action that maximizes its performance given the percept sequence and its knowledge of the environment.
- An agent program maps from a percepts to actions.
- There are a variety of designs
  - Reflex agents respond immediately to percepts
  - Goal-based agents work towards goals
  - Utility-based agents try to maximize their reward
  - Learning agents improve their behavior over time
- Some environments are more demanding than others.
- Environments that are partially observable, nondeterministic, strategic, dynamic, and continuous and multi-agent are the most challenging.

Societies of Agents (1)

- Conventional AI focuses on one agent, what happens when we consider more than one agent?
- An *intelligent agent* in a society is a rational agent with the following abilities:
  - Reactivity: the ability to react on changes in the environment in real time
  - Pro activeness: the ability to take the initiative with respect to the goals, e.g. not driven by events
  - Social ability: to interact (communicate, cooperate, collaborate) with other agents (and possibly humans) by some kind of agent-communication language

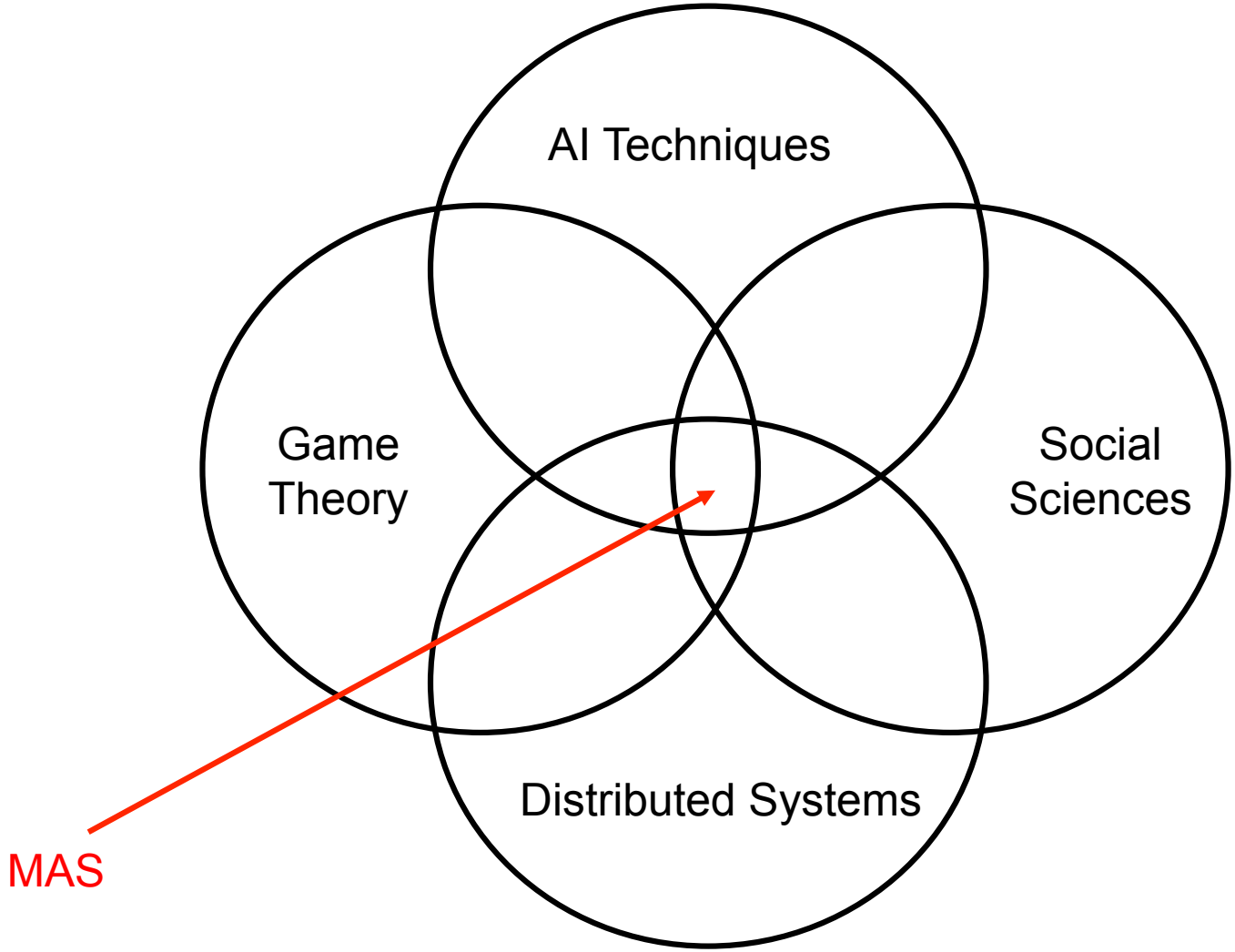
Societies of Agents (2)

