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

Andrea Omicini[1 [0000−0002−6655−3869]] and Roberta Calegari[1 [0000−0003−3794−2942]]

Dipartimento di Informatica, Scienza e Ingegneria (DISI)
Alma Mater Studiorum–Università di Bologna
{andrea.omicini, roberta.calegari}@unibo.it

**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 paper 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.

**Keywords:** pervasive system · MMAS · micro-intelligence · logic-based · LPaaS.

## 1 Introduction

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, along with the increasing diffusion of sensor networks, actuator devices, and computational resources in general, is rapidly transforming urban environments into wannabe-smart environments on a massively-large scale.

Whereas model, technology, and methodology aspects are nowadays the subject of many research activities [43,44], the issue of (*ubiquitous*) *intelligence* is the key to make environment really smart. Many novel approaches to machine intelligence nowadays are increasingly focussing on *non-symbolic* approaches – such as deep learning with neural neural networks, e.g., [38] – and how to make them work on the large scale. As promising as that may look – on the premise that those approaches have the potential minimise the engineering efforts towards large-scale intelligence – what we do also need is 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.

To this end, 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 are the most viable abstraction to encapsulate fundamental features such as *control*, *goals*, *mobility*, *intelligence* [44]. In particular, agents are widely recognised as the main abstractions to *distribute* intelligence in complex systems of any sort—e.g., [16]. Also, MAS abstractions such as *society* and *environment* [28] are essential to cope with the complexity of nowadays application scenarios—as well as to inject intelligence in complex computational systems [32].

On the other hand, declarative and logic-based technologies quite straightforwardly address issues such as *observability* and *explainability*, in particular when exploiting their inferential capabilities—e.g., [20]. Since logic-based approaches already have a well-understood role in building intelligent agents [36], we focus instead on the role that *logic-based* models and technologies can play when used to rule *agent societies* as well as to engineer *agent environment*.

In this paper we recap some of our research results about the role of logic-based models and technologies in MAS, discussing how they can be exploited to inject *micro-intelligence* [2] in large-scale scenarios. In particular, we show how Logic Programming as a Service (LPaaS [7]) can be used to distribute intelligence in KIE (knowledge-intensive environments), how Labelled Variables in Logic Programming (LVLP [4]) can help introducing *domain-specific* intelligence, and how the logic-based coordination language ReSpecT [30] can provide for social intelligence in MAS. Also, we show how logic-based approaches makes it possible to address in principle issues such as observability, formalisability, explainability, and accountability.

## 2 Background and Related Work

### 2.1 (M)MAS, IoT & Intelligence

Agent-oriented engineering and (massive) multi-agent systems ((M)MAS) have been already recognised as the most promising way for developing applications for the Internet of Tthings (IoT) as well as cyber-physical systems (CPS), since they are well-suited for supporting decentralised, loosely-coupled and highly dynamic, heterogeneous and open systems, in which components should be able to cooperate opportunistically [37]. Along that line, IoT and pervasive systems can be seen as (M)MASs, devoted to monitor and control our environments where the MAS abstractions, techniques, and methods are essential to cope with the complexity of the application scenarios. Among the many, [22,26,39] show how agents are the most natural way of approaching IoT systems featuring complexity, dynamism, situatedness, and autonomy. Moreover, [16] shows how 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.

In the following we focus our attention on the intelligence issue, a hot topic in current IoT research, according to the idea that devices in IoT pervasive

systems have to be massively networked and provided with (different degrees of) intelligence, in order to interoperate and cooperate to achieve different goals. To this end, agent-oriented models and technologies are gaining momentum for embedding decentralised intelligence—as discussed in [21].

### 2.2 Engineering Intelligence in the IoT: Key Challenges

The IoT extends the current Internet technology by connecting different types of things (objects or devices) with each other and enable them to be communicated *smartly* [42]. Consequently, this concept is designed to connect millions of things together, but things of this number need large storage spaces and generate heavy traffic, which could create many network issues. Furthermore, while the things are connected with each other, they are not necessarily able to *communicate* and *interoperate* with each other [40]. Their ability to communicate with each other depends on the similarity of the service they are assigned to do [23]. These deficiencies are due to that the things lack the ability to reason on their environments and to subsequently make intelligent decisions and actions to achieve their objectives. Often, the objects used in the IoT concept – e.g., RFID, sensors, televisions, washing machines, etc. – lacks intelligence due to hardware and software limitations.

