Norms for Typing MAS Multi-Agent Systems

Normative Multi-Agent Systems: NorMAS 2013

Norms for Typing MAS

M. Baldoni[1] C. Baroglio[1] F. Capuzzimati[1]

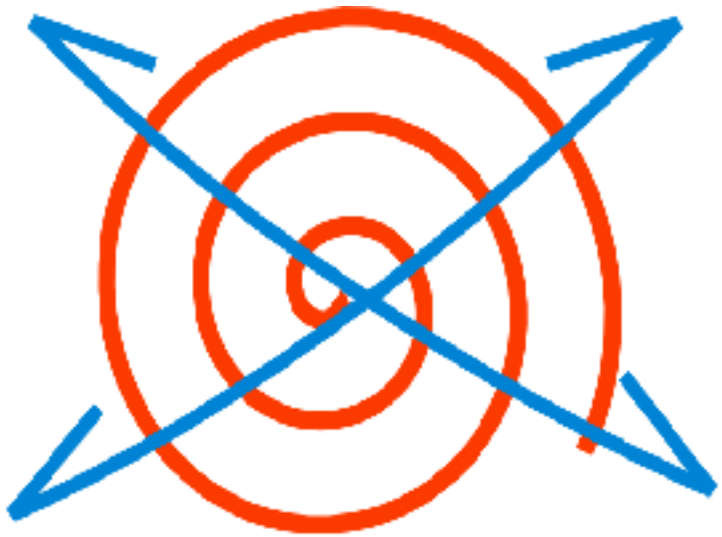
[1]Dipartimento di Informatica, Università degli Studi di Torino

Leiden, August 19-23, 2013

Overview

B2B, cross-business, open environment systems

- Software infrastructures: more and more global, pervasive and autonomic
- Computing is becoming ubiquitous, with embedded and distributed devices interacting with each other
- MAS have been recognized to be a promising paradigm for this kind of scenarios



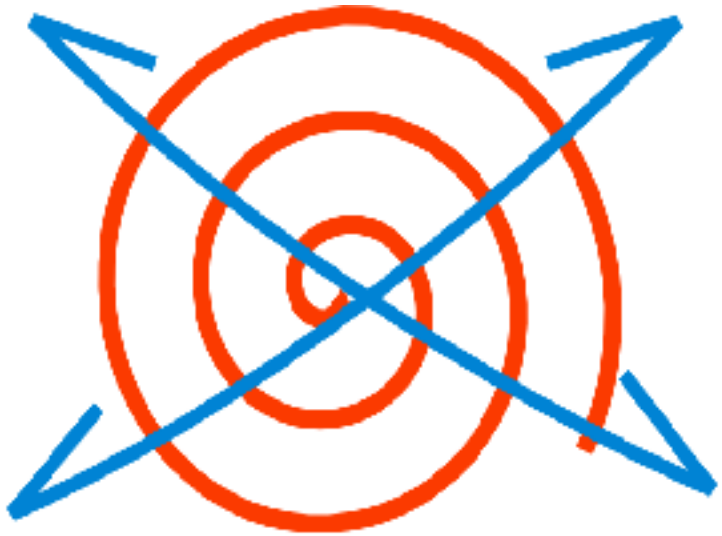
Typing Systems

HOWEVER

The more the complexity of programming these systems will increase, the more the need for effective tools for reasoning on properties of programs is noticed

Types

provide abstractions to perform sophisticated forms of program analysis and verifications that help programmers to face the complexity of their job



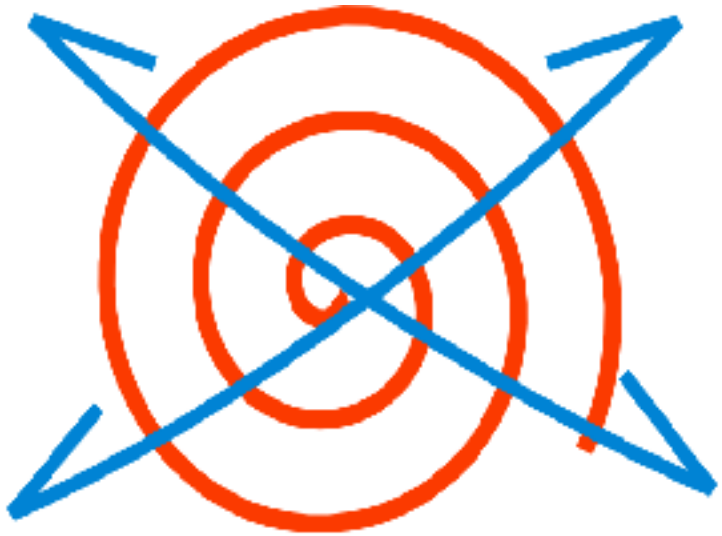
Typing Systems

HOWEVER

The more the complexity of programming these systems will increase, the more the need for effective tools for reasoning on properties of programs is noticed

