**Lecture 4**

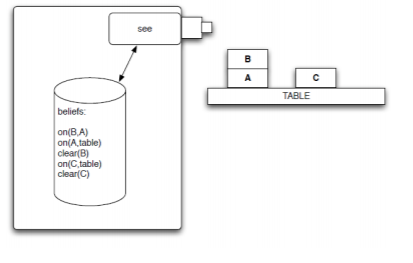
Agent Architectures

* An agent architecture is a software design for an agent.
* We have already seen a top-level decomposition, into:
  + Perception -> State -> Decision -> Action
* An agent architecture defines:
  + Key data structures;
  + Operations on data structures;
  + Control flow between operations

Types of Agents

* 1956 -> present: Symbolic Reasoning Agents proposes that agents use explicit logical reasoning in order to decide what to do
* 1985->present: Reactive Agents proposes that problems with symbolic reasoning requires a reaction against this; led to reactive agents movement.
* 1990->present: Hybrid Agents architectures attempt to combine the best of symbolic and reactive architectures.

Symbolic Reasoning Agents

* Classical approach to building agents is to view them as a particular type of knowledge based system, and bring all the associated methodologies of such systems to bear.
* This paradigm is known as symbolic AI.
* We define a deliberative agent or agent architecture
  + Contains an explicitly represented, symbolic model of the world;
  + Makes decisions via symbolic reasoning

The Transduction Problem

The problem of translating the real world into an accurate, adequate symbolic description, in time for that description to be useful. E.g. vision, speech, understanding, learning.

Representation/Reasoning Problem

How to symbolically represent information about complex real-world entities and processes, and how to get agents to reason with this information in time for the results to be useful.

Deductive Reasoning Agents

Deductive Reasoning Agents use logic to encode a theory defining the best action to perform in any given situation.

What is practical Reasoning

* Practical reasoning is reasoning directed towards actions a the process of figuring out what to do:
  + Practical reasoning is a matter of weighing conflicting considerations for and against competing options, where the relevant considerations are provided by what the agent desires/values/cares about and what the agent believes. (Bratman)
* Distinguish practical reasoning from theoretical reasoning. Theoretical reasoning is directed towards beliefs.

Components of practical reasoning

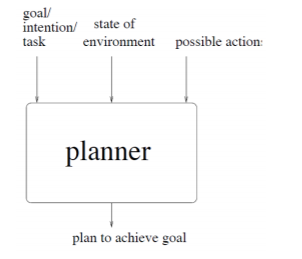
Human practical reasoning consists of two activities:

* Deliberation
  + Deciding what state of affairs we want to achieve:
  + The outputs of deliberation are intentions;
* Means-ends reasoning
  + Deciding how to achieve these states of affairs
  + : the outputs of means0ends reasoning are plans.

Intentions in Practical Reasoning

* Intentions pose problems for agents. Who need to determine ways of achieving them.
  + If I have an intention to a, you would expect me to devote resources to deciding how to bring about a.
* Intentions provide a “filter” for adopting other intentions, which must not conflict.
  + If I have an intention a, you would not expect me to adopt an intention b that was incompatible with a.
* Agents track the success of their intentions and are inclined to try again if their attempts fail.
  + If an agents first attempt to achieve a fails, then all other things being equal, it will try an alternative plan to achieve a.
* Agents believe their intentions are possible
  + That is, they believe there is at least some way that the intentions could be brought about
* Agents do not believe they will not bring about their intentions.
  + It would not be rational of me to adopt an intention to a if I believe that under “normal circumstances” I will succeed with a.
* Agents need not intend all the expected side effects of their intentions.
  + If I believe a->b and I intend that a, I do not necessarily intent b also.

Means-ends Reasoning/Planning

* Planning is the design of a course of action that will achieve some desired goal.
* Basic idea is to give a planning system:
  + (representation of) goal/intention to achieve
  + (representation of) actions it can perform
  + (representation of) the environment
* And have it generate a plan to achieve the goal.
* This automatic programming.

Representations

* How do we represent
  + The goal to be achieved
  + The state of the environment
  + The actions available to agent
  + The plan itself
* Weˆall illustrate the techniques with reference to the blocks world.
* Contains a robot arm, 2 blocks (A and B) of equal size, and a table-top.
* To represent this environment we need an **ontology**.
  + On(x,y) obj x on top of obj y
  + OnTable(x) obj x is on the table
  + Clear(x) nothing is on top of the obj x
  + Holding(x) arm is holding x

Automated Planning

* Automated planning is a basic AI problem
* We have:
  + An initial state
  + A goal state
  + A number of actions which modify the state
* We need to find (if it exists) a sequence of actions which link the beginning state to the goal state

Classical AI planning using PDDL

* PDDL(Planning Domain Definition Language) is a means of describing states, goals and actions
* A PDDL domain defines a problem and the actions that may be applied to it
* A PDDL problem defines the beginning state and goal state of a problem defined in a domain
* A PDDL solver attempts to find a sequence of actions that successfully transform the initial state into the goal state.

A PDDL Action

Action(Fly(p,from,to), PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)

EFFECT:¬At(p, from) ∧ At(p,to))

This action moves an aircraft p between from and to .

* A PDDL action consists of:
  + Name and a list of variables used
  + Precondition: defines the states in which the action can be executed
  + Effect: defines effect of the action on the variables

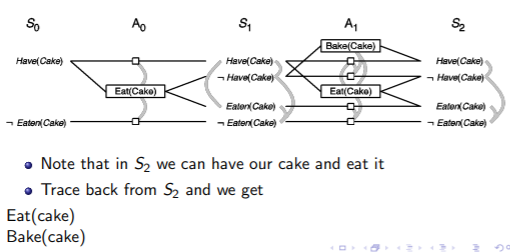
Automated Planners

* If we can specify a problem in PDDL then we can search for a solution automatically.
* Its not simple, there may exist a great number of possible plans
* Many will not take us to the goal state

Automated Problem Solving

* We need to search for a plan that takes us to the goal state
* The possibility is a tree search (state-space search)
  + Our root node is the initial state
  + We then have branch nodes for each state that we can reach using one valid action from the current state
* The tree will grow very quickly and become very large for anything except the most trivial problem!
* One option is to start from the goal state and the initial state and construct trees concurrently
  + When they both find the same state, they have collectivelyfound a plan
  + Also know as Backward search or regression
* Some will take us to the goal state via many unnecessary states
* As we know the beginning and end states can be thought of as a graph search
  + Nodes represent problem states
  + Arcs represent actions between problem states
* It is possible to use A\* to find a plan from within a graph
  + The heuristic is problematic
  + Need to be able to define how aclosea state is to solving the problem

GraphPlan

* A planning graph is a directed organised graph
* It comprises levels
  + S0 represnting the initial state
  + A0 represents those actions that are applicable in S0
* Keep adding levels until the graph has alevelled offa
* Levelling off occurs when consecutive levels of the graph are the same

Implementing Practical Reasoning Agents

* A first pass at an implementation of a practical reasoning agent:
* While true
  + Observe the world
  + Update internal world model;
  + Deliberate about what intention to achieve next;
  + Use means-ends reasoning to get a plan for the intention;
  + Execute the plan
* End while

Deliberation

* How does an agent deliberate?
  + Begin by trying to understand what the options available to you are;
  + Choose between them, and commit to some.
* Chose options are then intentions.
* The deliberate function can be decomposed into two distinct functional components:
  + Option generation
  + Filtering
* Chosen options are then intentions.

