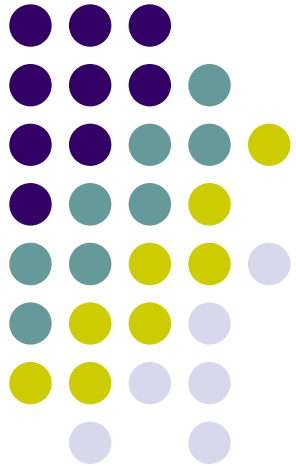
MULTI-AGENT SYSTEMS

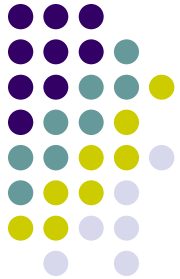
Cooperation, Coordination, Noegotiation

MIDDLE AGENTS

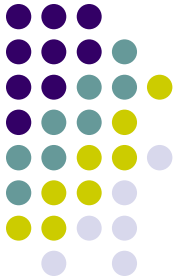
(source: Fasli – Agents in eCommerce
www.wileyeurope.com/college/fasli)



ASSIGNMENTS (due Oct.



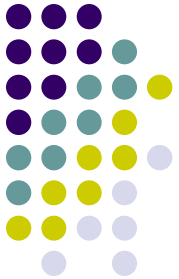
1. Give an example for each: cooperation, coordination, negotiation (from your project)
2. Give an example for each: matchmaker, broker, broadcaster (from your project)
3. Make a qualitative comparison between the three kinds of matching
4. Give a specification example for your project similar to the LARKS example on slide 38
5. Create your own example of Request and Offer and calculate the *distance* using the filter of your choice
6. Give an ontology example (service grounding, model and profile – slide 54) for your project



COORDINATION, COOPERATION, NEGOTIATION AND PLANNING

Coordination, Cooperation and Negotiation are aspects of group or joint activity of agents. Planning can be an individual activity of an agent, but in a group it is normally desirable that the plans be coherent and contribute towards the achievement of goals or objectives.

The focus in this part of the course will therefore be how can agents work together towards the achievement of objectives.



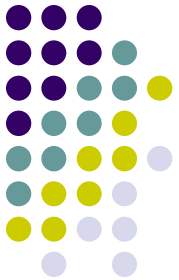
NOTE:

1. The following draws significantly from the text:

   O'Hare, G.M.P. and Jennings, N.R., "Foundations of Distributed Artificial Intelligence", John Wiley, 1996.

2. References cited in the following sections, unless otherwise indicated, may be found in that volume.

3. Unless otherwise indicated, quotations are from this source.



COORDINATION

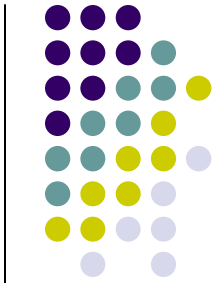
- *Coordination* is a mechanism for ensuring that agents' activities retain some desired relationship/s (sequence, complementarity, etc.).
- *Control* is the extent to which coordination information *must* be implemented by a recipient agent.
- The range of control is from *none* to *total*. Control is inversely related to autonomy (for the recipient agent, no control corresponds to *total* autonomy, and being totally controlled corresponds to *zero* autonomy.

- The fundamental coordination mechanisms are:
  - Mutual adjustment

    Agents share information and resources to achieve a common goal, adjusting their behaviour according to the behaviour of the other agents. No agent has prior control and decision making is joint. Coordination in peer groups and markets is usually by mutual adjustment.

  - Direct supervision

    One agent has some degree of control over others and can control information, resources, and behaviour. Often established through mutual adjustment (e.g. following acceptance of employment or a contract).

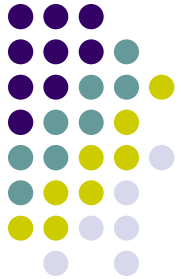


- Coordination by standardization

  Standard procedures are established for agents to follow. In mutual adjustment, are implemented by acceptance. In direct supervision, are implemented through (mandatory) requests.

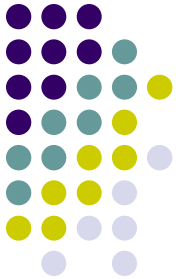
- Mediated coordination

  A mediator acts as a facilitator (e.g. finds/routes information, etc.), a broker (a ‘go-between’ and advisor on resource negotiations, etc.), and a supervisor (exercising some degree of direct supervision). The first role is mandatory, but the others are optional. A mediator facilitates or brokers mutual adjustment between agents and may also use direct supervision.



- Coordination by reactive behaviour

  Agents react to particular stimuli (“situations) with specific behaviours (“actions”). With appropriately selected or evolved stimuli-behaviour groupings and distributions, system-level patterns of coordinated behaviour emerge which contribute to the achievement of common or system goals.



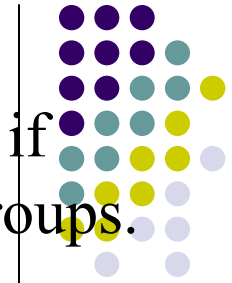
Coordinated Systems

Mechanisms

Mutual adjustment operates through peer-to-peer interactions; Direct supervision uses the ‘master-slave’ approach (in pure case, no peer-to-peer interactions allowed); A mediator facilitates or brokers mutual adjustment between agents and may exercise some degree of supervision. Reactive coordination depends on appropriate combinations of stimuli-behaviour patterns.

Coordinated Groups and Organization Structure

- Mutual adjustment can work well in small groups but the number of information links and amount of information increases rapidly with size.



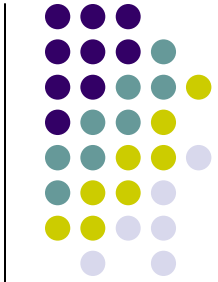
- Large groups can be effectively divided into sub-groups if most of the information exchange can occur in the subgroups.
- Effective coordination of subgroups requires each to have a coordinator/supervisor/mediator, who work together in one or more coordination groups.
- Each coordination group as well as each lower-level sub group can be internally coordinated either by mutual adjustment, or direct supervision, mediation, or reactive behaviour.
- A pure hierarchy of the groups results if direct supervision is the only coordination mechanism used.
- A (pure) heterarchy of agents results if only mutual adjustment is used as the mechanism, with no groups existing.
- Other intermediate forms of organization result from the use of hierarchies/heterarchies; a hierarchy of mediated groups, etc.

