Multi-agent system - COMP310

Notes for preparing MAS exams.

Chapter2: Intelligent Agents

- **Predicate Task Specifications**: specification that assigns Boolean utilities to agent runs, depending on whether the agent succeeded in its task for a run (i.e. the utility is or if it failed (i.e. the utility is 0).
- Task Environments: a pair of an environment and a Predicate Task Specification
- Tasks:
    - achievement tasks: The agent succeeds in an achievement task if it can force the environment into one of the goal states
    - maintenance goal: the agent succeeds in a maintenance task if it ensures that it is never forced into one of the fail states
- **An agent** is a computer system that is situated in some environment, and that is capable of **autonomous** action in this environment in order to meet its delegated objectives
- Properties of Environments
- Fully observable vs. partially observable
    - **Fully observable**: the agent can obtain complete, accurate, up-to-date information about the environment's state
    - Most moderately complex environments are partially observable
- Deterministic vs. non-deterministic
    - **Deterministic**: the environment in which any action has a single guaranteed effect – there is no uncertainty about the state that will result from performing an action
    - Environments **stochastic**: the non-determinism using probability theory
- Static vs. dynamic
    - **Static**: the environment that can be assumed to remain unchanged except by the performance of actions by the agent.
    - A **dynamic** environment is one that has other processes operating on it, and which hence changes in ways beyond the agent's control
- Discrete vs. continuous
    - **discrete**: there are a fixed, finite number of actions and precepts in it
- Episodic vs. non-episodic
    - **Episodic**:
        - 1. Each state is independent of each other.
          2. An agent's current action will not affect a future action
          3. The performance of an agent is dependent on a number of discrete episodes
          4. agent can decide what action to perform based only on the current episode
    - **non-episodic** or **sequential**
        - The current decision affects future decisions

- **Real time**: time plays a crucial part in evaluating an agent's performance
- Intelligent Agents
- exhibiting three types of behaviour:
    - **Reactive** (environment aware): maintains an ongoing interaction
    - **Pro-active** (goal-driven): generating and attempting to achieve goals
    - **Social ability**: the ability to interact with other agents via **cooperation**, **coordination**, **negotiation**
- Cooperation: working together as a team to achieve a shared goal
- Coordination: managing inter-dependencies between the activities of agents
- Negotiation: reach agreements on matters of common interest
- **intentional stance**: humans use different strategies or stances to explain and predict other entities' behaviour, **intentional stance** is **an abstraction tool**
- **intentional system**: whose behaviour can be predicted by the method of attributing belief, desires and rational acumen
    - A **first-order** intentional system has beliefs and desires (etc.) but no beliefs and desires about beliefs and desires
    - A **second-order** intentional system is more sophisticated; it has beliefs and desires (and no doubt other intentional states) about beliefs and desires (and other intentional states) – both those of others and its own
- To be remembered:

  The intentional stance description is useful when it helps us understand the structure of the machine, its past or future behaviour, or how to repair or improve it. It is useful when it helps us explain and predict a complex system's behaviour without having to understand how the mechanism actually works.
  The main reason that the intentional stance description is useful is that it is a powerful abstraction tool. It allows us to abstract what is happening at the low level within a machine. So instead of describing knowledge a machine has as data stored within a data-structure, or concerning ourselves with the format, we can simply think of that knowledge as a set of beliefs.
- A system is a **pair** containing **an agent** and **an environment**
- Model an agent as a function which maps runs: $Ag : R_E \rightarrow Ac$
- runs of agent: $R(Ag, Env)$
- **behaviourally equivalent** iff: $R(Ag_1, Env)$= $R(Ag_2, Env)$
- **purely reactive**: They base their decision making entirely on the present $Ag : E \rightarrow Ac$
- perception function: $see : E \rightarrow Per$
- action-selection function $action$: $action : I \rightarrow Ac$
- next state function: $next : I \times Per \rightarrow I$
- task specification: $u : E \rightarrow \mathbb{R}$
- achievement tasks vs maintenance tasks
    - **achievement tasks**: specified by a set of "good" or "goal"
    - **maintenance goal** : specified by a set of "bad" states

Chapter3: Deductive Reasoning Agents

- **Symbolic Reasoning Agents**: 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
- Issues of Symbolic Reasoning Agents:
  - transduction problem: translating the real world into an accurate adequate useful symbolic description
  - representation / reasoning problem: how to symbolically represent and process. How to get agent reason with the information
    - In general, many (most) research-based symbol manipulation algorithms of interest are **highly intractable**
    - Hard to find compact **representations**
- **Deductive Reasoning Agents** : use logic to encode a theory stating the best action to perform in any given situation

Δ be a logical database that describes the current internal state of an agent;

- A set of formulae of *first-order predicate logic*

ρ be the *theory* (typically a set of *deduction rules*);

- The deduction rules have the form: $\psi(\ldots) \rightarrow \phi(\ldots)$

$$In(0,0) \wedge Facing(north) \wedge \neg Dirt(0,0) \quad \longrightarrow \quad Do(forward)$$

Ac be the set of actions the agent can perform;

$\Delta \vdash_\rho \varphi$ means that φ can be proved from Δ using ρ.

- **Agent-oriented programming**: Directly programming agents in terms of **intentional notions**
- (Not in Exam)An **AGENT0 program** consists of two parts: **initialisation** and **commitment rules**, AGENT0 has 4 components:
  - A set of **capabilities**
  - A set of **initial beliefs**
  - A set of **initial commitments**
  - A set of **commitment rules**, Each commitment rule contains: a **message condition**, a **mental condition**, an **action**
- (Not in Exam) **Her Planning Communicating Agents (PLACA)** solves The inability of agents to plan, and communicate requests for action via high-level goals for AGENT0.   **Mental states** are expanded to include **plans** and **intentions**

Chapter4: Practical Reasoning Agents

- **Practical reasoning** is reasoning directed towards **actions** - the process of figuring out what to do
- **Theoretical reasoning** is directed towards **beliefs**
- The Components of Practical Reasoning:
  - **Deliberation**: Deciding what state of affairs we want to achieve, output **intentions**

- **Means-ends reasoning**: Deciding how to achieve these states of affairs, output **plans**

- **Practical Reasoning**
  - Intentions:
    - pose problems for agents
    - provide a "filter" for adopting other intentions
  - Agents:
    - track the success of their intentions, and are inclined to try again if their attempts fail
    - believe their intentions are possible
    - do not believe they will not bring about their intentions in **normal case**
    - need not intend all the expected side effects of their intentions
- Example Learning in Chapter 4 p15 -19 , p24-p26
- Formal representation:
  - **A action** $\alpha_i \in Ac : \alpha_i = \langle P_{\alpha_i}, D_{\alpha_i}, A_{\alpha_i} \rangle$, where is preconditions, D_{α_i} is delete list, A_{α_i} is add list
  - **A plan** is just a sequence of actions from Ac: $\pi = (\alpha_1, \ldots, \alpha_n)$
  - **A planning problem**: $\langle B_0, Ac, I \rangle$, where B_0 is the set of beliefs, Ac is the set of actions, I is the goal(or intention)
  - **A plan for a given planning problem**: $B_0 \xrightarrow{\alpha_1} B_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} B_n$
- **Deliberate function**:
  - **Option generation**: generates a set of possible alternatives
    $options : \mathscr{P}(Bel) \times \mathscr{P}(Int) \rightarrow \mathscr{P}(Des)$
  - **Filtering**: chooses between competing alternatives
    $options : \mathscr{P}(Bel) \times \mathscr{P}(Des) \times \mathscr{P}(Int) \rightarrow \mathscr{P}(Int)$
- Degrees of Commitment:

- **Blind commitment**: continue to maintain an intention **until** it believes that the intention has actually been achieved

  Agent Control Loop Version 3

  ```
  B := B0;
  I := I0;
  while true do
        get next percept ρ;
        B := brf(B, ρ);
        D := options(B, I);
        I := filter(B, D, I);
        π := plan(B, I, Ac);
        execute(π)
  end while
  ```

- **Single-minded commitment**: 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.

  $hd(\pi)$ returns the first action in the plan

  $tail(\pi)$ returns the last action in the plan

  $sound(\pi, I, B)$ means that π is a correct plan for I given B

```
Agent Control Loop Version 5

1.  B := B0;
2.  I := I0;
3.  while true do
4.      get next percept ρ;
5.      B := brf(B, ρ);
6.      D := options(B, I);
7.      I := filter(B, D, I);
8.      π := plan(B, I, Ac);
9.      while not(empty(π)
                or succeeded(I, B)
                or impossible(I, B)) do
10.         α := hd(π);
11.         execute(α);
12.         π := tail(π);
13.         get next percept ρ;
14.         B := brf(B, ρ);
15.         if not sound(π, I, B) then
16.             π := plan(B, I, Ac);
17.         end-if
18.     end-while
19. end-while
```

- **Open-minded commitment**: maintain an intention as long as it is still **believed possible**. Example with reconsideration

```
Agent Control Loop Version 7

B := B0;
I := I0;
while true do
    get next percept ρ;
    B := brf(B, ρ);
    D := options(B, I);
    I := filter(B, D, I);
    π := plan(B, I, Ac);
    while not(empty(π)
            or succeeded(I, B)
            or impossible(I, B)) do
        α := hd(π);
        execute(α);
        π := tail(π);
        get next percept ρ;
        B := brf(B, ρ);
        if reconsider(I, B) then
            D := options(B, I);
            I := filter(B, D, I);
        end-if
        if not sound(π, I, B) then
            π := plan(B, I, Ac);
        end-if
    end-while
end-while
```

- An agent has commitment both to:
    - **Ends**: wished state
    - **Means**: the mechanism to wished state
- Intention Reconsideration fn:

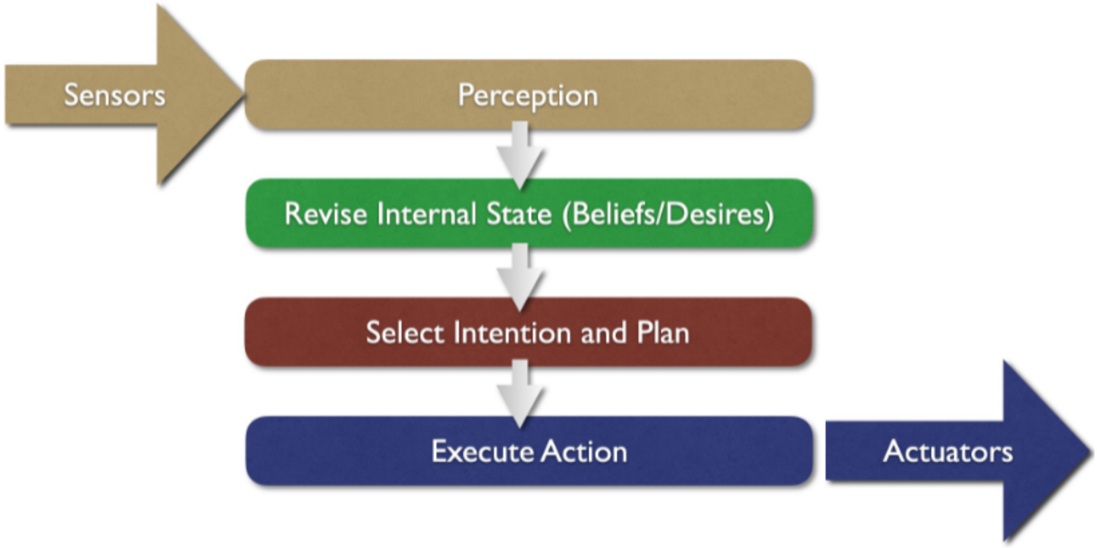
Situation number	Chose to deliberate?	Changed intentions?	Would have changed intentions?	$reconsider(\ldots)$ optimal?
1	No	—	No	Yes
2	No	—	Yes	No
3	Yes	No	—	No
4	Yes	Yes	—	Yes