Option Generation

* In which the agent generates a set of possible alternatives
* Represent option generation via a function, options, which takes the agents current beliefs and current intentions, and from them determines a set of options.

Filtering

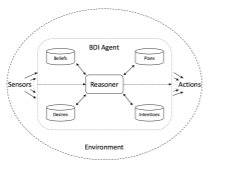
* In which the agent chooses between competing alternatives and commits to achieving them.
* In order to select between competing options, an agent uses a filter function.

Commitment

* Blind commitment
  + A blindly committed agent will continue to maintain an intention until it believes the intention has actually been achieved. Blind commitment is also sometimes referred to as fanatical commitment.
* Single-minded commitment
  + A single-minded agent will continue to maintain an intention until it believes that either the intention has been achieved, or else that it is no longer possible to achieve the intention.

Summary: BDI agents

* The BDI model is a well-established paradigm for the development of agents
* A BDI Agent has knowledge about its world (beliefs) and pursues goals (desires) while following given strategies (intentions).
* Agents must be able to act according to their knowledge of their internal and external world which must therefore be expressed in an appropriate format.



* Beliefs represent the informational state, which comprise its knowledge about the world (knowledge base)
* Desires represent the motivational state of the agent; in other words it defines the things that the agent may wish to achieve – goals.
* Plan is a set of available actions. A goal can be achieved by using one or more plans.
* Intentions of an agent are the particular plans that the agent has committed to performing in order to achieve its goals.
* Reasoner is the engine which bundles together the previous four components. It receives data (e.g. communication), updates beliefs and goals, selects next agent actions.

**Lecture 5 – Reactive and hybrid agents**

Problems with reasoning agents

* Logic and reasoning pose substantial problems
  + Transduction problem
  + Representation/reasoning problem
* -> some researchers completely reject symbolic approach and syntactic reasoning on symbolic representations.

Ractive Behaviours

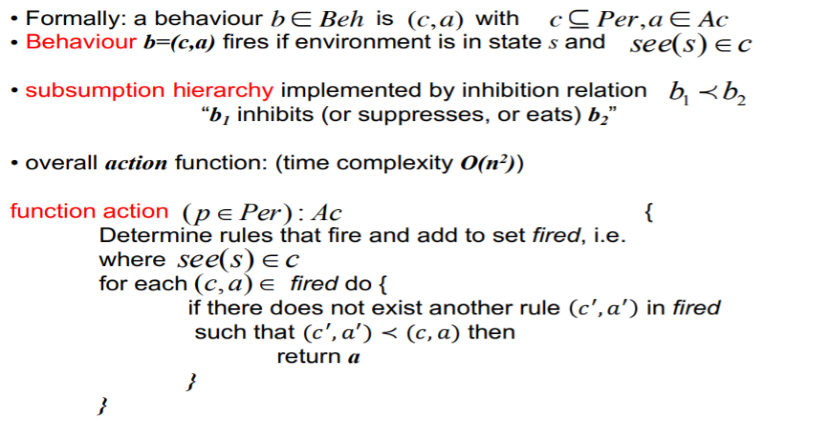
* An alternative approach is to build simple reactive behaviours,
* Complex intelligent behaviour emerges from simple interactions with environment and other agents.
* Rodney Brook’s Subsumption architecture.

Subsumption Architecture

Subsumption architecture thesis-

1. Intelligent behaviour without symbolic representations as proposed by AI
2. Intelligent behaviour without syntactic symbolic reasoning as proposed by AI
3. Intelligent behaviour is an emergent property of certain complex systems

* Subsumption architecture : Task accomplishing behaviours. Situation -> Action
* Behaviour: maps percept directly to action. Agent may have multiple behaviours. Original architecture: Behaviours implemented as finite state machines.
* Multiple behaviours may fire at the same time
  + ->mechanism to choose is necessary
  + Subsumption hierarchy
* Behaviours organized in layers. Lower layer behaviours inhibit higher level ones. E.g. “Avoid obstacles” lower layer(higher priority) than “drive to goal”

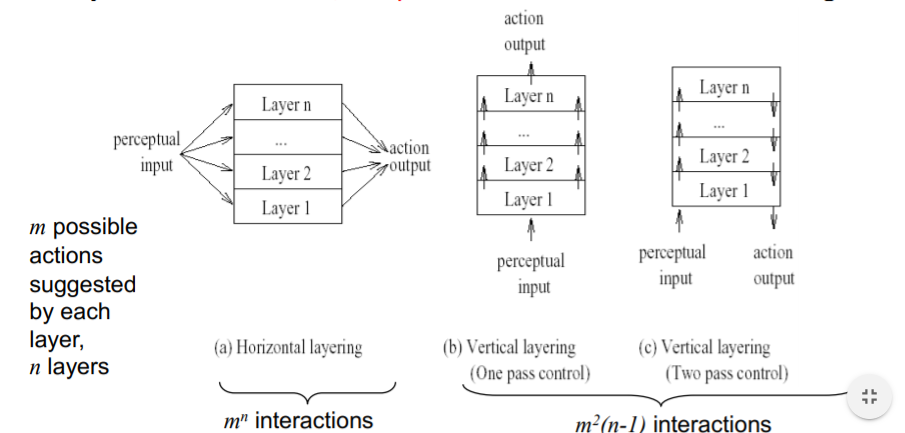


Limitations of reactive agents

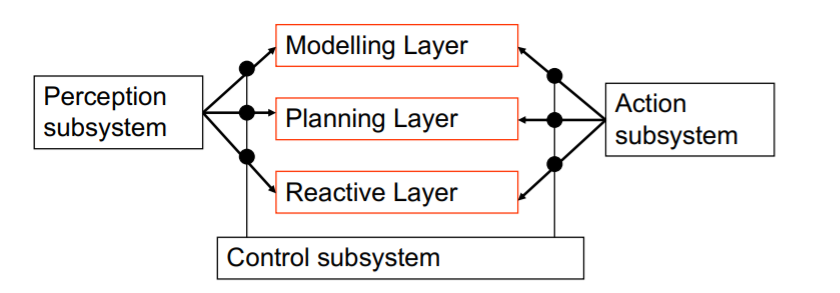
* Model of environment hard wired in rules. No symbolic representation means only (local) precepts -> can be inflexible.
* No learning
* Agents with many layers in their Subsumption hierarchy -> complex precedence interactions between layers -> hard to engineer.
* Only local decisions -> emergent global/long term behaviour which is difficult to engineer(no established methodologies; -> trial and error)