Organization, Information, and Coordination

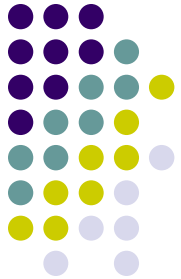
An organization is composed of organizational entities whose coordination is implemented through information flows and exchanges:

- **Organizational Entities**
  - Tasks, parts, machines, tools, ----- (“what”)
  - Functions, processes, ----- (“how”)
  - Schedules, schedulers, ----- (“when”)
  - Locations, destinations, ----- (“where”)

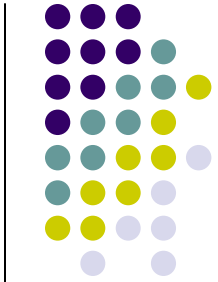
NOTE: All of these organizational entities can be represented by agents, in a multi-agent system.



- **Information**
  - Properties, status, -----
  - Conflicts, problems, -----
  - Goals, objectives, -----
  - Structures, membership, -----
  - Relationships, dependencies, -----
  - etc.
- **Coordination**
  - mutual adjustment
  - direct supervision
  - coordination by standardization
  - mediated coordination
  - reactive coordination



NOTE: *The structure of the organization can be based on one or more of the organizational entities, the information infrastructure, or the coordination mechanisms.*

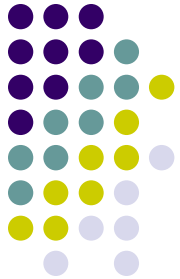


COOPERATION

Of the five coordination mechanisms considered, only two (mutual adjustment, mediated coordination) allows for agents to have autonomy, i.e. the freedom to make choices and determine their own actions. Agents which have some degree of autonomy can *cooperate* with other such agents on tasks and activities, for their own or mutual benefit.

Advantages of Cooperation

- Complete tasks quicker through shared effort
- By sharing resources, achieve tasks otherwise not possible
- Make use of complementary capabilities
- Avoid harmful interactions

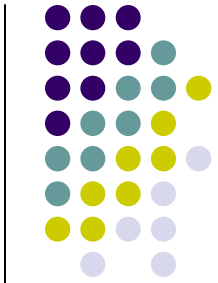


Modes of Cooperation

- accidental: not intended
- unilaterally intended: one agent intentionally helps another
- mutual cooperation: two or more agents intentionally collaborate

Degrees of Cooperation

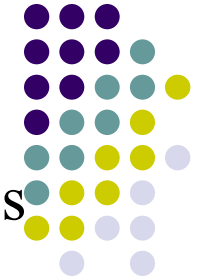
- Fully cooperative (benevolent): Agents always attempt to assist other agents that request or need their help
- Antagonistic: Agents do not cooperate with others and may even try to block their goals
- Partly cooperative: Agents sometimes or to some extent will assist other agents



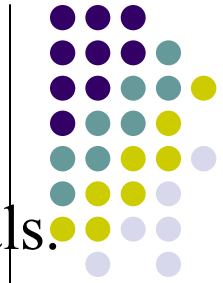
NEGOTIATION

The process of improving agreement (reducing inconsistency and uncertainty) on common viewpoints or plans through the structured exchange of relevant knowledge (Durfee, 1989)

- Bargaining (promising something in exchange for something else), bidding (offering a service or capability at a specific ‘price’), and contracting (committing to provide a service or capability at a specific ‘price’) may be part of the negotiation process.
- Prior to negotiation: an agent needs to identify a resource or service it requires but cannot supply and then it needs to identify an agent/s which it believes potentially could.

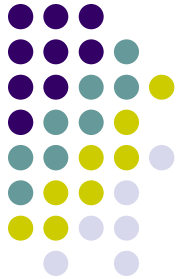


- Negotiation: Agent contacts other agents and identifies its need or requests resource or service (often under specified conditions); other agent/s identify what can be supplied under what conditions; initiating agent/s accepts conditions; supplying agent/s commit to provide resource or service.
  - In *single-stage* negotiation, initiating agent makes a request and the respondent accepts or rejects this.
  - In *multi-stage* negotiation, agents iterate through more than one stage of offer/counter-offer.
  - Negotiation protocols: a structured procedure for one or more stages of the negotiation process.



- Contract Net Protocol: “Agents coordinate their activities through contracts to accomplish specific goals. An agent acting as a *manager* decomposes its contract (the task or problem it was assigned with) into *subcontracts* to be accomplished by other *potential contractor* agents. For each subcontract, the manager announces a task to the network of agents. Agents receive and evaluate the announcement. Agents with appropriate resources, expertise, and information reply to the manager with *bids* that indicate their ability to achieve the announced task. The manager evaluates the bids received and awards the task to the most suitable agent, called the contractor. Finally, manager and contractor exchange information together during the accomplishment of the task.” (Moulin and Chaib-Draa (Course Text)

- In Contract Net, a contractor sends status (interim) reports, then final report (task completion) to manager. The manager may terminate a contract prematurely. Idle agents may broadcast their availability.
- Basic contract net model does not consider interacting tasks (e.g. two agents each need all of an available resource; several agents all need a resource or another agent's help at or about the same time)
- Possible approaches to interacting tasks:
  1. Agents negotiate directly on resolving problem.
  2. A mediator coordinates the interaction to reduce or eliminate the conflict, or negotiates with agents on resolving it.



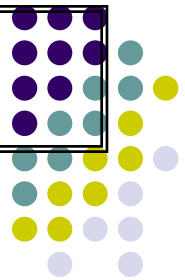
3. The agents not only *plan* how to accomplish their tasks/goals but *communicate* these to others and *negotiate* potential conflicts, modifying plans accordingly.

NOTE:

(a) Approach 3 is proactive. Approaches 1 and 2 can be simply reactive ways of dealing with a conflict problem *after* it has become evident (e.g. through conflicting interactions), or can be negotiation approaches used in 3's pro-active procedures.