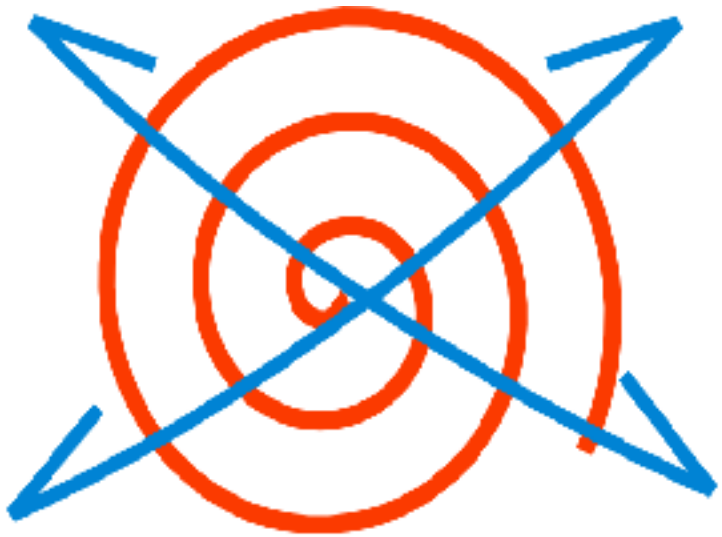
Types

provide abstractions to perform sophisticated forms of program analysis and verifications that help programmers to face the complexity of their job



Typing Systems

- Enable compile time/runtime error checking
- Conceptual and abstraction tools for modeling
- Documentation
- Conformance and compliance
- Reasoning about programs and components
- Type checking as a simple form of (a priori/runtime) verification



Typing Systems for MAS

- Here we focus on two more recent proposals:

Global Session Types in Jason

- By D. Ancona, S. Drossopoulou, and V. Mascardi [Ancona et al., 2012, Ancona et al., 2013]
- Behavioral types for multiparty interactions
- Monitoring agent that verifies (dinamically) the conformance of interacting agents w.r.t. a global session type

simpAL

- By A. Ricci and A. Santi [Ricci and Santi, 2012a, Ricci and Santi, 2012b]
- An agent-oriented programming language with types checking inspired by main stream OO languages
- Static type checking for error detection

Typing Systems for MAS

- Here we focus on two more recent proposals:

Global Session Types in Jason

- By D. Ancona, S. Drossopoulou, and V. Mascardi [Ancona et al., 2012, Ancona et al., 2013]
- Behavioral types for multiparty interactions
- Monitoring agent that verifies (dinamically) the conformance of interacting agents w.r.t. a global session type

simpAL

- By A. Ricci and A. Santi [Ricci and Santi, 2012a, Ricci and Santi, 2012b]
- An agent-oriented programming language with types checking inspired by main stream OO languages
- Static type checking for error detection

Typing Systems for MAS

- Here we focus on two more recent proposals:

Global Session Types in Jason

- By D. Ancona, S. Drossopoulou, and V. Mascardi [Ancona et al., 2012, Ancona et al., 2013]
- Behavioral types for multiparty interactions
- Monitoring agent that verifies (dinamically) the conformance of interacting agents w.r.t. a global session type

simpAL

- By A. Ricci and A. Santi [Ricci and Santi, 2012a, Ricci and Santi, 2012b]
- An agent-oriented programming language with types checking inspired by main stream OO languages
- Static type checking for error detection

Global Session Types in Jason

- Inspired by Scribble, in [Ancona et al., 2012, Ancona et al., 2013] protocols are the key aspect
- Protocols are expressed by means of global session types
- Jason is extended in order to automatically generate an agent monitor for dynamic conformance (compliance) verification
- Cyclic Prolog terms