In addition, the issues of security, governance and standardisation have to be taken in consideration [40]. No consolidated set of software engineering best practices has emerged so far in the IoT world [24,43] in order to face this issue: so, properly engineering such a new generation of scalable, highly-reactive, (often) resource-constrained software systems is still a challenge from the SE viewpoint.

Furthermore, *sociotechnical features* such as observability, explainability, and accountability have to be addressed whenever the complexity of automated reasoning goes far beyond the human ability to understand—and we are already there, basically. Understanding at some level how huge aggregates of intelligent devices and agents evolve and affect our social and organisation processes is to become essential for technical, social, normative, and ethical reasons. Apparently, this is where symbolic approaches like logic-based ones are going to be of help.

Along that line, in the following we focus on the practice of engineering and design intelligence in distributed systems, in particular in the IoT systems, by discussing our latest results in the field based on logic-based models and technologies.

## 3 Logic-based Approaches for (M)MAS and IoT

In order to face the challenges and the open issues highlighted in Subsection 2.2 we proposed a logic-based approach for injecting *micro-intelligence* in large-scale MAS, such as IoT pervasive systems.

Logic-based languages and technologies represent in principle a natural candidate for injecting intelligence within computational systems [1]: yet, many issues

have to be addressed—among these, the computational costs, the machinery often not suited for programming in the large (the intrinsic modularity provided by predicates does not scale up effectively, and modules are not enough for the purpose), the "no types" approach that makes it difficult to deal with domain-specific applications, the distance from mainstream programming paradigms, the integration with mainstream technologies. Moreover, MMAS and IoT inherently call for a fully distributed architecture, which is why the relationship between logic and physical distribution needs to be addressed and investigated in depth. Classical logic approaches apparently do not cope well with the current perception of distributed systems—for instance, the universal notion of consistency of the logic theory does not fit the incompleteness and inconsistency intrinsically implied by distributed scenarios.

Anyway, overall, we believe that a logic-based approach can bring some remarkable benefits in pervasive system, in particular dealing with the AI fears, by promoting observability, malleability, understandability, formalisability, and norm compliance—yet, a basic re-interpretation of some basic concepts of logic programming is clearly needed in order to cope well with the aforementioned issues.

Along with the reinterpretation of classical logic approach under the IoT vision, we define the concept of *micro-intelligence* [2]: small chunks of machine intelligence, spread all over the system, capable to enable the individual intelligence of any sort of devices, promoting coordination and interoperation among different entities. The micro-intelligence vision promote the ubiquitous distribution of intelligence in large pervasive systems such as IoT ones, in particular when coupled with agent-based technologies and methods, at both the individual and the collective level—when combined with an overall architectural view of large-scale systems exploiting logic-based technologies. 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.

As a source of intelligence, we focus here on logic-based engines–in particular LP (logic programming) engines, offering inference capabilities spread all over the network. The potential of logic-based model and its extensions is first of all related to the observability and understandability of the entire system. Declarativeness and explicit knowledge representation of LP enable knowledge sharing at the most adequate level of abstraction, while supporting modularity and separation of concerns [27], which are specially valuable in open and dynamic distributed systems. As a further element, LP sound and complete semantics naturally enables intelligent agents to reason and infer new information in a sound way. Traditional LP has well proven to work well both as a knowledge representation (KR) language and as an inference platform for rational agents. Logic agents may interact with an external environment by means of a suitably defined observe–think–act cycle. Finally, LP extensions or logic-based computational models – such as meta-reasoning about situations [25] – could be incorporated so as to enable complex behaviours tailored to the situated components.

Accordingly, our result in the field are *(i)* the Logic Programming as a Service (LPaaS henceforth [7]) model for distributing logic programs and logic engines accordingly to the SOA architecture; *(ii)* the Labelled Variables in Logic Programming (LVLPhenceforth [4]) extension to the LP model to answer the domain specificity issue of pervasive system; *(iii)* the possibility of exploiting logic-based coordination artefacts and logical tuples with the ReSpecT coordination language [30] upon the TuCSoN MAS coordination middleware [35].

In the following we shortly discuss each contribution, while trying to provide a general view of how logic-based models and technologies can be exploited to inject intelligence into (M)MAS.

## 4 LPaaS & LVLP for Environment Intelligence

### 4.1 Vision

The novel LPaaS & LVLP models and architectures – and the corresponding technology– express the concept of micro-intelligence defined above. In particular, we define two different, integrated models, namely Logic Programming as a Service and Labelled Variables in Logic Programming designed so as to act synergistically in order to support the distribution of intelligence in pervasive systems.