(b) Above considers *negative* interactions of tasks, i.e. conflicts. Interactions of tasks can also be *positive*, by providing beneficial synergies. While that can be accidental, it can also be an advantage of using planning.

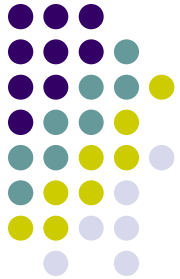
Middle Agents



Locating agents

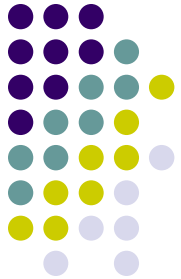
- In open systems, agents do not necessarily know others
- The connection problem: finding an appropriate agent to do a given task or provide a service
- Facilities and appropriate infrastructure are required to enable software agents to locate and come into contact with others

Matching



A MAS consists of

- End agents which can be
  - Providers of services/products
  - Requesters
- Middle agents
  - Enable interactions among end agents
  - **Purpose**: ***to match requesters with providers***

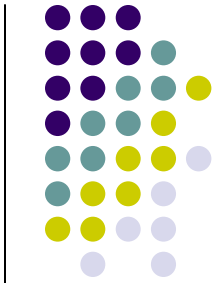


- Providers send advertisements or capability specifications which may include additional information on conditions of service
- Requesters send request specifications for required services which may include preference parameters
- Matching: the middle agent’s task is to match advertisements with request specifications as closely as possible

Two issues:

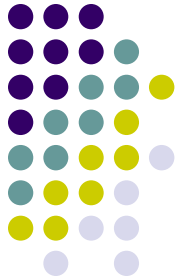
(i) Agents need to be able to communicate their advertisements and requests in a suitable language

(ii) Efficient matching mechanisms are required



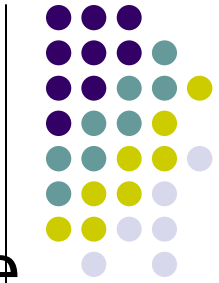
- An advertisement matches a request when the service described by the provider is *sufficiently similar* to the service requested by the requester
- Similarity can be defined at different levels

Requirements for matching engines



- Flexibility in recognizing the degree of similarity between advertisements and requests
- Make use of ontologies
- Minimize the false negative and false positive matches
- Encourage providers and requesters to be honest with their capabilities and needs respectively
- Operate on a nondiscriminatory basis and provide accurate information to all requesters

Interacting through a middle agent



The matching process can be divided into three phases

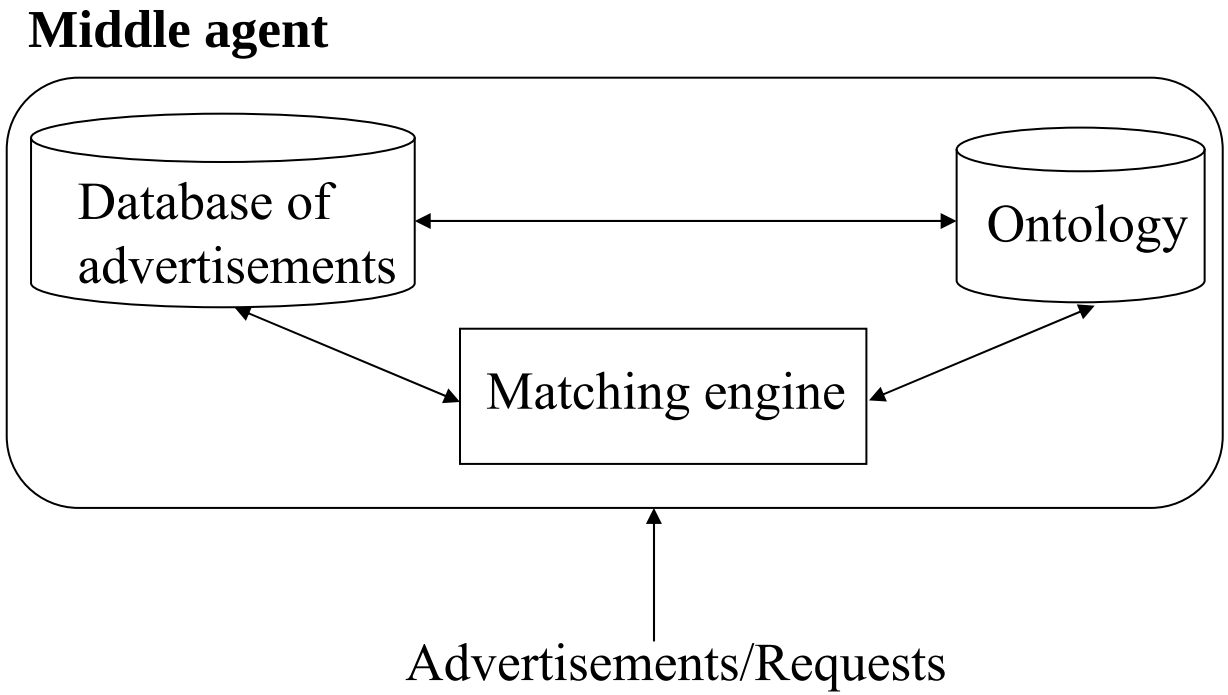
- Location
- Transaction
- Feedback

Some middle agents may take part in some of these phases or all

The transaction phase may take two forms:

- Providers and requesters interact with each other directly
- Providers and requesters interact via the middle agent

