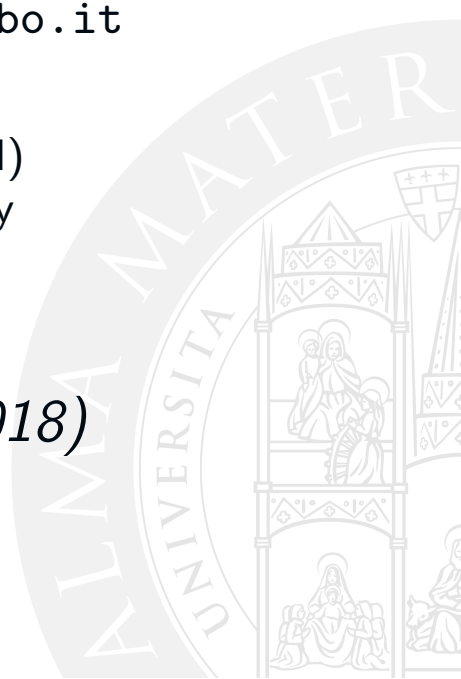
Injecting (Micro)Intelligence in the IoT: Logic-based Approaches for (M)MAS

Andrea Omicini
andrea.omicini@unibo.it

Roberta Calegari
roberta.calegari@unibo.it

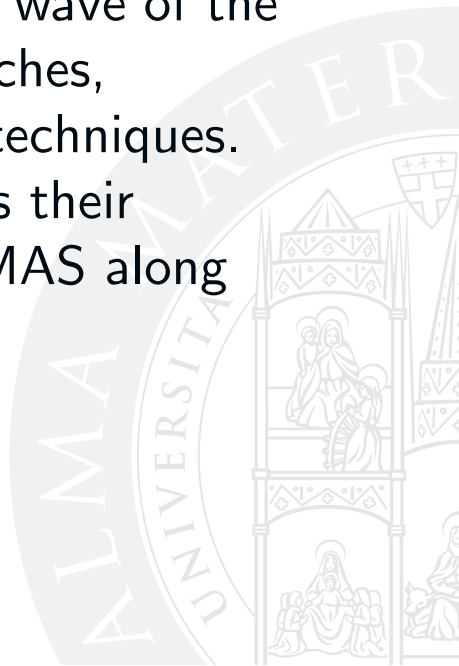
Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum—Università di Bologna, Italy

Invited Talk @ *International Workshop on Massively Multi-Agent Systems (MMAS 2018)*
Stockholm, Sweden, 14 July 2018



Abstract

Pervasiveness of ICT resources along with the promise of ubiquitous intelligence is pushing hard both our *demand* and our *fears* of AI: demand mandates for the ability to inject (*micro*)*intelligence* ubiquitously, fears compel the behaviour of intelligent systems to be *observable*, *explainable*, and *accountable*. Whereas the first wave of the new "AI Era" was mostly heralded by non-symbolic approaches, features like explainability are better provided by symbolic techniques. In this talk we focus on logic-based approaches, and discuss their potential in pervasive scenarios like the IoT and open (M)MAS along with our latest results in the field.



Or, What are We Going to Talk About?

Some basic questions

- how are we going to shape the world around our multitude of agents?
- how are we going to spread knowledge and intelligence for agents (and humans) to make the best use of it?

Some less basic questions

- nowadays, the AI pendulum is oscillating towards *non-symbolic* approaches: do we really think it is going to *stop* there?
- what have *symbolic* techniques to say before we hit the next AI wall?

Disclaimer

These slides are mostly for after-talk readers

- not really made for attendants following the presentation
  - they are *waaaay* too many for the presentation
- but, they typically work quite well *after* the presentation
  - since basically no one takes notes during the presentation, nowadays
- you can find them
  - on SlideShare
  - on our APICe server
  - or, just ask us
- a *pre-emptive sorry* to those who hate that sort of presentation!

1. Scenarios
   - IoT & (M)MAS
   - Distributed Knowledge & Intelligence
2. Machine Intelligence
   - Micro-intelligence
   - (Socio-)Technical Requirements
   - Symbolic vs. Non-symbolic
3. The LP Road to Micro-intelligence
   - LP for Intelligent Systems
   - LP for the IoIT

Next in Line...

Focus on...

ICT & Human Environment

Changing human environment

- human environment is more and more affected and even shaped by the increasing availability of ICT resources
- in particular within the constantly-growing urban areas all over the world
- the ubiquitous availability of personal devices, sensors networks, actuators, and computational resources in general, is affecting human environment
- in particular, changing urban environments into wannabe-*smart environments* on a massively-large scale

Internet of Things (IoT)

Computing & connecting myriad of devices

- the Internet of Things (IoT) [Atzori et al., 2010] is designed to connect millions of (diverse) things (objects, devices) altogether
- things that *compute* and *connect* with each other
- in a smart way [Xiang and Li, 2012]

IoT & (M)MAS

IoT as a (M)MAS scenario

- (massive) multi-agent systems ((M)MAS) have been already recognised as the most promising way for developing applications for the IoT and cyber-physical systems (CPS)
- since agents support decentralised, loosely-coupled and highly dynamic, heterogeneous and open systems, where components cooperate opportunistically [Savaglio et al., 2018]

→ IoT and pervasive systems can be seen as (M)MAS

- monitoring and controlling our environments
- where MAS abstractions, techniques, and methods cope with the complexity of the application scenarios

Intelligence & IoT

Agents & intelligence for IoT

- devices in IoT systems have to be massively networked and provided with (different degrees of) intelligence, in order to interoperate and cooperate in a smart way
- agents are the most natural way of approaching IoT systems featuring complexity, dynamism, situatedness, and autonomy

  [Kato et al., 2015, Manzalini and Zambonelli, 2006, Spanoudakis and Moraitis, 2015]
- agents are the most viable abstraction to encapsulate fundamental features such as *control*, *goals*, *mobility*, and intelligence, in the development of proactive, cooperating, and context-aware smart objects [Fortino et al., 2012]
- agent-oriented models and technologies are gaining momentum for embedding decentralised intelligence [Jamont and Occello, 2016]

Focus on...

Knowledge-Intensive Environments (KIE)

Information galore

- ubiquitous *knowledge sources* growing in number and size
- *knowledge prosumers* in socio-technical systems (STS) [Whitworth, 2006]
- application *goals* more and more *depending on information* available in the environment [Mariani and Omicini, 2013]

Pervasive Intelligence

Intelligence everywhere

- computation is everywhere around us
- software components are required to behave intelligently, understanding their own goals and the context where they have to work
- integration of software components is supposed to add social intelligence, possibly through coordination [Castelfranchi, 1998]
- pervasive computing nowadays calls for *ubiquitous intelligence*

Internet of Intelligent Things (IoIT)

Intelligent objects in the IoT

- our everyday physical objects should be able to network in the Internet of Things (IoT) [Gubbi et al., 2013, Atzori et al., 2010, Fortino et al., 2014]
- they are required to understand each other, to learn, to understand situations, to understand us [Lippi et al., 2017]
- our *everyday object* should be(come) intelligent in the *Internet of Intelligent Things* (IoIT) [Arsénio et al., 2014]

Next in Line...