One one side, the LPaaS architecture is designed so that LP can act as a source of distributed intelligence for the IoT world, by providing an abstract view of LP inference engines in terms of service. It exploits the XaaS (everything as a service) metaphor to promote maximum availability and interoperability while promoting context-awareness: any resource of any sort should be accessible as a service (possibly an intelligent one) via standard network operations. From the MAS viewpoint, LPaaS takes care of distributing knowledge as well as reasoning capabilities in the agent environment based on LP.

On the other side, LVLP extend the LP model to enable diverse computational models, each tailored to a specific situated component, to coherently and fruitfully coexist and cooperate within the same (logic-based) framework, so as to cope with *domain-specific* aspects. From the MAS viewpoint, LVLP takes care of embedding domain specific knowledge and reasoning capabilities at the micro-level [44].

The added value of such a hybrid approach is to make it possible to exploit LP for what it is most suited for, such as symbolic computation, delegating other aspects – such as situated computations – to other languages or other computational levels.

### 4.2 LPaaS in a nutshell

The LPaaShas been introduced in [5,6]; a detailed discussion on the technology can be found in [3], whereas a comprehensive account was presented in [7].

The main idea behind LPaaS is to embed a (possibly situated) logic theory in every computational device composing the MAS environment, along with a

working logic engine providing basic inferential capabilities. Multiple theories are intended to be consistent under the fundamental assumption that each logic theory describes axiomatically what is locally true—so, preventing logical inconsistency a priori. So, agents exploit both knowledge representation provided by logic theories and the inferential capabilities provided by logic engines as a distributed service, injecting intelligence in MAS environment.

To do so, LPaaS promote a radical re-interpretation of some basic facets of LP, moving LP towards the notion of *situated service*. Such a notion articulates along four major aspects: *i* the preservation (with re-contextualisation) of the SLD resolution process; *ii* stateless interactions; *iii* time-sensitive computations; *iv* space-sensitive computations. The SLD resolution process remains a staple in LPaaS: yet, it is re-contextualised in the situated nature of the specific LP service. This means that, given the precise *spatial*, *temporal*, and *general* contexts within which the service is operating *when the resolution process starts*, the process follows the usual rules of SLD resolution: situatedness is accounted for through the service abstraction with respect to such three contexts.

From an architectural viewpoint, SOA the standard approach for distributed system engineering: so, LPaaS adopts the *Software as a Service* architecture as its architectural reference [12], moreover LP services in LPaaS can be fruitfully interpreted as *microservices* [15].

In such a vision, the LPaaS abstraction represents a form of micro-intelligence, enabling situated reasoning, interaction and coordination in distributed systems, as the process by which an entity reasons about its local actions and the (anticipated) actions of others to try and ensure the community acts in a coherent manner. LPaaS means to enable reasoning in distributed systems taking into account the explicit definition of the spatial-temporal structure of the environment in which situated entities act and interact, thus promoting an approach based on key features descending from the inner nature of pervasive systems like coordination, interaction and environment awareness.

### 4.3 LVLP in a nutshell

Specificity of local domains, however, might not be easily addressed by the general-purpose approach of standard LP—in terms of both specific domain knowledge and domain-specific inference. For instance, the typical LP language, Prolog, is even untyped—which, roughly speaking, is good for generality, bad for making applications domain specific.

To this end, specific domain intelligence can be injected in MAS environment via Labelled Variables in Logic Programming, formally discussed in [4] and fully developed in [4]. Basically, the LVLP approach is consistent with that part of AI literature that has established that domain-specific knowledge is a major determinant of the success of KIE systems such as expert systems [11].

LVLP builds upon the general notion of *label* as defined by Gabbay [17], and adopts the techniques introduced by Holzbaur [19] to develop a generalisation of LP where labels are exploited to define computations in domain-specific contexts. LVLP allows heterogeneous devices in large-scale applications to have specific

application goals and manage specific sorts of information, enabling reactivity to environment change while capturing diverse logic and domains exploiting the concept of labelled variable.

An LVLP program is a collection of LVLP *rules* of the form $Head \leftarrow Labelling, Body$ to be read as "*Head* if *Body* given *Labelling*". There, *Head* is an atomic formula, *Labelling* is the list of labelled variables in the clause, and *Body* is a list of atomic formulas. By design, only variables can be labelled, there, given two generic LVLP terms, the unification result is represented by the extended tuple $(true/false, \theta, \ell)$ where $true/false$ represents the existence of an answer, $\theta$ is the *most general unifier* ($mgu$), and $\ell$ is the new label associated to the unified variables defined by the user defined (label-)combining function. The unification process is extended by two functions: namely, the (label-)combining function exploited during the unification of two labelled variables and the compatibility function exploited during the unification between a ground term and a labelled variable, ensuring the term is compatible with the label of the variable when interpreted in the domain of labels.