Middle agent architecture



Middle agent functionality

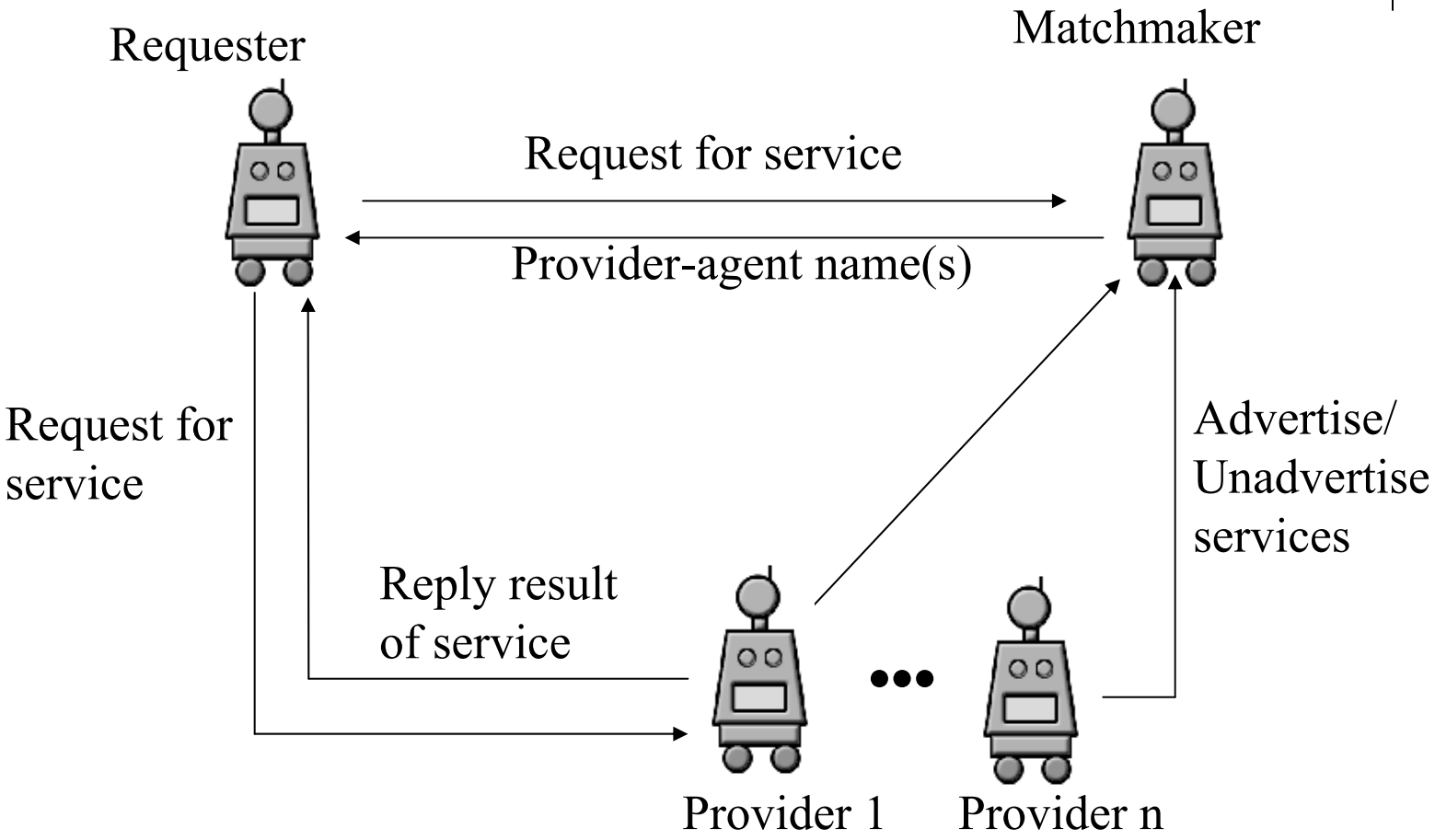
A middle agent's functionality can be characterized along the following dimensions:

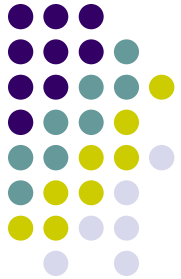
Who provides information to the middle agent

- How much and what sort of information is sent to the middle agent
- What happens to the information that the middle agent receives
- How is the information accessed
- How much information is specified in a query to the middle agent
- Mediation in the end agent transactions
- Collection of feedback

Different types of middle agent → different privacy guarantees!