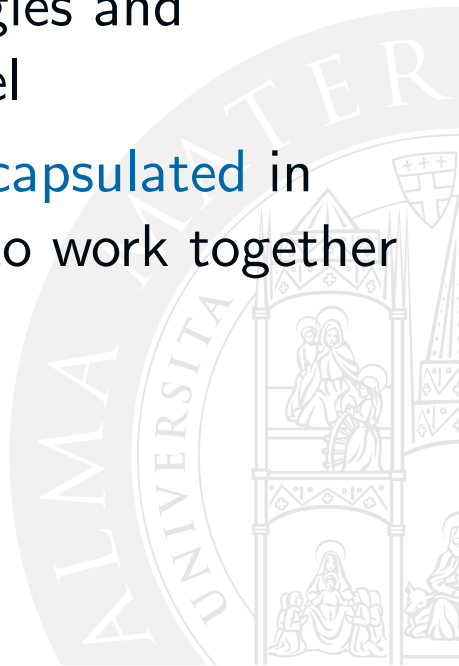
Focus on...

Which Intelligence for the IoT?

- we need *small chunks of machine intelligence*
- spread all over the system
- capable to enable individual intelligence of any sort of device
- promoting *smart* interoperation, collaboration, and coordination among different entities

Vision

- the micro-intelligence vision promote ubiquitous distribution of intelligence in large pervasive systems [Calegari, 2018]—such as IoT ones [Calegari et al., 2018a]
- in particular when coupled with agent-based technologies and methods, at both the individual and the collective level
- the idea behind micro-intelligence is that it can be encapsulated in devices of any sort, making them smart, and capable to work together in groups, aggregates, societies



Focus on...

Socio-technical Systems & Requirements I

Socio-technical systems (STS) [Whitworth, 2006]

- artificial systems where both *humans* and *artificial* components play the role of system components
  - from online reservation systems to social networks
- most of nowadays systems are just STS
  - or, at least, cannot be engineering and successfully put to work without a proper socio-technical perspective in the engineering stage

Socio-technical Systems & Requirements II

Socio-technical requirements

- some system requirements are not just merely *technical*
- they concern the way in which humans *perceive* them, *participate* them, *understand* them, include and exploit them into they own processes, including cognitive ones—not just use them
- these are what we call here socio-technical requirements
- even though, of course, they *do* have (strong) technical counterparts

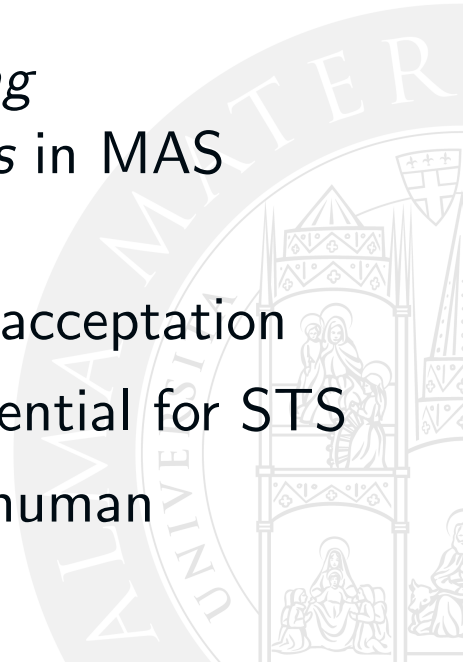
Observability

- *observability* is essential for understanding execution of distributed systems [Tanenbaum and van Steen, 2007]—as in the case of stateless communication in REST
- *mutual observation* is a pre-condition for many forms of cognitive agent coordination—e.g., behavioural implicit communication (BIC) [Castelfranchi et al., 2010]
- inspectability an essential feature for *online engineering* [Fredriksson and Gustavsson, 2004]—in particular for *artefacts* in MAS environment [Omicini et al., 2008]

! observability has a classical, straightforward technical acceptation

→ a more articulated socio-technical interpretation is essential for STS

e.g. making the portion of a system required for effective human interaction *readable* for humans at run time



Malleability

- the ability of changing / adapting at run time is a pre-condition for most forms of self-adaptive / self-organising systems [Omicini et al., 2008]

e.g. malleability of coordination artefacts allows for self-organisation patterns in MAS [Viroli et al., 2005]

e.g. malleability of a Prolog engine allow for adaptive intelligent middleware [Piancastelli et al., 2008]

! malleability may in principle allow both humans and intelligent agents to affect the system behaviour at run time [Omicini et al., 2006a]

Explainability

We need explanations

- it is not enough for STS to just work
- we need to understand them—otherwise, technology is not much different from magic

e.g. decision support systems need not just to propose solutions, but also to *motivate* them

e.g. the problem with deep learning till now

e.g. engineering systems with self-organisation techniques mandates some for of system *predictability*

! as a socio-technical requirement, explanation needs to be understandable by humans

Predictability

We need to be sure

- it is not enough for systems to work
- we need to know how they will behave—otherwise, it could be dangerous, or, unacceptable, or, ...

e.g. we like to add features of *self-organising systems* to our systems—but, we need to foresee their evolution over time, at least in general terms

e.g. complex systems are generally *unpredictable*, however we would need at least partial predictability of their behaviour when we engineer them

! as a socio-technical requirement, predictability mandates at least for a general expression of macro-level effects that could be appreciated by a human

Accountability

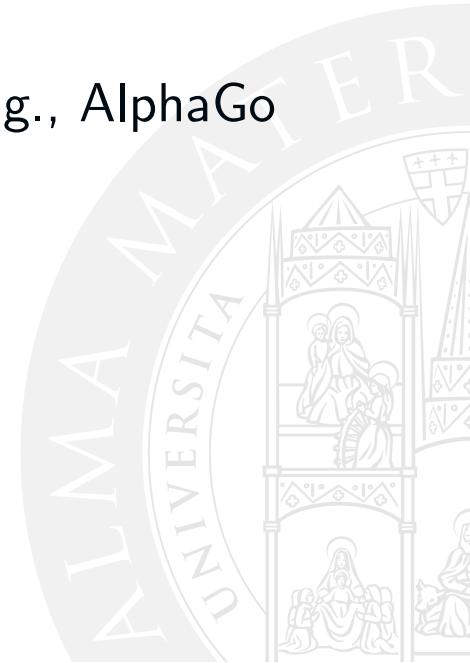
Systems & norms

- if computational systems are pervasive, and affect every aspects of our everyday life, they are subject of all the *norms* ruling human individual and social behaviour
- *norm compliance* should be enforced by computational systems, starting from the earliest phases of software engineering
- *normative reasoning* should become part of the intelligent behaviour of components and systems
- accountability, traceability, and responsibility should be part of each component and every system
- humans need someone to *blame* for anything

Focus on...

Symbolic vs. Non-symbolic Approaches to AI

- symbolic vs. non-symbolic is a *leitmotiv* in AI research since Brooks' robotics [Brooks, 1991b, Brooks, 1991a]
- many novel approaches to machine intelligence nowadays are increasingly focussing on *non-symbolic* approaches
  - such as deep learning with neural neural networks—e.g., AlphaGo [Silver et al., 2016]
- and how to make them work on the large scale



Back to Symbolic AI?