Among the many differences w.r.t. the approaches in the state of the art is the fact that our approach does not change the basic of the logic language, which remains the same, but allows for different specific extensions tailored to local needs. Overall, the main idea behind LVLP is to enable diverse computational models via labelled variables: each logic engine can exploit its own local label systems tailored to the specific needs of situated components, to coherently and fruitfully coexist side by side, interacting within a logic-based framework.

### 4.4 LPaaS & LVLP in (M)MAS

In [8] we discuss how the LPaaS architecture can be exploited to inject micro-intelligence in MAS , by enriching the overall MAS architecture with the notion of LPaaS agent / service, which allows for situated reasoning on locally-available data by design. The LPaaS model can be further extended towards domain-specific computations via LVLP.

As mentioned above, the multi-agent paradigm offers a powerful mechanism for *autonomous* and *situated* behaviour, *social* and *cooperative* exchanges within *organisations* that are required for large-scale systems. Besides autonomy, situatedness, and sociality, large-scale application scenarios such as the IoT may benefit from other agent features – e.g., *mobility* and *intelligence* – that could straightforwardly map onto the multitude of heterogeneous devices. However, whereas mobility may come at a reasonable cost, intelligence is considerably a more challenging issue, in particular when computationally expensive technologies – such as machine learning, common-sense reasoning, natural language processing, advanced situation recognition and context awareness – are involved. Along this line, whenever local agent intelligence cannot be available for any reason – i.e. memory constraints hindering the opportunity to have a local KB, CPU constraints limiting efficiency of reasoning, etc. – a given agent may simply request to another, "more intelligent" one, to perform some intelligent activity on its behalf.

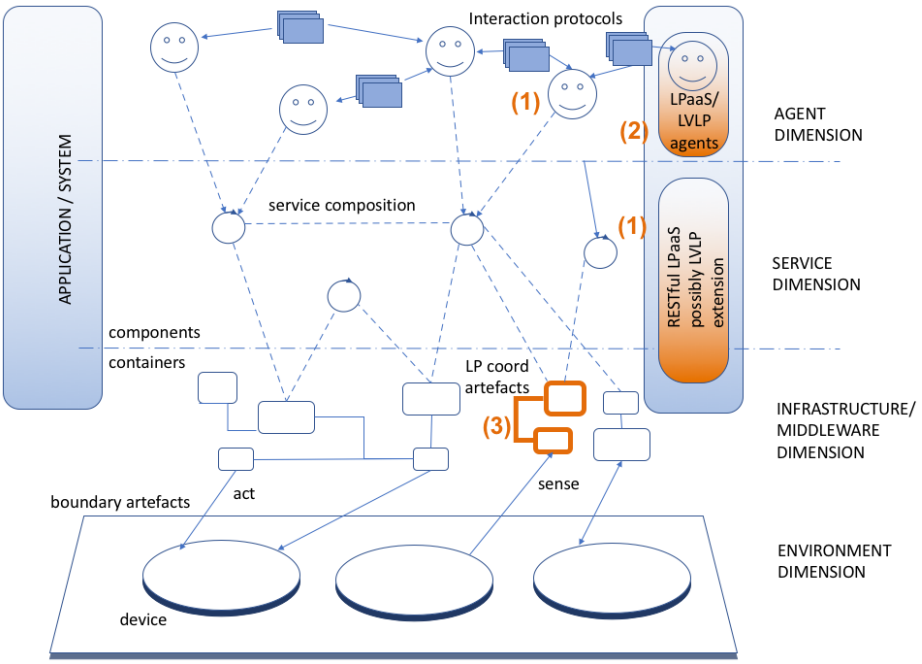


**Fig. 1.** Overview of a LPaaS-LVLP MAS

More interestingly here, intelligent activity could be also delegated to the environment—for instance, relying on LPaaS-LVLP services instead of the agents. In such a scenario, agents are always computationally efficient and responsive, since they are able to delegate reasoning-related tasks – such as situation recognition, planning, inference of novel information, etc. – to dedicated infrastructural services based on LPaaS-LVLP.

Fig. 1 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). In particular, at the bottom layer, the physical / computational environment lives, with *boundary artefacts* [33] 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* [33] 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.