Is it not all just Artificial Intelligence (AI) ?

- Do we need to solve all the problems of AI itself, e.g. to solve the planning problem, the learning problem, ... in order to build an agent?
  - ... In short, while we may draw upon AI techniques to build agents, we do not need to solve all the problems of AI to build an agent ...
  - Intelligent agents are *99%* computer science and *1%* AI (Etzioni, 1996)
  - "We made our agents dumber and dumber and dumber… until finally they made money." (Etzioni speaking about the commercial experience with NETBOT)
- Classical AI ignored *social* aspects of agency. These are important parts of intelligent activity in real-world settings

Societies of Agents (3)

Influencing Disciplines

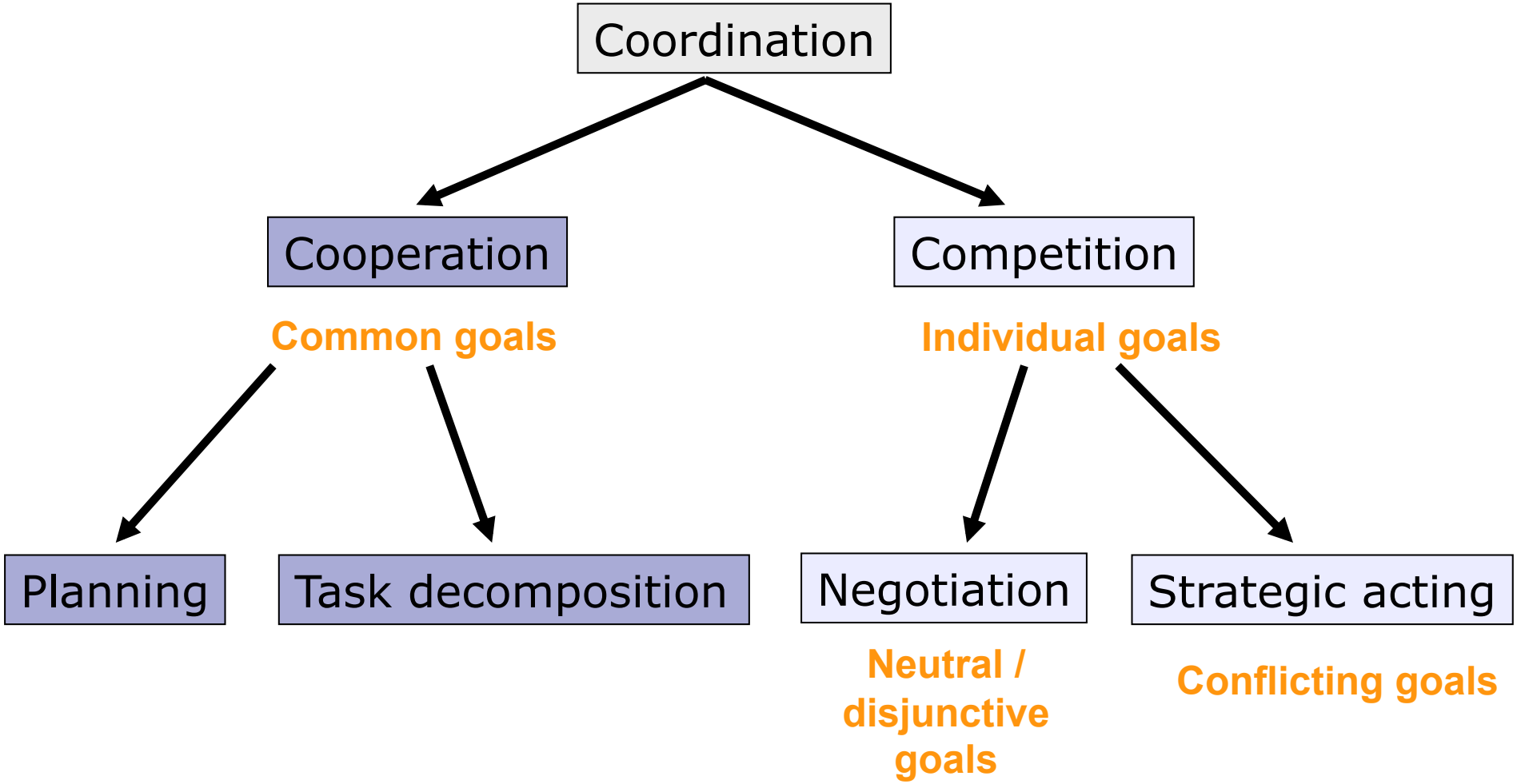


Attributes of MAS

	attribute	range
agents	number	from two upward
	uniformity	homogeneous / heterogeneous
	goals	contradictory / complementary
	architecture	reactive / deliberative
	abilities (sensors etc.)	simple / advanced
interaction	frequency	high / low
	persistence	short-term / long-term
	level	signal level / knowledge level
	pattern	decentralized / hierarchical
	variability	fixed / changeable
	purpose	competitive / cooperative
environment	predictability	foreseeable / unforeseeable