- non-symbolic approaches have the potential minimise the engineering efforts towards large-scale intelligence
- yet, we do also need that our large-scale intelligent systems exhibit *socio-technical features* such as *observability*, *explanability*, and *accountability*
- to make *ubiquitous intelligence* actually work in human organisations and societies

→ more classic AI approaches to intelligence can be of help

  - such as *agents* and *multi-agent systems* (MAS)
  - as well as *declarative* and *logic-based* approaches

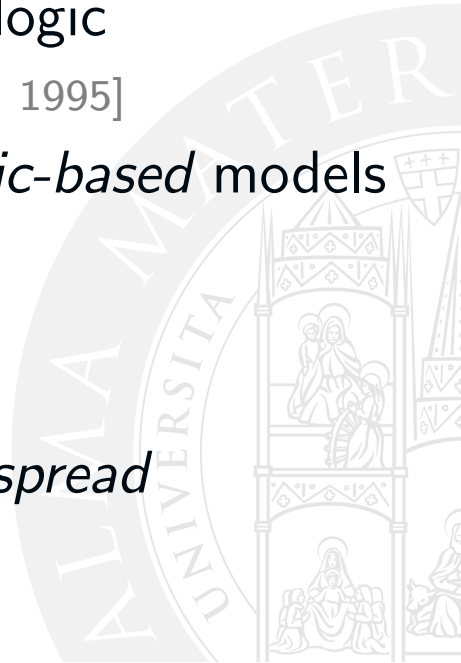
Agents

Agents & multi-agent systems (MAS)

- agents are the most viable abstraction to encapsulate fundamental features such as *control*, *goals*, *mobility*, *intelligence* [Zambonelli and Omicini, 2004]
- MAS abstractions such as *society* and *environment* [Omicini, 2001] are essential to cope with the complexity of nowadays application scenarios [Omicini and Mariani, 2013]
- agent-oriented models and technologies are gaining momentum for embedding decentralised intelligence [Singh and Chopra, 2017] and to *distribute* intelligence in complex systems of any sort [Fortino et al., 2012]

Logic-based Approaches

- declarative and logic-based technologies quite straightforwardly address issues such as *observability* and *explainability*
- in particular when exploiting their inferential capabilities—e.g., [Idelberger et al., 2016]
- logic-based approaches already have a well-understood role in building intelligent agents [Omicini and Zambonelli, 2004]—e.g., the logic [Rao and Georgeff, 1998] behind BDI agents [Rao and Georgeff, 1995]
- instead, in the following we focus on the role that *logic-based* models and technologies can play when used
  - to rule *agent societies*
  - to engineer *agent environment*
- so, again, to *shape the world* around the agents, and *spread intelligence* around them



Next in Line...

Focus on...

Why Logic Programming for Intelligent Systems? I

General features of Logic Programming (LP)

- computation as deduction
- declarative and procedural interpretation match
- reasoning about what is true rather than about what to do
- logic theories as programs and knowledge bases
- programming with relations and inferences
- non-typed tree structures for uniform data representation
- unification as a non-directional mechanism for message passing
- single-assignment variables, step-by-step refinement
- reasoning with incomplete information
- non determinism
- interactive computational model
- ...

Why Logic Programming for Intelligent Systems? II

Overall

LP languages and technologies

represent in principle the most natural candidates for

injecting intelligence

within computational systems

Why Logic Programming for Distributed Systems? I

Early days

- since the early days of logic programming, features like
  - high-level languages
  - non-determinism
  - referential transparency

  made the potential for exploitation of parallelism in the execution of logic programs clearly emerge [Gupta et al., 2001]
- at the same time, the articulation of logic computation, with the frequent occurrence of
  - heavy computational load
  - non-trivial computational structures

  made the computational techniques developed for logic languages relevant for the general scenario of concurrent / parallel computation

Why Logic Programming for Distributed Systems? II

Or- & and-parallelism

- logic languages like Prolog allow for parallel evaluation
  - the proof tree could be explored parallel / concurrently
  - even more, the potential computational complexity of the proof tree actually *encourages* parallel exploration
- basically, Prolog supports two sorts of parallelism
  - *and-parallelism*
  - *or-parallelism*

fitting distributed computing

Logic Programming: Issues I

Why did LP never *really* take off?

- computational costs too high, machinery too huge
- not really suited for programming in the large—intrinsic modularity (predicates) does not really scales up, modules are not enough
- a novel programming paradigm requires huge efforts by programmers
- "no types" makes it difficult (also) to deal with domain-specific applications
- how to deal with non determinism?
- distance from mainstream programming paradigms potentially harms conceptual integrity
- troublesome integration with mainstream technologies
- ...

Logic Programming: Issues II

LP for distributed systems?

- mostly, distributed computing for LP—to share the huge computational load
- distributed computing is *not* the same as distributed systems, at least according to their historical meanings—per se, LP does not work for distributed systems
- interactive computational model works locally—one user querying one engine
- the universal notion of consistency of logic theory does not cope well with the incompleteness and inconsistency intrinsically implied by the needs of distributed pervasive systems
- LP technology does not always integrate well with available tools and technologies for distributed pervasive systems

Ubiquitous Declarative Languages and Technologies

Declarative languages and technologies

- SQL and its derivatives are everywhere in databases and knowledge sources of any sort
- XML and its applications are the de facto standard for the Web as well as for most of the novel application scenarios at any level
- even the shift in object-oriented programming from classes to interfaces in some sense represents a shift towards declarativeness
- declarative models and technologies play an essential roles in agent-oriented computing [Omicini and Zambonelli, 2004]

New LP Technologies

Essential features

- minimal and configurable engines running without heavy computational costs [Denti et al., 2001]
- multi-platform technology, including resource-constrained devices [Denti et al., 2013]
- interoperability with widespread languages, technologies, and standards
- clean model integration for conceptual integrity [Denti et al., 2005]

e.g. tuProlog `http://tuprolog.unibo.it`

Distributing Logic Programs

Concurrent logic processes

- processes represented by logic *processes* in concurrent systems
  - logic *agents* in the acceptation of coordination models [Ciancarini, 1996]
- there, concurrent / distributed systems are first of all collection of logic engines running in a concurrent / distributed way [Robertson, 2004]

e.g. Shared Prolog [Brogi and Ciancarini, 1991], $\mathcal{ACLT}$ [Denti et al., 1996]

$\rightarrow$ moving LP towards *distributed systems* and MAS [Baldoni et al., 2010]

Logic-based Coordination

Logic tuple spaces

- the space of interaction is built around *logic tuple spaces*
  [Ciancarini, 1994, Denti et al., 1996]
- allowing for heterogeneous distributed processes to be coordinated
- blackboards and programmable logic tuple spaces promote logic-based coordination of distributed systems

e.g. Shared Prolog [Brogi and Ciancarini, 1991], ReSpecT [Omicini and Denti, 2001]

Focus on...

Micro-intelligence for the IoT

- IoT scenarios (and CPS as well) mandates for distributed & situated micro-intelligence
  - huge numbers of small unit of computation
  - situated within a spatially-distributed environment
  - promoting the local exploitation of high-level symbolic languages
  - with inferential capabilities

? what role could the LP models and technologies play in such scenarios?