## 5 ReSpecT & Logical Tuples for Social Intelligence

*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 [33]. This is in fact the role of *coordination* models and technologies [9], where *coordination media* provided by the agent middleware are the basic abstractions around which agent societies can be designed [10]. The notion of *coordination as a service* expresses precisely that: autonomous agents submit themselves to the coordination policies embedded in a coordination medium by choosing to interact with other agents through the medium itself, thus constituting an agent society [41].

The TuCSoN coordination model [35] provides MAS with *tuple centres* [30] that extend LINDA tuple spaces [18] with *programmability* [13] based on the ReSpecT [29] logic-based language. In particular, ReSpecT tuple centres contain *logic tuples* in the tuple space, and 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.

More precisely, a ReSpecT specification tuple is a special kind of first-order logic tuple of the form `reaction(E,G,R)`, where: `E` is the *triggering event* of the reaction, that is, the coordination-related event – represented by the coordination primitive invoked – whose occurrence triggers evaluation of the *reaction*; `G` is the (set of) *guard predicate*(s) which must evaluate to `true` for the reaction to actually execute—enabling fine-grained control over reactions execution; `R` is the *reaction body*, that is, the set of Prolog computations and ReSpecT primitives to execute to bring about the reaction effects.

Whereas the ReSpecT tuple space contains the (logic) tuples used for the *communication* among agents, the ReSpecT specification space contains the (logic) specification tuples used for the *coordination* of agents. So, ReSpecT tuple centres exploit logic for both knowledge representation (in the communication space) and coordination policies (in the coordination space) in agent societies.

Overall, ReSpecT tuple centres are *programmable* – via ReSpecT specification tuples –, *observable* – in that both the basic and the specification tuples can be accessed by the agents – and *malleable*—in that suitable specification primitives can be used to change the ReSpecT behaviour specification at run-time. Programmabilty of the coordination medium – along with the Turing-equivalence of ReSpecT [14] – makes it possible to embed any computable coordination policy within the coordination media, possibly allowing for *intelligent social behaviour*—e.g., [31]. 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*.

Moreover, a typical feature of tuple-based middleware – which clearly fits large-scale scenarios – is the multiplicity of distributed coordination media made available to the MAS: each group of agents can interact within a shared environment via a locally-deployed coordination medium. Along this line, TuCSoN

middleware supports multiple ReSpecT tuple centres, which can be spatially distributed and connected via *linkability* [34]. Accordingly, 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, and 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*.

## 6 Conclusion

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. In this invited paper we intentionally ignore 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 LPaaS, LVLP, and ReSpecT as our reference logic-based models and technologies, we discuss their potential impact within large-scale MAS for complex application scenarios such as the IoT.

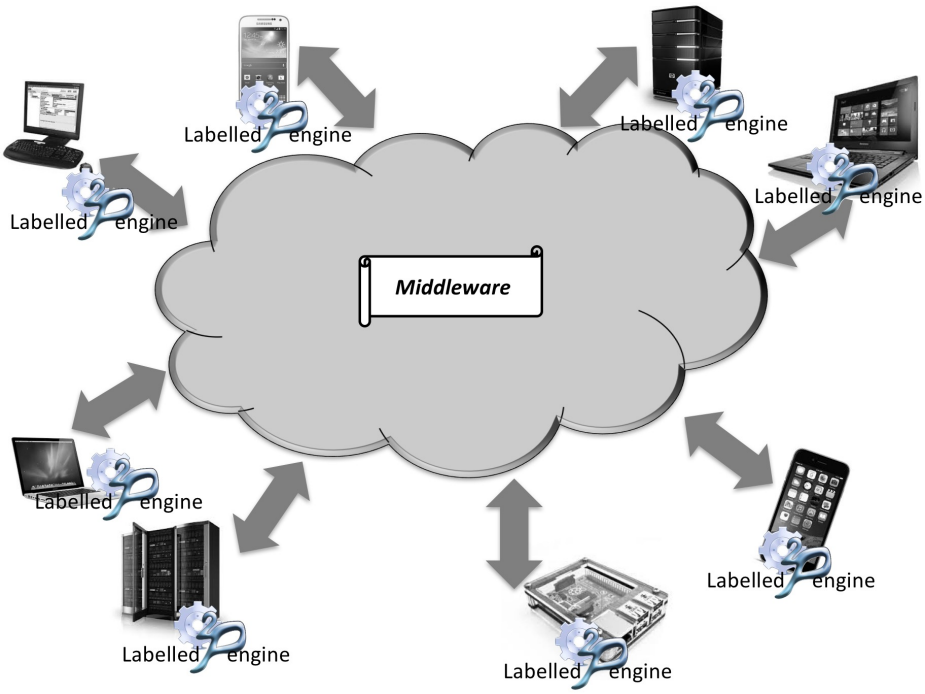


**Fig. 2.** Overview LPaaS-LVLP-ReSpecT MAS Architecture

Altogether, the logic-based approaches discussed in this paper can lead to the overall architecture depicted in Fig. 2. 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

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, still have the potential to be key players in the future of large-scale intelligent systems.