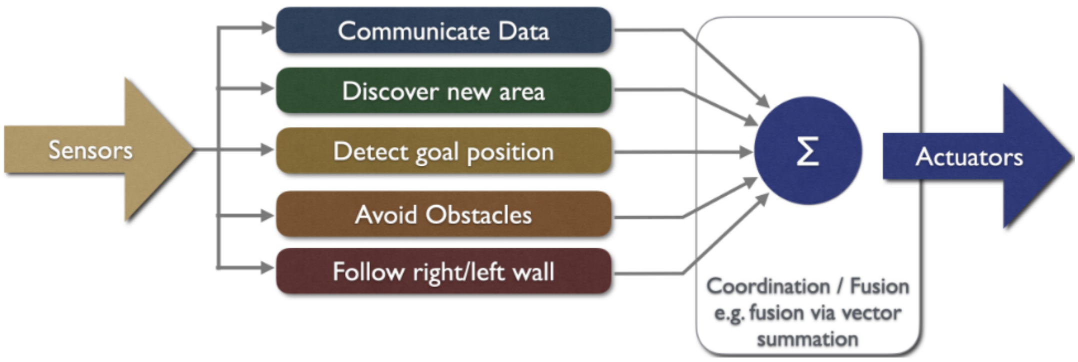
- Two different types of reconsideration strategy:
    - **Bold** agents: never pause to reconsider intentions before their current plan fully executed
    - **Cautious** agents: stop to reconsider after every action
- **Dynamism**: the environment is represented by the rate of world change,. γ , Bold agents are better when low γ, cautious agents are better when high γ

Chapter5: Reactive and Hybrid Agents

Reactive Systems

- Agent Control Loop as Layers

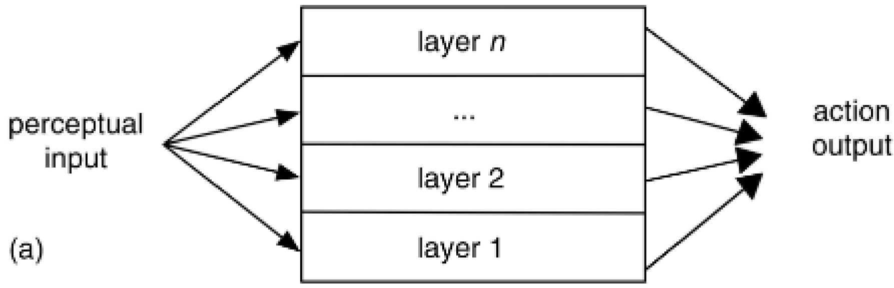


- Behaviours:
    - Complex behaviour emerges from simple components.;
    - layer is independent;
    - Can then assemble them into a complete system
    - 

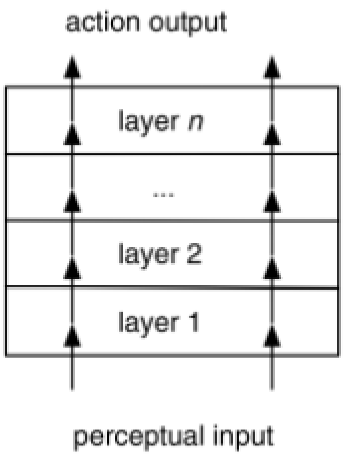
- Brooks Behavioural Languages:
    - Situatedness and embodiment
    - Intelligence and emergence
- Two defining characteristics of **Subsumption Architecture**:
    - An agent's decision-making process is realised through a set of **task-accomplishing behaviours**
    - Many behaviours can 'fire' **simultaneously**.
- Limitations of Reactive Systems:
    - the need of sufficient information of the local environment for determining actions
    - As actions are based on local information, such agents inherently take a "short-term" view.
    - Emergent behaviour is very hard to engineer or validate
    - models using many layers are inherently complex and difficult to understand.

Hybrid Architectures

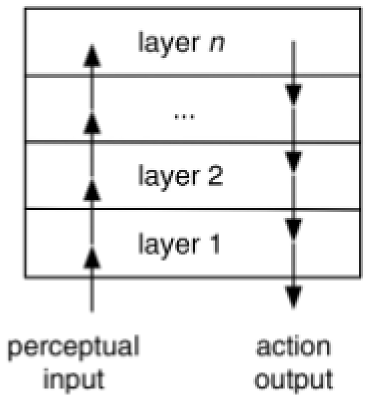
- **deliberative** Agents, containing a symbolic world model, which develops plans and makes decisions in the way proposed by symbolic AI
- **reactive** Agents, which can react to events without complex reasoning, they base their decisions on pre-defined rules.
- **Hybrid agents** try to combine the speed of reactive agents with the power of deliberative agents Hybrid Architectures:
    - Horizontal layering:
        - Layers are each directly connected to the sensory input and action output
        - each layer itself acts like an agent, producing suggestions as to what action to perform.



    - Vertical layering:
        - Sensory input and action output are each dealt with by at most one layer each

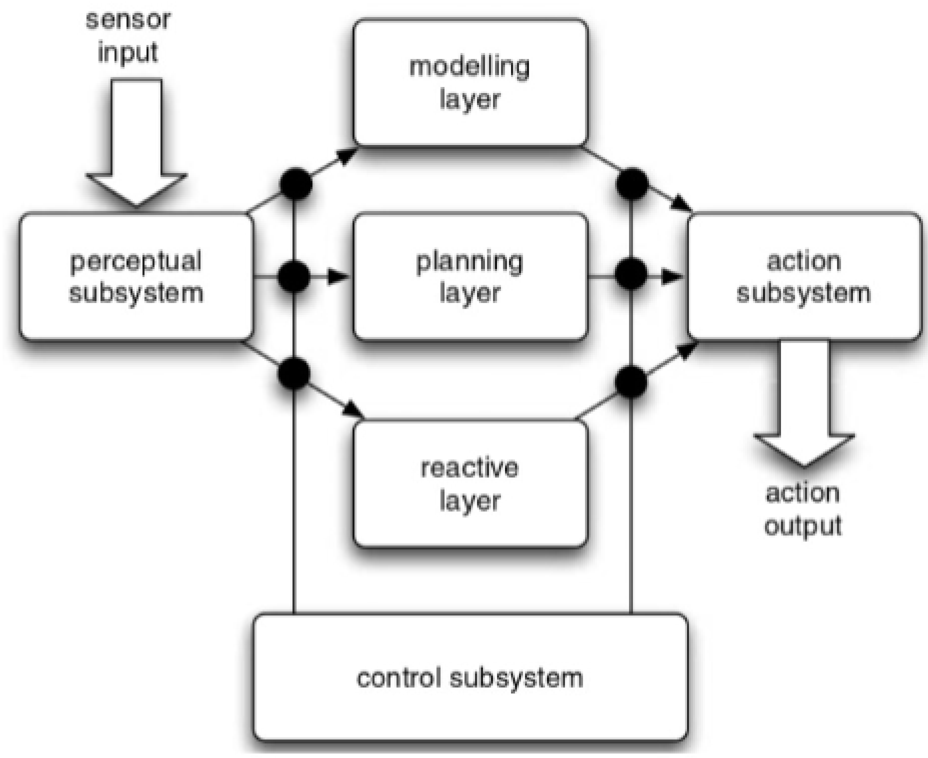


(b) Vertical layering (One pass control)

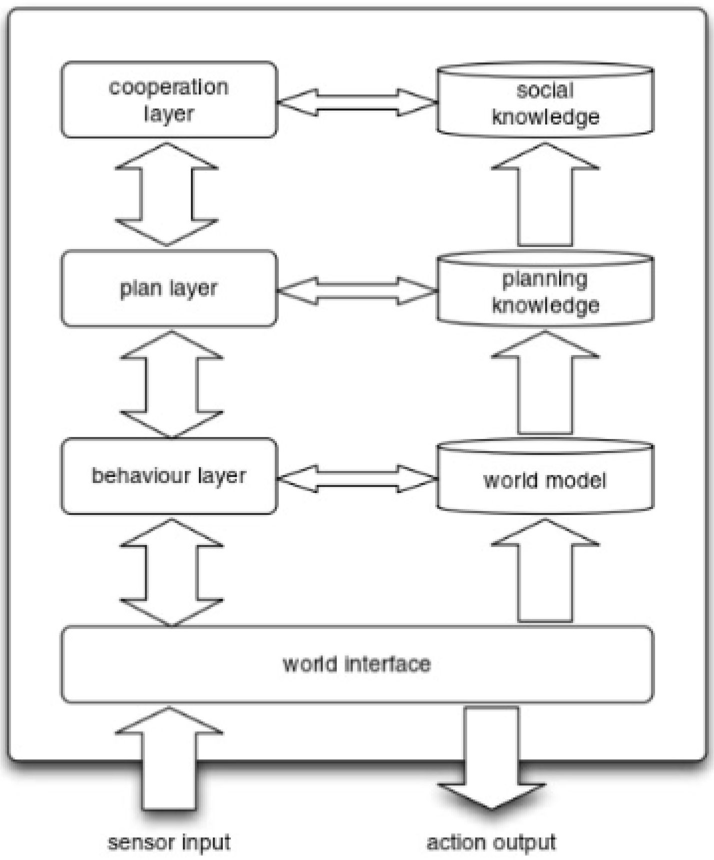


(c) Vertical layering (Two pass control)

- **TouringMachines**:
    - **reactive layer** is implemented as a set of situation-action rules
    - **planning layer** achieves the agent’s proactive behaviour
    - **modelling layer** contains symbolic representations of the ‘cognitive state’ of other entities in the agent’s environment

