AGENT INTERACTION MODELS: THE CURRENT LANDSCAPE

GIUSEPPE VIZZARI

8/3/2019

Agents *interaction*

"An interaction occurs when two or more agents are brought into a dynamic relationship through a set of reciprocal actions"

(J. Ferber, Multi Agent Systems - 1999)

Interaction is required when it is not possible for a single agent to carry out its task(s)

Main reasons

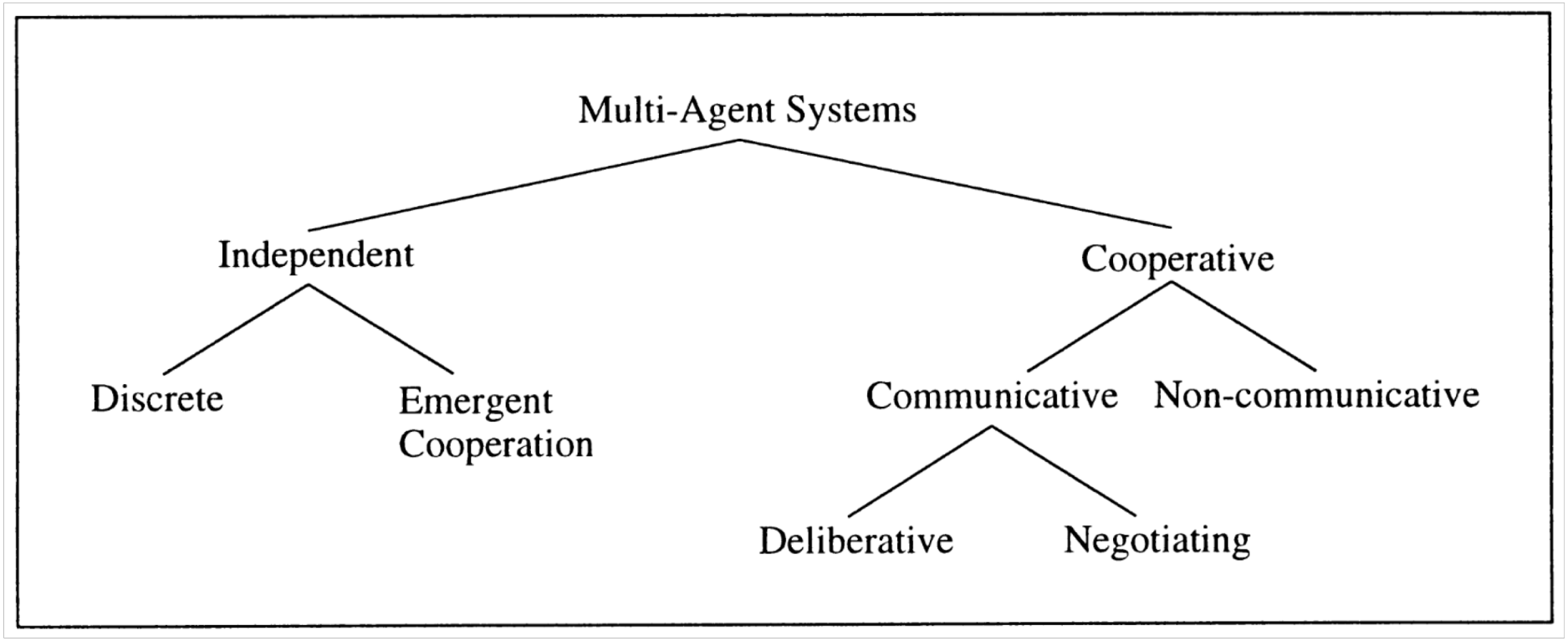
- Insufficient resources (e.g. information)
- Resources may be intrinsically distributed in an environment
- Insufficient abilities (e.g. not enough knowledge, computational power)
- Convenience of a distributed approach (suitability of the modelling approach, suitability to the application scenario)

Interactions: a possible schema

Goals	Resources	Abilities	Type
Compatible	Sufficient	Sufficient	Independence
Compatible	Sufficient	Insufficient	Simple collaboration
Compatible	Insufficient	Sufficient	Obstruction
Compatible	Insufficient	Insufficient	Coordinated collaboration
Incompatible	Sufficient	Sufficient	Pure individual competition
Incompatible	Sufficient	Insufficient	Pure collective competition
Incompatible	Insufficient	Sufficient	Individual conflicts over resources
Incompatible	Insufficient	Insufficient	Collective conflicts on resources

[Ferber – Multi agent systems - 1999]