	accessibility	limited / unlimited
	dynamics	low / high
	diversity	poor / rich
	availability of resources	restricted / ample

From Huhns & Singh 1998, "Agents and multi-agent systems: Themes, approaches and challenges"

Coordination Through Interaction (1)



Coordination Through Interaction (2)

- Benevolent agents
  - e.g. team of fire brigades, robots exploring unknown terrain
  - Agents are assumed to act truthfully
  - Cooperative distributed problem solving: agents can be designed to help whenever asked for
  - Cooperation mechanisms are for example contract nets, and blackboard system
- Self-interested agents
  - e.g. from different organizations, Internet markets, computer games
  - Agents assumed to work for their own benefit, possibly at expense of others
  - Coordination by adequate mechanism design, e.g. Game theory, Auctions

Task Decomposition and Assignment: Contract Nets (1)

- An agent that wants a task to be solved is the manager
- Agents able to solve the task are potential contractors
- The manager:
  - announces a task (the task specification)
  - receives and evaluates bids from potential contractors
  - awards a contract to a suitable contractor
  - receives and synthesizes the results

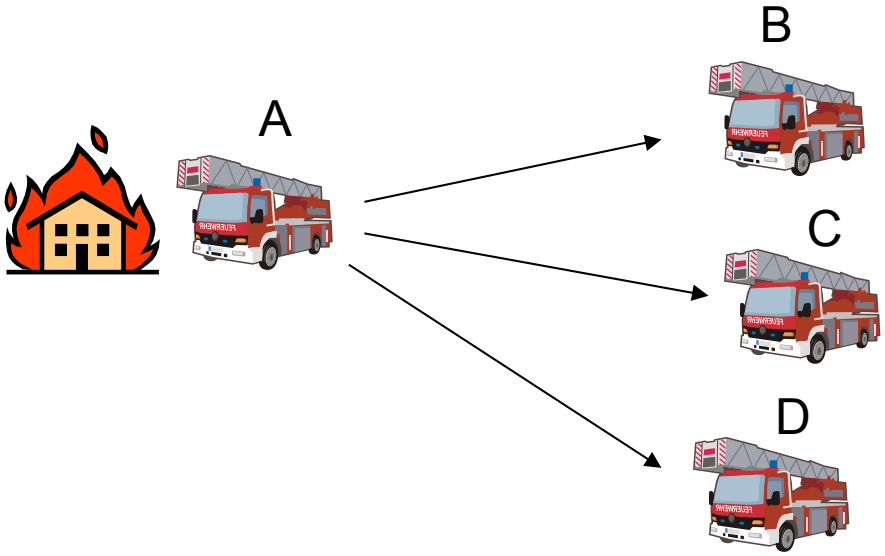
Contract Nets (2)