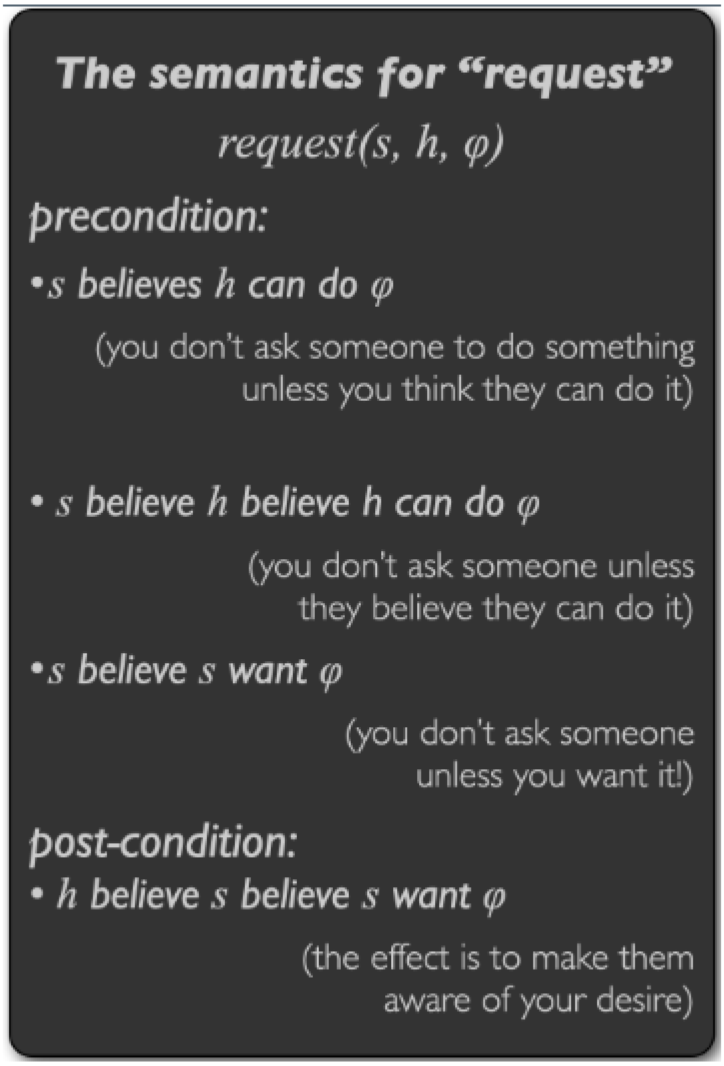


- **InteRRaP**: a vertically layered two-pass agent architecture
  - Three control layers: behaviour layer, plan layer and cooperation layer
  - Each layer is associated with a knowledge base
  - Bottom-up activation: a lower layer passes control to a higher layer since it is not competent to deal with the current situation
  - Top-down execution: higher layer makes use of the facilities provided by a lower layer to achieve one of its goals



Chapter6&7: Ontologies & Communication

- **Speech act theory**: A theory of how utterances are used to achieve intentions
  - Representatives: are used to inform an agent about a fact, e.g., 'It is raining'
  - Directives: are used to ask an agent to do something, e.g., 'please make the tea'
  - Commisives: are used to notify an agent (i.e. the hearer) that the communicating agent (i.e. the speaker) is committed to doing something, e.g., 'I promise to…'
  - Expressives: are used when an agent expresses a mental state, e.g., 'thank you!'
  - Declarations: are used to make a statement which could have implications, e.g. declaring war
- **Semantics**: **precondition-delete-add** list formalism of planning research



- **Agent Communication Languages (ACLs)**: standard formats for the exchange of messages
- **Knowledge Sharing Effort (KSE)** 's method:

- The message itself: **Knowledge Query and Manipulation Language (KQML)**, that defines various acceptable ‘communicative verbs’, or **performatives**

  An example KQML message:

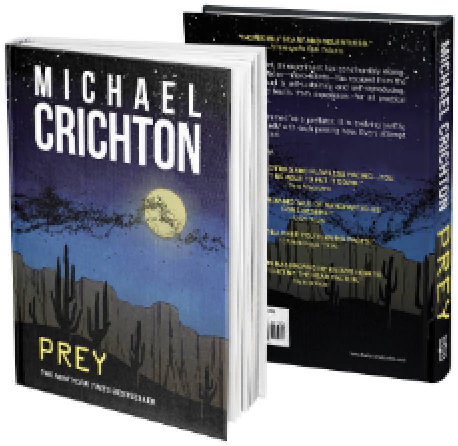
  ```
  (ask-one
      : content   (PRICE  IBM  ?price)
      : receiver   stock-server
      : language   LPROLOG
      : ontology NYSE-TICKS
  )
  ```

- The body of the message: **Knowledge Interchange Format (KIF)**, a language for expressing **message content**, or **domain knowledge**

- The role of an **ontology** is to fix the meaning of the terms used by agents. An **ontology** is a formal definition of a body of knowledge

Alice:

- *Did you read "Prey"?*

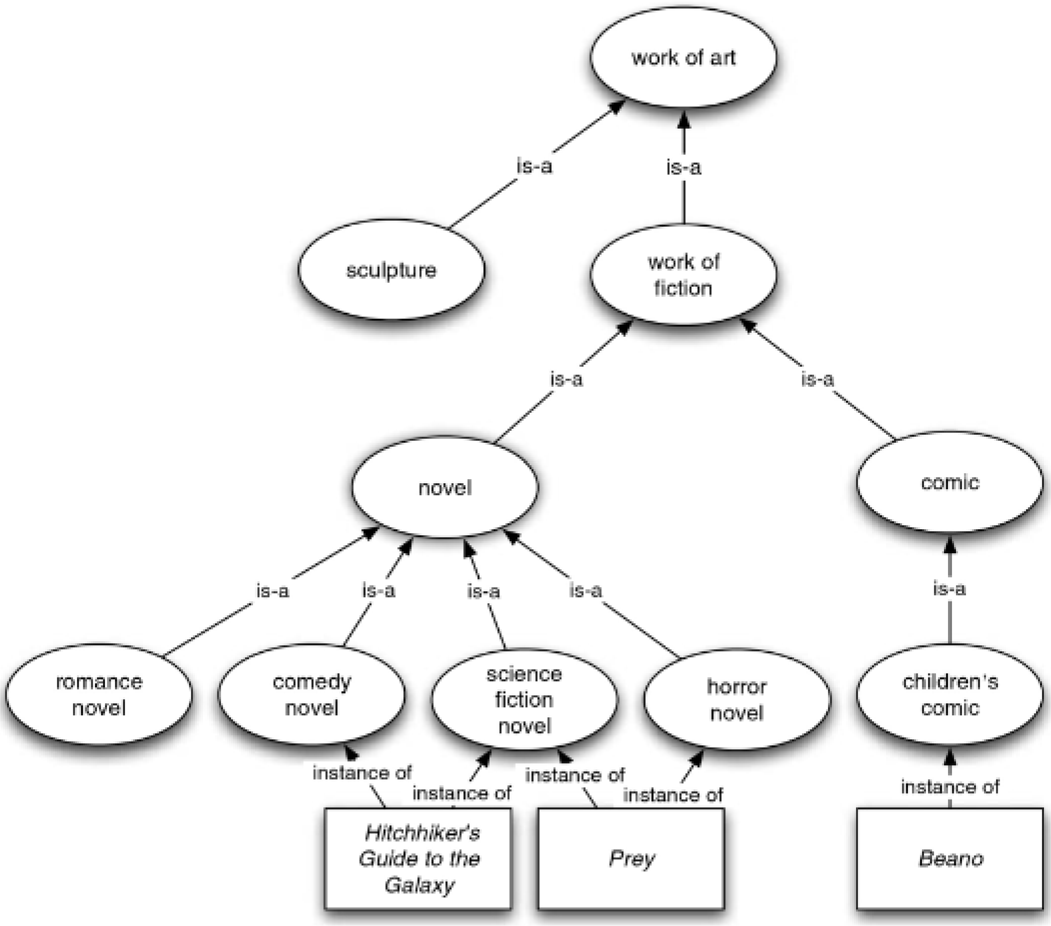


Bob:

- *No, what is it?*

Alice:

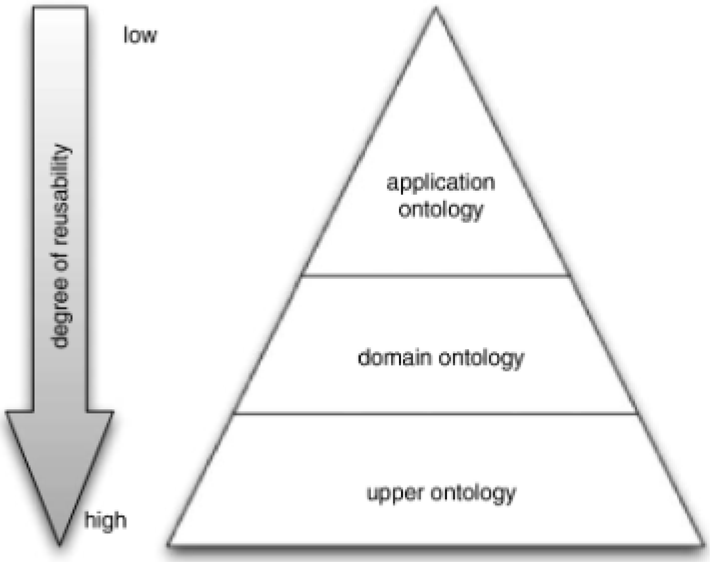
- *A science fiction novel. Well, it is also a bit of a horror novel. It is about multi-agent systems going haywire.*



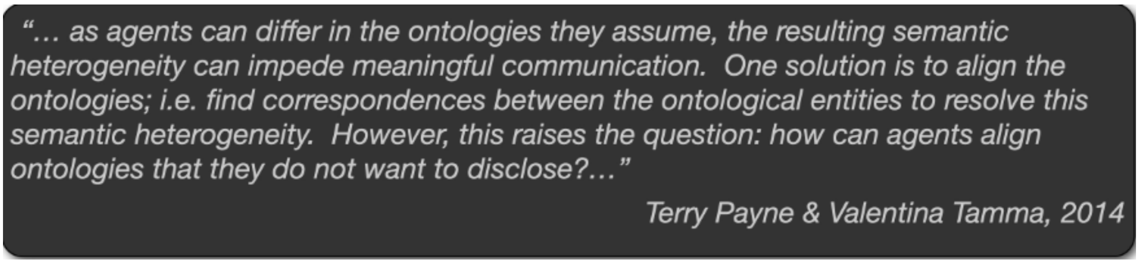
- Multi-levels of Ontologies:
  - **Upper ontology**: Contains the most general information about the world.
  - **Domain ontology**: Defines concepts appropriate for a specific application domain. For example it might define concepts relating to medical terminology, and be used by a number of applications in the area of medicine. Note that a domain ontology will typically build upon and make use of concepts from an upper ontology; this idea of

reuse of ontologies is very important as the more applications use a particular ontology the more agreement there will be on terms.
    - **Application ontology**: Defines concepts used by a specific application. Again, it will typically build upon a domain ontology and in turn upon some upper ontology. Concepts from an application ontology will not usually be reusable; they will typically be of relevance only within the application for which they were defined.