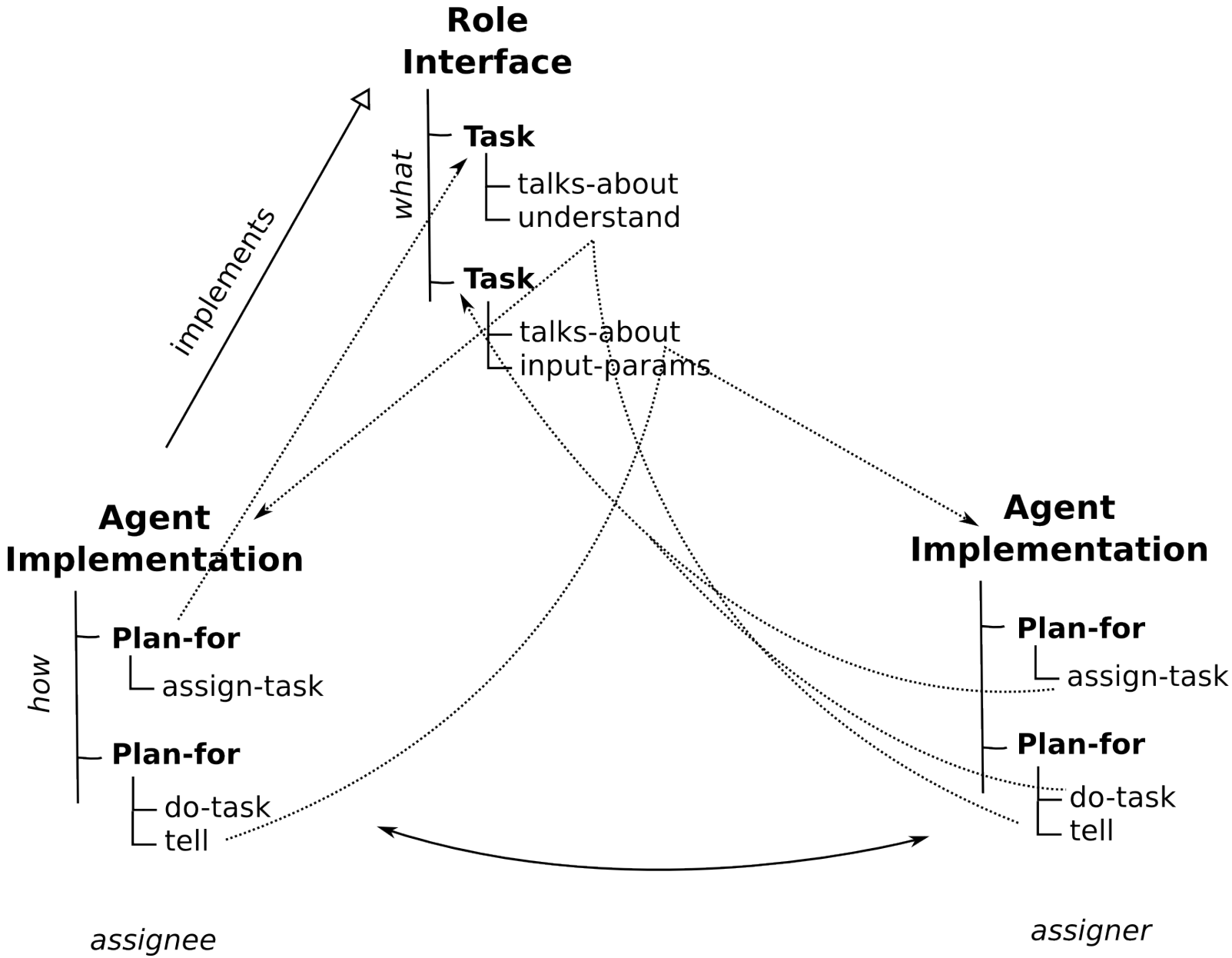
Global Session Types in Jason

- Global session types are "*procedural*" types (process abstraction), so they do not:
  - respect autonomy of agents
  - clearly espress what is expected from a role and what is possible for a role
- Who is the agent monitor? Who trusts it? Should all messages/actions be notified to it? How to guarantee this fact?
- Lack of a *normative* characterization of coordination [Castelfranchi, 1997, Singh, 1999], so that the publicly acceptance of the regulation allows reasoning about agents' behavior [Conte et al., 1998]

simpAL

- Inspired by main stream OO languages, in [Ricci and Santi, 2012a, Ricci and Santi, 2012b] static type checking for error detection is the key aspect
- Builds on the experience of JaCaMo
- role, usage-interface, org-model: interfaces
- agent-script, artifact, org: implementations

simpAL


Role
Interface
what
Task
talks-about
understand
Task
talks-about
input-params
implements
Agent
Implementation
how
Plan-for
assign-task
Plan-for
do-task
tell
Agent
Implementation
Plan-for
assign-task
Plan-for
do-task
tell
assignee
assigner

simpAL

- Static vs dynamic type systems: is compile time checking the key point? Sometimes this is not good also for OO languages (eg. downcasting)
- Types or ontological reasoning?
- Are roles mere agent interfaces?
- What is the semantics of types? Is type checking only a syntactic matching?

Commitment

Commitment

$$C(x, y, r, p)$$

represents the engagement from x to y, to bring about the consequent condition p when the antecedent condition r holds.

- Commitments have a normative nature: agents are liable for the violation of the commitments they have taken
- Commitment protocols allow for flexible behaviours: x is free to choose its actions
- The agent's compliance can be verified by *observing* the interaction

Commitment-based protocols

A commitment-based protocol is a **set of actions** whose meaning in terms of effects on the social state is agreed upon by all the interacting agents.