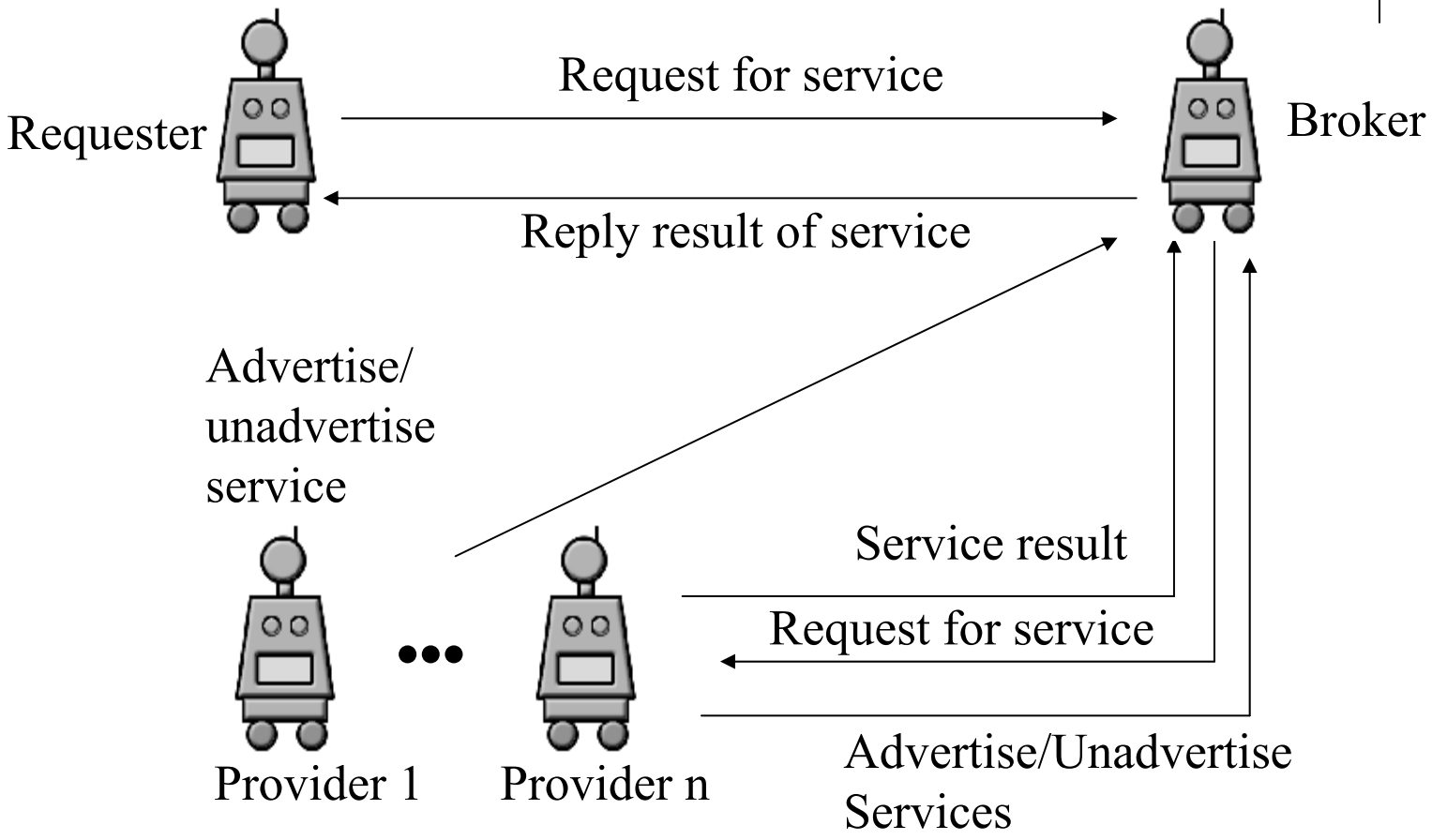
Matchmaker


Requester
Matchmaker
Request for service
Provider-agent name(s)
Request for service
Advertise/ Unadvertise services
Reply result of service
Provider 1
Provider n

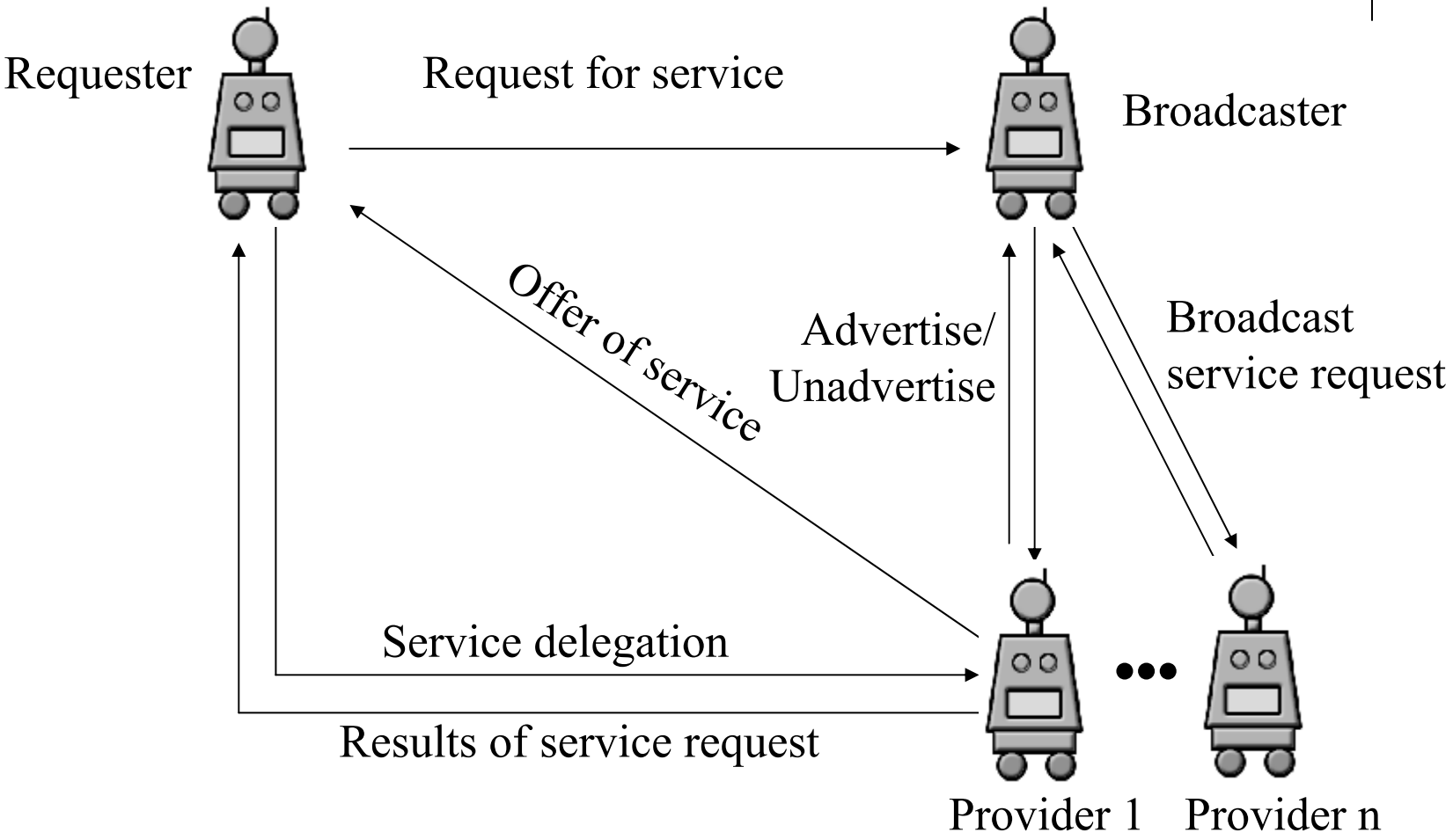


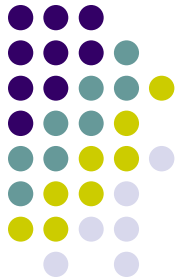
- The matchmaker can operate in two modes:
  - It attempts to perform an exact match and return the result to the requester – no local processing at the requester
  - It attempts to find all those advertisements that closely match the request. This list can be sorted according to the degree of match. Local processing is required at the requester to decide on further action

Broker


Requester
Request for service
Broker
Reply result of service
Advertise/
unadvertise
service
Service result
Request for service
Provider 1
Provider n
Advertise/Unadvertise
Services