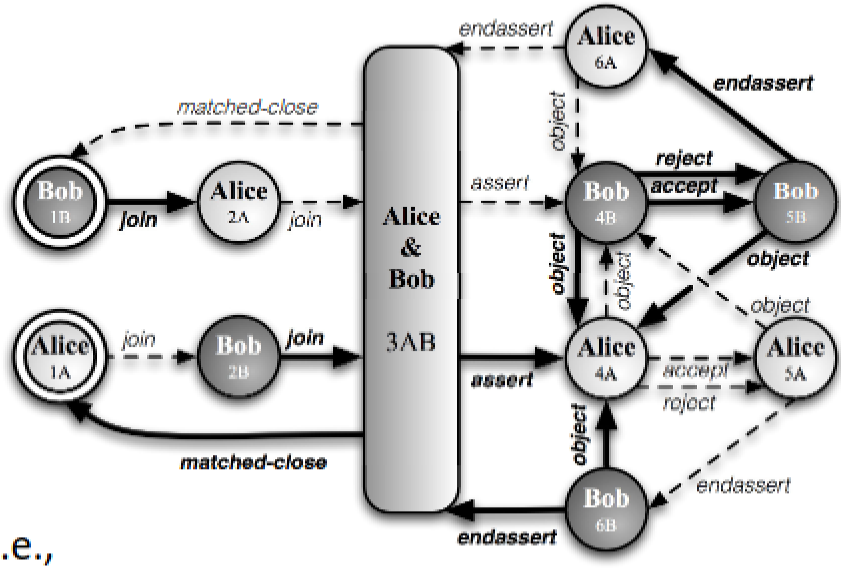
The more specific an ontology, the less reusable it is.



- Aligning Agents' Ontologies:



- (Not in Exam) Correspondence Inclusion Dialogue (CID): Allows two agents to **exchange knowledge** about **ontological correspondences** to agree upon a **mutually acceptable** final alignment.
  Assumptions:
    - Each agent knows about different **correspondences** from different **sources**
    - This knowledge is **partial**, and possibly **ambiguous**; i.e., more than one correspondence exists for a given entity
    - Agents associate a utility (**Degree of Belief**) to each **correspondence**



.e.,

- (Not in Exam) KQML/KIF Example Chapter 6&7 p37

Chapter8: Multi Agent Coordination

- Agent Motivations
  - **Benevolent** Agents:
    - Share the same optimisation target
    - Problem-solving in benevolent systems is **Cooperative Distributed Problem Solving (CDPS)**
  - Self Interested Agents:
    - everyone optimize their own interests
    - Strategic behaviour to solve potential conflicts
- **Cooperative Distributed Problem Solving:** studies how a loosely coupled network of problem solvers can work together to solve problems that are beyond their individual capabilities.
- **Coherence**: how well the MA system behaves as a unit along some dimension of evaluation
- **Coordination**: managing inter-dependencies between the activities of agents, coordination relationships between activities could be either positive or negative
- Contract Net:
  - **Recognition**: an agent recognises it has a problem it wants help with
  - **Announcement**: the agent with the task sends out an announcement of the task which includes a specification of the task to be achieved
  - **Bidding**: Agents that receive the task announcement decide for themselves whether they wish to bid for the task.
  - **Awarding** : The agent that sent task announcement must choose between bids and decide who to "award the contract" to
  - **Expediting**: The agent that won the contract now needs to achieve the task and return the results to the agent with the original task
- Bid decision:

- Contractor i receives an announcement of task specification ts, which is for a set of tasks $\tau(ts)$, the cost to i to carry out is $c_i^t(\tau)$
  - The **marginal cost** of carrying out τ will be:
    $\mu_i(\tau(ts)|\tau_i^t) = c_i(\tau(ts) \cup \tau_i^t) - c_i(\tau_i^t)$
  - Contractor i's resource e_i is $\mu_i(\tau(ts)|\tau_i^t) < e_i$ means the agent can afford to do the new work
- Handling Inconsistency:
  - Forbidden, For example, in the contract net the only view that matters is that of the manager agent.
  - Resolve inconsistency, argumentation until resolve
- **Social Norms**: rules of behaviour. Achieved by the **Constraint pairs**: $< E', \alpha >$, a **social law** is **a set of these constrains**
- **Focal states**: $F \subseteq E$ are the states we want our agent to be able to get to. A **useful social law** is the one that does not prevent agents from getting from one focal state to another
- **Joint Persistent Goal (JPG)** : A group of agents have a collective commitment to bring about some goal φ and motivation ψ

Chapter11: Multi-Agent Competitive

- A **multi-agent system** contains a number of agents that:
  - Interact through communication
  - Are able to act in an environment
  - Have different “spheres of influence”
  - Will be linked by other (organisational) relationships
- Payoff Matrices:
  - Agent i is the **column player** and gets the **upper reward** in a cell
  - Agent j is the **row player** and gets the **lower reward** in a cell

		i defect	i coop
j	defect	1 1	4 1
	coop	1 4	4 4

- **Dominant Strategies**: a strategy s_i is **dominant** for agent i if no matter what strategy s_j agent j choose, i will do at least as well playing s_i as it would doing anything else. e.g. the coop is dominates defect for both agents
- A rational agent will never play a dominated strategy
- **Nash Equilibrium (NE)** : Under the assumption that agent i plays s_1 , agent j can do no better than play s_2 , either to agent j for agent i
- Formally: A strategy (i^*, j^*) is a **pure strategy Nash Equilibrium** solution to the game (A, B) if :

    - $\forall i, a_{i^*,j^*} \geq a_{i,j^*}$ and $\forall j, b_{i^*,j^*} \geq b_{i^*,j}$.
    - This NE may not unique or exist.
- **mixed strategy**: allows to choose between possible choices by introducing randomness into the solution. Nash proved that **Every finite game has a Nash Equilibrium in mixed strategies**
- **Pareto optimal** or **Pareto efficient**: there is no other outcome that makes one agent **better off** without making another agent **worse off**. That means, even if I don't directly benefit from outcome w', you can benefit without me suffering
- **social welfare**: the sum of utilities that each agent gets from outcome w : $\sum_{i \in Ag} u_i(w)$
- Zero-Sum Interactions: $u_i(w) + u_j(w) = 0$, for all $w \in \Omega$
- Payoff matrix for prisoner's dilemma:

Defect: confess

Coop: not confess

j \ i	defect	coop
defect	i: 2, j: 2	i: 1, j: 4
coop	i: 4, j: 1	i: 3, j: 3

- Cooperation is (provably) the **rational choice** in the **infinitely repeated** Prisoner's Dilemma. If finite, raise **backwards induction problem**
- Strategies in Axelrod's Tournament:

ALL-D

"Always defect" — the hawk strategy;

JOSS

As TIT-FOR-TAT, except periodically defect.

Tit-For-Tat

1. On round u = 0, cooperate.
2. On round u > 0, do what your opponent did on round u – 1.

Tester

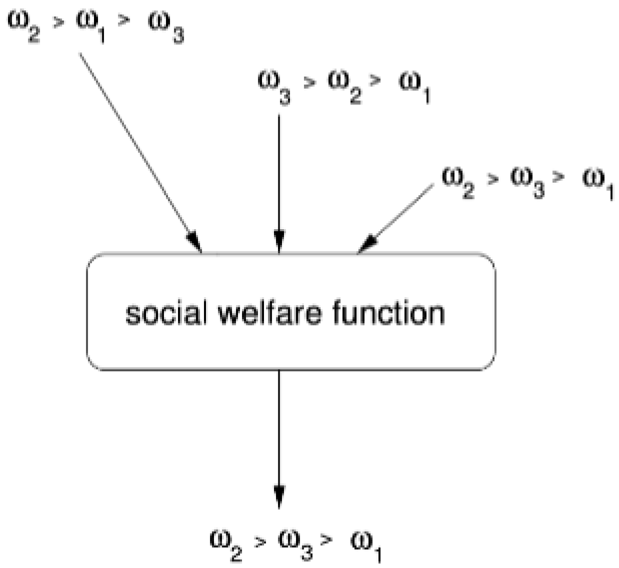
On 1st round, defect. If the opponent retaliated, then play TIT-FOR-TAT. Otherwise intersperse cooperation & defection.

Chapter12: Making Group Decisions

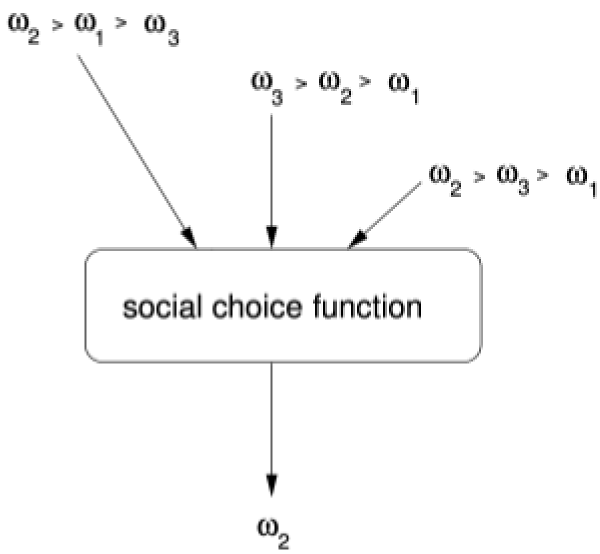
- **social preference order**: $w_a ,_i w_c ,_i w_d, \ldots ,_i w_n$, where $\Omega = \{w_1, w_2, ..\}, |\Omega| = n$ of outcomes for agent i preference
- Two variants of preference aggregation:

- **Social welfare functions**: produces a social preference order

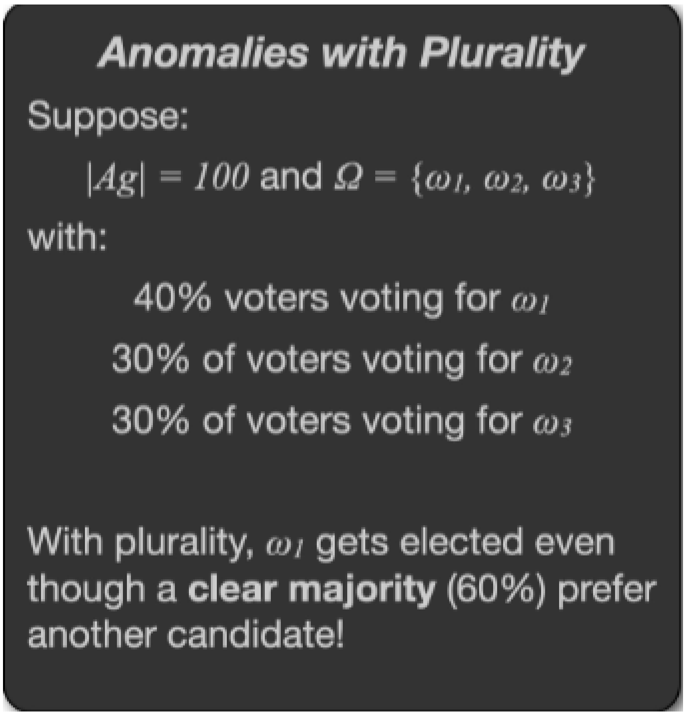
$$f : \underbrace{\Pi(\Omega) \times \cdots \times \Pi(\Omega)}_{n \text{ times}} \mapsto \Pi(\Omega)$$



- **Social choice function**: get a single choice



- **Plurality**: majority selection

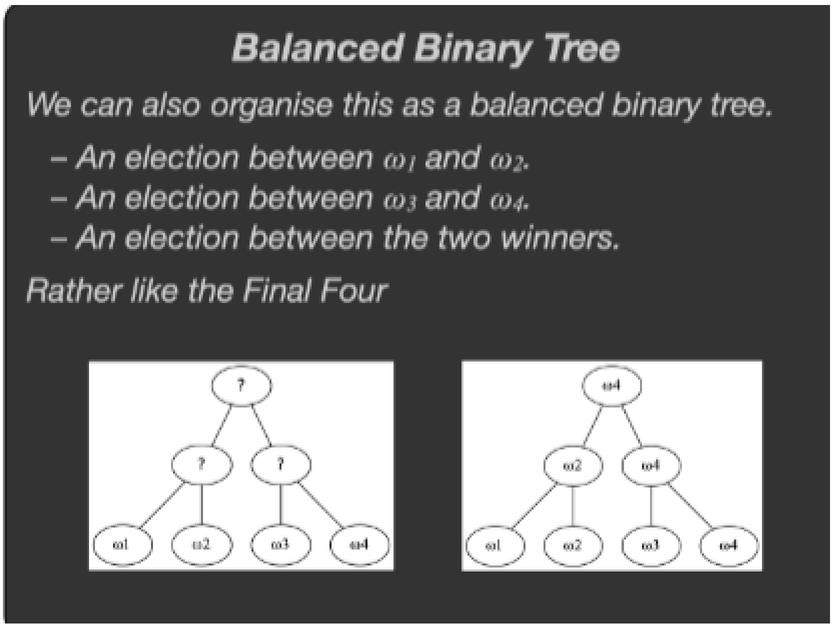


- **Condorcet's Paradox**: no matter which outcome we choose, a majority of voters will be unhappy with the outcome chosen, solutions: **Sequential Majority Elections**

Suppose $Ag = \{1, 2, 3\}$ and $\Omega = \{\omega_1, \omega_2, \omega_3\}$ with:

$$\omega_1 \succ_1 \omega_2 \succ_1 \omega_3$$
$$\omega_3 \succ_2 \omega_1 \succ_2 \omega_2$$
$$\omega_2 \succ_3 \omega_3 \succ_3 \omega_1$$

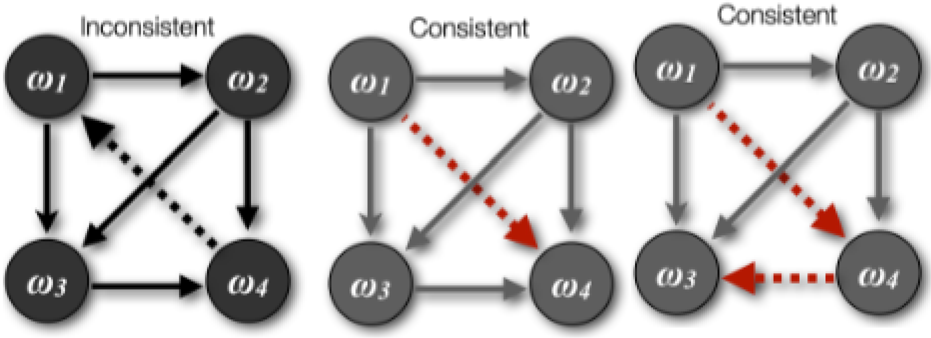
- **Linear Sequential Pairwise Elections**: reduce a general voting scenario to a series of pairwise voting scenarios of an ordering of the outcomes – the **agenda**



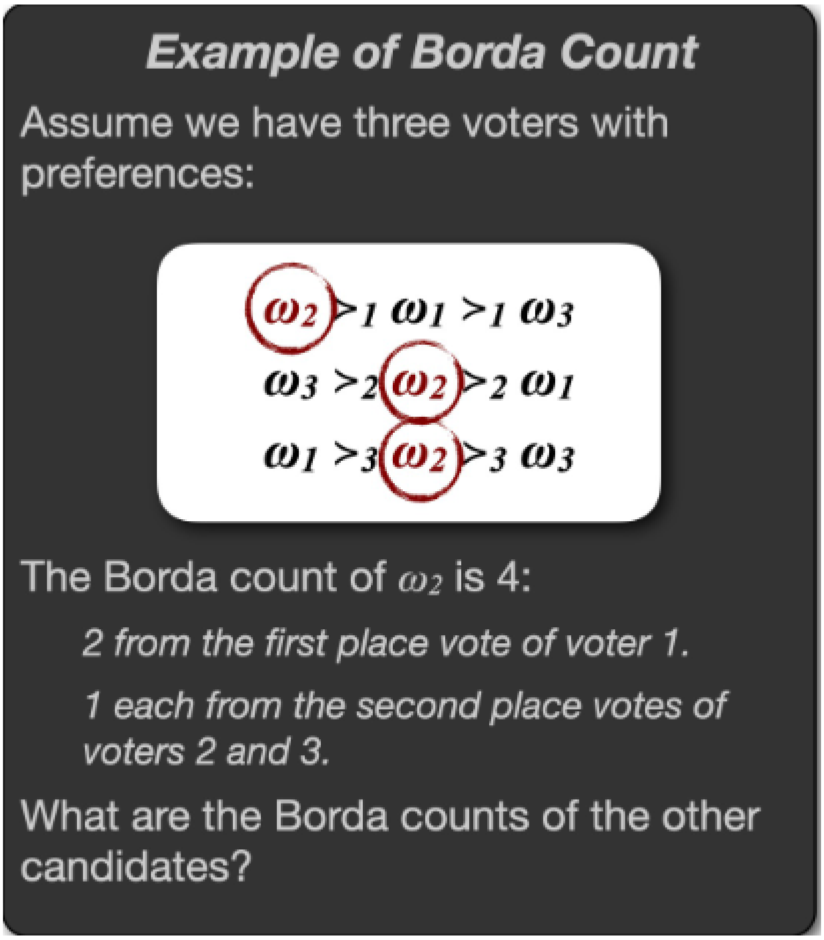
- **majority graph**: An edge (i, j) if i would beat j in a simple majority election, an odd number of voters of a graph called **tournament**
  - **possible winner** if there is an agenda that will result in it winning overall. In majority graph, a possible winner w_i will have a **path** from w_i to w_j for every other w_j

- **Condorcet winner**: if it is the overall winner for every possible agenda. In majority graph, a possible winner w_i will have a **edge** from w_i to w_j for every other w_j

- Slater Ranking: If we reserved some edges in a graph, which ordering minimises this inconsistency measure. For last 2, a **cost** of 1 and 2 are produced. **Slater ranking** is the one with **minimal cost**



- **Borda Count**: takes the whole preference order into account



- **social choice voting** or **Instant Runoff Voting**: Counting procedures in rounds, with the last place candidate being eliminated, until there is a majority vote. Offers a solution to Condorcet's paradox. Example is given in Chapter 12 P31-33.

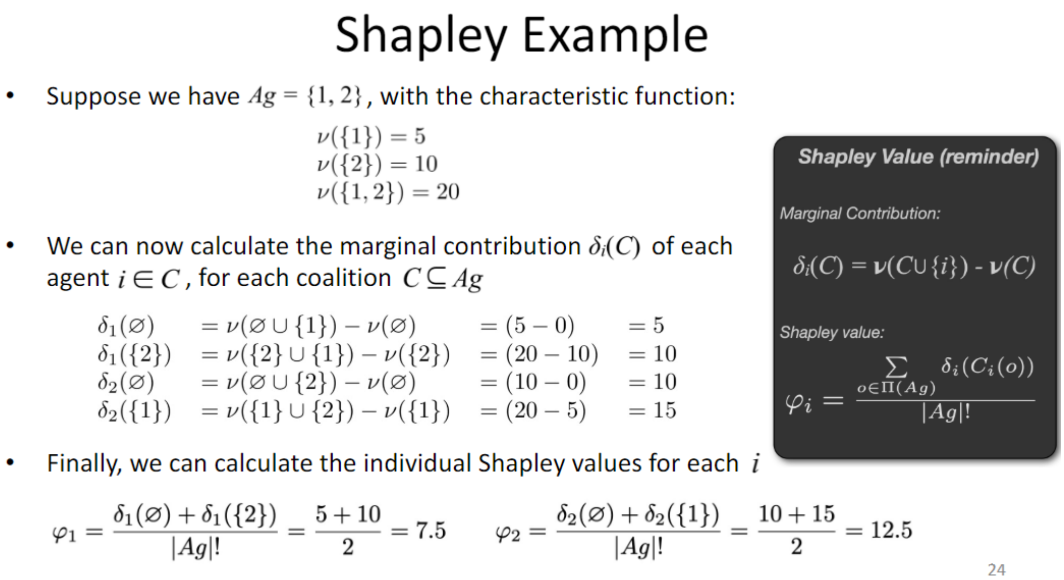
- Desirable Properties of Voting Procedures:
  - **Pareto property**: if everybody prefers w_i over w_j, then w_i should be ranked over w_j in the social outcome. Satisfied by plurality and Borda but not by sequential majority.
  - **Condorcet Winner condition**: if w_i is a condorcet winner, the w_i should always be ranked first. However, of the ones we've seen, only sequential majority satisfies it
  - **Independence of Irrelevant Alternatives**: the social ranking of w_i and w_j should depend **only** on the way they are ranked in the $\succ$ relations of the voters. Plurality, Borda, and sequential majority do not satisfy IIA.