Hybrid agents

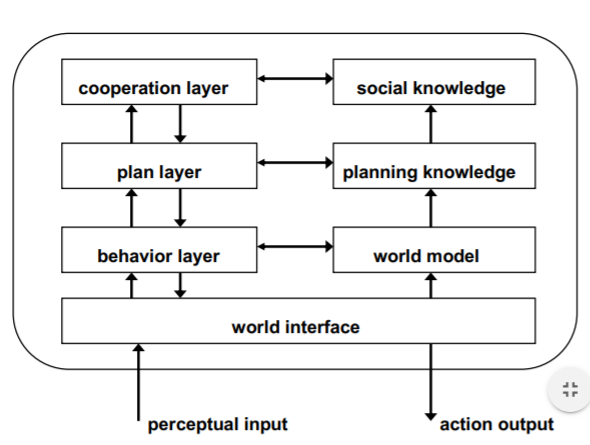
* Purely reactive agents lack proactive behaviour -> design two sub-systems: One reactive, one proactive. -> Combine them into an agent.



* All layers have input (perceptions) and output (actions)
* Horizontal layering: Behaviours of layers may conflict -> Mediator (overall control function) -> must consider m^n possible layer of interactions -> can be bottleneck
* Vertical layering: Contol flow: perceptions and action suggestions are passed up (one-pass) or up and then down (two pass). -> fewer interactions to consider
* Vertical layering: not fault tolerant: failure in one layer -> failure of whole agent.

Touring machines (Ferguson)

* Horizontal layer architecture
* Each layer continually produces suggestions for what action the agent should perform.
* Based on autonomous vehicles driving through streets.
* Planning Layer: Long term behaviour(day to day); chooses (sub) plans out of plan library
* Modelling layer: keeps and modifies environmental model(e.g. represents facts about environment (including other agents)); selects new goals for planning layer.
* Control subsystem: exercises control e.g. by suppressing information input to certain layers (censorship) in order to prevent the triggering of certain actions

InteRRap

* Vertically layered, two pass architecture.
* Behaviour layer: Similar to reactive layer in Touring Machines.
* Planning layer: Planning to reach own goals
* Cooperation layer: Planning for cooperation to reach goals
* Each layer: Represents world in own knowledge base (growing degree of abstraction) : Behaviour Knowledge Base: Raw facts about environment; Planning Knowledge Base: Plans of own actions; Cooperative Knowledge Base: (Presumed) plans of other agent’s actions.
* Bottom-Up-Activation: If lower level layer cannot handle situation à pass control to higher level layer
* Top-Down-Execution: Higher level layers make use of “facilities” provided by lower level layer

Agent communication

* What agents can do:
  + Perform communication acts.
  + Goal: influence other agents
  + E.g. to make them perform actions or to male them believe certain propositions
  + Other agent decides whether to perform action or believe proposition

Speech Act

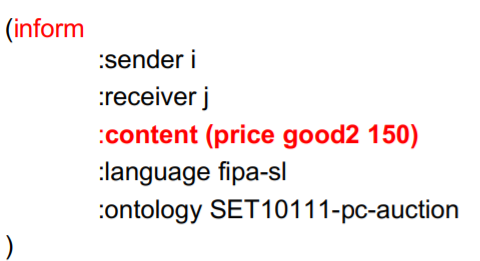
* Most treatments of communication in multi-agent systems borrow their inspiration from speech act theory.
* Speech act theories are pragmatic theories of language, i.e., theories of language use.
* Three aspects of a speech act
  + Locutionary act – making the utterance
  + Illocutionary act – the action performed in saying something
  + Perlocution – the effect of the act
* Speech acts can be seen to have two main components:
  + A performative verb (e.g. request, inform etc)
  + Propositional content (e.g. “the door is closed”)

Representing messages

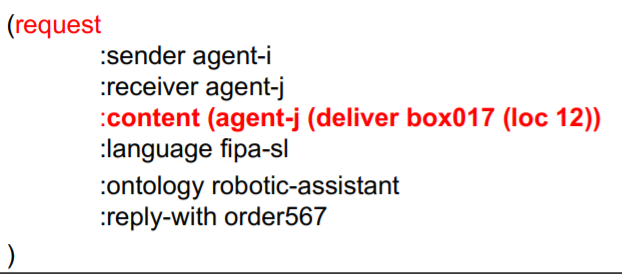
* Communication involves a combination of
  + A message envelope/template
  + The performative e.g. inform, request
  + A content language
  + An ontology that defines the concepts being referred to
* Allows agents built by different people to communicate (open multi-agent systems)
* Ideally allows the possibility for communications to be verified by third parties – but this is problematic.

Agent communication language

* An agent communication language is a standard format for the exchange of messages.
* In this module we are using the FIPA Agent Communication Language.
* This is composed of two parts:
  + The ACL that defines the message envelope and performatives (and the semantics/meaning of the performatives).
  + SL- the inner language that defines the message content.
* FIPA SL (Semantic Language)

FIPA SL Content Language

* A content expression in SL can be one of two types (int the subset used in this module):
* Predicate (a logical formula that can be assigned a truth value)
* Paired with an inform performative



* Action (something to be performed)
* Paired with a request performative

FIPA Interaction Protocols

* Patterns of messages exchange between agents
* Concurrent interaction Protocols are called conversations
* There is a basic set of predefined standard IPs
* Ad hoc IPs can be defined

Problems with the FIPA Agent Communication Language

* The semantics say nothing about the social context.
  + E.g. “I inform you that you are now husband and wife”
* The semantics are defined in terms of internal mental states
  + How can we verify an agent is complying with languages semantics?
  + A real problem in open systems
  + In an open system, any agent can join, and the agents may have different owners/ designers.

**Lecture 6 Ontologies**

* How do we represent a book we wish to order?
* Ideally want to be able to add semantic meaning to our message, e.g. that every book must have a title, that paperbacks and hardbacks are both books, that year must be an integer.
* An ontology lets us do this.