A different schema

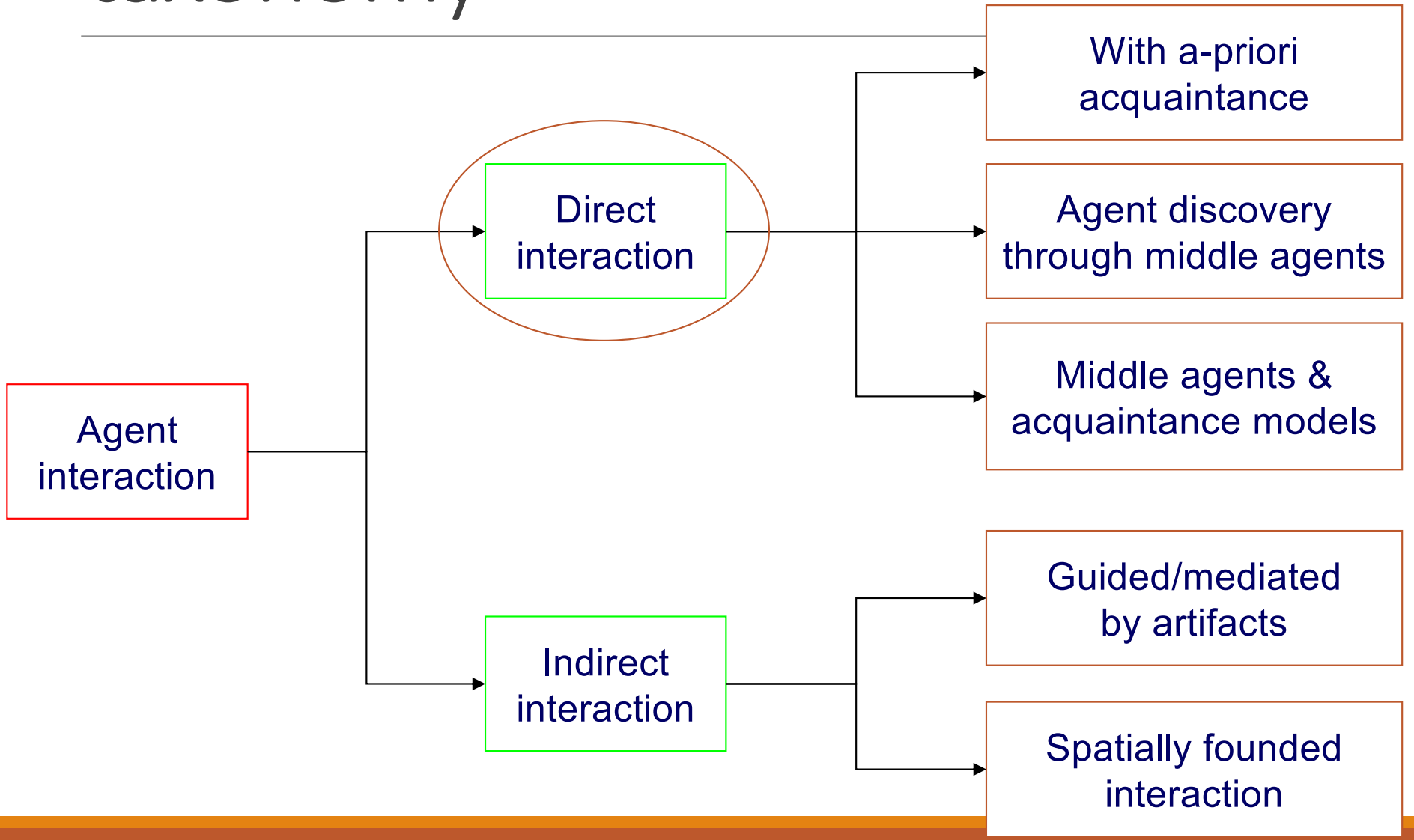


DORAN, J., FRANKLIN, S., JENNINGS, N., & NORMAN, T. (1997). On cooperation in multi-agent systems. *The Knowledge Engineering Review, 12*(3), 309-314.

Interaction models

- Agent internal architecture can be separated by the (*interaction*) model that defines the way agents communicate
- This approach allows the modelling, design and implementation of heterogeneous entities, sharing an *environment* in which they can interact
- Many different interaction models have been defined and implemented
- Often inspired by other disciplines (e.g., social science, linguistics, biology)

Interaction models: a taxonomy



Direct interaction models

Agents are able to *directly exchange information*

Information exchange

- *Communication/conversation rules* ("protocol") → Agent Communication Language (ACL)
- *Message structure* (shared ontology) → Content Language

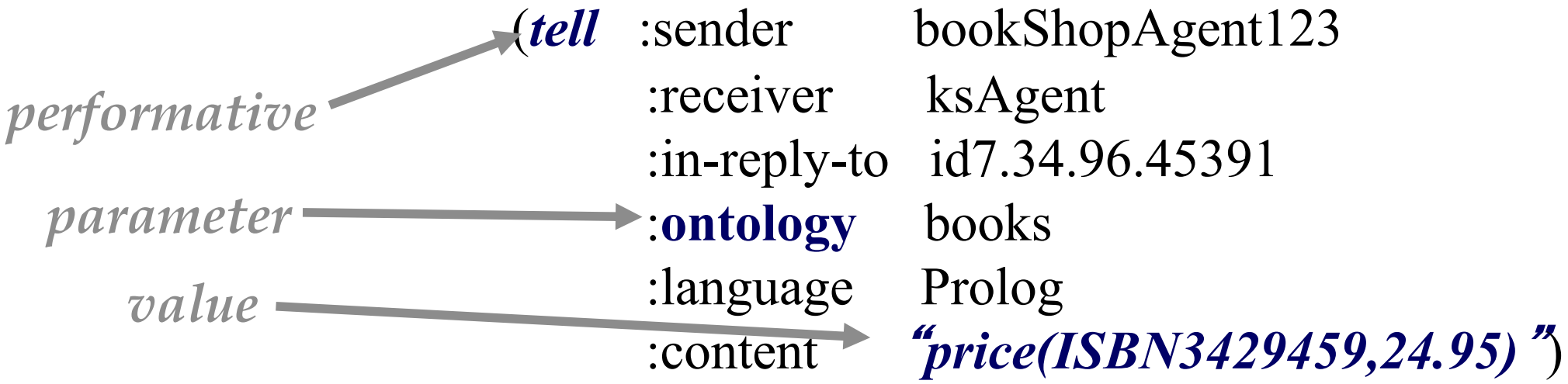
Information exchange is *indiscriminate*

- Once an agent knows another one, it will be able to communicate with it
- No external, contextual factors are considered

Direct interaction model example: KQML

- *Knowledge Query and Manipulation Language (KQML)* and *Knowledge Interchange Format (KIF)* are results of the ARPA Knowledge Sharing Effort
- KQML is an ACL, a high level interaction language
- KIF is a content language, defining syntax of contents
- KQML defines *performatives*, that is, basic messages to compose conversations among agents
- KIF allows to represent *information and knowledge* about agents, beliefs, desires, intentions, perceptions plans and thus their environment
- Agents must share an *ontology*, in terms a common vocabulary and agreed upon meanings to describe a subject domain

A KQML Message



The above message represents a single KQML *speech act* described by a list of *attribute/value pairs e.g.* :content, :language, :from, :in-reply-to.