? what LP for the IoIT? [Arsénio et al., 2014]

Distributing Intelligence through Logic Engines I

Distributed logic engines

- lightweight and interoperable Prolog engines could be distributed even on resource-constrained devices [Denti et al., 2013]
- *multiple logic theories* are then scattered around, encapsulated in each engine, and associated to individual computational devices and *things* in the IoT
- each logic theory is *situated*, and represents what is true *locally*, according to a simple paraconsistent interpretation [Blair and Subrahmanian, 1989]
- LP *resolution process* is local to each theory / engine, so it is both standard and consistent [Robinson, 1965]

Distributing Intelligence through Logic Engines II

Our approach: tuProlog

tuProlog `http://tuprolog.unibo.it`

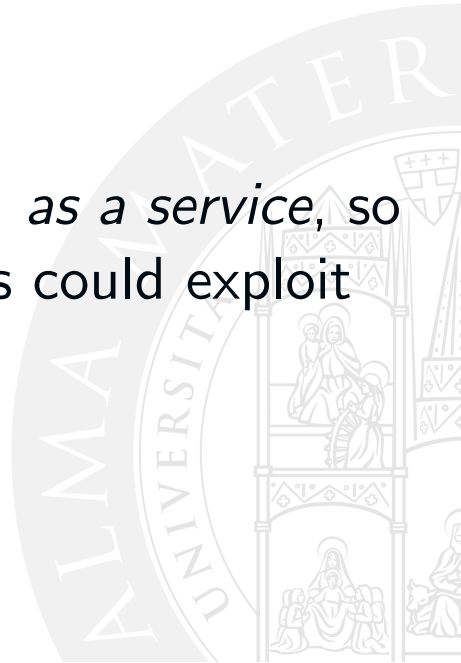
- is a light-weight Prolog system for distributed applications and infrastructures [Denti et al., 2001]
- is intentionally designed around a minimal core
- can be either statically or dynamically *configured* by loading/unloading libraries of predicates
- natively supports multi-paradigm programming [Denti et al., 2005], providing a clean, seamless integration model between Prolog and mainstream object-oriented languages
- runs on most known platforms and devices

Making LP Available in Distributed Systems I

- LP promotes *interactive programming*, even though one user / one engine sort
- in order to meet the needs of nowadays distributed systems, a more *standard* approach should be adopted, in terms of both architecture and technology
- possibly subsuming standard LP interaction

e.g. providing logic engines (theories, inference, resolution) *as a service*, so that standard distributed components and applications could exploit them

$\rightarrow$ exploiting a XaaS (everthing as a service) architecture



Making LP Available in Distributed Systems II

Our approach: Logic Programming as a Service (LPaaS)

Logic Programming as a Service (LPaaS) [Calegari et al., 2018c]

- provides an abstract view of an LP inference engine in terms of service
- promotes interoperability, encapsulation, and situatedness, as well as context-awareness
- each logic engine exposes its services concurrently to multiple clients, via two interfaces (*client* and *configurator*)
- preserving the standard resolution process
- providing for both *stateless* and *stateful* client-server interaction
- promoting *time-sensitive* and *space-sensitive* computation

Making LP Available in Distributed Systems III

Main ideas

- embedding a logic theory in every computational device composing the MAS environment
- along with a working logic engine providing basic inferential capabilities
- every local service is *situated* in that it describes what is locally true
- agents exploit both knowledge representation provided by logic theories and the inferential capabilities provided by logic engines as a distributed service
- thus injecting intelligence in MAS environment

Dealing with Situatedness & Domain-specific Scenarios I

- pervasive applications typically demands for domain-specific approaches
- multiple, heterogeneous devices in the IoT usually have specific application goals and manage specific sorts of information
- situatedness require reactivity to environment change

! LP is general enough to deal with whatever, but never specific enough to do it well

- many specific logics exist to deal with many diverse application scenarios—e.g., S4 modal logic for spatial computing
- LP can be extended to capture different logic and domains

Dealing with Situatedness & Domain-specific Scenarios II

Our example: Labelled Variables in Logic Programming

In Labelled Variables in Logic Programming (LVLP) [Calegari et al., 2018b]

- logic variable can be associated to a *label*
- labels are domain-specific
- each logic engine can be configured with its own set of domain-specific label, along with the specific functions to interfere with the logic resolution process, without losing declarativeness
- labels allows for a customisable computational model (domain-specific *labelled system*) bound to the standard LP computation

e.g. a wardrobe could host a LVLP engine whose labelled system represent its content, and helps in dress selection [Calegari et al., 2018b]

Dealing with Situatedness & Domain-specific Scenarios III

LPaaS & LVLP in (M)MAS

- LPaaS can be extended towards domain-specific computations via LVLP
- each LPaaS service can be enriched with its local LVLP domain-specific model

Dealing with Situatedness & Domain-specific Scenarios IV

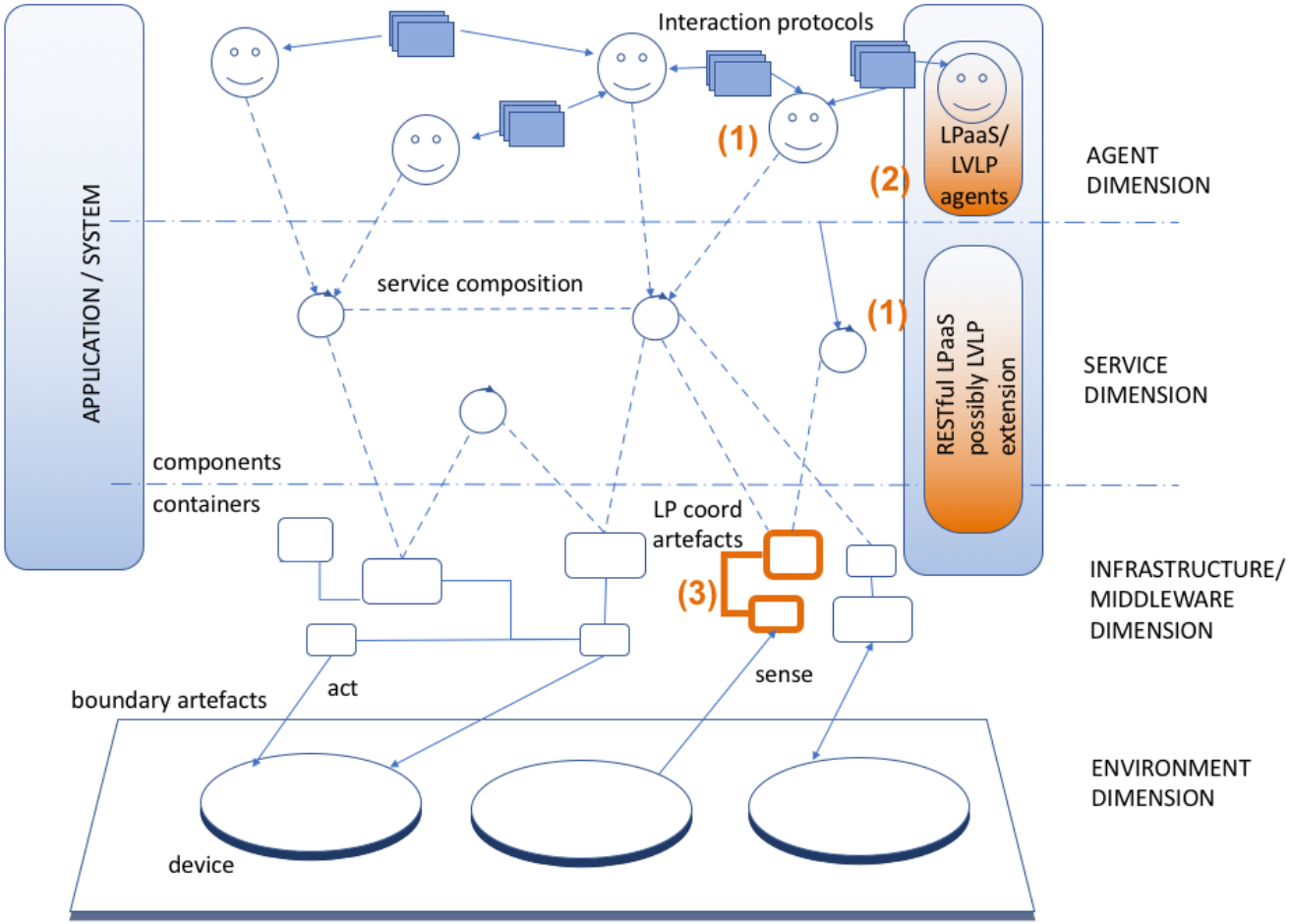
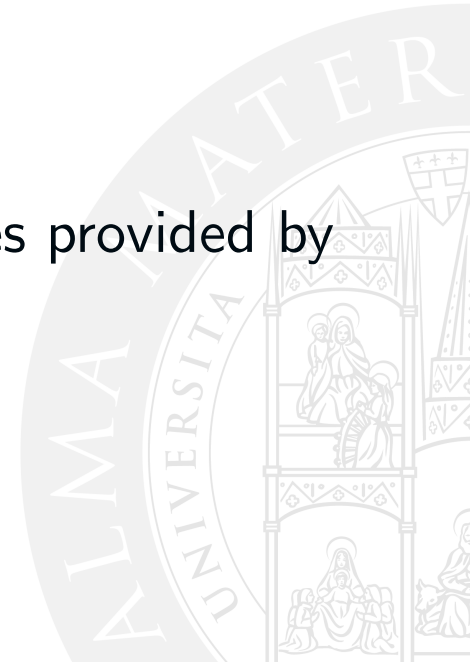