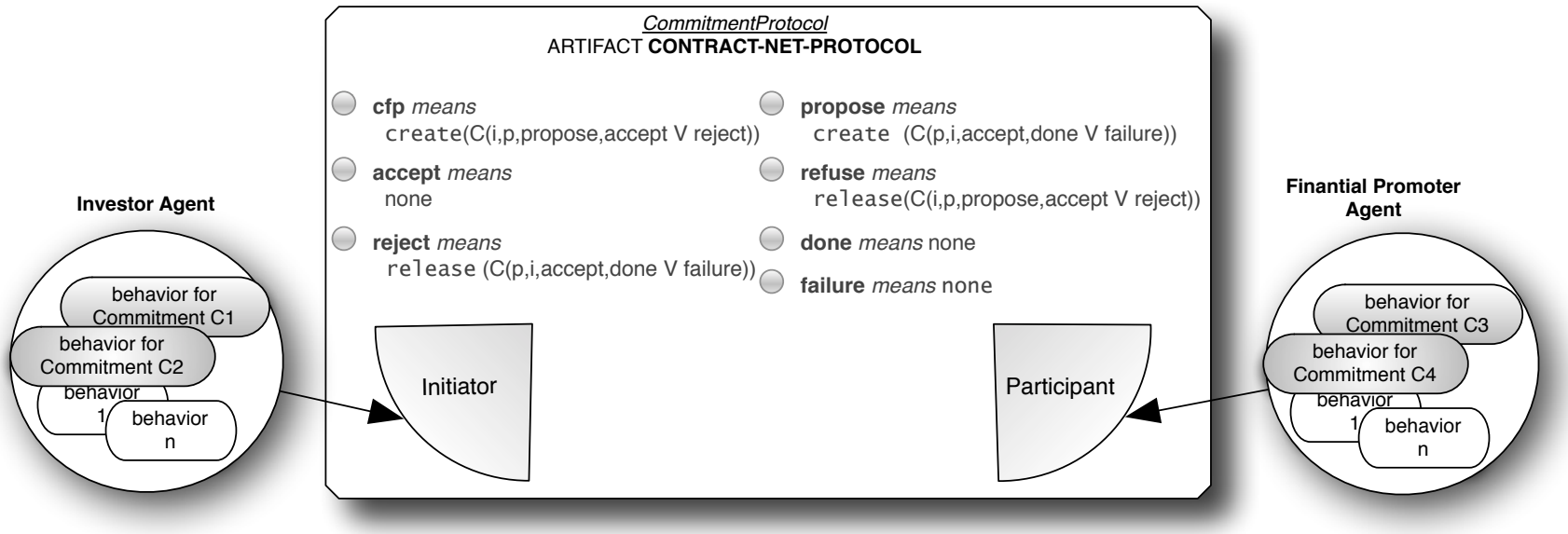
Actions definition

action **means** effects **if** *condition*

The *means* construct captures which *physical* events count as which *social* events

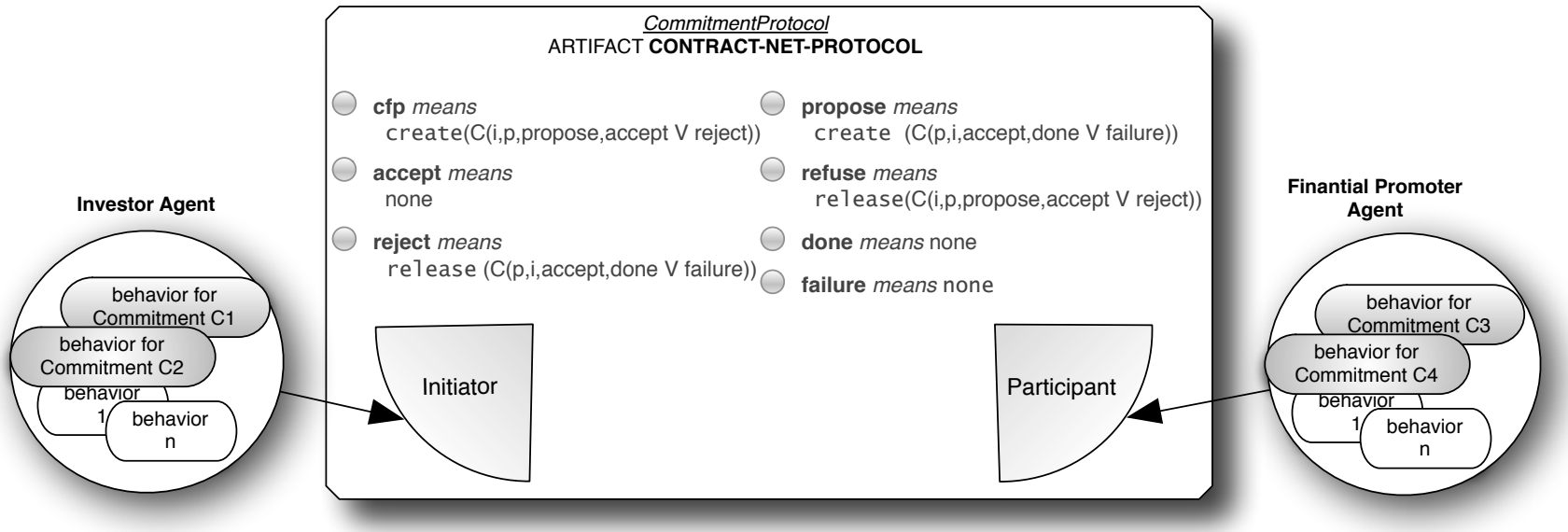
- **means:** introduces the social effects
- **if:** condition for the action to have the intended meaning

Commitment-based protocols



- An agent/initiator should be able to *accept* or *refuse* a proposal
- An agent/participant should be able to complete the assigned task (*done*) or to communicate its *failure*