- The potential contractor:
  - receives task announcements
  - evaluates the capability to respond
  - responds with a bid or declines
  - perform task if the bid is accepted
  - report the results back
- Roles are not specified in advance, but are dynamic
- In particular, a contractor might further decompose a task and give some parts away to other contractors!

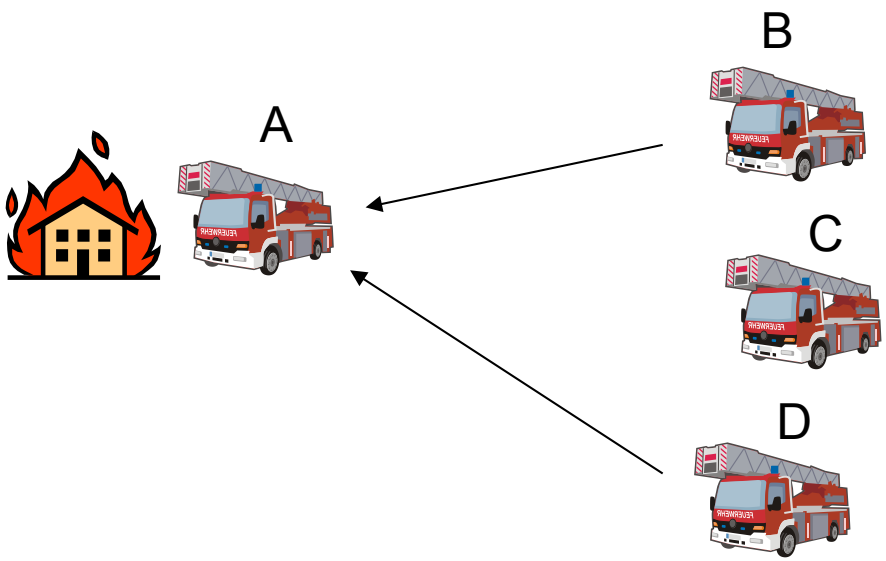
Contract Nets (3)

Fire Brigade example

- Fire brigade *A* needs help to extinguish a building
  - Task specification: needed amount of water, the location of the fire, and a deadline