Ontologies for agents

* Each agent has to know something about the domain it is working in.
  + What is a smartphone?
  + What specifications are valid?
* An agent can only communicate about facts expressed in some ontology.
* This ontology must be agreed and understood among the agent community to enable agents to understand messages from each other.

What is an ontology?

* Used to capture knowledge about some domain of interest.
* Describes the concepts in a domain, and the relationships between those concepts
  + An Intel PC is a PC
  + Every PC has one or more CPUs
  + A pepperoni pizza is a meat pizza.
* A set of classes (concepts) in a domain, and the hierarchical relationships between them
* A set of properties(slots) belonging to each class
  + A property can have constraints on its values(type, cardinality).

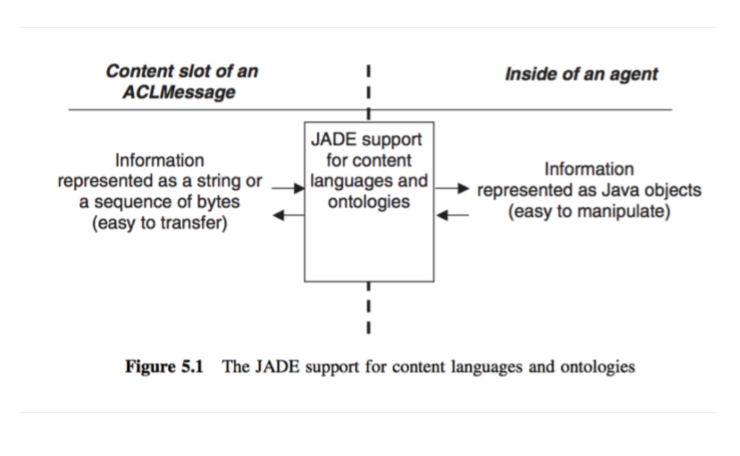
Types of ontology

* Informal (no automated reasoning)
  + Controlled vocabulary
  + Terms/glossary
  + Thesaurus
  + Informal ‘is-a’ taxonomies
* Formal
  + Formal ‘is-a’ taxonomies
  + Properties
  + Value restrictions on properties
  + Arbitrary logical constraints

Advantages of ontologies

* Allow agents written by different developers to communicate
* Integrates with JADE
* Makes domain knowledge explicit
* Allows for automated reasoning (e.g. Infer properties)

Ontologies in JADE



Designing an ontology for JADE

* Identify the predicates, agent actions and concepts.
* Predicates are valid content for an INFROM or QUERY\_IF message.
* Agent actions are valid concepts for a REQUEST.
* Concepts are not valid inside a FIPA message by themselves (only valid wrapped inside predicates or agent actions).

**Lecture 7 – Multi-agent coordination**

Open vs closed multiagent systems

* Closed system
  + All agents known at design time
  + All owned by individual(s) sharing a common goal
* Open System
  + New agents can appear at runtime
  + Owned by different individuals that may have conflicting goals

The benevolence assumption

* Important to make a distinction between:
  + benevolent agents
  + Self-interested agents
* Problem-solving in benevolent systems is cooperative distributed problem solving (CSPS)
* Benevolence simplifies the system design task enormously.

Contrast with self interested agents

* If agents represent the interests of individuals or organisations, then we cannot make the benevolence assumption.
* Agents will be assumed to act to further there own interests, possibly at expense of others.
* Complicates the design task.

Coherence

* Criteria for assessing an agent-based system.
* Coherence
  + How well the multiagent system behaves as a unit along some dimension of evaluation.
* We can measure coherence in terms of solution quality, how efficiently resources are used, conceptual clarity and so on.
* Good Coordination ensures that agents will not clobber each other’s sub-goals

Task sharing and result sharing

* How does a group of agents work together to solve a problem?
* There are three stage
  + **Problem decomposition**
  + **Sub-problem solution**
  + **Answer synthesis**

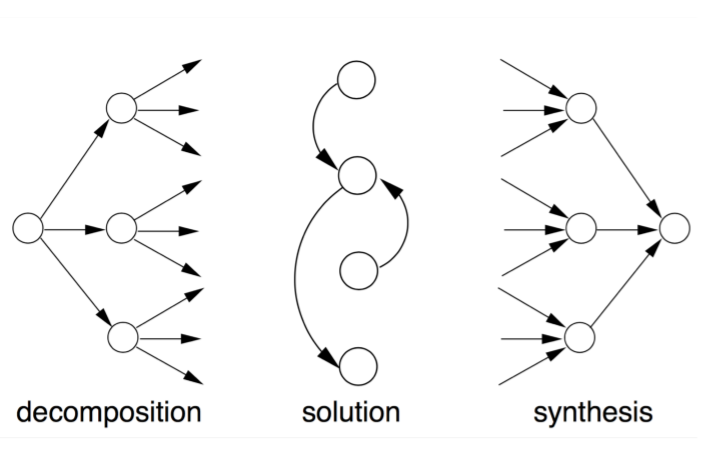
Problem decomposition

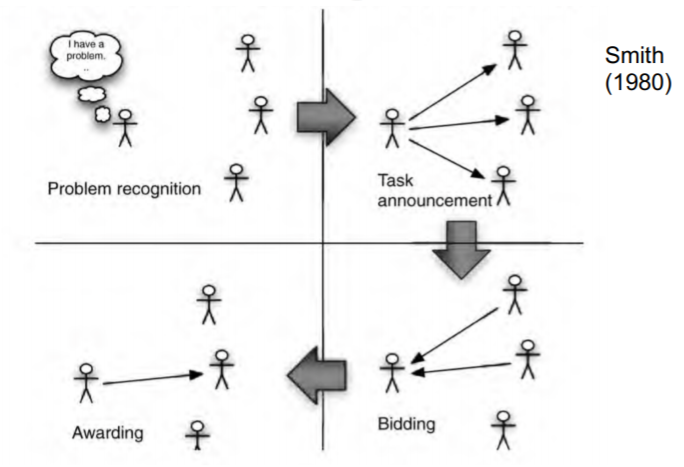
* The overall problem to be solved is divided into smaller sub problems.
* This is typically a recursive/hierarchical process. Sub problems get divided up also
  + In ACTORS, this is done until we are at the level of an individual program instructions
* Clearly requires some processing to do the division. How this is done is a design choice.

Sub-problem solution

* The sub-problems derived in the previous stage are solved.
* Agents typically share some information during this process.
* A given step may involve two agents synchronizing their actions.

Solution synthesis

* In this stage solutions to sub problems are integrated.
* Again this may be hierarchical
  + Different solutions at different levels of abstraction.