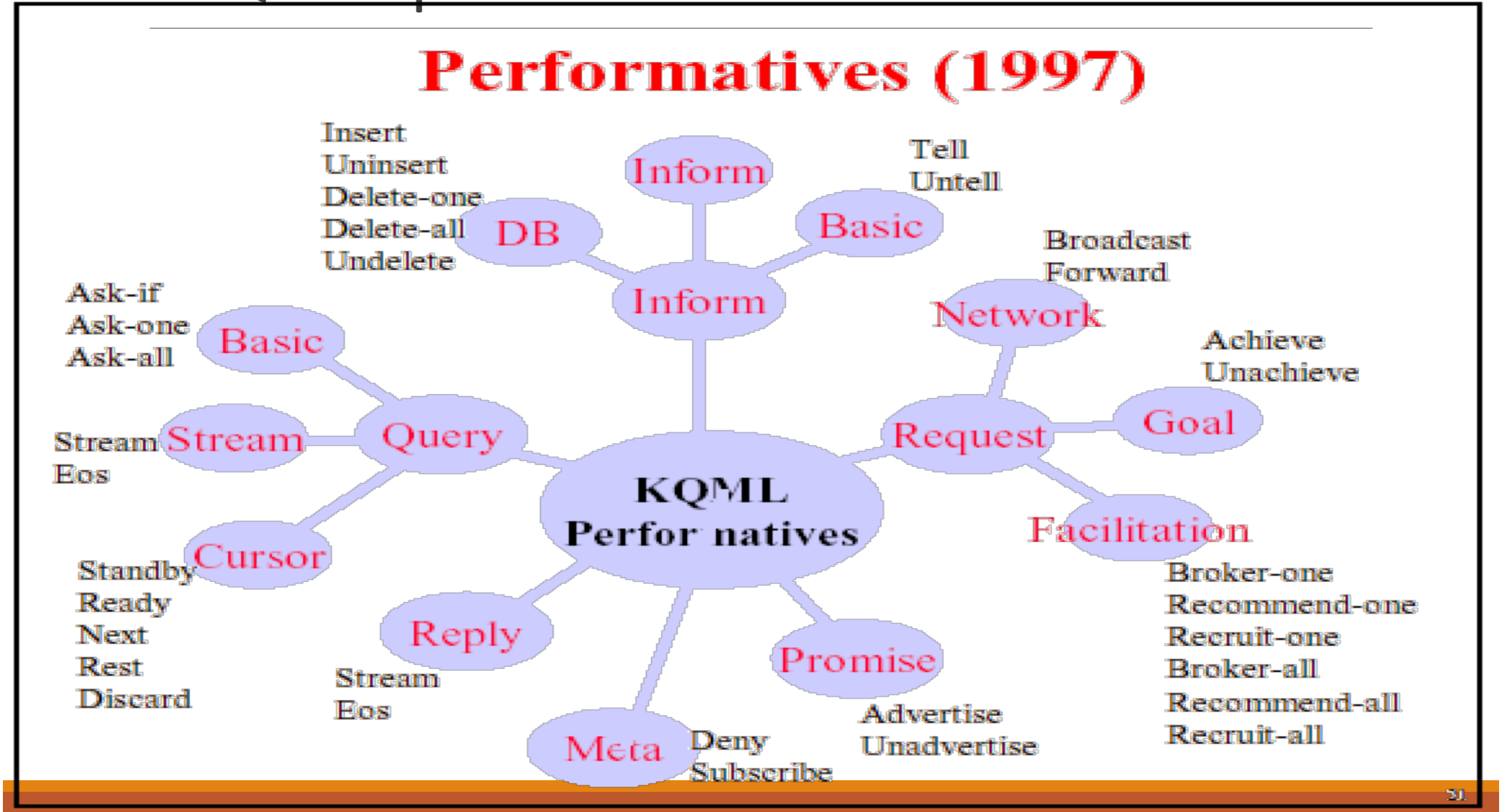
A KQML Dialogue

Consider agents A and B "talking" about the prices of books bk1 and bk2:

```
A to B: (ask-if (>(price bk1)(price bk2)))
B to A: (reply true)
B to A: (inform (= (price bk1) 25.50))
B to A: (inform (= (price bk2) 19.99))
```

For convenience message format above is simplified and attribute/value pairs for :ontology etc. are omitted.

KQML performatives



Some requirements

Agents need to *know their communication partners*

- Common approach is to have *specific facilitators* that are known by every agent and allow them to get acquainted
- Problems: how many of those 'middle agents' (robustness)? How to keep the aligned?

A semantic must be defined to obtain/enforce meaningful conversations

- Agent considered as a *logical reasoner* with beliefs, desires and intentions
- Pre and post conditions defined in terms of a of *logic formalization*
- Actualization of postconditions triggers preconditions of other performatives
- What about autonomy?

Other tools for communication semantics

The specification of conversations can be done through several formal models

- *Finite State Machines* based
- *Petri nets* based

The former approach has been widely used to model, analyze and demonstrate properties of network protocols

These appraches also limit agents' autonomy

Direct interaction models: pros

Similarity to existing protocols for distributed systems

- Point-to-point massage passing
- Easy implementation on top of existing middleware platforms

Simple *integration with deliberative agents* approach

- Agents exchange facts conforming to some kind of formally defined ontology

Formal semantics of ACLs can be easily specified

- Communication semantics is related to agents' beliefs, decisions, intentions

Direct interaction models: cons

Information exchange occurs according to specific rules
- Network protocol like issues (conversation rules, message formats)

→ *Semantical issues*
- communication semantics related to agent internals (beliefs, decisions, intentions)
- normative semantics limits agents' autonomy

Exchanged information must conform to an ontology that is somehow shared by the agents