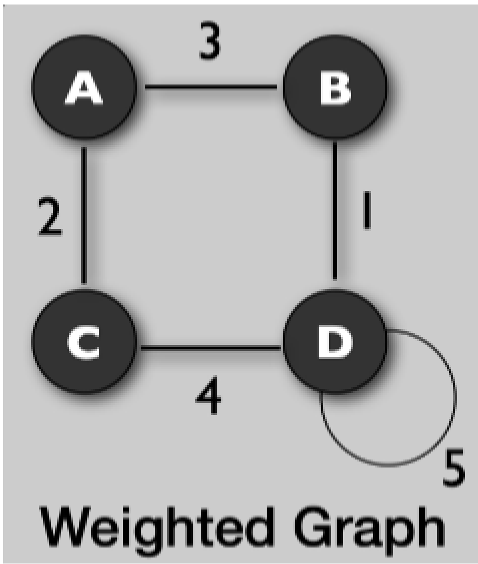
- A social welfare function f is a **dictatorship** if for some agent i: $f(\Pi(\Omega_1), \Pi(\Omega_2), \Pi(\Omega_3), \ldots) = \Pi(\Omega_i)$
- **Arrow's Theorem**: For elections with more than 2 candidates, the **only voting procedure satisfying the Pareto condition and IIA is a dictatorship**
- **Gibbard-Satterthwaite Theorem**: **The only non-manipulable voting method** satisfying the Pareto property for elections with more than 2 candidates is a **dictatorship**

Chapter13: Forming Coalitions

- three stages of cooperative action:
  - Coalitional structure generation
  - Solving the optimisation problem of each coalition
  - Dividing the benefits
- A Coalitional Game: $\mathcal{G} = \langle Ag, v \rangle$, where a set of agents: $Ag = \{1, \ldots, n\}$, the characteristic function of the game: $v : 2^{Ag} \to \mathbb{R}$. A coalition $C \subseteq Ag$
  - **Singleton Coalition**: a coalition consists of a single member
  - **Grand Coalition**: $C = Ag$, for all agents
  - Each coalition has a payoff value k, defined by the characteristic function $v(C) = k$
- game property:
  - **superadditive**: if $v(U) + v(V) \leq v(U \cup V)$, where The coalition that maximizes social welfare is the **Grand Coalition**
  - **subadditive**: $v(U) + v(V) > v(U \cup V)$, where The coalition that maximizes social welfare is the **Singleton Coalition**
  - neither subadditive nor superadditive, The **characteristic function value calculations** need to be determined **for each of the possible coalition**
- **Stability** is a **necessary** but **not sufficient** condition for coalitions to form
- the **core** of a coalitional game: the set of **feasible distributions of payoff** to members of a coalition that **no sub-coalition can reasonably object** to.
- the stability of the grand coalition:
  - An **outcome** x for a coalition C in game $\langle Ag, v \rangle$ is a distribution of C 's utility to members of C , $x = \langle x_1, \ldots, x_k \rangle$, which is both **feasible** for C and **efficient**: $v(C) = \sum_{i \in C} x_i$
  - **Feasible**: C really could obtain the payoff indicated in the outcome x
  - **Efficient**: all of the payoff is allocated
- **Shapley value**: the **average** amount that agent i is expected to contribute to a coalition. It is based on the **marginal contribution** of the agent to a coalition: $\delta(C) = v(C \cup i) - v(C)$, where the $\delta(C)$ is the amount of value that agent i adds by joining a coalition $C \subseteq Ag$
- A **fair** distribution of coalitional values should satisfy:
  - **symmetry**: Agents that make the same contribution should get the same payoff. **It states**: if i and j are interchangeable, the $\varphi_i = \varphi_j$
  - **Dummy player**: Agents that never have any synergy with any coalition, and thus only get what they can earn on their own. **It states**: if i is a dummy player, then $\varphi_i = v(i)$

    - **Additivity**: if two game are combined, the value an agent gets should be the sum of the values it gets in the individual games. **It states**: the value φ_i^{1+2} of agent i in game $\mathcal{G}^{1+2}$ should be $\varphi_i^1 + \varphi_i^2$
- marginal contribution can be **dependent on the order** in which an agent joins a coalition. It better to consider the average value it adds **over all possible positions that it could enter the coalition**:
    - the **Shapley value** for agent i :
      $\varphi_i = \frac{1}{|Ag|!} \sum_{o \in \Pi(Ag)} \delta_i(C_i(o))$,
      where $o \in \Pi(Ag)$ denotes the ordering and $C_i(o)$ denotes the agents that appear before agent i



- An n-player game consists of $2^n - 1$ coalitions, problem raised in obvious representation, solutions could be:
    - a **complete** representation that is **succinct in "most"** cases
    - a representation that is **not complete** but is **always succinct**
- **Induced Subgraph**: succinct, but not complete
    - the value of a coalition $C \subseteq Ag$: $v(C) = \sum_{\{i,j\} \subseteq C} w_{i,j}$
    - Shapley value: $\varphi_i = \frac{1}{2} \sum_{j \neq i} w_{i,j}$
    - 

- **Marginal Contribution Nets**: complete, but not succinct

- characteristic function v is represented as **a set of rules**: $pattern \rightarrow value$
- the value of a coalition $C \subseteq Ag$: $v(C) = \sum_{rule_C \in rules} value$

Rule set (rs) 2:

$$\begin{aligned} a \wedge b &\rightarrow 5 \\ b &\rightarrow 2 \\ c &\rightarrow 4 \\ b \wedge \neg c &\rightarrow -2 \end{aligned}$$

$v_{rs2}(\{a\}) = 0$	no rules apply
$v_{rs2}(\{b\}) = 2 + -2 = 0$	2nd and 4th rules
$v_{rs2}(\{c\}) = 4$	3rd rule
$v_{rs2}(\{a, b\}) = 5+2+-2 = 5$	1st, 2nd and 4th rules
$v_{rs2}(\{a, c\}) = 4$	3rd rule
$v_{rs2}(\{b, c\}) = 2+4 = 6$	2nd and 3rd rules
$v_{rs2}(\{a, b, c\}) = 5+2+4 = 11$	1st, 2nd and 3rd rules

- the Shapley value: $\varphi_i = \sum_{r_C \in rules} \varphi_i^r$, where $\varphi_i^{1 \wedge \ldots \wedge l \rightarrow x} = \frac{x}{l}$

$$\begin{aligned} a \wedge b &\rightarrow 5 \\ b &\rightarrow 2 \\ c &\rightarrow 4 \end{aligned}$$

$$\varphi_A = \sum_{r \in rs; A \text{ occurs in lhs of } r} \varphi_A^r = \frac{5}{2} = 2.5$$

$$\varphi_B = \sum_{r \in rs; B \text{ occurs in lhs of } r} \varphi_B^r = \frac{5}{2} + 2 = 4.5$$

$$\varphi_C = \sum_{r \in rs; C \text{ occurs in lhs of } r} \varphi_C^r = 4$$

- **Coalition Structure Generation**: maximize the social welfare of the system
  - The optimal coalition structure: $V(CS) = \sum_{C \in CS} v(C)$, where $CS^* = \underset{CS}{argmax} V(CS)$

Chapter 14: Allocating Scarce Resources

- **auction**: takes place between an agent known as the **auctioneer** and a collection of agents known as the **bidders**
- Types of Value:
  - **Private Value**: good has a value to me that is independent of what it is worth to you
  - **Common Value**: good has the same value to all of us, but we have differing estimates of what it is
  - **Correlated Value**: values are related defined
- **winner's curse**: the winning bid in an auction exceeds the intrinsic value or true worth of an item
- Auction Protocol Dimensions:
  - Winner Determination
  - Open Cry vs. Sealed-bid
  - One-shot vs. Iterated Bids
- Dutch Auction: Starts at a **high price**, and the auctioneer calls out **descending prices**

- Vickrey auction: **second-price one-shot sealed-bid** auction.
  - Each bidder submits a single sealed bid, Bidders do not see the bids made by others; The winning bid is the **highest bid** and pay with **second highest bid**
  - The payoff of bidder i: $p_i = \begin{cases} v_i - max_{j \neq i} b_j & \text{if } b_i > max_{j \neq i} b_j \\ 0 & \text{otherwise} \end{cases}$
  - a **dominant strategy** for each bidder to bid their true valuation, making it possible for **antisocial** behaviour
- None of the auction types discussed so far are immune to **collusion**:
  - A grand coalition of bidders can agree beforehand to collude
  - An auctioneer could employ bogus bidders
- **Combinatorial Auctions**: auction for bundles of goods
  - valuation function: the value of bundles of goods
  - free disposal: an agent is never worse off having more stuff
  - outcome: an allocation of goods being auctioned among the agents, no need to be allocated
- social welfare function: $sw(\underbrace{Z_1, \ldots, Z_n}_{allocations}, \underbrace{v_1, \ldots, v_n}_{valuations}) = \sum_{i=1}^n v_i(Z_i)$
- Combinatorial auction: $Z_1^*, .., Z_n^* = \underset{(Z_1, .., Z_n) \in alloc(\mathcal{Z}, Ag)}{argmax} sw(Z_1, \ldots, Z_n, v_1, \ldots, v_n)$
  - where $\mathcal{Z}$ is a set of goods, the goal is to find an allocation $Z_1^*, .., Z_n^*$ that maximizes sw, the collection of valuation functions $v_1, .., v_n$ for each agent $i \in Ag$
  - Determining the winner is a **combinatorial optimization problem** (NP-hard)
- An **atomic bid** $\beta = (Z, p)$, where $p = v_\beta(Z)$ is the price the bidder is prepared to pay for the bundle Z, a bundle Z' **satisfies** an atomic bid (Z, p) if $Z \subseteq Z'$
- XOR Bids: pay for at most one of the bundles

$\beta = (Z_1, p_1) XOR \ldots XOR (Z_k, p_k)$

B_1 =({a,b}, 3) XOR ({c, d}, 5)

"…I would pay 3 for a bundle that contains a and b but not c and d. I will pay 5 for a bundle that contains c and d but not a and b, and I will pay 5 for a bundle that contains a, b, c and d..."

From this we can construct the valuation:

$$\begin{aligned} v_{\beta_1}(\{a\}) &= 0 \\ v_{\beta_1}(\{b\}) &= 0 \\ v_{\beta_1}(\{a,b\}) &= 3 \\ v_{\beta_1}(\{c,d\}) &= 5 \\ v_{\beta_1}(\{a,b,c,d\}) &= 5 \end{aligned}$$

- OR Bids: prepared to pay for more than one bundle
  - $\beta = (Z_1, p_1) OR \ldots OR (Z_k, p_k)$
  - OR bids are strictly less expensive than XOR bids
  - OR bids also suffer from computational complexity
  -

B_1 =({a,b}, 3) OR ({c, d}, 5)

"…I would pay 3 for a bundle that contains a and b but not c and d. I will pay 5 for a bundle that contains c and d but not a and b, and I will pay 8 for both bundles that contain a combination of a, b, c and d..."