Contract Net protocol for task sharing

Recognition

* In this stage, an agent recognises it has a problem it wants help with.
* Agent has a goal and either
  + Realises it cannot achieve the goal in isolation – does not have the capability;
  + Realised it would prefer not to achieve he goal in isolation (typically because of solution quality, deadline, etc)
* As a result, it needs to involve other agents.

Announcement

* In this stage, the agent with the task sends out an announcement of the task which incuded a specification of the task to be achieved.
* Specification must encode:
  + Description of task itself(may be executable)
  + Any constraints(e.g. deadlines, quality constraints)
  + Meta-task information(e.g. bids must be submitted by…)
* The announcement is then broadcast

Bidding

* Agents that receive the announcement decide for themselves whether they wish to bid for the task.
* Factors:
  + Agent must decide whether it is capable of expediting task;
  + agent must determine quality constraints & price information (if relevant).
* If they do choose to bit they submit a tender.

Awarding and expediting

* Agent that task announcement must choose between bids & decide who to “award the contract” to.
* The result of this process is communicated to agents that submitted a bid.
* The successful contractor then expedites the task.
* May involve generating further manager-contractor relationships: sub-contracting
  + May involve another contract net recursively.

Limitations of the Contract Net

* Before sub-problems can be distributed (-> announcements can be made), problem decomposition needs to be performed (highly non-trivial)
* Communication produces overhead, is slow.
* Problems must have right granularity.
* Recognition stage (agent realises that it needs help with a problem)is not explicitly covered

Result sharing

* In results sharing, agents provide each other with information as they work towards a solution.
* It is generally accepted that results sharing improves problem solving by:
  + Confidence
  + Completeness
  + Precision
  + Timeliness

Result sharing in blackboard systems

* The first scheme for cooperative problem solving: was the blackboard system.
* Results shared via shared data structure (BB).
* Multiple agents can read and write to BB.
* Agents write partial solutions to BB.
* BB may be structure into a hierarchy.
* Mutual exclusion over BB required => bottleneck.
* Not concurrent activity.

Result sharing in Subscribe/Notify pattern

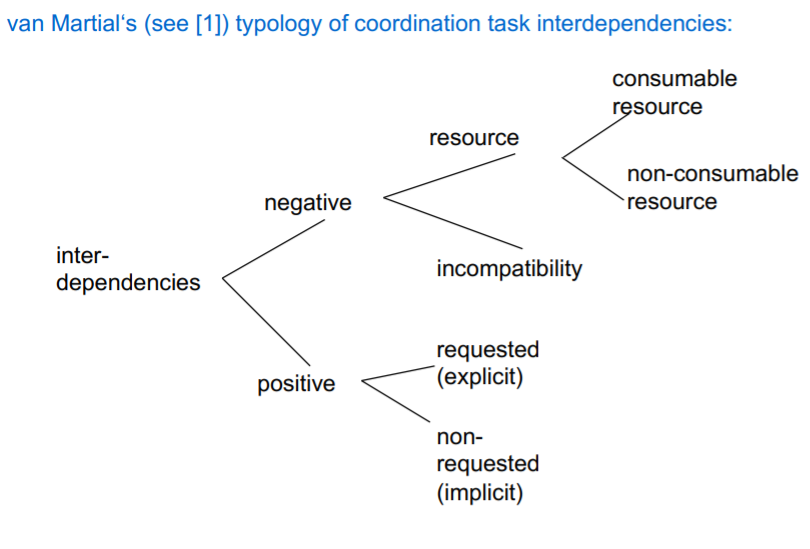
* Common design pattern in OO.
* An object subscribes to another object, saying “tell me when event e happens”.
* When event e happens, original object is notified.
* Information pro-actively shared between objects.
* Objects required to know about the interests of other objects => inform object when relevant information arises.

Handling inconsistency

* A group of agents may have inconsistencies in their:
  + Beliefs
  + Goals or intentions
* To handle this we could
  + Not allow it
  + Resolve inconsistency
  + Build systems that degrade gracefully in the face of inconsistency.

Coordination and task interdependencies

* Coordination is managing dependencies between agents.



Types of interdependencies

* Three types of non-requested interdependencies:
  + Action-equality-interdependence: both agents need to have action a done -> one of them can do it
* Consequence-interdependence: Actions of one agent’s plan have side effect of achieving other agent’s goal.
* Favour-Interdependence: Actions of one agents plan have side effect of partially achieving other agents goal (positively contributing to it)

Coordination by mutual modelling

* Having model of other agents (beliefs, intentions, goals) and of cooperative conventions enables coordination without communication.
* Example: Younger and older person approach door -> resource conflict-> possible solution: both wait -> waste of resource -> knowing usual convention and having a model of each other (“other agent is conservatively polite”) ->older person will go first
* Model could be induced by game-theoretic payoff-matrix -> agents will know rational action

Coordination by social norms and laws

* In human multi-agent systems we have norms and laws for coordination.
* In software multi-agent systems, norms and laws can be implemented.
  + At design (offline design) (easier, more straightforward, more direct control)
  + As an emergent behaviour (more flexible in unforeseeable environments, producing possibly better coherence)

Candidate strategies

* Simple majority: change from strategy s1 to s2 if more agents with s2 were seen.
* Simple majority with agent types (wrt. Strategy) and communication (exchange of complete memory with agents of same type)(-> broadening their statistical basis for majority decision)
* Simple majority with communication on success: when agent has reached certain success-level with current strategy -> communicate memory related to successful strategies to other agents (-> only successful strategies are communicated)
* Highest cumulative reward: Record payoff of current strategy; choose strategy with highest cumulative payoff

Evaluating strategy performance

* Efficiency of strategy update function measured e.g. by convergence time in t-shirt game.
* Experiments: all strategies were successful
* How could we get agents to move to a different convention?