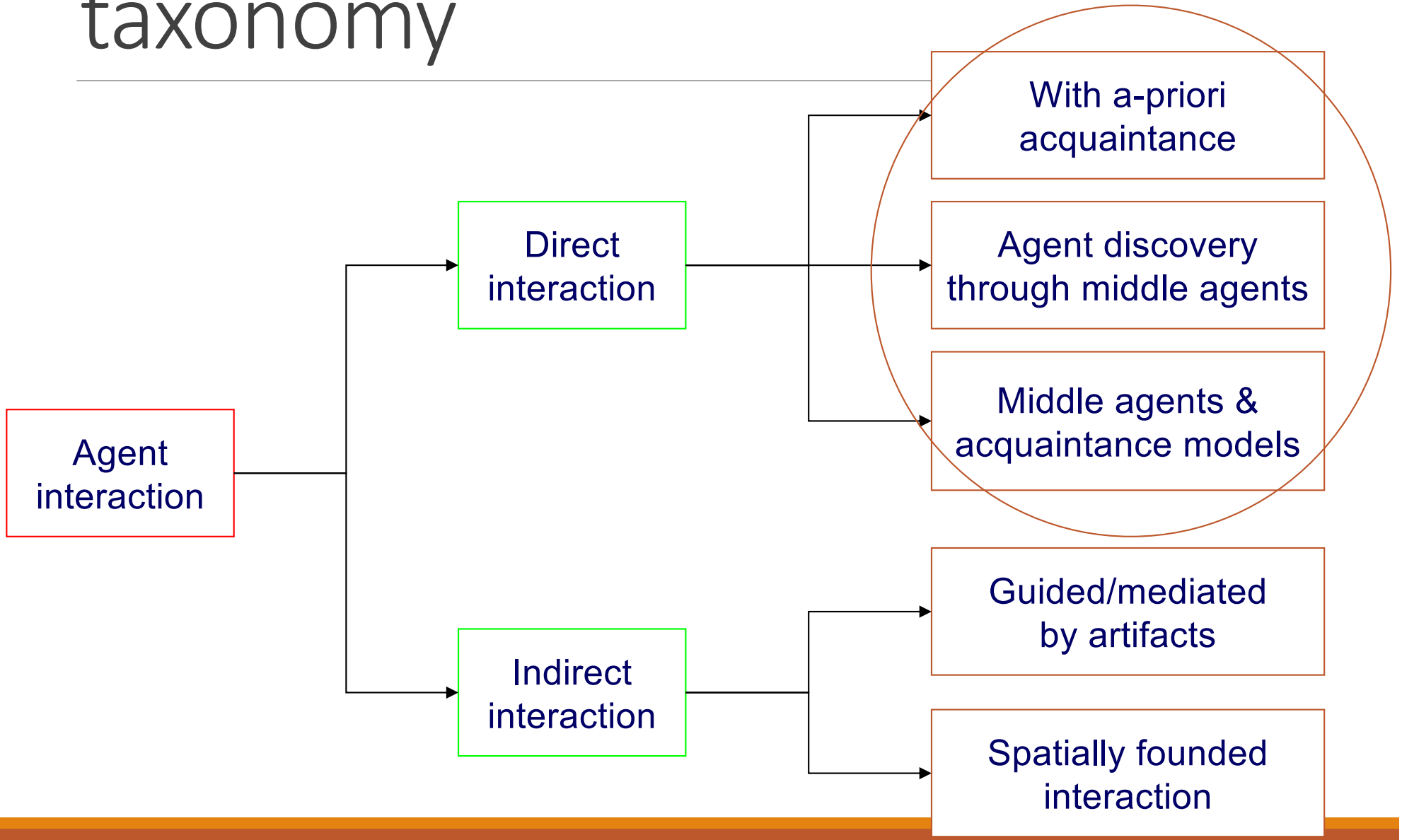
→ *Ontology issue*

Agents need to be aware of the presence of a communication partner

→ *Discovery issue*

Direct interaction models do not provide abstractions to represent elements of agents *context*

Interaction models: a taxonomy

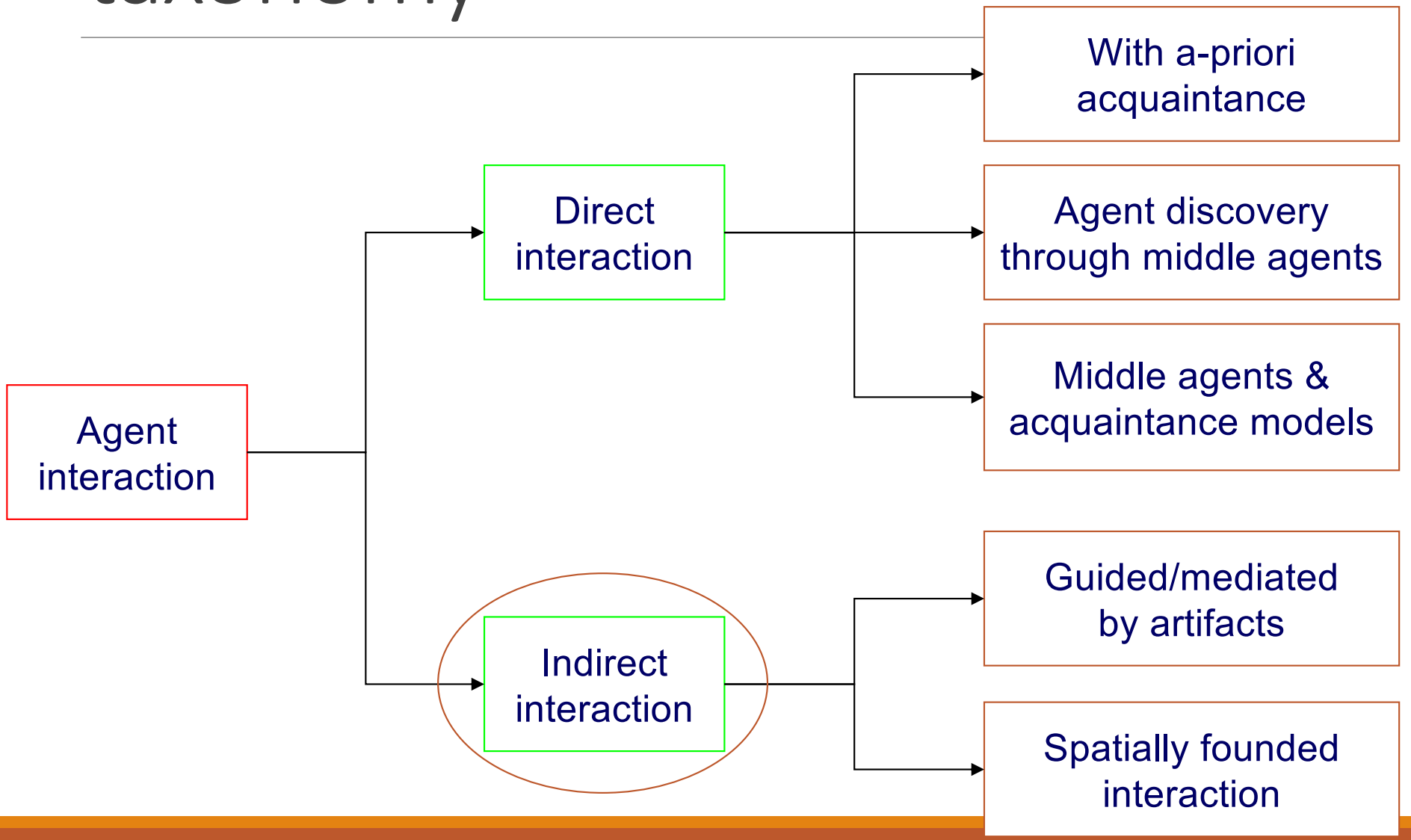


Direct interaction models: some enhancements

Discovery issue and agent context

- *Middle agents* as specific agents collecting and providing acquaintance information to entities of the system
- Not a single middle agent, but a *network* of them, organized in order to provide robustness and structure
- Not just mere agent name service, but information on *provided services* (and possibly other contextual information?)