Broadcaster





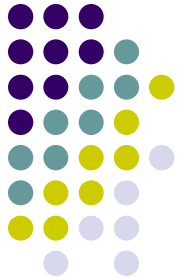
The broadcaster can operate in two modes:

- General broadcast: all agents get all requests – no processing of the requests
- Specialized broadcast: only those requests judged relevant to a provider are forwarded – processing of the requests is required, i.e. matching

FIPA Directory Facilitator

- The FIPA Directory Facilitator (DF) is an essential component of an Agent Platform
- A DF provides a yellow pages directory service to other agents
- A trusted entity and its remit is to maintain an accurate and complete list of agents and provide the most current information on a nondiscriminatory basis
- The act of registering with a DF, does not imply any further obligations for the registered agent: it can refuse a request
- The information registered with the DF is not checked, therefore no guarantees are provided regarding its accuracy

Agent Capability Description Languages

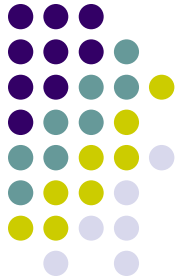


- A language that enables providers and requesters to express their capabilities and needs is needed
- Desired properties for a capability description language
  - Flexibility and expressiveness
  - Ability to express semi-structured data
  - Ability to make inferences and comparisons
  - Ability to express constraints
  - Ease of use

LARKS

- The Language for Advertisement and Request for Knowledge Sharing (LARKS) is an Agent Capability Description Language
- A specification in LARKS is a frame which is wrapped up in a KQML performative which indicates whether it is an advertisement or a request

LARKS specification



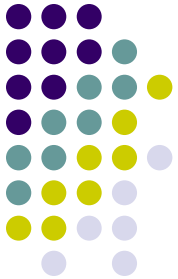
A LARKS specification includes:

- The context (Context) of the advertisement or request specification, i.e. the semantic domain of the service
- The data types used in the specification (Types)
- The input and output parameters (Input, Output) as well as
- The constraints on these which take the form of Horn clauses (InConstraints, OutConstraints)
- A textual description of the specification of service or request (TextDescription)
- The concept descriptions (ConcDescriptions)

Example specification

FindLaptopInfo	
Context	Laptop*Laptop
Types	Infolist=Listof(model:Model*LaptopModel,brand:Brand*Brand, price:Price*Money, colour:Colour*Colours);
Input	brands: SetOf Brand*Brand; processor: SetOf CPU*CPU; priceLow*LowPrice:Integer; priceHigh*HighPrice:Integer;
Output	Info:InfoList;
InConstraints	
OutConstraints	sorted(Info)
ConcDescriptions	Laptop=(and Product (exists has processor CPU) (all has-memory Memory) (all is-model LaptopModel)); LowPrice=(and Price (ge 700) (exists in-currency aset (GBP))); HighPrice=(and Price (le 2500) (exists in-currency aset (GBP))); LaptopModel=aset(Dell,IBM,Samsung,Sony,Toshiba); CPU=aset(PentiumM,Centrino,AMDAthlon);
TextDescription	Find information about laptops.

Types of matching

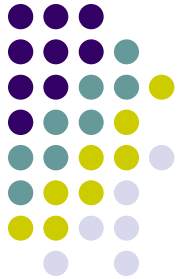


Exact match

- Descriptions are equivalent (literally equivalent, by renaming the variables, or equivalent logically)
- Most restrictive type of matching, but most accurate

Plug-in match

- Less accurate but more useful type of match
- The agent whose capability description matches a given request can be ‘plugged into the place’ where the request was made
- A pair of request and advertisement specifications can differ in the signatures of the their inputs/outputs, the number of constraints, or the constraints themselves



Relaxed match

- Least accurate, but most useful match
- It determines how close the advertisement and the request are by returning a numerical distance value: if the distance value is smaller than a preset threshold, then the two specifications match

Matching methods in LARKS

- Proposed as a part of a matchmaker agent, the following filters are used in LARKS to provide for accurate, efficient, and effective matching:
  - Context matching
  - Profile comparison
  - Similarity matching
  - Signature matching
  - Constraint matching

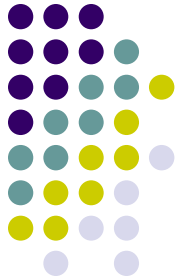
Relaxed match: Context matching, Profile comparison

Plug-in match: Similarity matching, Signature matching

Exact match: Context matching, Profile comparison, Similarity matching, Signature matching, Constraint matching

- The user may decide to use any combination of filters

Context matching filter

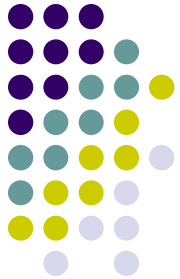


- Filters out any capability specifications that are not relevant to the current request
- The similarity of the semantic domain of two specifications is checked in two steps

Step 1

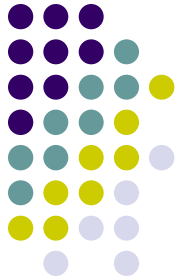
- For every pair of words *u*, *v* that are part of the Context slots in the specifications, the word distances $d_w(u,v) \in [0,1]$ are computed as real values
- The most similar matches for any word *u* are determined by selecting words *v* with the minimum distance $d_w(u,v)$
- The word distance is computed using the trigger-pair model

Step 2



- For every pair of most similar matching words, the semantic distance among the attached concepts is checked
- To compute the semantic distance, a weighted associative network with directed edges between concepts as nodes is used
- The edges denote the kind of binary relation (generalization, specialization, and positive association) between two concepts and also include a numerical weight which indicates the strength of belief in the relations

Profile comparison filter



- The profile comparison filter uses the term frequency-inverse document frequency weighting (TF-IDF) technique
- Advertisements and requests are considered to be documents
- $h(w,d) = wf(w,d) \cdot \log(\frac{|D|}{df(w)})$ word *w* appears throughout the collection of documents *D* is called the document frequency *df*(*w*) of *w*
- A weight determines the significance of the classification of *w* for *d*:

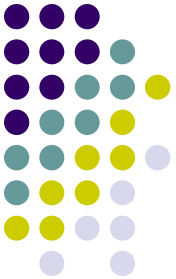
- The weighted keyword representation *wkv*(*d,Dictionary*) of *d* contains for every word *w* in dictionary *Dictionary* the weight *h*(*w*,*d*) as an element
- The similarity between a request and an advertisement is:

$$dps(Request, Advertisement) = \frac{Request \cdot Advertisement}{|Request| \cdot |Advertisement|}$$

where *Request* ·*Advertisement* is the inner product of the weighted keyword vectors

- If the similarity value exceeds a threshold then the specifications are considered similar

Similarity matching filter



- A combination of distance values are calculated for pairs of input and output declarations and constrains
- The distance values are computed in terms of the distance between concepts and words that occur in the particular parts of the

$$Similarity(E_i, E_j) = 1 - ((\sum_{(u,v) \in S(E_i) \times S(E_j)} d_w(u,v)/|S(E_i) \times S(E_j)|))$$

- The similarity between two variable declarations or constraints E_i and E_j where $S(E)$ denotes the set of words in E:

$$Similarity(S_a, S_b) = \frac{\sum_{(E_i,E_j)\in(D(S_a)\times D(S_b))\cup(C(S_a)\times C(S_b))} Similarity(E_i, E_j)}{|(D(S_a) \times D(S_b)) \cup (C(S_a) \times C(S_b))|}$$

- The similarity between two specifications *Similarity*(S_a,S_b) is calculated as the average of the sum of similarity computations among all pairs of declarations and constraints

where

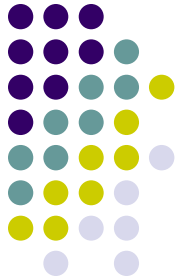
D(*S*) is the input/output declaration

C(*S*) is the input/output constraint parts of a specification *S*

Signature matching filter

- It first checks the declaration parts of the request and advertisement specifications and determines pairwise if their signatures of input and output variable types match
- This is carried out by checking a set of subtype inference rules that determine when a type is a subtype of another
- The matching of two signatures *sig*$_1$ and *sig*$_2$ is defined by *fsm*

$$fsm(sig_1, sig_2) = \begin{cases} sub\ sig_2 \preccurlyeq_{subtype} sig_1 \\ Sub\ sig_1 \preccurlyeq_{subtype} sig_2 \\ eq\ sig_1 =_{subtype} sig_2 \\ disj\ else \end{cases}$$



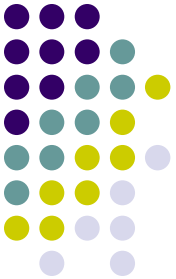
- Two declarations D_i and D_j syntactically match if they are sufficiently similar (where γ is a threshold value):

  $Similarity(D_i, D_j) \geq \gamma \wedge fsm(D_i, D_j) \neq disj$

- Two constraints C_i and C_j syntactically match if they are sufficiently similar

  $Similarity(C_i, C_j) \geq \gamma$

Syntactical matching of specifications



- The profile comparison, similarity and signature matching methods contribute to determine the syntactical matching of two specifications
- Two specifications S_a and S_b syntactically match if
  1. Their profiles match, i.e. $dps(S_a,S_b) \geq \beta$
  2. The declarations and constraints match

  $Syntactical(E_i, E_j) \wedge Similarity(E_i, E_j) =$

  $max\{Similarity(E_i, E_j), k \in \{1,\ldots,n_b\}\}$
  3. For each pair of declarations (D_i, D_j) determined in (2) the matching of their signatures is of the same type, i.e. the value of $fsm(D_i, D_j)$ is the same
  4. The similarity value $Similarity(S_a, S_b)$ exceeds a threshold

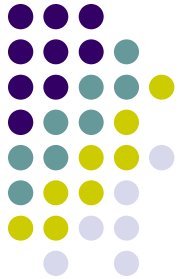
Constraint matching filter

- Two specifications $S_a(Pres_{S_a}, Post_{S_a})$ and $S_b(Pres_{S_b}, Post_{S_b})$ match in terms of constraints if:

  $(Pre_{S_a} \Rightarrow Pre_{S_b}) \wedge (Post_{S_a} \Rightarrow Post_{S_b})$

- Logical implication among constraints in LARKS is computed using polynomial subsumption checking for Horn clauses (since logical implication of clauses is undecidable)