Multi-agent planning

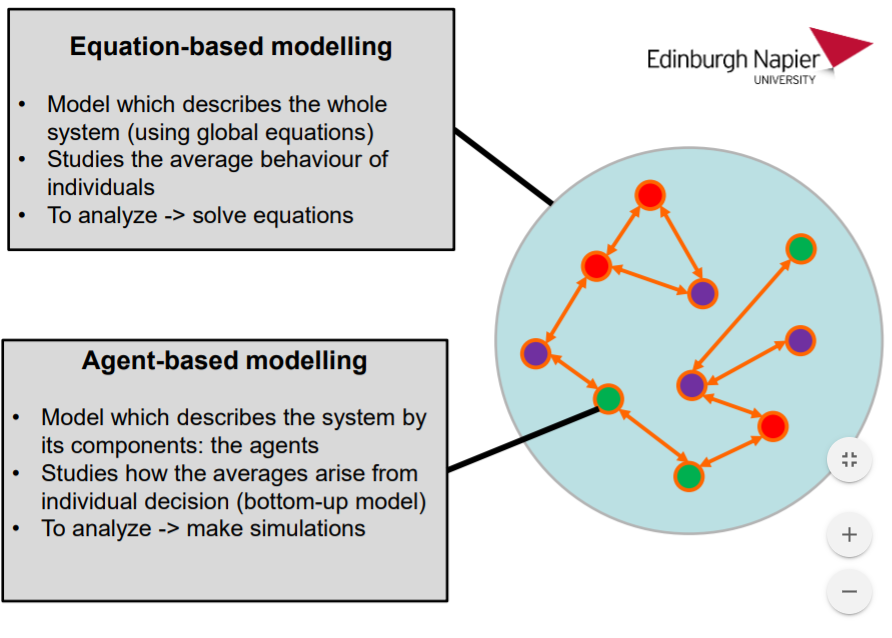
* Centralised planning by one agent
* Distributed planning: multiple agents work together to form one plan.
* Distributed planning for distributed plans: multiple agents cooperate to form multiple plans
  + What to do if agent are self-interested

**Lecture 8 Agent based modelling**

Why do we need modelling?

* Models are simplified versions of real-world system.
* Because humans are bad at understanding complex systems
  + Science against common sense
* Models to understand are different from models to predict
  + Aim for simplicity

Equation and agent-based modelling



Agent-based models

* Use the simulation to gain an understanding of what happens in the real world.
  + Prediction vs understanding (qualitive vs quantitative)
* Produce a set of agents that interact to model a real-world problem
  + Buyers and sellers in an auction
  + Shoppers in a supermarket
  + Vehicles in a city
  + Individuals in a crowd
* Model explicitly the agents
  + The properties of the agents
  + The interactions rules/decision-making
* Agent behaviours might change through time, by learning or evolution

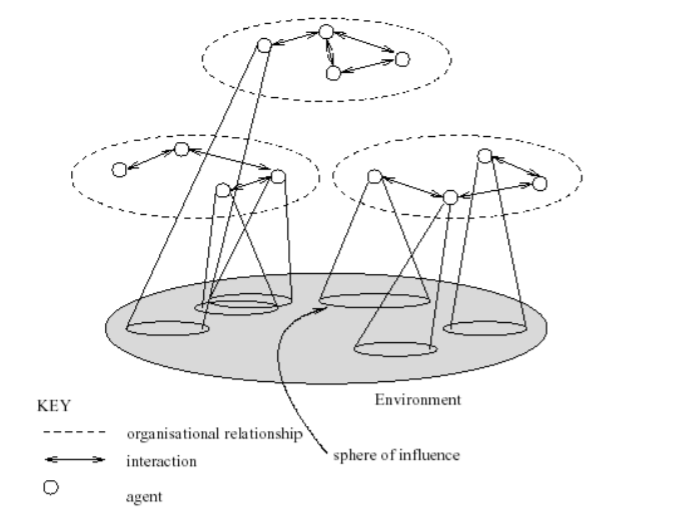
Advantages of agent-based models

* Can model systems that equation-based models cannot
* Provides a natural way to build a model (bottom-up from thinking about individual behaviour rather than global properties)
  + Easier to produce by domain experts.
* Easy to graphically present the results to aid intuitive understanding.

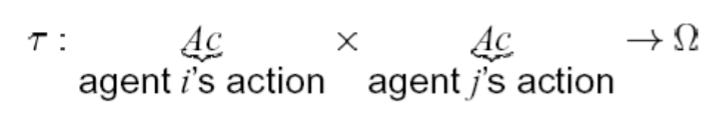
Disadvantages of agent-based models

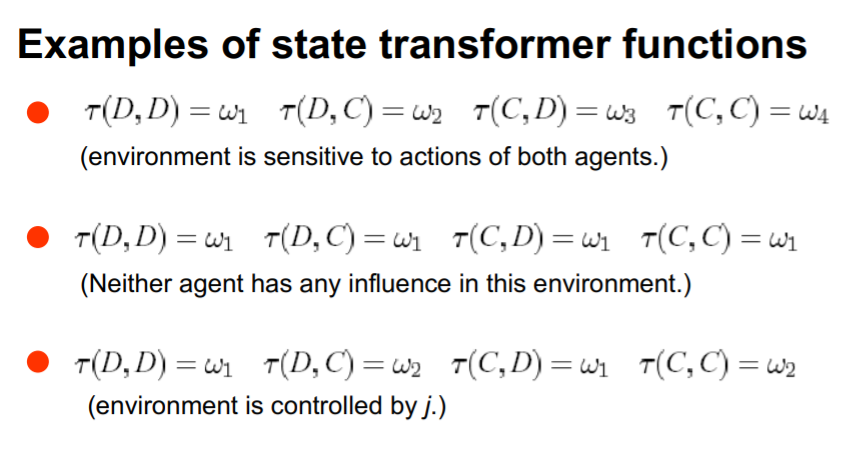
* Computationally expensive.
* Can produce non-intuitive results, but not always easy to understand why.
* Need to avoid temptation to try and model every aspect of the real-world system.

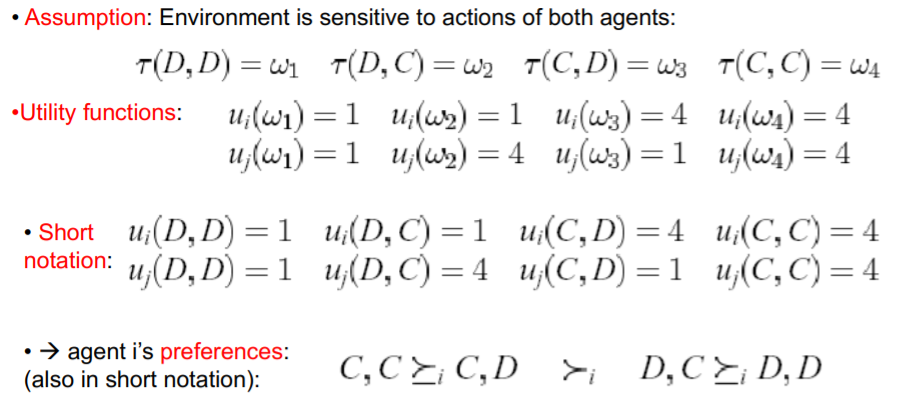
**Lecture 9 – Game Theory**

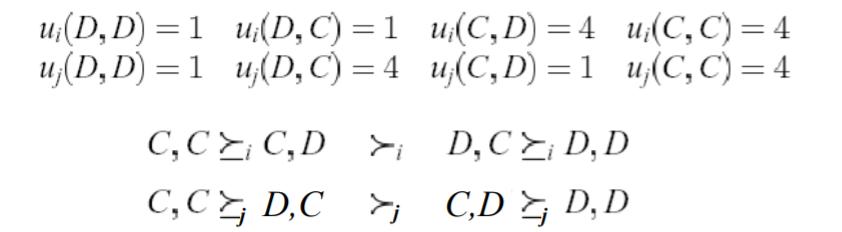
Agent societies

Multi-agent encounters

* Agents simultaneously choose action to perform -> result of the actions they select -> outcome in ohms.
* Actual outcome depends on the combination of actions
* Assume: each agent has just two possible actions C(“cooperate”) and D(“defect”).
* Environment behaviour given by state transformer function:



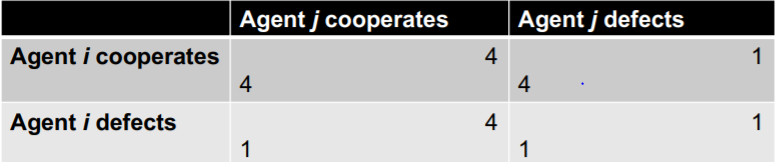
Rational behaviour

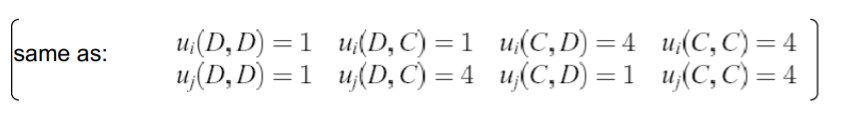


* “C” is the rational choice for i. (Because I (strongly) prefers all outcomes that arise through C over all outcomes that arise through D.)
* “C” is the rational choice for j. (Because J (Strongly) prefers all outcomes that arise through c over all outcomes that arise through D.)

Payoff matrices

* Game theory: characterize the previous scenario in a payoff matrix:





Solution concepts

* How will a rational agent behave in any given scenario?
* Answered in solution concepts:
  + Dominant strategy
  + Nash equilibrium strategy
  + Pareto optimal strategy
  + Strategies that maximise social welfare

Dominant strategies

* A strategy si is dominant for player I if no matter what strategy sj agent j chooses, I will do at least as well playing si as it would playing anything else.
* Unfortunately, there isn’t always a dominant strategy.

Nash Equilibrium

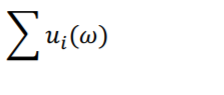
* In general, we will say that two strategies s1 and s2 are in Nash equilibrium if:
  + Under the assumption that agent I plays s1, agent j can do no better than play s2; and
  + Under the assumption that agent j plays s2, agent I can do no better than
* Neither agent has any incentive to deviate from a nash equilibrium
* Unfortunately:
  + Not every interaction has a (pure strategy) Nash equilibrium
  + Some interactions have more than one Nash equilibrium

Paeto optimality

* An outcome is said to be Pareto optimal (or Pareto efficient) if there is no other outcome that makes one agent better off without making another agent worse off.
* If an outcome is pareto optimal, then at least one agent will be reluctant to move away from it (because this agent will get worse off).
* If an outcome w is not Pareto optimal, then there is another outcome w1 that makes everyone as happy, if not happier than w.

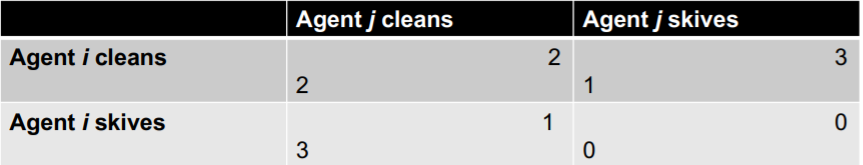
Social Welfare

* The social welfare of an outcome w is the sum of the utilities that each agent gets from w:



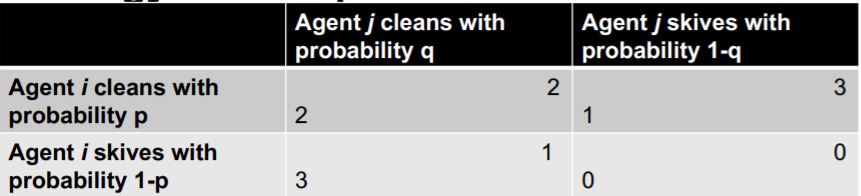
* Think of it as the “total amount of money in the system”.
* As a solution concept, may be appropriate when the whole system (all agents) has a single owner (then overall benefit of the system is important, not individuals).

The kitchen stand-off game



* Two (pure) Nash Equilibria: (skive, clean), (clean, skive)
  + Assuming the other skives you can do no better than clean
  + Assuming the other cleans you can do no better than skive
* There is no dominant strategy
* All outcomes except (skive, skive) are pareto optimal
* All outcomes except (skive, skive) maximise social welfare.

The kitchen stand-off game: Mixed strategy Nash Equilibrium

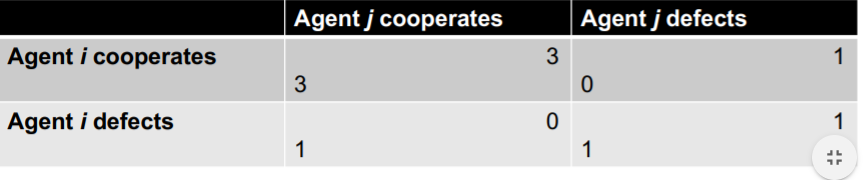


* What if agent I plays clean with probability, agent j with probability q?
* Expected payoffs for agent I given agent j’s strategy:
  + Agent I chooses clean: 2q +(1-q)1
  + Agent I chooses skive: 3q + (1-q)0
* At Nash Equilibrium the payoffs to cleaning and skiving for agent I must be equal given j’s strategy (otherwise I could benefit from increasing the probability of doing one of them):
* 2q + (1-q)1 = 3q + (1-q)0
* 2q + 1-q = 3q
* 3q – 2q +q = 1
* 2q = 1 • q = ½
* By the same reasoning p=½.
* Note social welfare is lower in the mixed strategy equilibrium.

Pure coordination game (t-shirt game)

* Two agents (human or artificial) do better if they use the same strategy.
* Both use VHS or Betamax.
* Both drive on the left.
* Both use Windows or Linux.
* Both follow the same social norm.
* Two Nash Equilibria (red, red) (blue, blue)
* There is no dominant strategy
* (red, red) and (blue, blue) are pareto optimal
* (red,red) and (blue, blue) maximise social welfare.