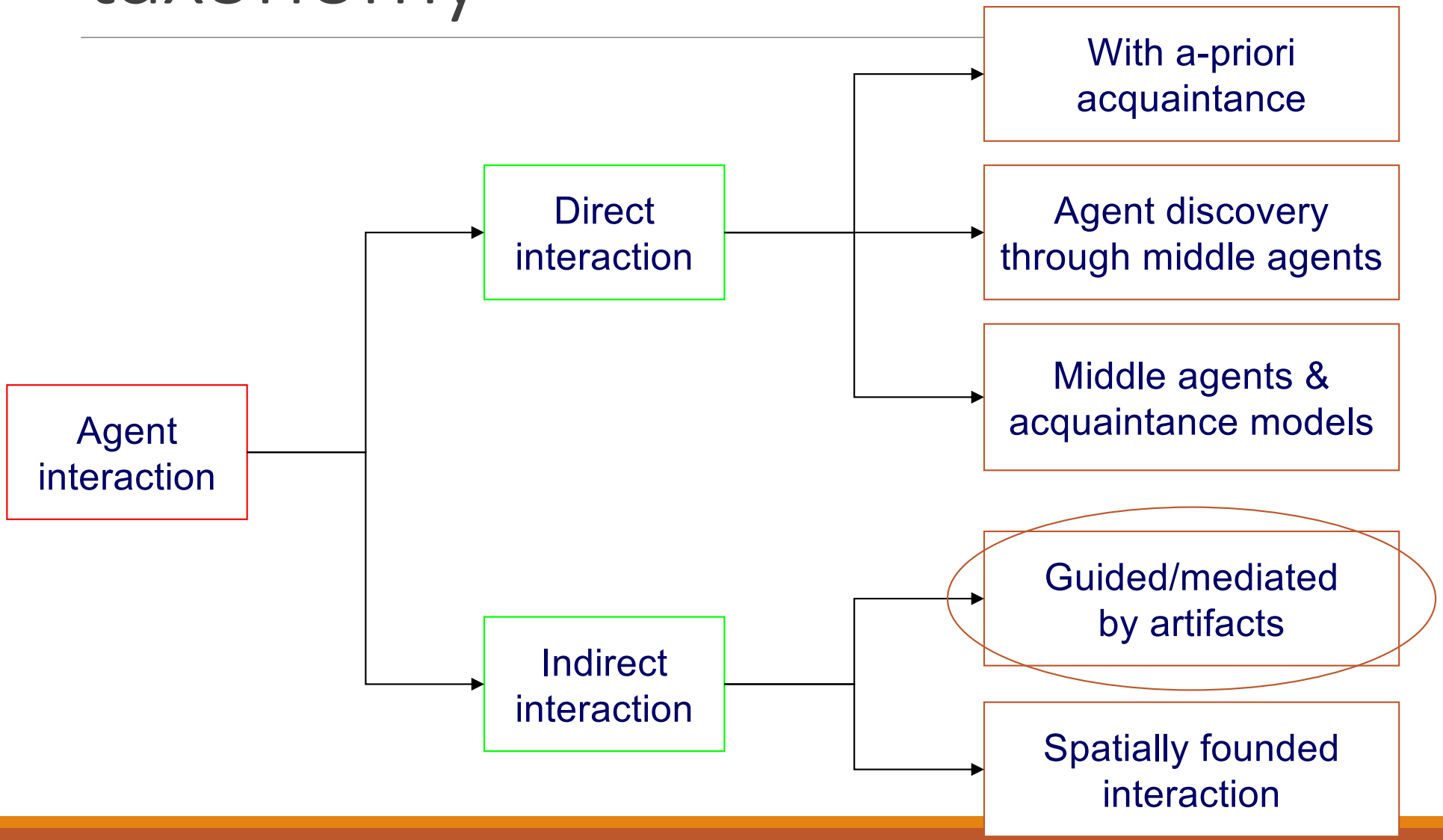
Interaction models: a taxonomy



Indirect interaction models

- Agents interact through an *intermediate* entity
- This medium supplies specific *interaction mechanisms* and *access rules*
- These rules and mechanisms define agent *local context* and perception
- *Time and space uncoupling*
- *Name uncoupling*

Interaction models: a taxonomy



Artifact-mediated interaction

Agents access a *shared artifact* that

- they can *observe*
- they can *modify*

Such artifact is a *communication channel* characterized by an intrinsically broadcast transmission

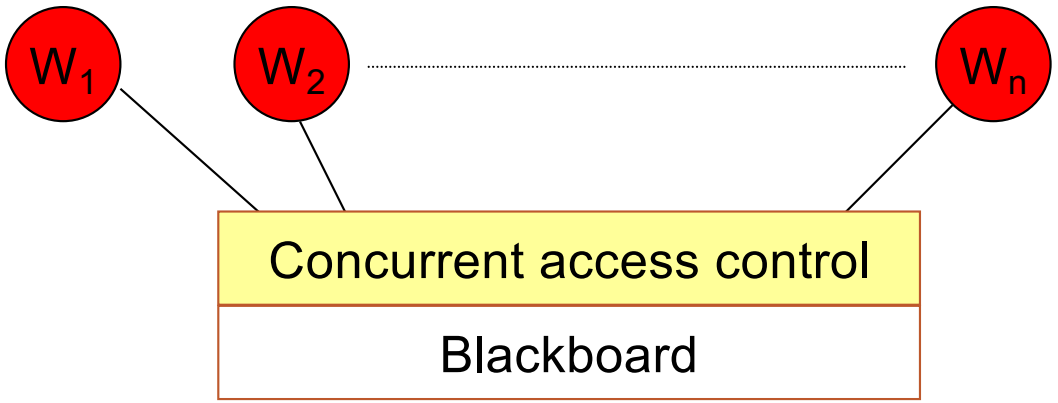
Specific *laws regulating access* to this medium