Commitment-based protocols

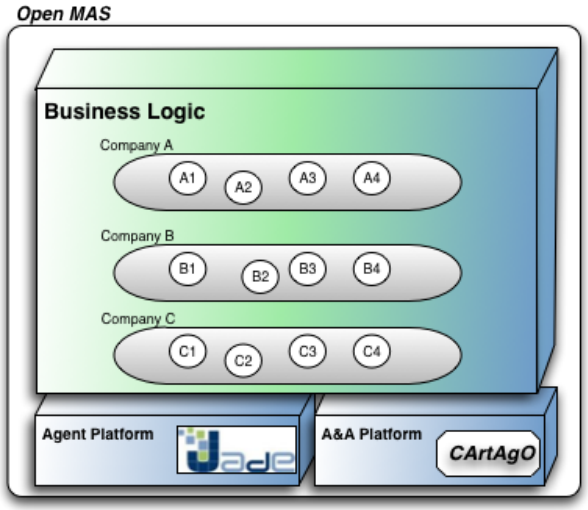


Organizational Roles [Baldoni et al., 2007, Boella and van der Torre, 2007]

- *Foundation*, *definitional dependence*, and *institutional empowerment*
- *Requirements* (ability to satisfy own commitments) and *powers* (action with a institutional meaning)

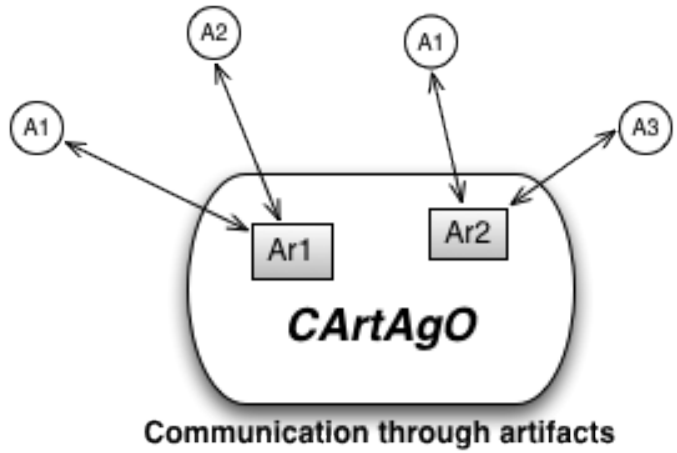
Mercurio [Baldoni et al., 2011]

- Integrating JADE with support for indirect communication
- According to Keil and Goldin, indirect communication fosters the collaboration and the coordination inside open systems
- PrThe adoption of programmable communication channels allows the specification of a normative facet that applies to the agents that are involved in the interaction



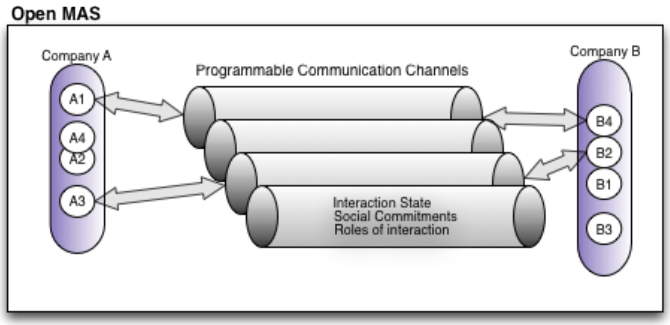
Mercurio [Baldoni et al., 2011]

- Artifact abstraction [Weyns et al., 2007, Omicini et al., 2008]: first-class entity, i.e. non-agent dynamic and programmable resource, that an agent can use, perceive, observe
- Artifacts can provide mediated, indirect communication to agents
- Designers can leverage artifacts to explicitly model interaction protocols, defining a social agreement accepted by agents using them