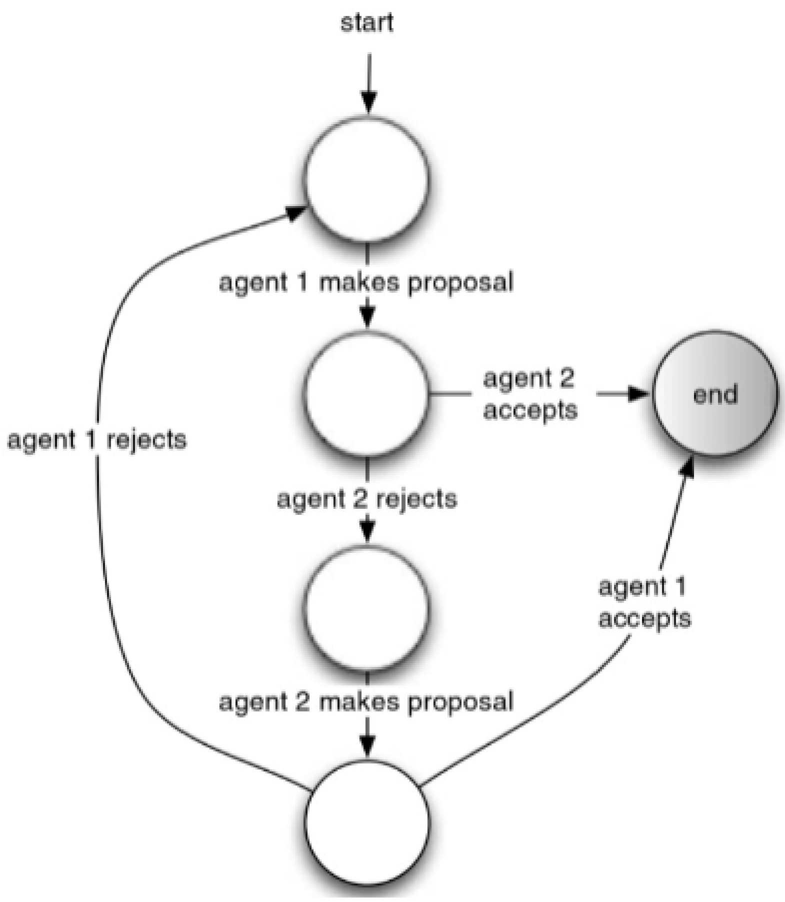
From this we can construct the valuation:

$$\begin{aligned} v_{\beta_1}(\{a\}) &= 0 \\ v_{\beta_1}(\{b\}) &= 0 \\ v_{\beta_1}(\{a,b\}) &= 3 \\ v_{\beta_1}(\{c,d\}) &= 5 \\ v_{\beta_1}(\{a,b,c,d\}) &= 8 \end{aligned}$$

Note that the **cost of the last bundle is different to that when the XOR bid** was used

- VCG Mechanism:
  - **Indifferent valuation** function: $v^0(Z) = 0$ for all Z
  - the **social welfare function without agent** i : $sw_{-i}(Z_1, \ldots, Z_n, v_1, \ldots, v_n) = \sum_{j \in Ag, j \neq i} v_j(Z_j)$, defines how well everyone expect agent i
  - the pays p_i for agent i is: $$p_i = sw_{-i}(Z_1, \ldots, Z_n, v_1, \ldots, v^0, \ldots, v_n) \\ -sw_{-i}(Z_1^*, \ldots, Z_n^*, v_1, \ldots, v_i, \ldots, v_n)$$

Chapter 15: Bargaining / Negotiation

- **Auctions** are only concerned with the **allocation of goods**; **Negotiation** is the process of **reaching agreements on matters of common**
- negotiation:
  - **negotiation set**: possible proposals that agents can make
  - **protocol**: defines what legal proposals an agent can make
  - **collection of strategies**: one for each agent, which are typically private
  - **rule**: determines **when** a deal has been struck and **what** the agreement deal is
  - Negotiation usually proceeds proceeds in a series of rounds
- **single-issue, symmetric, one-to-one negotiation** settings are in COMP310
- Rubinstein's alternating offers:
  - 

  - If there is **no agreement**, we say the result is the **conflict deal** Θ

- Impatient Players:

Impatient Players

Each agent has a **discount factor** δ_i where $i \in \{1, 2\}$, and $0 \leq \delta < 1$. The closer δ_i is to 1, the more patient the agent is.

If agent i is offered x, then the value of the slice is:

- x at time 0.
- $\delta_i x$ at time 1.
- $(\delta_i * \delta_i)x = \delta_i^2 x$ at time 2.
- $(\delta_i * \delta_i * \delta_i)x = \delta_i^3 x$ at time 3.
- ...
- $\delta^k x$ at time k

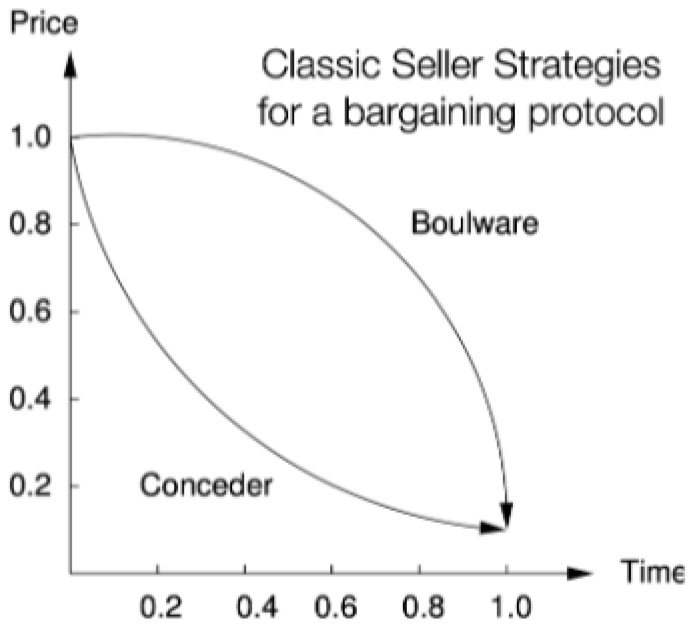
Now we can make some progress with the fixed number of rounds.

Division of the Pie with Discount Fctors

Discount factor δ_1 ↓ / Discount factor δ_2 →	0.8	0.6	0.4	0.2
0.8	(0.556,0.444)	(0.769,0.231)	(0.882,0.118)	(0.952,0.048)
0.6	(0.385,0.615)	(0.625,0.375)	(0.789,0.211)	(0.909,0.091)
0.4	(0.294,0.706)	(0.526,0.474)	(0.714,0.286)	(0.870,0.130)
0.2	(0.238,0.762)	(0.455,0.545)	(0.652,0.348)	(0.833,0.167)

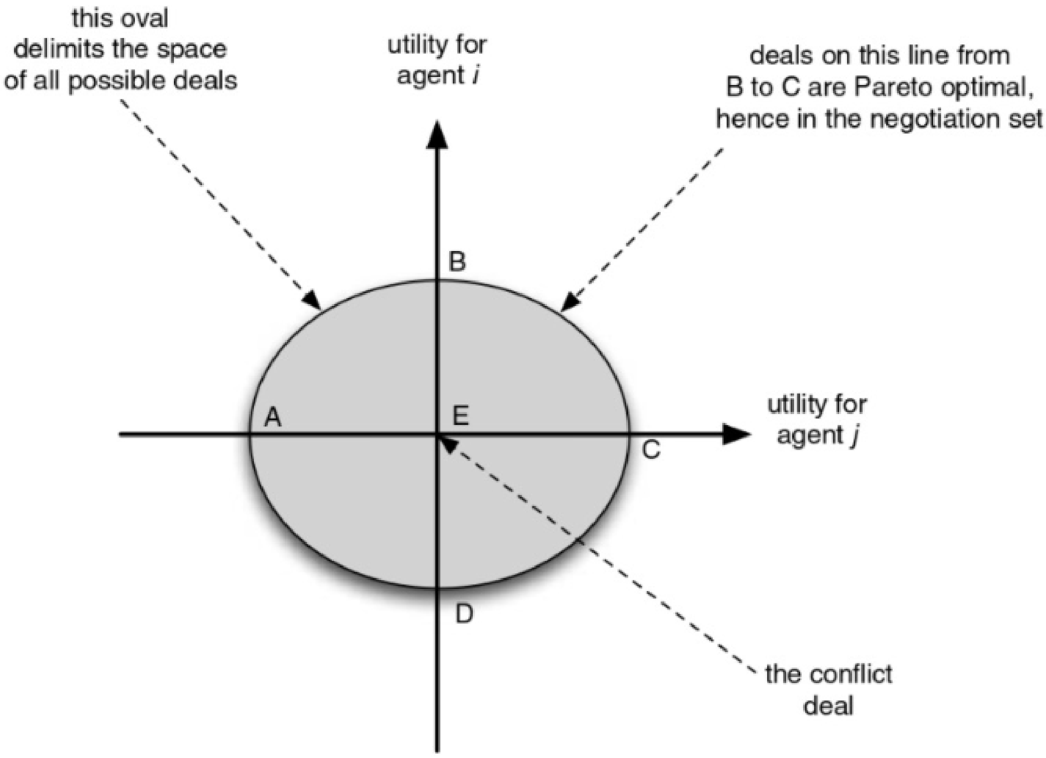
- Negotiation Decision Functions:
  - Boulware: Very **slow initial decrease** in price until close to deadline and then an exponential decrease

- Conceder: **Makes its concessions early**, and makes fewer concessions later in the negotiation



- **Task-Oriented Domains (TODs)**:
  - Formally: A TOD is a tuple $\langle T, Ag, c\rangle$,
    where, $Ag = \{1, \ldots, n\}$ denotes the participant agents,
    the task set for all possible tasks T
    the cost of each subtasks $c : 2^T \rightarrow \mathbb{R}^+$, is **monotonic** and $c(\emptyset) = 0$
  - An **encounter** is a collection of tasks $\langle T_1, \ldots, T_n\rangle$, where $T_i \subseteq T$ for each agent $i \in Ag$
- Deals in TODs"
  - deal $\delta = \langle D_1, D_2\rangle$ is a allocation of the tasks $T_1 \cup T_2$, for encounter $\langle T_1, T_2\rangle$, agent i cost: $c(D_i)$
  - utility of the deal δ for agent i : $utility_i(\delta) = c(T_i) - cost_i(\delta)$, where the $cost_i(\delta)$ denotes the cost of agent i in the deal
  - For conflict deal Θ , $untility_i(\Theta) = 0$, for all agents
  - **dominate deal** $\delta_1 \succ \delta_2$: $\begin{cases} \forall i \in \{1,2\}, utility_i(\delta_1) \geq utility_i(\delta_2) \\ \exists i \in \{1,2\}, utility_i(\delta_1) > utility_i(\delta_2) \end{cases}$
  - A deal that is **not dominated** by any other deal is **Pareto optimal**
- **Negotiation Set**:
  - **Individual rational**: agents won't be interested in deals that give negative utility since they will prefer the conflict deal