Figure: Overview of a LPaaS-LVLP MAS

Dealing with Situatedness & Domain-specific Scenarios V

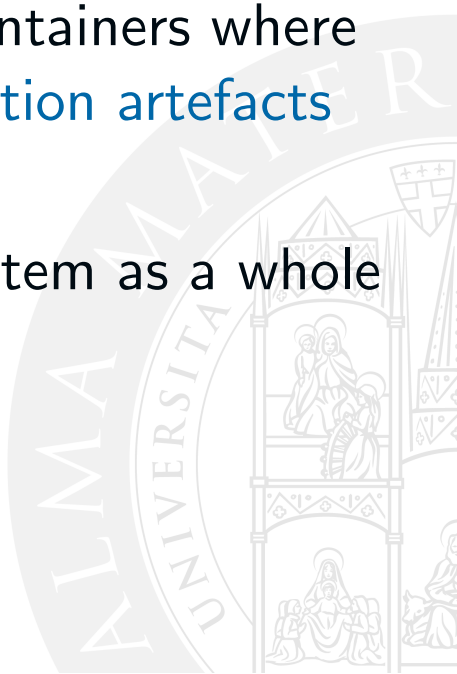
The picture above illustrates the model of the LPaaS-LVLP approach depicting the whole picture where

1. some agents are kept more lightweight and rely on infrastructural services (or other more "intelligent" agents) to get LPaaS functionalities
2. some agents embed the LPaaS-LVLP functionalities
3. some LP functionalities are embedded in some services provided by the middleware (namely by the containers)



Dealing with Situatedness & Domain-specific Scenarios VI

- at the bottom layer, the physical / computational environment lives, with boundary artefacts [Omicini et al., 2006a] taking care of its representation and interactions with the rest of the MAS
- then, typically, some middleware infrastructure provides common API and services to application-level software – i.e., the containers where service components live – there including the coordination artefacts [Omicini et al., 2004] governing the interaction space
- finally, on top of the middleware, the application / system as a whole lives, viewed as a *mixture* of *services* and *agents*



Logic-based Coordination Artefacts I

Social intelligence

- is another dimension that MAS models and technologies can fruitfully exploit
- in particular by building agent societies around *programmable*, *observable* and *malleable* coordination abstractions [Omicini et al., 2006a]

Logic-based Coordination Artefacts II

Coordination models and technologies

- play that role [Ciancarini, 1996] by providing *coordination media* as the basic abstractions around which agent societies can be designed [Ciancarini et al., 2000]
- via agent *coordination middleware*
- typically, a multiplicity of distributed coordination media are made available to the MAS: each group of agents can interact within a shared environment via a locally-deployed coordination medium

Logic-based Coordination Artefacts III

Coordination as a service [Viroli and Omicini, 2006]

- coordination artefacts are provided as services for the agents to participate to *properly-normed* social activities
- autonomous agents submit themselves to the coordination policies embedded in a coordination medium by choosing to interact with other agents through the service provided by the medium itself
- thus constituting an agent society

Logic-based Coordination Artefacts IV

Our approach: TuCSoN coordination middleware

The TuCSoN coordination model [Omicini and Zambonelli, 1999]

- provides MAS with *tuple centres* [Omicini and Denti, 2001] as the coordination media
- extending LINDA tuple spaces [Gelernter, 1985] with *programmability* [Denti et al., 1997]
- based on the ReSpecT [Omicini, 2007] logic-based language, where tuples are *logic tuples*
- TuCSoN middleware supports *multiple* ReSpecT tuple centres, which can be spatially distributed and connected via *linkability* [Omicini et al., 2006b]

Logic-based Coordination Artefacts V

Coordination artefacts [Omicini et al., 2004]

- encapsulate coordination policies for distributed systems
- can inject *social intelligence* within computational systems [Castelfranchi, 1998]
- can be *observable* and *malleable* [Omicini et al., 2008]

Logic-based coordination artefacts

- represent coordination policies declaratively
- pave the way towards (partial) formalisation of complex systems

Logic-based Coordination Artefacts VI

Our approach: ReSpecT coordination artefacts

ReSpecT coordination artefacts (ReSpecT) [Omicini, 2007]

- are *inspectable* and *malleable*
- can express any Turing-computable coordination policy [Denti et al., 1998]
- there, both *communication* and *coordination* theories are first-order logic theories
- can be used within the TuCSoN middleware for MAS coordination [Omicini and Zambonelli, 1999]

Logic-based Coordination Artefacts VII

ReSpecT tuple centres

- *logic tuples* in the tuple space
- ReSpecT *specification tuples* in the *specification space*
- which sets the coordinative behaviour of the medium by defining the *reaction* of the tuple centres to relevant MAS events

Logic-based Coordination Artefacts VIII

Features

- *programmabilty* of the coordination medium – along with the Turing-equivalence of ReSpecT [Denti et al., 1998] – makes it possible to embed any computable coordination policy within the coordination media, possibly allowing for *intelligent social behaviour*—e.g., [Omicini et al., 1995]
- *observability* of all tuples in a tuple centre makes it possible to reason about the state and behaviour of the corresponding agent society
- *malleability* of the tuple centre behaviour makes it possible to *change* (possibly intelligent) coordinative behaviours at run-time, paving the way towards *adaptability*

Logic-based Coordination Artefacts IX

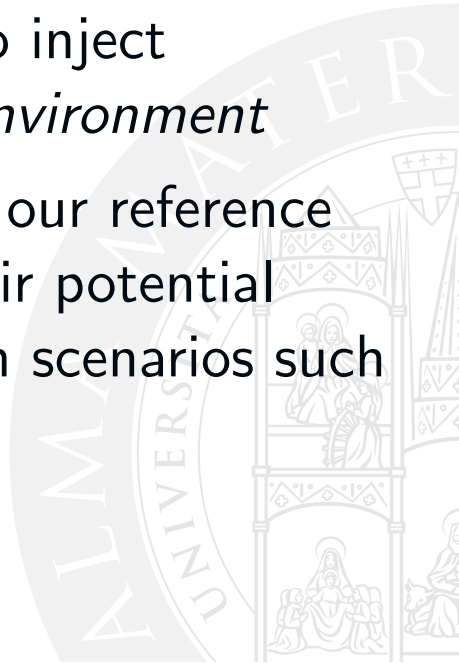
Scalability

- since ReSpecT tuple centres can be designed to be locally deployed within a physically-distributed environment
- with a (possibly huge) number of spatial containers and physical devices
- correspondingly embodying the laws for *local* coordination, ruling the social behaviour of the locally-interacting agents