## References


1. Brownlee, J.: Clever algorithms: nature-inspired programming recipes. Jason Brownlee (2011)
2. Calegari, R.: Micro-Intelligence for the IoT: Logic-based Models and Technologies. Ph.D. thesis, Alma Mater Studiorum—Università di Bologna, Bologna, Italy (2018)
3. Calegari, R., Ciatto, G., Mariani, S., Denti, E., Omicini, A.: Micro-intelligence for the iot: Se challenges and practice in lpaas. In: 2018 IEEE International Conference on Cloud Engineering (IC2E 208). pp. 292–297. IEEE Computer Society (17–20 Apr 2018). https://doi.org/10.1109/IC2E.2018.00061
4. Calegari, R., Denti, E., Dovier, A., Omicini, A.: Extending logic programming with labelled variables: Model and semantics. Fundamenta Informaticae **161**, 53—74 (2018). https://doi.org/10.3233/FI-2018-1695, Special Issue CILC 2016
5. Calegari, R., Denti, E., Mariani, S., Omicini, A.: Towards logic programming as a service: Experiments in tuProlog. In: Santoro, C., Messina, F., De Benedetti, M. (eds.) WOA 2016 – 17th Workshop "From Objects to Agents". CEUR Workshop Proceedings, vol. 1664, pp. 91–99. Sun SITE Central Europe, RWTH Aachen University (29–30 Jul 2016), `http://ceur-ws.org/Vol-1664/w14.pdf`, proceedings of the 17th Workshop "From Objects to Agents" co-located with 18th European Agent Systems Summer School (EASSS 2016)
6. Calegari, R., Denti, E., Mariani, S., Omicini, A.: Logic Programming as a Service (LPaaS): Intelligence for the IoT. In: Fortino, G., Zhou, M., Lukszo, Z., Vasilakos, A.V., Basile, F., Palau, C., Liotta, A., Fanti, M.P., Guerrieri, A., Vinci, A. (eds.) 2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC 2017). pp. 72–77. IEEE (May 2017). https://doi.org/10.1109/ICNSC.2017.8000070
7. Calegari, R., Denti, E., Mariani, S., Omicini, A.: Logic programming as a service. Theory and Practice of Logic Programming **18**(3-4), 1–28 (2018). https://doi.org/10.1017/S1471068418000364, Special Issue "Past and Present (and Future) of Parallel and Distributed Computation in (Constraint) Logic Programming"