It represents a part of agents' *environment*

Blackboard systems

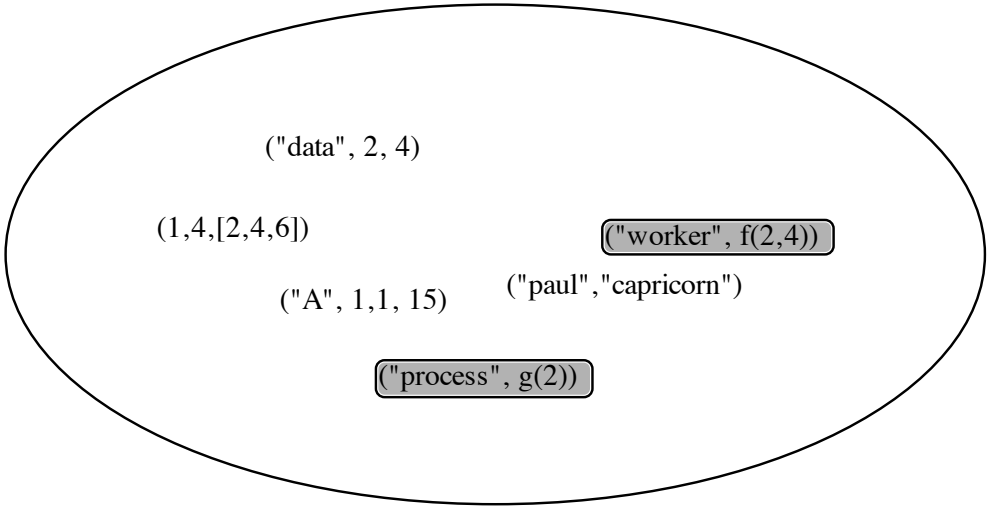
"Metaphorically we can think of a set of workers, all looking at the same blackboard: each is able to read everything that is on it, and to judge when he has something worthwhile to add to it."

(A. Newell, 1962)



Linda as a coordination model

The Tuple Space is a global computing environment conceptually including both data, in form of *passive* tuples, and agents, in form of *active* tuples



All tuples are created by agents; they are atomic data (they can only be created, read or deleted)

Agents cannot communicate directly:

an agent can only read (`rd`) or consume (`in`) a tuple, write (`out`) a new tuple or create (`eval`) a new agent that when terminates becomes a data tuple

Linda: a specific blackboard based system

Tuple space: a sort of blackboard in which *tuples* (record-like data structures) can be inserted, inspected and extracted by agents

Operations

- out(t) puts a new tuple in the Tuple Space, after evaluating all fields of t; the caller agent continues immediately
- in(t) looks for a tuple matching the pattern t in the Tuple Space; if not found the agent suspends; when found, reads and deletes it
- rd(t) looks for a tuple matching the pattern t in the Tuple Space; if not found the agent suspends; when found, reads it
- inp(t) looks for a tuple matching the pattern t in the Tuple Space; if found, deletes it and returns TRUE; if not found, returns FALSE
- rdp(t) looks for a tuple matching the pattern t in the Tuple Space; if found, copies it and returns TRUE; if not found, returns FALSE

Linda examples

Example:

```
out("string", 10.1, 24, "another
        string")

real f; int i;
rd("string", ?f, ?i, "another
        string")
```

succeeds

```
in("string", ?f, ?i, "another
        string")
```

succeeds

```
rd("string", ?f, ?i, "another
        string")
```

does NOT succeed

Example:

```
out(1,2)
rd(?i,?i)
```

does not succeed

Example:

```
eval("worker",7,exp(7))
```

creates an active tuple

```
in("worker",?i,?f)
```

succeeds when eval terminates

Example:

```
eval("double work", f(x), g(y))
in("double work", ?h, ?k)
```

succeeds when both active fields terminate

Example of coordination pattern: master worker

```
master(){

for all tasks {
   /* build task structure for this iteration */
   ...
   out("task", task_structure);
   }

for all tasks {
   in("result",?&task_id,?&result_structure);
   /* update total result using this result */
   ...
   }

}

worker(){
   while(inp("task",?&task_structure){
   /*exec task*/
   ... out("result,task_id,result_Structure);

}
```

Dining philosophers

```
#define NUM 5
phil(int i){
          while(1) {
                    think();
                    in("room ticket");
                    in("fork", i);
                    in("fork", (i+1)%NUM);
                    eat();
                    out("fork", i);
                    out("fork", (i+1)%NUM);
                    out("room ticket");
          }
}

real_main() {
          int i;
          for (i=0, i<NUM, i++){
                    out("fork", i);
                    eval(phil(i));
                    if (i<(NUM-1)) out("room ticket");
          }
}
```

From Linda, to mobility and beyond

Distributed tuple spaces: these systems allow to have a conceptually shared tuple space that is spread in a distributed environment

More than just distribution

- *Programmable, reactive* tuple spaces: adding a behaviour to tuple spaces
- Including *organizational abstractions* (roles, policies) to enhance access rules