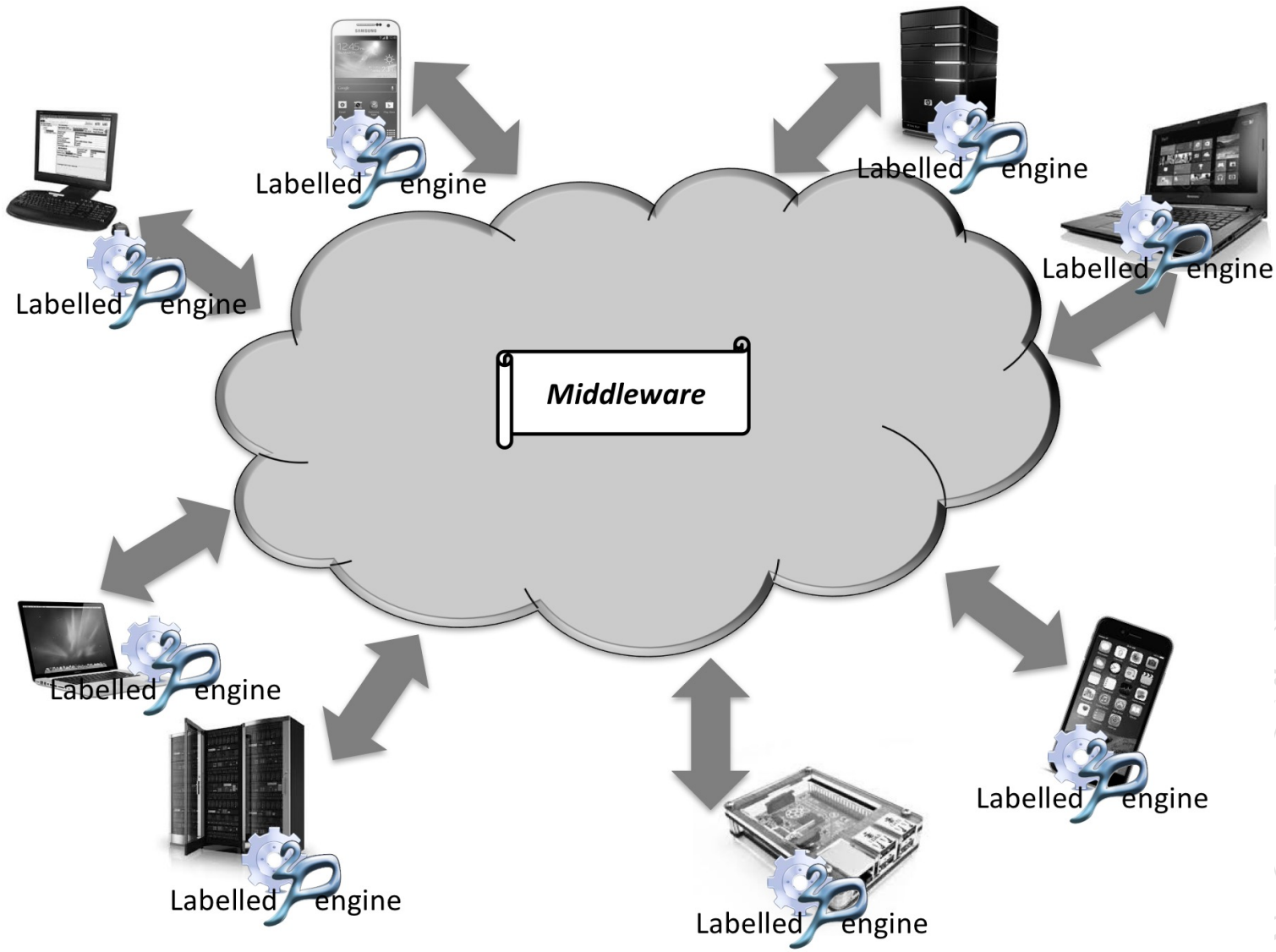
→ any coordination policy can then be tailored to the specific needs of every locality in a large-scale scenario—thus providing a way towards scalable coordination

Conclusion I

- when properly integrated within agent-based models and technologies, logic-based approaches have the potential to be exploited for *knowledge representation* and *reasoning* at the *large scale*
- we intentionally ignored the role of logic within agents, focussing instead on how logic-based models can be exploited to inject *micro-intelligence* through agent *societies* and MAS *environment*
- by adopting tuProlog, LPaaS, LVLP, and ReSpecT as our reference logic-based models and technologies, we discussed their potential impact within large-scale MAS for complex application scenarios such as the IoT



Conclusion II



Conclusion III

Altogether, the logic-based approaches discussed in this paper can lead to the overall architecture depicted in the picture above. There,

- distributed logic engines provide for widespread *micro-intelligence*
- exploited as standard *services* by components of any sorts, including intelligent agents via LPaaS
- situated and *domain-specific* extensions via LVLP
- coordinated via ReSpecT logic-based artefacts, encapsulating *social intelligence*
- based on a logic-based *middleware* like TuCSoN

Conclusion IV

In the end, whereas non-symbolic approaches to AI are currently taking the stage – especially in the eye of the public opinion – symbolic approaches, yet with a long read ahead of them, may still have the potential to be key players in the future of large-scale intelligent systems

References I

Arsénio, A., Serra, H., Francisco, R., Nabais, F., Andrade, J., and Serrano, E. (2014).
Internet of Intelligent Things: Bringing artificial intelligence into things and communication networks.
In *Inter-cooperative Collective Intelligence: Techniques and Applications*, volume 495 of *Studies in Computational Intelligence*, pages 1–37. Springer Berlin Heidelberg.

Atzori, L., Iera, A., and Morabito, G. (2010).
The Internet of Things: A survey.
Computer Networks, 54(15):2787–2805.

Baldoni, M., Baroglio, C., Mascardi, V., Omicini, A., and Torroni, P. (2010).
Agents, Multi-Agent Systems and Declarative Programming: Who, What, When, Where, Why, How?
In Dovier, A. and Pontelli, E., editors, *A 25 Year Perspective on Logic Programming. Achievements of the Italian Association for Logic Programming, GULP*, LNAI: State-of-the-Art Survey, chapter 10, pages 200–225. Springer.

Blair, H. A. and Subrahmanian, V. S. (1989).
Paraconsistent logic programming.
Theoretical Computer Science, 68(2):135–154.

References II

Brogi, A. and Ciancarini, P. (1991).
The concurrent language, Shared Prolog.
ACM Transactions on Programming Languages and Systems, 13(1):99–123.

Brooks, R. A. (1991a).
Intelligence without reason.
In Mylopoulos, J. and Reiter, R., editors, *12th International Joint Conference on Artificial Intelligence (IJCAI 1991)*, volume 1, pages 569–595, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Brooks, R. A. (1991b).
Intelligence without representation.
Artificial Intelligence, 47:139–159.

Calegari, R. (2018).
Micro-Intelligence for the IoT: Logic-based Models and Technologies.
PhD thesis, Alma Mater Studiorum—Università di Bologna, Bologna, Italy.