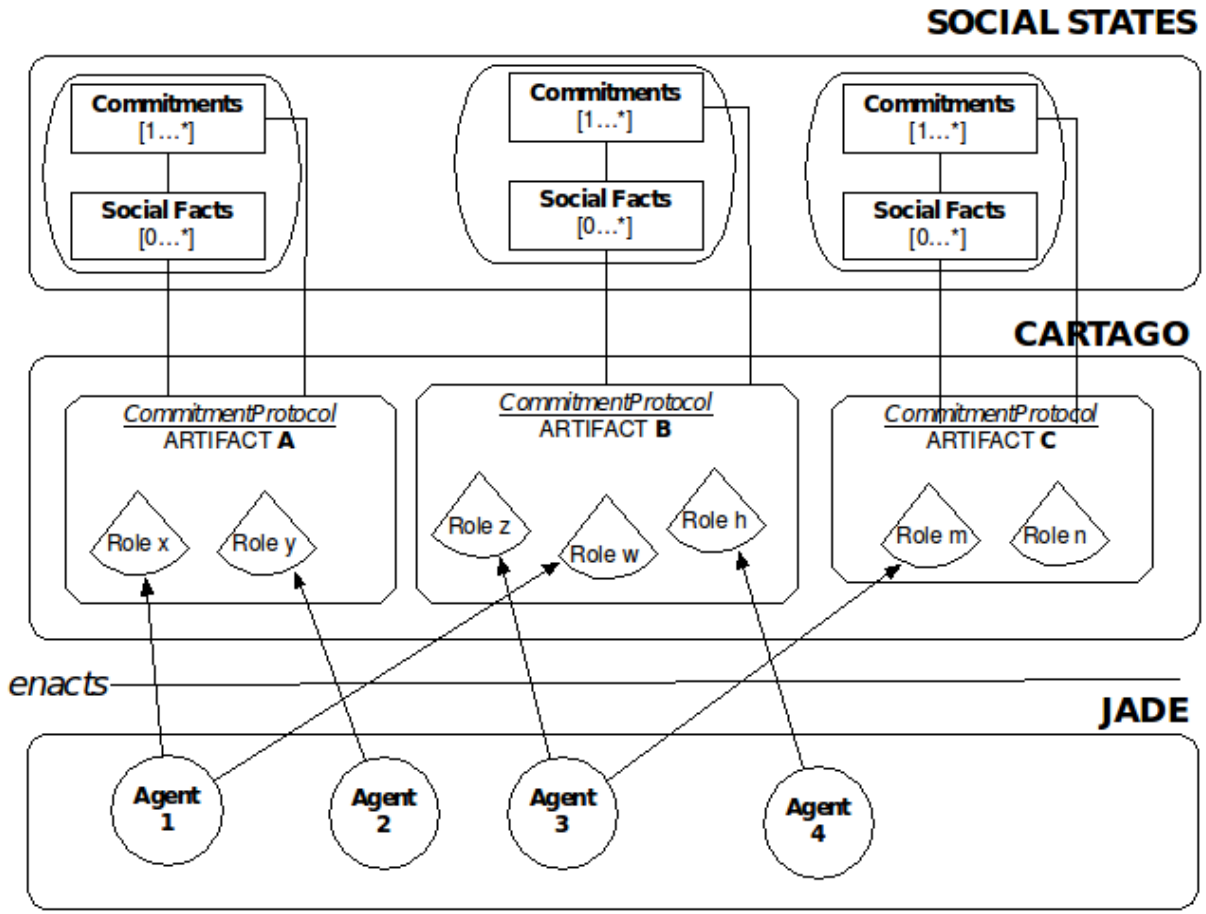
Mercurio [Baldoni et al., 2011]

- Artifacts, as programmable indirect communication channels, can reify and implement normative characterization and social expectation
- Such an artifact entails mutual, social dependencies between agents using it
- We model social dependencies as commitments



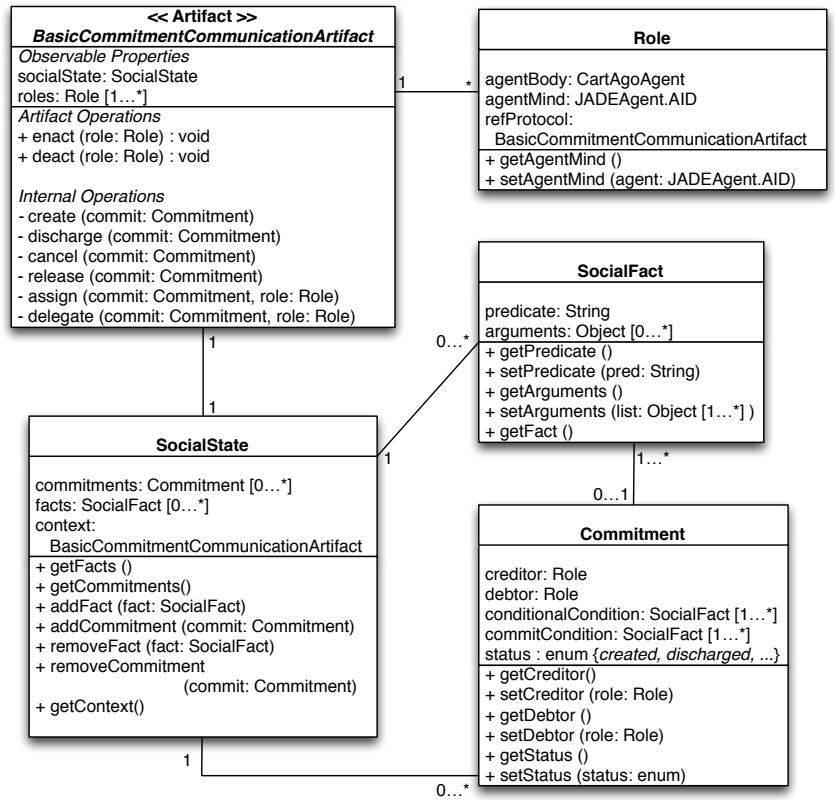
2COMM [Baldoni et al., 2013]

- 2COMM: reifying commitment protocols in JADE by means of CArtAgO framework [Ricci et al., 2009]