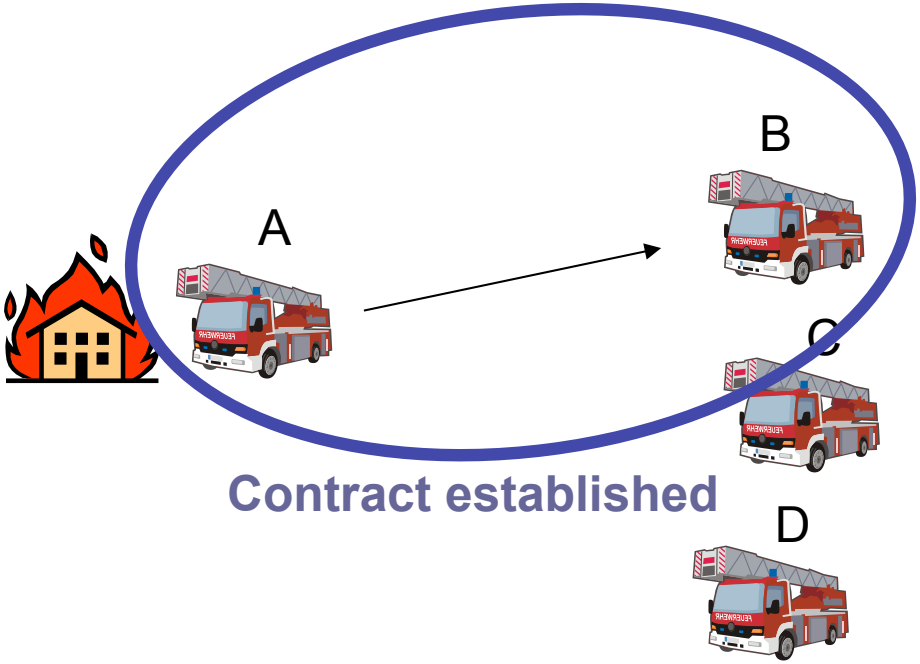
- Agent *B* and *D* submit their bits
  - The bit contains estimated costs for traveling to the location and for refilling the tank



Contract Nets (4)

Fire Brigade example

- The manager awards a contract to the most appropriate agent
  - For example, agent *B*, which is closer to the fire
- The contractor sends back a report after finishing the task or further subdivides the task …



Contract Nets (5)