Calegari, R., Ciatto, G., Mariani, S., Denti, E., and Omicini, A. (2018a).
Micro-intelligence for the iot: Se challenges and practice in lpaas.
In *2018 IEEE International Conference on Cloud Engineering (IC2E 208)*, pages 292–297. IEEE Computer Society.

References III

Calegari, R., Denti, E., Dovier, A., and Omicini, A. (2018b).
Extending logic programming with labelled variables: Model and semantics.
Fundamenta Informaticae, 161(1-2):53–74.
Special Issue CILC 2016.

Calegari, R., Denti, E., Mariani, S., and Omicini, A. (2018c).
Logic programming as a service.
Theory and Practice of Logic Programming, 18(3-4):1–28.
Special Issue "Past and Present (and Future) of Parallel and Distributed Computation in (Constraint) Logic Programming".

Castelfranchi, C. (1998).
Modelling social action for AI agents.
Artificial Intelligence, 103(1-2):157–182.

Castelfranchi, C., Pezzullo, G., and Tummolini, L. (2010).
Behavioral implicit communication (BIC): Communicating with smart environments via our practical behavior and its traces.
International Journal of Ambient Computing and Intelligence, 2(1):1–12.

References IV

Ciancarini, P. (1994).
Distributed programming with logic tuple spaces.
New Generation Computing, 12(3):251–283.

Ciancarini, P. (1996).
Coordination models and languages as software integrators.
ACM Computing Surveys, 28(2):300–302.

Ciancarini, P., Omicini, A., and Zambonelli, F. (2000).
Multiagent system engineering: The coordination viewpoint.
In Jennings, N. R. and Lespérance, Y., editors, *Intelligent Agents VI. Agent Theories, Architectures, and Languages*, volume 1757 of *LNAI*, pages 250–259. Springer.
6th International Workshop (ATAL'99), Orlando, FL, USA, 15–17 July 1999. Proceedings.

Denti, E., Natali, A., and Omicini, A. (1997).
Programmable coordination media.
In Garlan, D. and Le Métayer, D., editors, *Coordination Languages and Models*, volume 1282 of *LNCS*, pages 274–288. Springer-Verlag.
2nd International Conference (COORDINATION'97), Berlin, Germany, 1–3 September 1997. Proceedings.

References V


Denti, E., Natali, A., and Omicini, A. (1998).
On the expressive power of a language for programming coordination media.
In *1998 ACM Symposium on Applied Computing (SAC'98)*, pages 169–177, Atlanta, GA, USA. ACM.
Special Track on Coordination Models, Languages and Applications.


Denti, E., Natali, A., Omicini, A., and Venuti, M. (1996).
Logic tuple spaces for the coordination of heterogeneous agents.
In Baader, F. and Schulz, K. U., editors, *Frontiers of Combining Systems*, volume 3 of *Applied Logic Series*, pages 147–160, Munich, Germany. Kluwer Academic Publishers.


Denti, E., Omicini, A., and Calegari, R. (2013).
tuProlog: Making Prolog ubiquitous.
ALP Newsletter.


Denti, E., Omicini, A., and Ricci, A. (2001).
tuProlog: A light-weight Prolog for Internet applications and infrastructures.
In Ramakrishnan, I., editor, *Practical Aspects of Declarative Languages*, volume 1990 of *LNCS*, pages 184–198. Springer.
3rd International Symposium (PADL 2001), Las Vegas, NV, USA, 11–12 March 2001. Proceedings.

References VI

Denti, E., Omicini, A., and Ricci, A. (2005).
Multi-paradigm Java-Prolog integration in tuProlog.
Science of Computer Programming, 57(2):217–250.

Fortino, G., Guerrieri, A., and Russo, W. (2012).
Agent-oriented smart objects development.
In *2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2012)*, pages 907–912. IEEE.

Fortino, G., Guerrieri, A., Russo, W., and Savaglio, C. (2014).
Integration of agent-based and Cloud Computing for the smart objects-oriented IoT.
In *2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 493–498.

Fredriksson, M. and Gustavsson, R. (2004).
Online engineering and open computational systems.
In Bergenti, F., Gleizes, M.-P., and Zambonelli, F., editors, *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*, volume 11 of *Multiagent Systems, Artificial Societies, and Simulated Organization*, pages 377–388. Kluwer Academic Publishers.

References VII

Gelernter, D. (1985).
Generative communication in Linda.
ACM Transactions on Programming Languages and Systems, 7(1):80–112.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013).
Internet of Things (IoT): A vision, architectural elements, and future directions.
Future Generation Computer Systems, 29(7):1645–1660.

Gupta, G., Pontelli, E., Ali, K. A., Carlsson, M., and Hermenegildo, M. V. (2001).
Parallel execution of Prolog programs: a survey.
ACM Transactions on Programming Languages and Systems, 23(4):472–602.

Idelberger, F., Governatori, G., Riveret, R., and Sartor, G. (2016).
Evaluation of logic-based smart contracts for blockchain systems.
In Alferes, J. J., Bertossi, L., Governatori, G., Fodor, P., and Roman, D., editors, *Rule Technologies. Research, Tools, and Applications*, volume 9718 of *Lecture Notes in Computer Science*, pages 167–183. Springer.

Jamont, J.-P. and Occello, M. (2016).
Meeting the challenges of decentralised embedded applications using multi-agent systems.
International Journal of Agent-Oriented Software Engineering, 5(1):22–68.

References VIII

Kato, T., Chiba, R., Takahashi, H., and Kinoshita, T. (2015).
Agent-oriented cooperation of iot devices towards advanced logistics.
In *2015 IEEE 39th Annual Computer Software and Applications Conference (COMPSACW 2015)*, volume 3, pages 223–227.

Lippi, M., Mamei, M., Mariani, S., and Zambonelli, F. (2017).
Coordinating distributed speaking objects.
In *37th IEEE International Conference on Distributed Computing Systems (ICDCS 2017)*. IEEE Computer Society.

Manzalini, A. and Zambonelli, F. (2006).
Towards autonomic and situation-aware communication services: the CASCADAS vision.
In *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*, pages 383–388.

Mariani, S. and Omicini, A. (2013).
Molecules of Knowledge: Self-organisation in knowledge-intensive environments.
In Fortino, G., Bădică, C., Malgeri, M., and Unland, R., editors, *Intelligent Distributed Computing VI*, volume 446 of *Studies in Computational Intelligence*, pages 17–22. Springer.
6th International Symposium on Intelligent Distributed Computing (IDC 2012), Calabria, Italy, 24-26 September 2012. Proceedings.

References IX

Omicini, A. (2001).
SODA: Societies and infrastructures in the analysis and design of agent-based systems.
In Ciancarini, P. and Wooldridge, M. J., editors, *Agent-Oriented Software Engineering*, volume 1957 of *Lecture Notes in Computer Science*, pages 185–193. Springer-Verlag.
1st International Workshop (AOSE 2000), Limerick, Ireland, 10 June 2000. Revised Papers.

Omicini, A. (2007).
Formal ReSpecT in the A&A perspective.
Electronic Notes in Theoretical Computer Science, 175(2):97–117.
5th International Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA'06), CONCUR'06, Bonn, Germany, 31 August 2006.
Post-proceedings.