Artifact-mediated interaction models: pros and cons

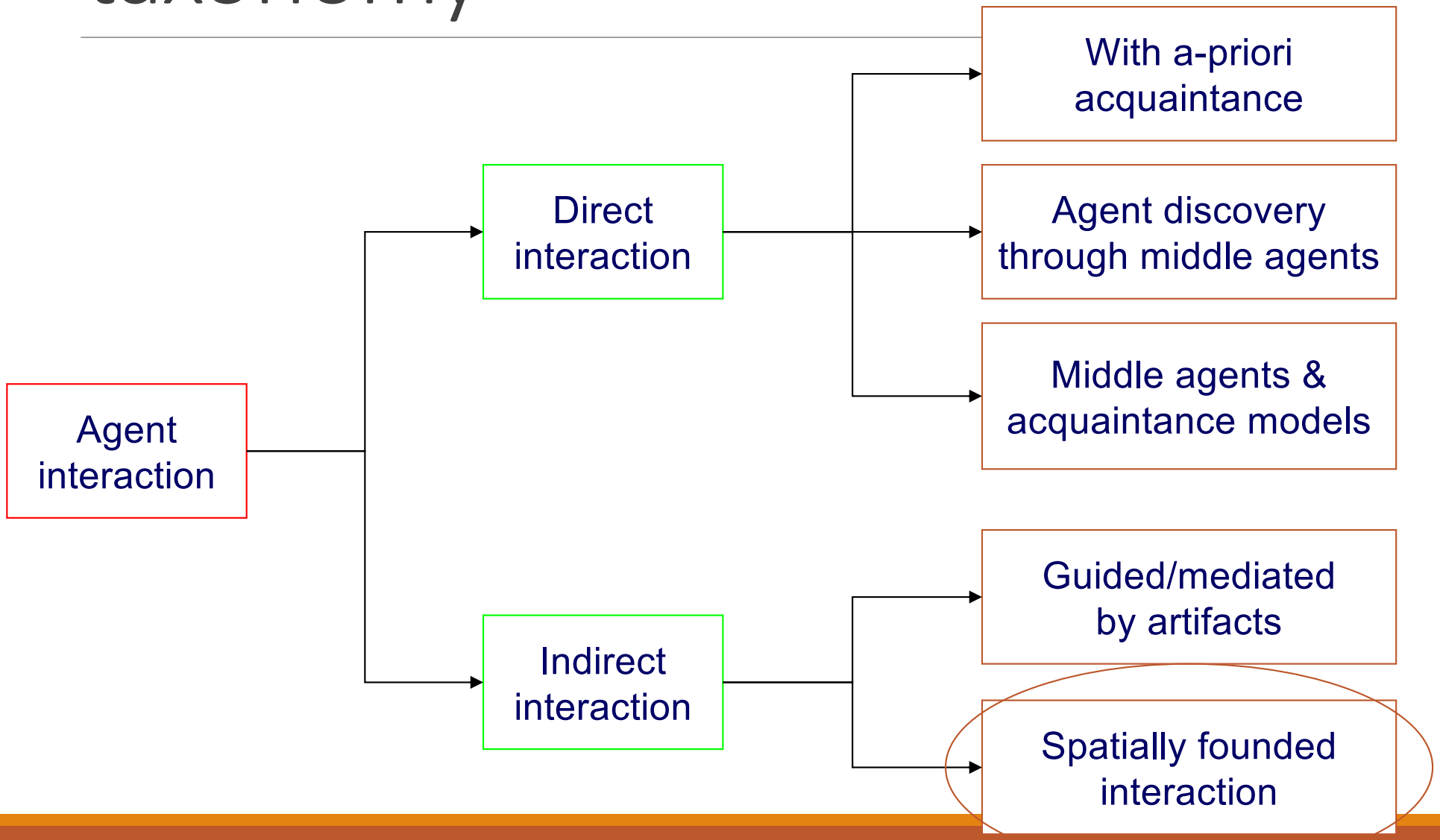
Advantages

- The artifact represents an *abstraction of agents' environment*, and the burden of interaction is moved from the agents to their environment
- Interaction is *mediated*, and can thus be *controlled* (enforcement/enactment of organizational rules)

Issues

- *Complex implementation* (in distributed environments)
- How to *integrate* different *artifacts and contexts*?

Interaction models: a taxonomy



Spatially founded interaction

Artifact mediated interaction are a first step in agents' *environment modelling*

Such *artifacts represent very focused parts of the environment*, and cannot consider the parts of agents' context that does not pertain the specific artifact

- They represent a single specific context of interaction

Other approaches bring the environment metaphor to a deeper level, providing *spatially founded interaction* mechanisms

Spatial features of the environment *are explicitily considered* by interaction mechanisms

Ancestors of *Spatial Interaction*: CAs

A *Cellular Automata* (CA) is a set of homogeneous cells, evolving in discrete time steps

Cells form a regular n-dimensional lattice

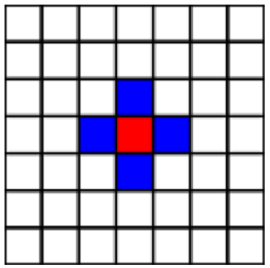
- Homogeneous neighborhood (e.g. Von Neumann, Moore)

Cells characterized by

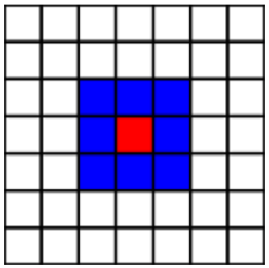
- A state, belonging to a finite set representing possible cell states
- A transition rule, describing cell state dynamics

Cell → sort of reactive agent

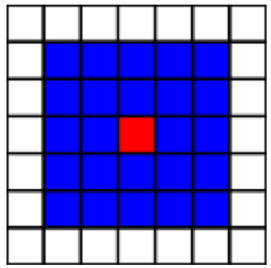
- That cannot move in the environment
- Can only interact with neighbouring cells according to precisely defined rules



von Neumann Neighbourhood



Moore Neighbourhood



Extended Moore Neighbourhood

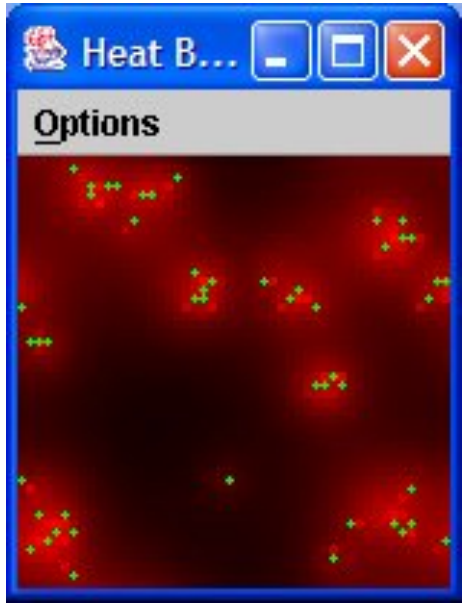
Swarm (and the likes) agent environment

Swarm and many derived projects provide specific *environments* in which agents may be placed and interact