8. Calegari, R., Denti, E., Mariani, S., Omicini, A.: Logic programming as a service in multi-agent systems for the Internet of Things. International Journal of Grid and Utility Computing (2018)
9. Ciancarini, P.: Coordination models and languages as software integrators. ACM Computing Surveys **28**(2), 300–302 (Jun 1996). https://doi.org/10.1145/234528.234732
10. Ciancarini, P., Omicini, A., Zambonelli, F.: Multiagent system engineering: The coordination viewpoint. In: Jennings, N.R., Lespérance, Y. (eds.) Intelligent Agents VI. Agent Theories, Architectures, and Languages, LNAI, vol. 1757, pp. 250–259. Springer (2000). https://doi.org/10.1007/10719619_19, 6th International Workshop (ATAL'99), Orlando, FL, USA, 15–17 Jul. 1999. Proceedings
11. Crevier, D.: AI: The tumultuous history of the search for artificial intelligence. Basic Books (1993)
12. Cusumano, M.: Cloud computing and SaaS as new computing platforms. Communications of the ACM **53**(4), 27–29 (Apr 2010). https://doi.org/10.1145/1721654.1721667
13. Denti, E., Natali, A., Omicini, A.: Programmable coordination media. In: Garlan, D., Le Métayer, D. (eds.) Coordination Languages and Models, LNCS, vol. 1282, pp. 274–288. Springer-Verlag (1997). https://doi.org/10.1007/3-540-63383-9_86, 2nd International Conference (COORDINATION'97), Berlin, Germany, 1–3 Sep. 1997. Proceedings
14. Denti, E., Natali, A., Omicini, A.: On the expressive power of a language for programming coordination media. In: 1998 ACM Symposium on Applied Computing (SAC'98). pp. 169–177. ACM, Atlanta, GA, USA (27 Feb–1 Mar 1998). https://doi.org/10.1145/330560.330665, special Track on Coordination Models, Languages and Applications
15. Familiar, B.: Microservices, IoT, and Azure: Leveraging DevOps and Microservice Architecture to Deliver SaaS Solutions. Apress, Berkely, CA, USA, 1st edn. (2015)
16. Fortino, G., Guerrieri, A., Russo, W.: Agent-oriented smart objects development. In: 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2012). pp. 907–912. IEEE (May 2012). https://doi.org/10.1109/CSCWD.2012.6221929
17. Gabbay, D.M.: Labelled Deductive Systems, Volume 1, Oxford Logic Guides, vol. 33. Clarendon Press (Sep 1996), `http://global.oup.com/academic/product/labelled-deductive-systems-9780198538332`
18. Gelernter, D.: Generative communication in Linda. ACM Transactions on Programming Languages and Systems **7**(1), 80–112 (Jan 1985). https://doi.org/10.1145/2363.2433
19. Holzbaur, C.: Metastructures vs. attributed variables in the context of extensible unification. In: Bruynooghe, M., Wirsing, M. (eds.) Programming Language Implementation and Logic Programming, Lecture Notes in Computer Science, vol. 631, pp. 260–268. Springer (1992). https://doi.org/10.1007/3-540-55844-6_141
20. Idelberger, F., Governatori, G., Riveret, R., Sartor, G.: Evaluation of logic-based smart contracts for blockchain systems. In: Alferes, J.J., Bertossi, L., Governatori, G., Fodor, P., Roman, D. (eds.) Rule Technologies. Research, Tools, and Applications. Lecture Notes in Computer Science, vol. 9718, pp. 167–183. Springer (2016). https://doi.org/10.1007/978-3-319-42019-6_11
21. Jamont, J.P., Occello, M.: Meeting the challenges of decentralised embedded applications using multi-agent systems. Int. J. Agent-Oriented Softw. Eng. **5**(1), 22–68 (Jan 2016). https://doi.org/10.1504/IJAOSE.2015.078435

22. Kato, T., Chiba, R., Takahashi, H., Kinoshita, T.: Agent-oriented cooperation of iot devices towards advanced logistics. In: 2015 IEEE 39th Annual Computer Software and Applications Conference (COMPSACW 2015). vol. 3, pp. 223–227 (Jul 2015). https://doi.org/10.1109/COMPSAC.2015.237
23. Khan, R., Khan, S.U., Zaheer, R., Khan, S.: Future internet: The internet of things architecture, possible applications and key challenges. In: 2012 10th International Conference on Frontiers of Information Technology. pp. 257–260 (Dec 2012). https://doi.org/10.1109/FIT.2012.53
24. Larrucea, X., Combelles, A., Favaro, J., Taneja, K.: Software engineering for the Internet of Things. IEEE Software **34**(1), 24–28 (Jan 2017). https://doi.org/10.1109/MS.2017.28
25. Loke, S.W.: Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective. The Knowledge Engineering Review **19**(3), 213–233 (Sep 2004). https://doi.org/10.1017/S0269888905000263
26. Manzalini, A., Zambonelli, F.: Towards autonomic and situation-aware communication services: the CASCADAS vision. In: IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06). pp. 383–388 (Jun 2006). https://doi.org/10.1109/DIS.2006.71
27. Oliya, M., Pung, H.K.: Towards incremental reasoning for context aware systems. In: Abraham, A., Lloret Mauri, J., Buford, J.F., Suzuki, J., Thampi, S.M. (eds.) Advances in Computing and Communications: First International Conference, ACC 2011, Kochi, India, July 22-24, 2011. Proceedings, Part I, Communications in Computer and Information Science, vol. 190, pp. 232–241. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22709-7_24
28. Omicini, A.: SODA: Societies and infrastructures in the analysis and design of agent-based systems. In: Ciancarini, P., Wooldridge, M.J. (eds.) Agent-Oriented Software Engineering, Lecture Notes in Computer Science, vol. 1957, pp. 185–193. Springer-Verlag (2001). https://doi.org/10.1007/3-540-44564-1_12, 1st International Workshop (AOSE 2000), Limerick, Ireland, 10 Jun. 2000. Revised Papers
29. Omicini, A.: Formal ReSpecT in the A&A perspective. Electronic Notes in Theoretical Computer Science **175**(2), 97–117 (Jun 2007). https://doi.org/10.1016/j.entcs.2007.03.006, 5th International Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA'06), CONCUR'06, Bonn, Germany, 31 Aug. 2006. Post-proceedings
30. Omicini, A., Denti, E.: From tuple spaces to tuple centres. Science of Computer Programming **41**(3), 277–294 (Nov 2001). https://doi.org/10.1016/S0167-6423(01)00011-9
31. Omicini, A., Denti, E., Natali, A.: Agent coordination and control through logic theories. In: Gori, M., Soda, G. (eds.) Topics in Artificial Intelligence. LNAI, vol. 992, pp. 439–450. Springer-Verlag (1995). https://doi.org/10.1007/3-540-60437-5_43, 4th Congress of the Italian Association for Artificial Intelligence (AI*IA'95), Florence, Italy, 11–13 Oct. 1995, Proceedings
32. Omicini, A., Mariani, S.: Agents & multiagent systems: En route towards complex intelligent systems. Intelligenza Artificiale **7**(2), 153—164 (Nov 2013). https://doi.org/10.3233/IA-130056, Special Issue Celebrating 25 years of the Italian Association for Artificial Intelligence
33. Omicini, A., Ricci, A., Viroli, M.: *Agens Faber*: Toward a theory of artefacts for MAS. Electronic Notes in Theoretical Computer Science **150**(3), 21–36 (29 May 2006). https://doi.org/10.1016/j.entcs.2006.03.003