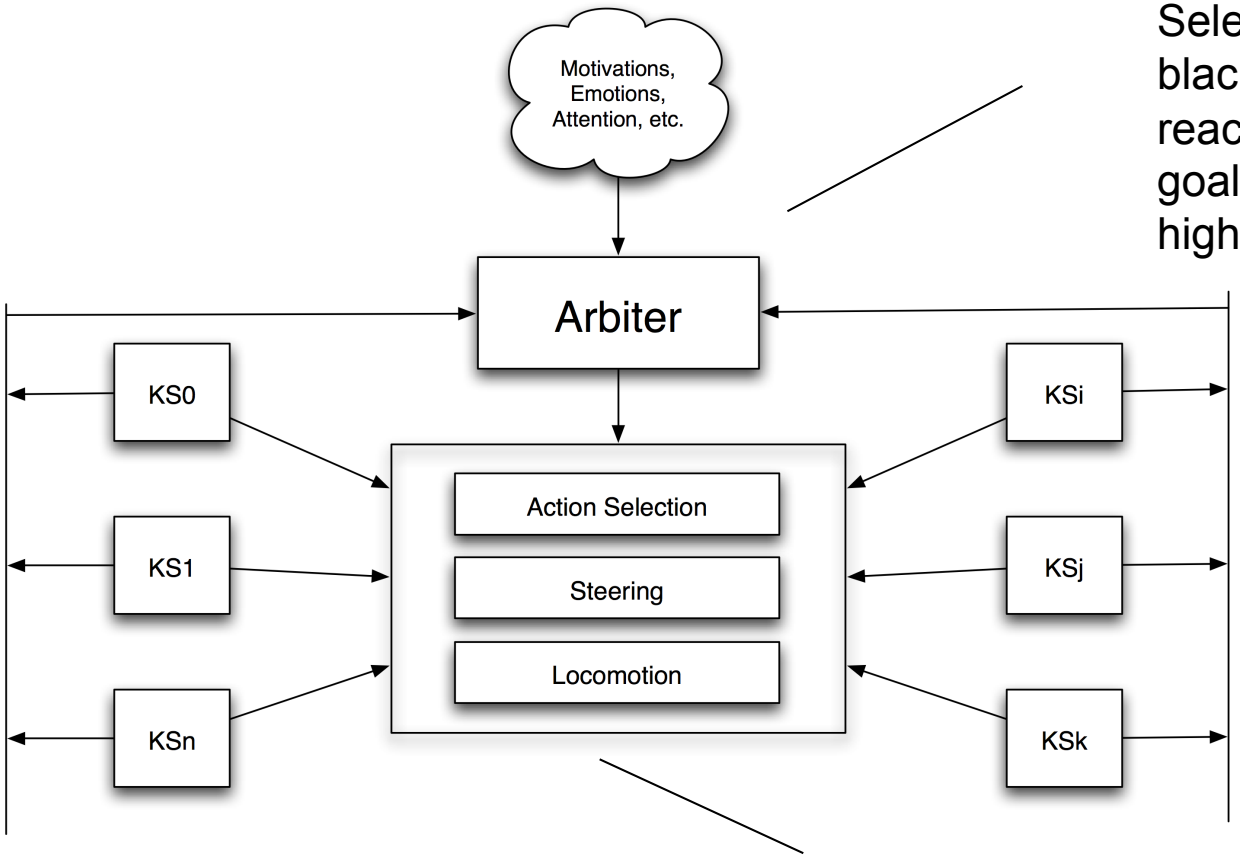
Limitations

- Limitations:
  - Task decomposition and problem syntheses can be non-trivial
  - Communication overhead
  - The awarded contractor might not be the best choice, a better candidate could be temporarily busy during award time
- Efficiency modifications:
  - Focused addressing / direct contracts (e.g. team structure)
  - Agent send status message, e.g. *eligible but busy*, *ineligible*, *uninterested*, …

Task Decomposition and Assignment: Blackboard Systems (1)

- Data-driven approach to task assignment
  - A number of "experts" are sitting next to a blackboard
  - When one of the experts sees that she can contribute something, she writes this on the blackboard
  - This continues until the "solution" comes up on the blackboard
- Mainly used for *distributed problem solving*, e.g. speech recognition
- Requires a common interaction language
- Event-based activation
- Can have different levels of abstraction

Blackboard systems (2)



Arbiter

Selects "winning" KS for accessing blackboard. Mechanism can be reactive (data-driven) but also goal-driven, e.g. select KS with highest expected future outcome

Knowledge sources (KSs)

A series of components that are able to operate on the blackboard

Blackboard

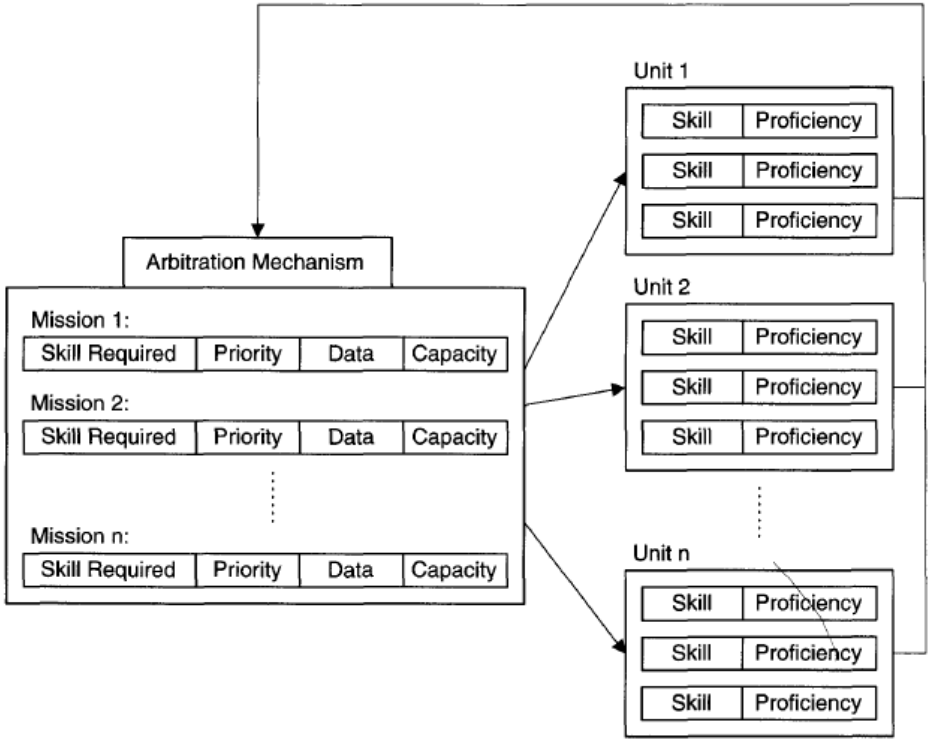
publicly read/writeable data structure (e.g. shared memory)