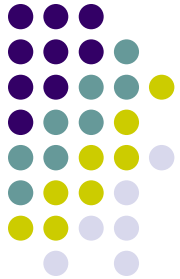
Semantical plug-in match



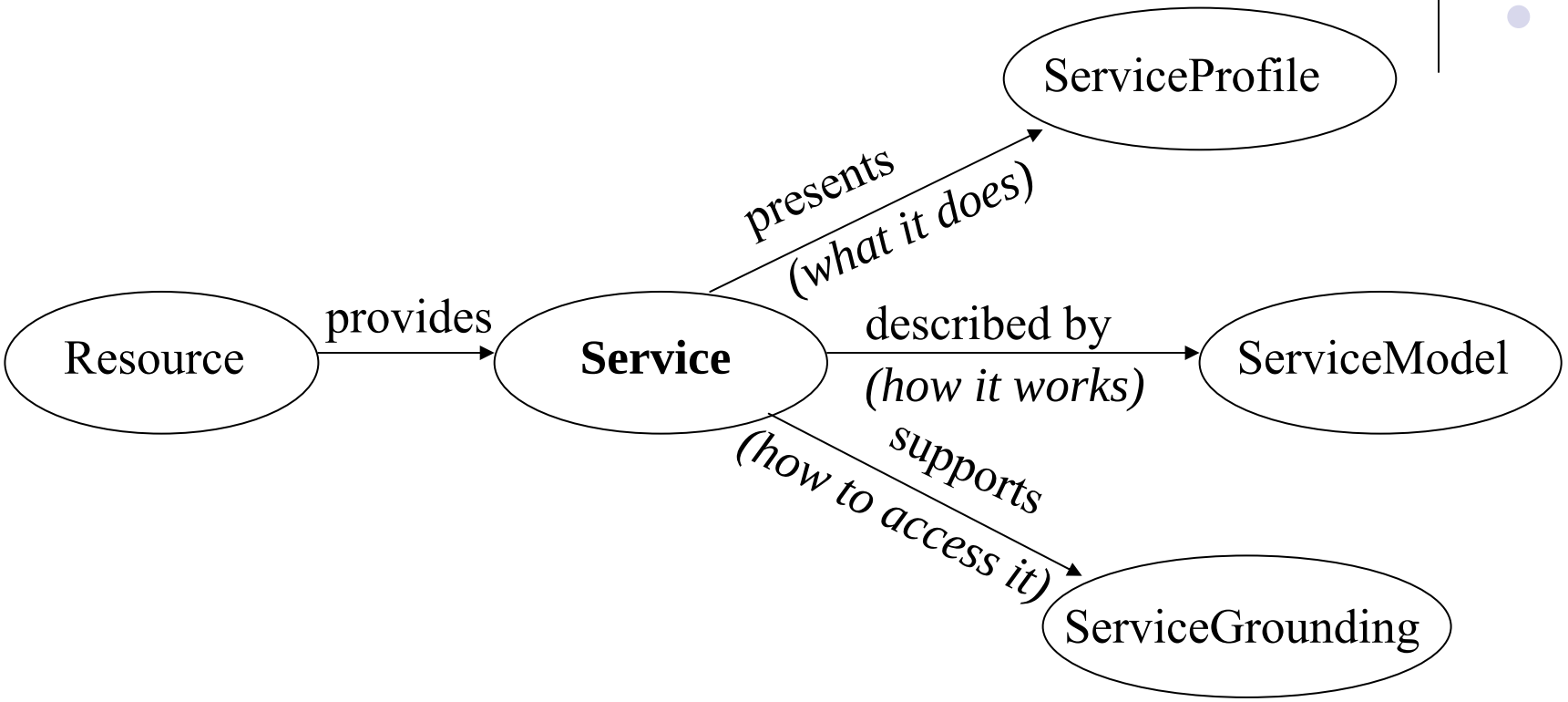
A specification S_b semantically plug-in matches a specification S_a if:

- The signatures of their variable declaration parts match
- The set of input constraints of S_a logically implies that of S_b
- The set of output constraints of S_b logically implies that of S_a

OWL-S

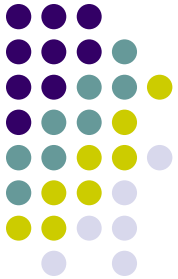


- Advertising capability specifications and submitting requests is a related problem in the context of web services
- The Ontology Web Language for Services (OWL-S) is a framework for describing the capabilities of web services
- OWL-S is organized around three interrelated sub-ontologies
  - The *ServiceProfile*
  - The *ServiceModel*
  - The *ServiceGrounding*



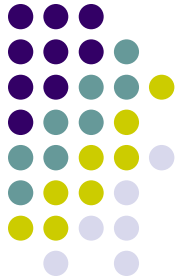
- A service can be described by at most one service model
- A grounding must be associated with exactly one service

ServiceProfile



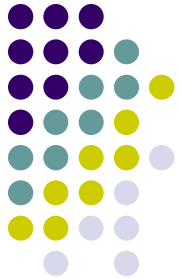
- ‘What the service does’
- This is the capability specification of the service; essential for matching purposes
- May include limitations on service applicability and quality of service as well as any requirements that the requester must satisfy to use the service successfully (authentication etc.)

ServiceModel



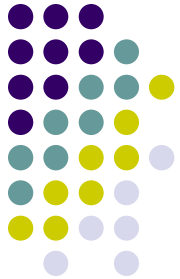
- ‘How a service works’
- How to ask for it and what happens when the service is carried out
- Used to enable invocation, enactment, composition, monitoring and recovery
- A requester or middle agent can use the *ServiceModel* to:
  - examine in-depth whether the service meets its needs
  - compose service descriptions from multiple services to perform a specific task
  - coordinate the activities of the different participants during the course of the service enactment
  - monitor the execution of the service

ServiceGrounding



- Provides the details of how a requester or middle agent can interact with or access the service
- It specifies the communication protocol, message formats and other service-specific details such as port numbers
- This information can be expressed in the Web Services Description Language (WSDL)
- For each semantic type of input or output specified in the *ServiceModel* it must also specify an unambiguous way of exchanging data elements of that type with the service, i.e. the serialization techniques employed

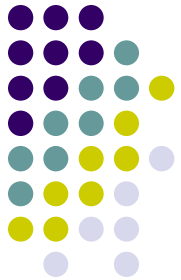
Matching in OWL-S



An advertisement specification matches a request when

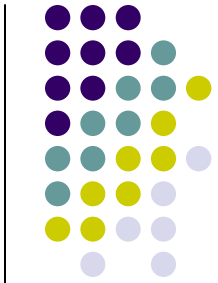
(i) all the outputs of the request are matched by the outputs of the advertisement

(ii) all the inputs of the advertisement are matched by inputs of the request



Matching between output parameters

```
outputMatch(outputsRequest,outputsAdvertisement){
    globalDegreeMatch = exact
    forall outputR in outputsRequest do {
        find outputA in outputsAdvertisement such that
            degreeMatch = MaxDegreeMatch(outputR,outputA)
            if (degreeMatch<globalDegreeMatch)
                globalDegreeMatch = degreeMatch
    return sort(recordMatch);}
```



The degree or level of a service match is determined by the degree of parameter matches:

```
degreeOfMatch(outputR, outputA):
    if outputA = outputR then return exact
    if outputR subclassOf outputA then return exact
    if outputA subsumes outputR then return plugIn
    if outputR subsumes outputA then return subsumes
        else return fail
```

The degree of match of the input of an advertisement and the input of a request is decided in a similar way

- The ensuing degrees of match between a request and a set of advertisement specifications are organized from most to least preferable (exact, plug-in, subsumes, fail)
- The resulting matches are then sorted using as criterion the highest score in the outputs as this indicates the extent to which the provider can carry out the requested service
- The matching of input parameters is only taken into account in order to break possible ties