- **Pareto optimal**: agents can always transform a non-Pareto optimal deal into Pareto optimal deal by making one agent happier and none of the others worse off



- Monotonic Concession Protocol:
  - Negotiation proceeds in rounds u where $u \geq 0$
  - Procedure:
    - On round 1, Both agents simultaneously propose a deal from the negotiation set
    - Agreement is reached if one agent finds that the deal proposed by the other is at least as good or better than its proposal
    - If no agreement is reached, then negotiation proceeds to another round of simultaneous proposals
    - In round $u + 1$, no agent is allowed to make a proposal that is less preferred by the other agent than the deal it proposed at time u.
    - If neither agent makes a concession in some round $u > 0$, then negotiation terminates with the conflict deal
  - negotiation is guaranteed to end after a finite number of rounds
- Zeuthen strategy: determine **who** should concede and **how much** in each round, in **Nash Equilibrium**
  - Key idea: evaluate an agent's **willingness to risk conflict**
  - Agent i's **willingness to risk conflict** at round t, denoted $risk_i^t$, can be measured as:

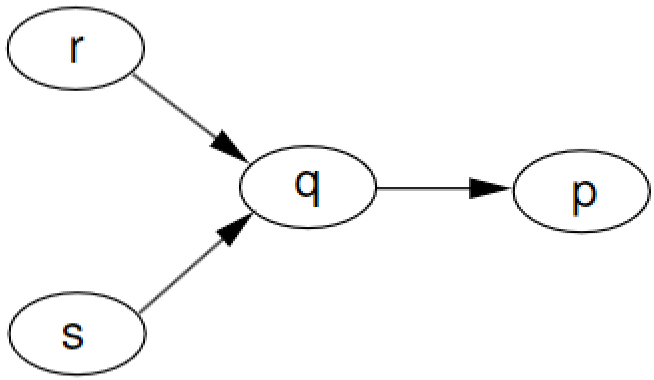
$$risk_i^t = \begin{cases} 1 & \text{if } utility_i(\delta_i^t) = 0, \\ \dfrac{utility_i(\delta_i^t) - utility_i(\Theta)}{utility_i(\delta_i^t)} & otherwise \end{cases}$$

  - agent can start by proposing its **most preferred deal**, Concede whenever its **willingness to risk conflict value** $\leq$ to the other agent's
  - If an agreement is reached, it is guaranteed that this agreement will be **individual rational** and **Pareto optimal**

- Ways to be benefited from Deception in TODs:
  - Phantom and Decoy tasks: Pretending that you have been allocated tasks that you have not, can be prevented by ensuring that tasks are **verifiable**
  - Hidden tasks: Pretending not to have been allocated tasks that you have

Chapter 16: Argumentation

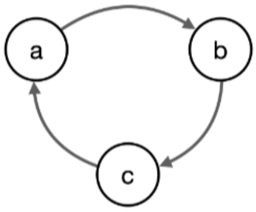
- Argumentation: principled techniques for resolving with **inconsistencies** with beliefs of multiple agents
- **Dung's Argumentation System (Abstract argumentation)**:
  - $q \rightarrow p$, means argument q attacks argument p
  - a set of Dung-style arguments: $\langle \sum, \rhd \rangle$:
    - $\sum$: set of arguments
    - $\rhd$: set of attacks between arguments in $\sum$
    - $(\varphi, \psi) \in \rhd$: the relationship: φ attacks ψ
  - Example: $\langle \{p, q, s, r\}, \{\langle r, q \rangle, \langle s, q \rangle, \langle q, p \rangle\} \rangle$



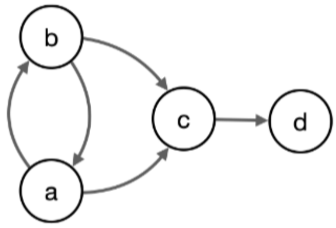
- **Type of Positions**:
  - **Position**: $S \subseteq \sum$ denotes a set of arguments
  - **Conflict free positions**: no member of S attacks an other member of S. e.g. $\emptyset, \{p\}, \{q\}, \{r\}, \{s\}, \{r, s\}, \{p, r\}, \{p, s\}, \{r, s, p\}$ in previous $\sum$
  - **mutually defensive positions**: every element of S that is attacked is defended by some element of S.
    e.g. $\emptyset, \{r\}, \{s\}, \{r, s\}, \{p, r\}, \{p, s\}, \{r, s, p\}$
  - **Admissible Positions**: both conflict free and mutually defensive, it is a minimal notion of a reasonable position

- **Preferred Extension**: a maximal admissible set S. if S is admissible and **no superset** of S is admissible .
  e.g. $\{r, s, q\}$ is a preferred extension, others are not

These two arguments are mutually attacking. As either could attack the other, there are two preferred extensions: {a} and {b}

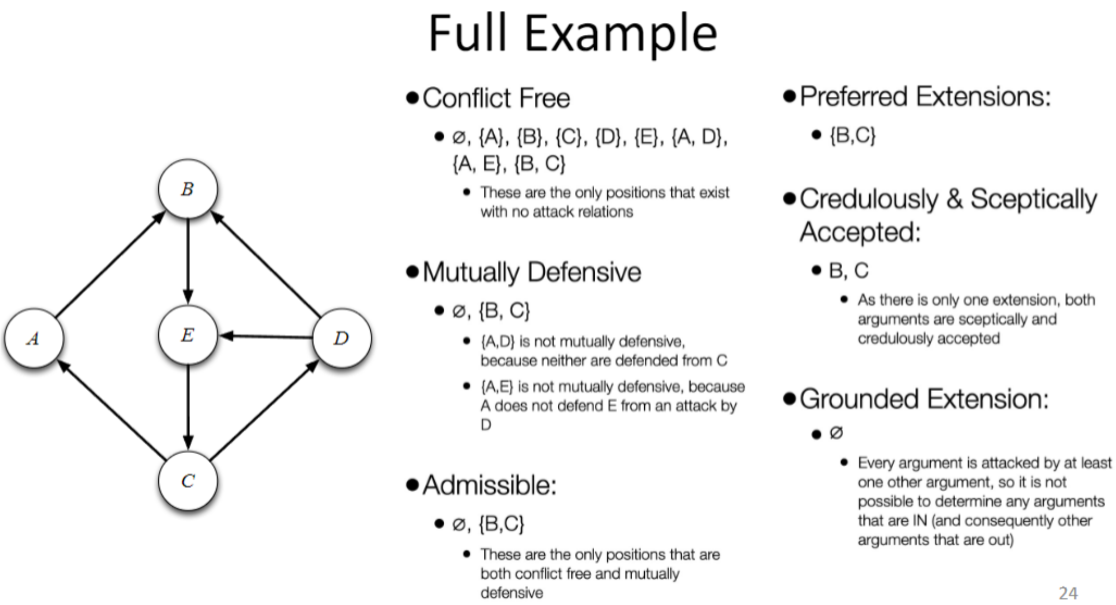


With an odd number of arguments attacking in a cyclic pattern, there can be no consistent state. Thus, the preferred extension is ∅.

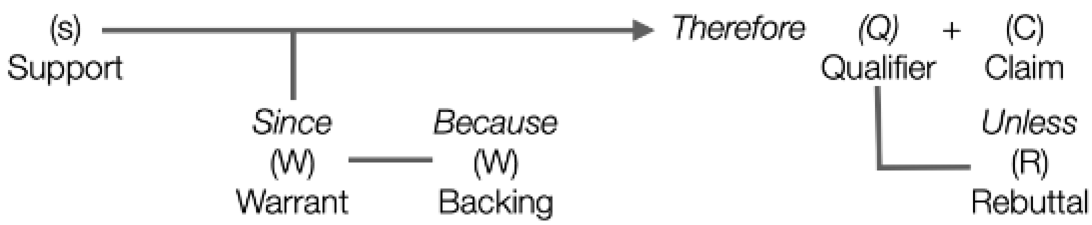


In this case, a and b are mutually attacking, and thus there will be at least two preferred extensions. As they both attack c, d is defended. Therefore, we have the two extensions: {a,d} and {b,d}

- Argument acceptance:
  - **sceptically** accepted argument: a member of **every preferred extension**
  - **credulously** accepted argument: a member of **at least one preferred extension**
- Grounded Extensions - a progress:
  - keep INs, that have no attackers
  - eliminate OUT, that is attacked by INs
  - There is **always** a grounded extension, and it is **always unique**
- Full example:



- **Deductive Argumentation**, are defeasible reasoning by logical formulating arguments



  - Formally, a **deductive argument** is a pair (S, p) over a database $\sum$
  - $\sum$: a set of logical formulae, the available 'evidence'
  - p : a conclusion, a logical formulae

- S: the grounds or support, a set of logical formulae
  - $S \subseteq \sum$
  - $S \vdash p$
  - no such S' that $S' \subset S$ and $S' \vdash p$

Example

Σ = {

human(*Socrates*)
human(*Heracles*)
father(*Heracles*, *Zeus*)
father(*Apollo*, *Zeus*)

divine(*X*) → ¬ *mortal*(X)
human(X) → *mortal*(X)
father(X, *Zeus*) → *divine*(X)
¬(*father*(X, *Zeus*) → *divine*(X)

}

Therefore, the following argument Arg_1 holds:

Arg_1= ({*human*(*Socrates*), *human*(X) → *mortal*(X)}, *mortal*(*Socrates*))

I.e.

S = {*human*(*Socrates*), *human*(X) → *mortal*(X)}
p = *mortal*(*Socrates*)

- **Rebut**: (S_2, p_2) rebuts (S_2, p_1) if $p_2 \equiv \neg p_1$

- **Undercut**: (S_2, p_2) undercuts (S_1, p_1) if $p_2 \equiv \neg q_1$ for some $q_1 \in S_1$

Example

Σ = {

human(*Heracles*)
father(*Heracles*, *Zeus*)
father(*Apollo*, *Zeus*)

divine(*X*) → ¬ *mortal*(X)
human(X) → *mortal*(X)
father(X, *Zeus*) → *divine*(X)
¬(*father*(X, *Zeus*) → *divine*(X)

}

Given the argument Arg_2:

Arg_2= ({*human*(*Heracles*), *human*(X) → *mortal*(X)}, *mortal*(*Heracles*))

The argument Arg_3 rebuts Arg_2:

Arg_3= ({*father*(*Heracles*, *Zeus*), *father*(X, *Zeus*) → *divine*(X), *divine*(*X*) → ¬ *mortal*(X)}, ¬ *mortal*(*Heracles*))

The argument Arg_4 undercuts Arg_3:

Arg_4= ({¬(*father*(X, *Zeus*) → *divine*(X))}, ¬(*father*(X, *Zeus*) → *divine*(X)))