34. Omicini, A., Ricci, A., Zaghini, N.: Distributed workflow upon linkable coordination artifacts. In: Ciancarini, P., Wiklicky, H. (eds.) Coordination Models and Languages, LNCS, vol. 4038, pp. 228–246. Springer (Jun 2006). https://doi.org/10.1007/11767954_15, 8th International Conference (COORDINATION 2006), Bologna, Italy, 14–16 Jun. 2006. Proceedings
35. Omicini, A., Zambonelli, F.: Coordination for Internet application development. Autonomous Agents and Multi-Agent Systems **2**(3), 251–269 (Sep 1999). https://doi.org/10.1023/A:1010060322135, Special Issue: Coordination Mechanisms for Web Agents
36. Omicini, A., Zambonelli, F.: MAS as complex systems: A view on the role of declarative approaches. In: Leite, J.A., Omicini, A., Sterling, L., Torroni, P. (eds.) Declarative Agent Languages and Technologies, Lecture Notes in Computer Science, vol. 2990, pp. 1–17. Springer Berlin Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-25932-9_1, 1st International Workshop (DALT 2003), Melbourne, Australia, 15 Jul. 2003. Revised Selected and Invited Papers
37. Savaglio, C., Fortino, G., Ganzha, M., Paprzycki, M., Badica, C., Ivanovic, M.: Agent-based computing in the internet of things: A survey. In: Ivanović, M., Bădică, C., Dix, J., Jovanović, Z., Malgeri, M., Savić, M. (eds.) Intelligent Distributed Computing XI, Studies in Computational Intelligence, vol. 737, pp. 307–320. Springer (2018). https://doi.org/10.1007/978-3-319-66379-1_27
38. 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., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. Nature **529**, 484–489 (Jan 2016). https://doi.org/10.1038/nature16961
39. Spanoudakis, N., Moraitis, P.: Engineering ambient intelligence systems using agent technology. IEEE Intelligent Systems **30**(3), 60–67 (May 2015). https://doi.org/10.1109/MIS.2015.3
40. Tan, L., Wang, N.: Future Internet: The Internet of Things. In: 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE). vol. 5, pp. (5)376–(5)380 (Aug 2010). https://doi.org/10.1109/ICACTE.2010.5579543
41. Viroli, M., Omicini, A.: Coordination as a service. Fundamenta Informaticae **73**(4), 507–534 (2006), `http://content.iospress.com/articles/fundamenta-informaticae/fi73-4-04`, Special Issue: Best papers of FOCLASA 2002
42. Xiang, C., Li, X.: General analysis on architecture and key technologies about internet of things. In: 2012 IEEE International Conference on Computer Science and Automation Engineering. pp. 325–328 (June 2012). https://doi.org/10.1109/ICSESS.2012.6269471
43. Zambonelli, F.: Key abstractions for IoT-oriented software engineering. IEEE Software **34**(1), 38–45 (Jan 2017). https://doi.org/10.1109/MS.2017.3
44. Zambonelli, F., Omicini, A.: Challenges and research directions in agent-oriented software engineering. Autonomous Agents and Multi-Agent Systems **9**(3), 253–283 (Nov 2004). https://doi.org/10.1023/B:AGNT.0000038028.66672.1e, Special Issue: Challenges for Agent-Based Computing