Stag-hunt game (risky coordination game)

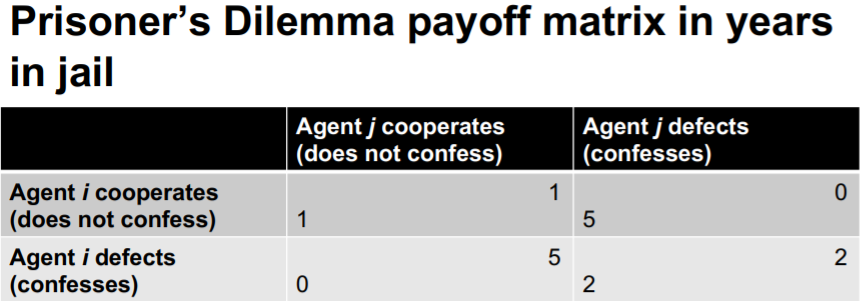
* You and a friend are thinking of dressing up as Batman and Robin to lectures.
* If both of you do it – result!
* If only one of you does it, that one gets laughed at alone and suffers a negative payoff.
*  If neither of you do it your payoff remains unchanged
* Two Nash equilibria: (cooperate, cooperate), (defect, defect)
* (defect, defect) is risk-dominant – it has the largest basin of attraction (is less risky).
* (cooperate, cooperate) is Pareto optimal.
* (cooperate, cooperate) maximises social welfare.

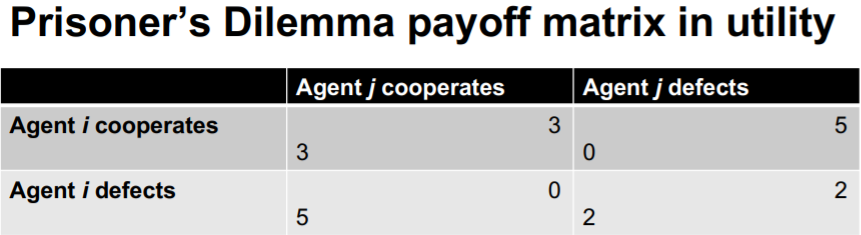
Stag-Hunt mixed strategies

* Let I play cooperate with probability p, and j play cooperate with probability q.
* If I plays cooperate they get 3q
* If I plays defect they get q + (1-q)
* At Nash equilibrium 3q = q +(1-q)
* So q = 1/3
* Likewise for p.

**Lecture 9 – The prisoners’ dilemma and repeated games**

The Prisoner Dilemma

* You and a “partner” are collectively charged with a crime and held in separate cells, with no way of meeting or communicating.
* You are told by the police that:
  + if you confess and the other does not, then you will be freed, and the other will be jailed for five years;
  + if you both confess, then each of you will be jailed for two years.
* Both of you know that if neither confesses, then you will be jailed for one year.
* What do you do?
* NB cooperation is cooperating with the other suspect by not confessing.



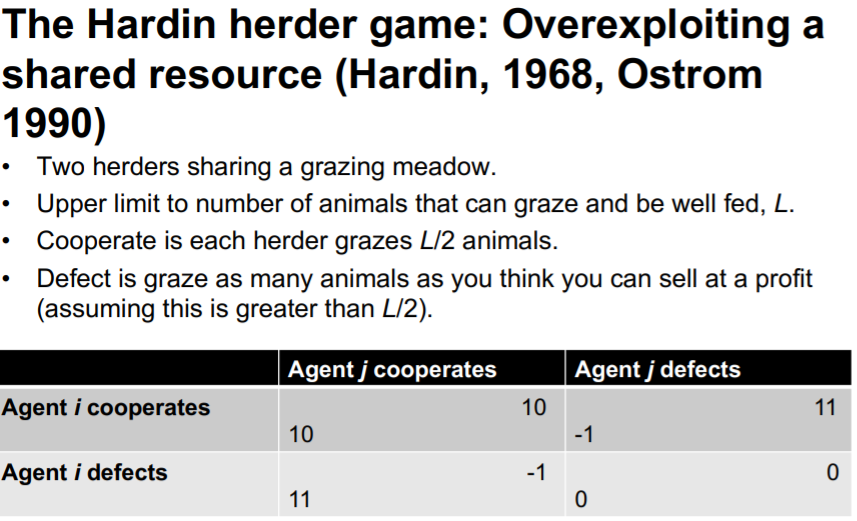
* Utility is a decreasing function of yours spent in jail.
* D is a dominant strategy.
* (D,D) is the only Nash equilibrium.
* All outcomes except (D,D) are Pareto optimal.
* (C,C) maximises social welfare.

Dominant strategies

* A strategy si is dominant for player I if no matter what strategy sj agent j chooses, I will do at least as well playing si as It would be playing anything else.
* Unfortunately sometimes there isn’t always a dominant strategy.

Rational behaviour

* Only one Nash Equilibrium (D,D) (“Under the assumption that the other does D, one can do no better than do D”).
* Intuition says: (C,C) is better than (D,D) so why not (C,C)? -> but if agent assumes that other agent does c it is BEST to do D! -> seemingly “waste of utility”.
  + (D,D) is Pareto inferior to (C,C)
* “Shocking” truth: defect is rational, cooperate is irrational
* And the outcome that maximises social welfare is not a Nash equilibrium.

The Hardin herder Game: Overexploiting a shared resource

* “Defect more rational than cooperate” -> Humans: Machiavellism (opposed to real altruism)
* Philosophical question: isn’t even altruism ultimately some kind of optimisation towards OWN goals?

How to engineer cooperation?

* Could allow the agents to communicate beforehand to make an agreement
* Buy how to enforce the agreement?
* Enforcement by third party?
  + But how is this enforcement paid for?
* Multiple sequential games -> “The shadow of the future”.

The shadow of the future: Iterated Prisoners Dilemma

* Game is played multiple times. Agents can see past actions of other agent.
* Agents care about future payofs, and hence the (discounted) sum of payoffs received over t rounds.
* Course of reasoning
  + If I defect, the other agent can punish me by defecting in the next round(cannot happen in the single shot prisoners dilemma game)
* -> in an iterated Prisoners Dilemma game: cooperation is rational (under certain conditions)

The folk Theorem of game theory

* In a game repeated for an indefinite number of times, with information about the past actions of other agents, cooperation will be Nash equilibrium through conditional strategies.
  + “Cooperate if my partner cooperated last time, otherwise defect”
* Technically, any strategy gives more than the minimax payoff will be an equilibrium.
* Doesn’t work if:
  + End time known
  + Insufficient information about past actions of other agents.