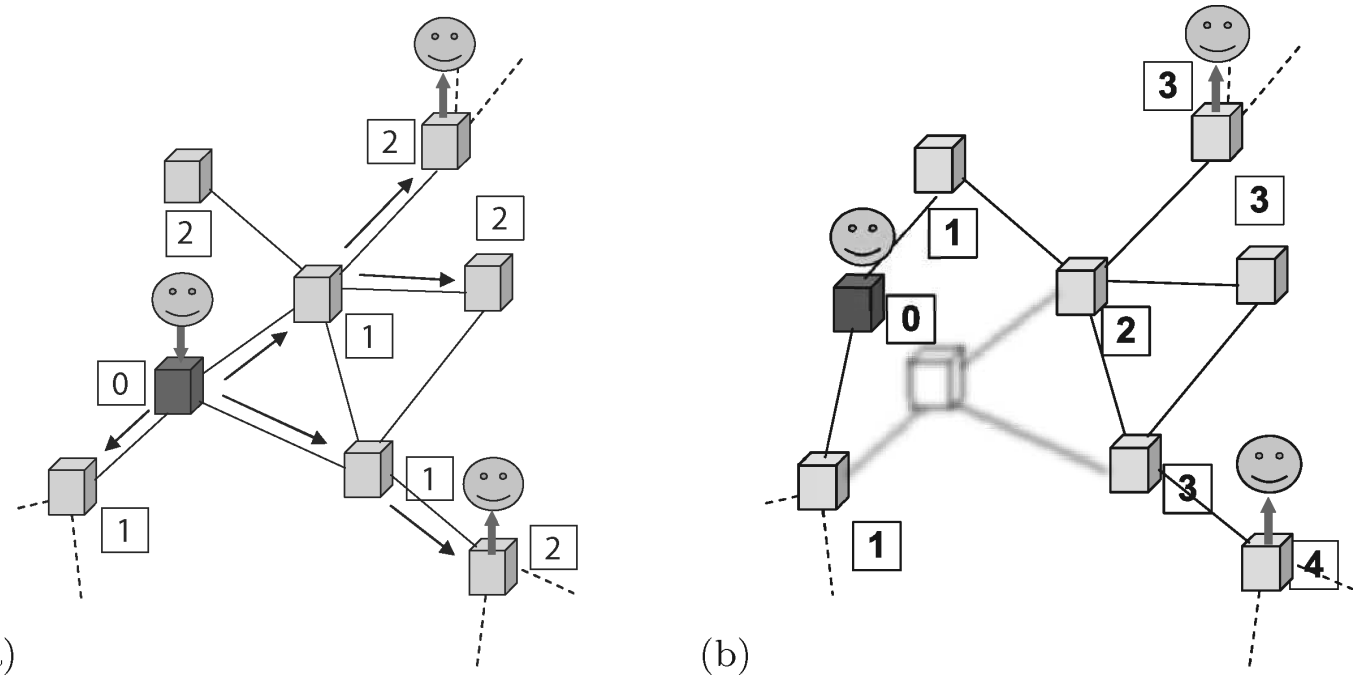
Regular lattices supporting *diffusion of signals* that are

- Emitted by entities
- Spread in the spatial structure
- Affecting other entities
- Evaporating over time

Diffusion strictly related to specific environmental structures

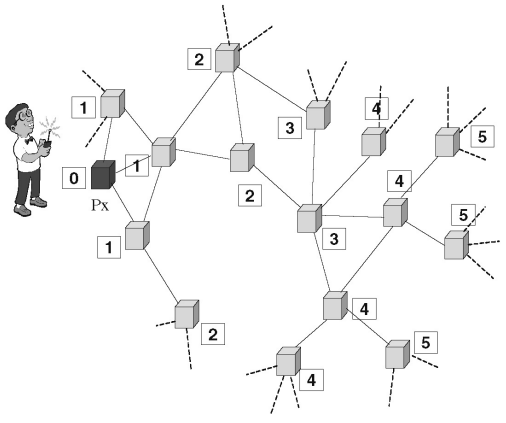


CoFields

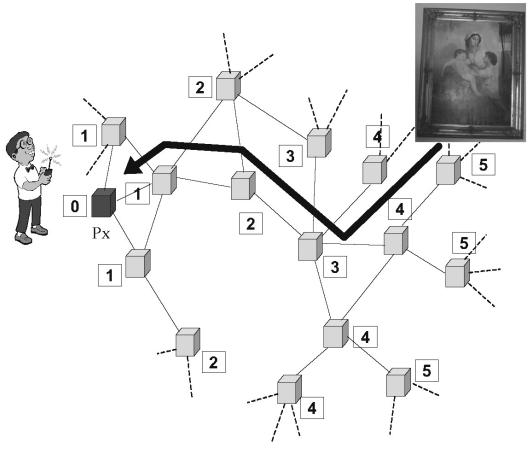


- Each node is a tuple space
- Nodes can be connected according to some networked topology, for instance in a manet scheme (what's a manet?)
- Similar considerations can be done for the organization of peer-to-peer network management systems

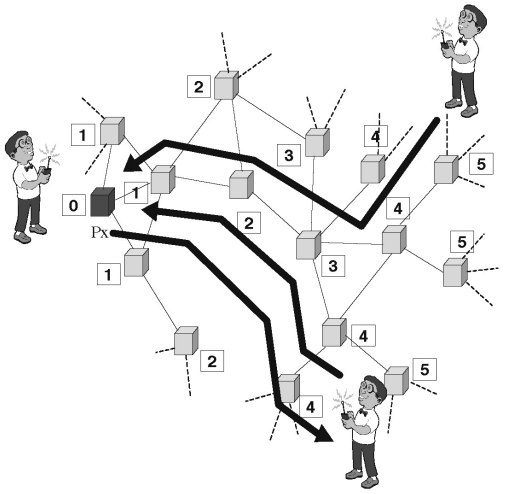
CoFields



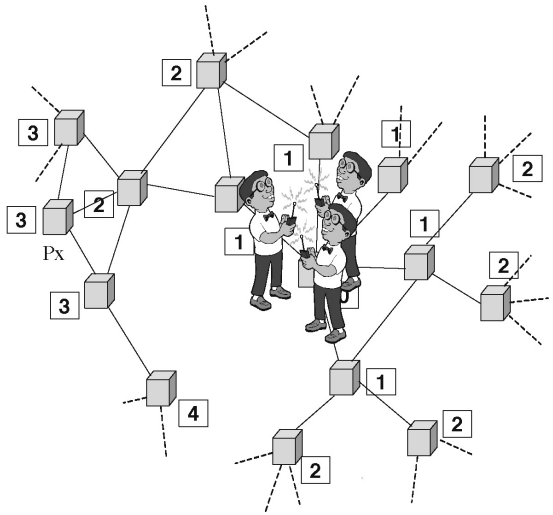
(a)



(b)



(c)



(d)

Concluding remarks

- The new scenario allows/requires the design and development of *context-aware* systems
- *Agent interaction models* can be exploited to develop such systems
- Not all these models offer abstractions for the modelling of *contextual elements*
- Spatially founded interaction models can suitably represent and exploit contextual information