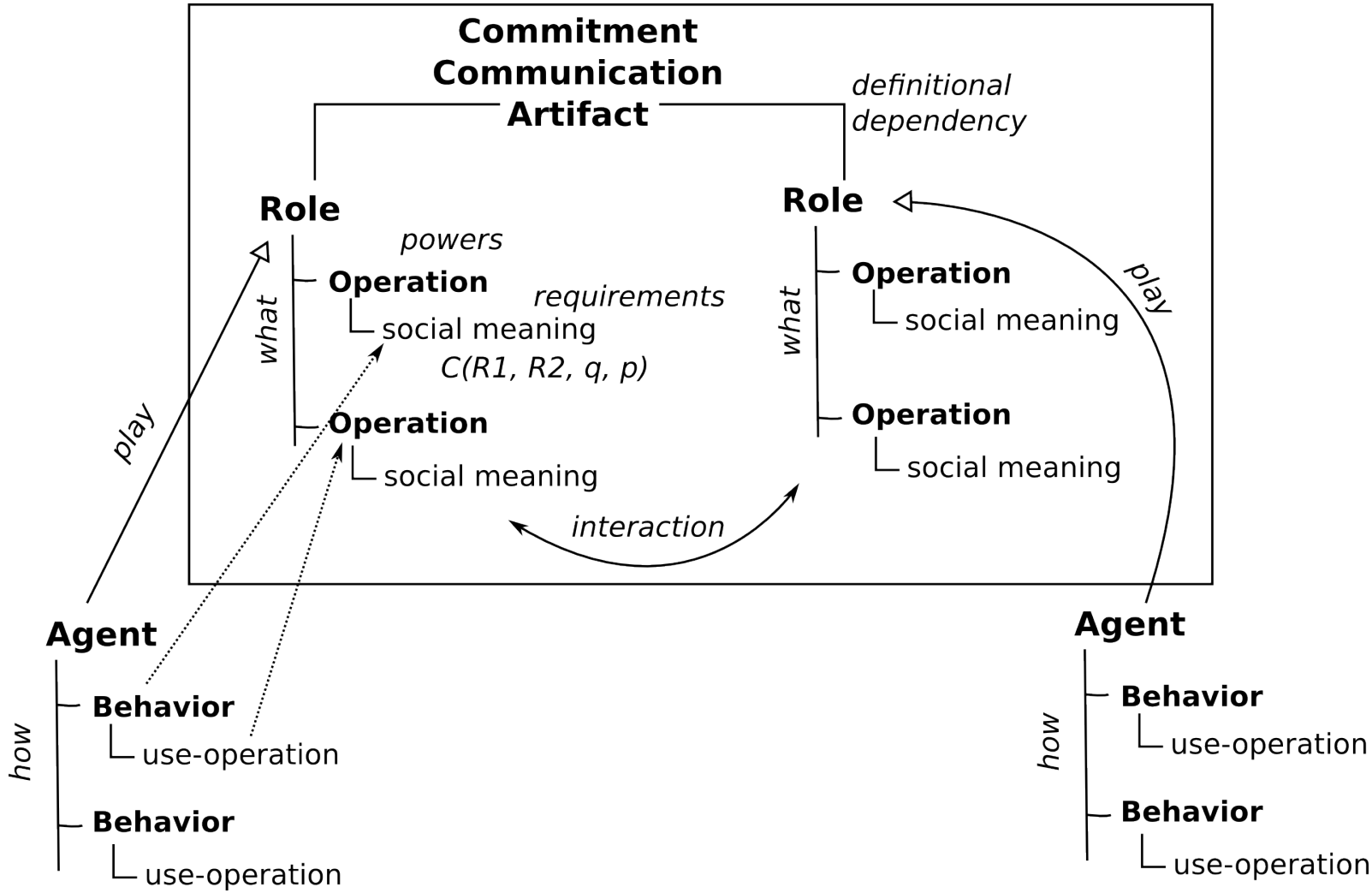


2COMM [Baldoni et al., 2013]

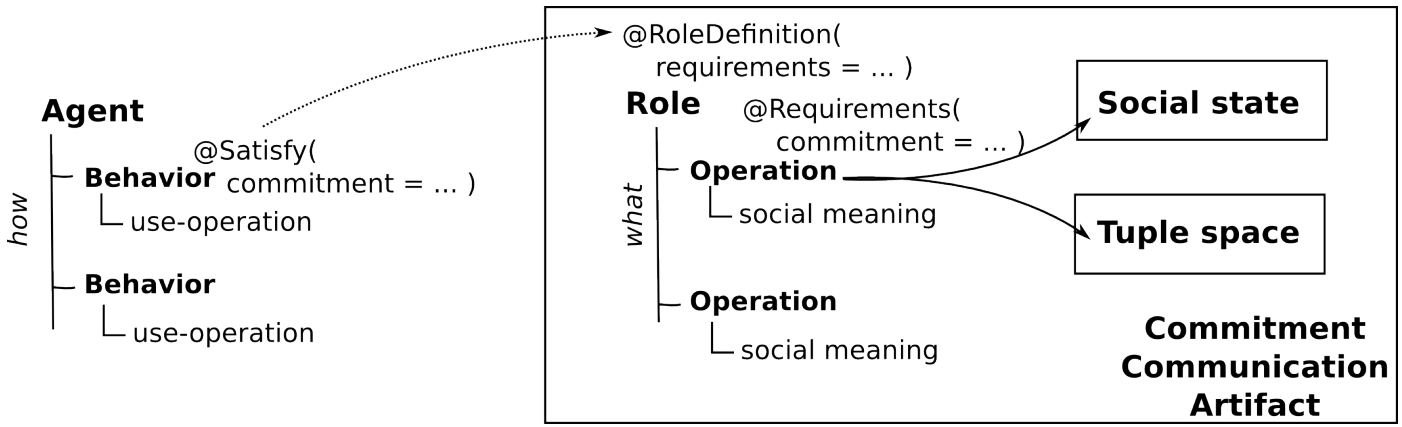
- An agent can use a communication artifact enacting a role
- A role represents the interface between the artifact and the agent using it
- When acting as a certain role, an agent's actions impact on the interaction state