"Blackboard Architectures," AI Game Programming Wisdom, Volume 1, pp. 333 - 344

Blackboard systems (3)

Example: RTS game *BBWar* using the C4 blackboard architecture (MIT 2001)

- The KSs are individual units that have special skills that can be executed on demand
- The blackboard contents take the form of open missions
- Units from different levels of the hierarchy pay attention to different types of postings
  - Commanders look for *ATTACK-CITY* missions and *create* *ATTACK-LOCATION* missions
  - Soldiers look for *ATTACK-LOCATION* missions
  - …
- Implemented as a hash table mapping skill names to open missions



"Blackboard Architectures," AI Game Programming Wisdom, Volume 1, pp. 333 - 344

Blackboard systems (4)

- Advantages:
  - Simple mechanism for cooperation and coordination
  - KSs do not need to know about other KSs they are cooperating with
  - Postings can be overwritten by different systems, e.g. units can be replaced
  - Can also be used for inter-agent communication
- Disadvantages:
  - Mainly suitable for agents executed on the same architecture (possible solution: decentralized hash tables)

Self-interested Agents (1)

- What happens when agents are not benevolent?
  - Why should they report their capabilities truthfully?
  - Why should they actually complete contracted tasks?
- Cooperation works fine if we can design the entire system by ourselves
  - We can then try to maximize some performance measure and guarantee that all member of a team of agents work towards the common goal
- If agents work for different parties the common goal might not be the goal of the single agents
  - e.g., assume an arrival management system for airports with a number of different airlines or the Internet
- If an MAS becomes large and complex the overall goal is not evident (e.g. in an intelligent house)
  - ➢ It might be more robust to design agents as self-interested agents

Self-interested Agents (2)

- What is the self-interest of a competitive agent?
- The agent tries to maximize expected utility!
- AI techniques are good for that, but …
- … here we have other agents that also act
- All agents know (to a certain extend) what their options are and what the payoff will be

➢ Strategic deliberation and decision making
  - ➢ Choose the option that maximizes own payoff under the assumption that everybody also acts rationally
  - ➢ Does not maximize social welfare but is robust

Game Theory

- Game Theory is the field that analyzes strategic decision situations
  - economic settings
  - military contexts
  - social choices
- Usual assumption: All agents act rationally
  - Unfortunately, humans do not follow this pattern all the time
  - Often change their utility function on the way or simply do not maximize or do not assume that all others act rationally
- Nevertheless: For designing MAS it might just be the right theoretical framework because we can design our agents to act rationally.

Summary

- MAS focus on the interaction between agents as opposed to AI, which focuses on single agents
- There are two main strands:
  - Cooperative agents, which work together to achieve a common goal
  - Competitive agents, which try to maximize their own expected utility
  - The latter might also be useful in cooperative settings, because it leads to particularly robust behavior
- Game Theory is the right theoretical framework to deal with strategic decision situations appearing in groups of self-interested agents

Literature

- Russel, S. and Norvig, P. **Artificial Intelligence: A Modern Approach**, second edition, *Prentice Hall*, 2003
- Davis, R. and Smith, R. **Negotiation as a Metaphor for Distributed Problem Solving** *Artificial Intelligence* 20, pp. 63-109, 1983. Winner of the 2006 Influential Paper Award
- Corkill, D. **Blackboard Systems**. *AI Expert*, 6(9):40-47, September, 1991
- Isla D. and Blumberg, B. **Blackboard Architectures**, *AI Game Programming Wisdom*, Volume 1, pp. 333 - 344