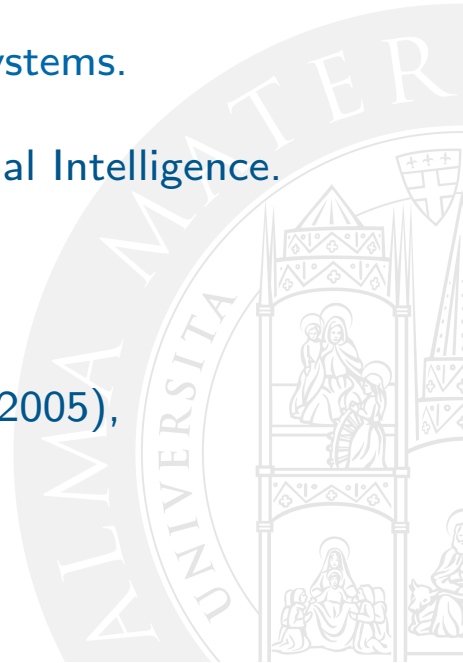
Omicini, A. and Denti, E. (2001).
From tuple spaces to tuple centres.
Science of Computer Programming, 41(3):277–294.

References X

Omicini, A., Denti, E., and Natali, A. (1995).
Agent coordination and control through logic theories.
In Gori, M. and Soda, G., editors, *Topics in Artificial Intelligence*, volume 992 of *LNAI*, pages 439–450. Springer-Verlag.
4th Congress of the Italian Association for Artificial Intelligence (AI*IA'95), Florence, Italy, 11–13 October 1995, Proceedings.

Omicini, A. and Mariani, S. (2013).
Agents & multiagent systems: En route towards complex intelligent systems.
Intelligenza Artificiale, 7(2):153—164.
Special Issue Celebrating 25 years of the Italian Association for Artificial Intelligence.

Omicini, A., Ricci, A., and Viroli, M. (2006a).
Agens Faber: Toward a theory of artefacts for MAS.
Electronic Notes in Theoretical Computer Science, 150(3):21–36.
1st International Workshop "Coordination and Organization" (CoOrg 2005), COORDINATION 2005, Namur, Belgium, 22 April 2005. Proceedings.



References XI

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.
Special Issue on Foundations, Advanced Topics and Industrial Perspectives of Multi-Agent Systems.

Omicini, A., Ricci, A., Viroli, M., Castelfranchi, C., and Tummolini, L. (2004).
Coordination artifacts: Environment-based coordination for intelligent agents.
In Jennings, N. R., Sierra, C., Sonenberg, L., and Tambe, M., editors, *3rd international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, volume 1, pages 286–293, New York, USA. ACM.

Omicini, A., Ricci, A., and Zaghini, N. (2006b).
Distributed workflow upon linkable coordination artifacts.
In Ciancarini, P. and Wiklicky, H., editors, *Coordination Models and Languages*, volume 4038 of *LNCS*, pages 228–246. Springer.
8th International Conference (COORDINATION 2006), Bologna, Italy, 14–16 June 2006. Proceedings.

References XII

Omicini, A. and Zambonelli, F. (1999).
Coordination for Internet application development.
Autonomous Agents and Multi-Agent Systems, 2(3):251–269.
Special Issue: Coordination Mechanisms for Web Agents.

Omicini, A. and Zambonelli, F. (2004).
MAS as complex systems: A view on the role of declarative approaches.
In Leite, J. A., Omicini, A., Sterling, L., and Torroni, P., editors, *Declarative Agent Languages and Technologies*, volume 2990 of *LNCS*, pages 1–17. Springer.
1st International Workshop (DALT 2003), Melbourne, Australia, 15 July 2003. Revised Selected and Invited Papers.

Piancastelli, G., Benini, A., Omicini, A., and Ricci, A. (2008).
The architecture and design of a malleable object-oriented Prolog engine.
In Wainwright, R. L., Haddad, H. M., Menezes, R., and Viroli, M., editors, *23rd ACM Symposium on Applied Computing (SAC 2008)*, volume 1, pages 191–197, Fortaleza, Ceará, Brazil. ACM.
Special Track on Programming Languages.

References XIII

Rao, A. S. and Georgeff, M. P. (1995).
BDI agents: From theory to practice.
In Lesser, V. R. and Gasser, L., editors, *1st International Conference on Multi Agent Systems (ICMAS 1995)*, pages 312–319, San Francisco, CA, USA. The MIT Press.

Rao, A. S. and Georgeff, M. P. (1998).
Decision procedures for BDI logics.
Journal of Logic and Computation, 8(3):293–342.

Robertson, D. (2004).
Multi-agent coordination as Distributed Logic Programming.
In Demoen, B. and Lifschitz, V., editors, *Logic Programming. 20th International Conference, ICLP 2004, Saint-Malo, France, September 6-10, 2004. Proceedings*, volume 3132, pages 416–430. Springer.

Robinson, J. A. (1965).
A machine-oriented logic based on the resolution principle.
Journal of the ACM, 12(1):23–41.

References XIV

Savaglio, C., Fortino, G., Ganzha, M., Paprzycki, M., Badica, C., and Ivanovic, M. (2018).
Agent-based computing in the internet of things: A survey.
In Ivanović, M., Bădică, C., Dix, J., Jovanović, Z., Malgeri, M., and Savić, M., editors, *Intelligent Distributed Computing XI*, volume 737 of *Studies in Computational Intelligence*, pages 307–320. Springer.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016).
Mastering the game of Go with deep neural networks and tree search.
Nature, 529:484–489.

Singh, M. P. and Chopra, A. K. (2017).
The Internet of Things and multiagent systems: Decentralized intelligence in distributed computing.
In *37th IEEE International Conference on Distributed Computing Systems (ICDCS 2017)*. IEEE Computer Society.

Spanoudakis, N. and Moraitis, P. (2015).
Engineering ambient intelligence systems using agent technology.
IEEE Intelligent Systems, 30(3):60–67.

References XV


Tanenbaum, A. S. and van Steen, M. (2007).
Distributed Systems. Principles and Paradigms.
Pearson Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition.


Viroli, M. and Omicini, A. (2006).
Coordination as a service.
Fundamenta Informaticae, 73(4):507–534.
Special Issue: Best papers of FOCLASA 2002.


Viroli, M., Omicini, A., and Ricci, A. (2005).
Engineering MAS environment with artifacts.
In Weyns, D., Parunak, H. V. D., and Michel, F., editors, *2nd International Workshop "Environments for Multi-Agent Systems" (E4MAS 2005)*, pages 62–77, AAMAS 2005, Utrecht, The Netherlands.


Whitworth, B. (2006).
Socio-technical systems.
In Ghaou, C., editor, *Encyclopedia of Human Computer Interaction*, pages 533–541. IGI Global.

References XVI

Xiang, C. and Li, X. (2012).
General analysis on architecture and key technologies about Internet of Things.
In *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE 2012)*, pages 325–328.

Zambonelli, F. and Omicini, A. (2004).
Challenges and research directions in agent-oriented software engineering.
Autonomous Agents and Multi-Agent Systems, 9(3):253–283.
Special Issue: Challenges for Agent-Based Computing.

URLs

Slides

- On APICe

→ `http://apice.unibo.it/xwiki/bin/view/Talks/MicrointelligencelpMmas2018`

- On SlideShare

→ `http://www.slideshare.net/andreaomicini/injecting-microintelligence-in-the-iot-logicbased-approaches-for-mmas`

Injecting (Micro)Intelligence in the IoT: Logic-based Approaches for (M)MAS

Andrea Omicini
andrea.omicini@unibo.it

Roberta Calegari
roberta.calegari@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum—Università di Bologna, Italy

Invited Talk @ *International Workshop on Massively Multi-Agent Systems (MMAS 2018)*
Stockholm, Sweden, 14 July 2018