Type checking via commitments



Type checking with commitments



- Type checking by means of Java Annotations
- Dynamic type checking: behaviors must comply to requirements (java reflection is used)
- Type system as a logic "theory" of commitments
- Commitment to regulation [Marengo et al., 2011]: regulating how

Conclusion

- Dynamic vs static type checking
- Conformance as logic entailment
- Compliance: programmable communication channel with monitoring functionalities
- A normative value thanks to commitment-based approach
- Flexibility and openess typical of MAS
- Modularity and compositionality typical of design and development methodologies
- Enable a business level architecture as fostered in [Chopra and Singh, 2009]


Ancona, D., Barbieri, M., and Mascardi, V. (2013).
Constrained global types for dynamic checking of protocol conformance in multi-agent systems.
In Shin, S. Y. and Maldonado, J. C., editors, *SAC*, pages 1377–1379. ACM.


Ancona, D., Drossopoulou, S., and Mascardi, V. (2012).
Automatic generation of self-monitoring mass from multiparty global session types in jason.
In Baldoni, M., Dennis, L. A., Mascardi, V., and Vasconcelos, W., editors, *DALT*, volume 7784 of *Lecture Notes in Computer Science*, pages 76–95. Springer.


Baldoni, M., Baroglio, C., and Capuzzimati, F. (2013).
2COMM: a commitment-based MAS architecture.
In Cossentino, M., El Fallah Seghrouchni, A., and Winikoff, M., editors, *Proc. of the 1st International Workshop on Engineering Multi-Agent Systems, EMAS 2013, held in conjuction with AAMAS 2013*, pages 17–32, St. Paul, Minnesota, USA.


Baldoni, M., Baroglio, C., Marengo, E., Patti, V., and Ricci, A. (2011).
Back to the future: An interaction-oriented framework for social computing.
In *RESC*, pages 2–5. IEEE.


Baldoni, M., Boella, G., and van der Torre, L. W. N. (2007).
Interaction between objects in powerjava.
Journal of Object Technology, 6(2):5–30.


Boella, G. and van der Torre, L. W. N. (2007).
The ontological properties of social roles in multi-agent systems: definitional dependence, powers and roles playing roles.
Artif. Intell. Law, 15(3):201–221.


Castelfranchi, C. (1997).
Principles of Individual Social Action.
In Holmstrom-Hintikka, G. and Tuomela, R., editors, *Contemporary action theory: Social action*, volume 2, pages 163–192, Dordrecht. Kluwer.

Chopra, A. K. and Singh, M. P. (2009).
Elements of a business-level architecture for multiagent systems.
In Braubach, L., Briot, J.-P., and Thangarajah, J., editors, *PROMAS*, volume 5919 of *Lecture Notes in Computer Science*, pages 15–30. Springer.

Conte, R., Castelfranchi, C., and Dignum, F. (1998).
Autonomous norm acceptance.
In Müller, J. P., Singh, M. P., and Rao, A. S., editors, *ATAL*, volume 1555 of *Lecture Notes in Computer Science*, pages 99–112. Springer.

Marengo, E., Baldoni, M., Baroglio, C., Chopra, A. K., Patti, V., and Singh, M. P. (2011).
Commitments with regulations: reasoning about safety and control in regula.
In Sonenberg, L., Stone, P., Tumer, K., and Yolum, P., editors, *AAMAS*, pages 467–474. IFAAMAS.

Omicini, A., Ricci, A., and Viroli, M. (2008).
Artifacts in the a&a meta-model for multi-agent systems.

Autonomous Agents and Multi-Agent Systems, 17(3):432–456.


Ricci, A., Piunti, M., Viroli, M., and Omicini, A. (2009).
Environment Programming in CArtAgO.
In *Multi-Agent Programming II: Languages, Platforms and Applications, Multiagent Systems, Artificial Societies, and Simulated Organizations*.


Ricci, A. and Santi, A. (2012a).
From actors to agent-oriented programming abstractions in simpal.
In Leavens, G. T., editor, *SPLASH*, pages 73–74. ACM.


Ricci, A. and Santi, A. (2012b).
Typing multi-agent programs in simpal.
In Dastani, M., Hübner, J. F., and Logan, B., editors, *ProMAS*, volume 7837 of *Lecture Notes in Computer Science*, pages 138–157. Springer.


Singh, M. P. (1999).
An ontology for commitments in multiagent systems.
Artif. Intell. Law, 7(1):97–113.


Weyns, D., Omicini, A., and Odell, J. (2007).
Environment as a first class abstraction in multiagent systems.
Autonomous Agents and Multi-Agent Systems, 14(1):5–30.