

# A Unifying Framework for Agency in Hypermedia Environments

Victor Charpenay<sup>1</sup>[0000–0002–9210–1583], Tobias Käfer<sup>2</sup>[0000–0003–0576–7457], and  
Andreas Harth<sup>3</sup>[0000–0002–0702–510X]

<sup>1</sup> Laboratoire d’informatique, de modélisation et d’optimisation des systèmes  
(LIMOS), Saint-Étienne, France

`victor.charpenay@emse.fr`

<sup>2</sup> Institute AIFB, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

`tobias.kaefer@kit.edu`

<sup>3</sup> Chair of Technical Information Systems, Friedrich-Alexander University (FAU)  
Erlangen-Nuremberg, Nuremberg, Germany

`andreas.harth@fau.de`

**Abstract.** One of the emerging trends in engineering multi-agent systems (MASs) is to use the Web as an environment. On the Web, hypermedia is the guiding principle of agent perception and action. Web standards allow agents to have a single uniform interface to their environment, be it real or simulated. Most recent proposals for hypermedia MASs tend, however, to introduce a coupling between agents and their environment.

This paper introduces a framework based on Semantic Web technologies to formalize interactions between agents and a hypermedia environment. Semantic Web technologies and, more specifically, Web ontologies guarantee interoperability on the Web and maximize uncoupling between components. We show how existing ontologies can be used to make existing MAS prototypes fit our framework.

Our framework formalizes state transfer, the guiding principle of agent-environment interactions in hypermedia, with respect to a reference formalism originally introduced by Genesereth and Nilsson. We also show the equivalence between the two in the paper, under certain conditions.

**Keywords:** Semantic Web · Linked Data · Hypermedia · Multi-Agent System

## 1 Introduction

“On the Internet, nobody knows you’re an *autonomous agent*.” The quote<sup>4</sup> emphasizes the fact that Web servers have no means to distinguish between human agents and autonomous agents (or ‘bots’) that perform request according to a

---

<sup>4</sup> originally about dogs, from a Peter Steiner cartoon published in *The New Yorker* in 1993.

predefined plan. Conversely, agents have no means to assert the origin of a resource, whether it was e.g. created by another agent in relation to physical world events or simulated/forged.

Generally speaking, the architecture of the Web provides a uniform information space that agents can manipulate, through hypermedia [12]. As such, the Web has been seen as a good candidate architecture for building multi-agent systems (MASs) since at least the 2000s [10]. At the time, it was envisioned that autonomous agents could browse the Web as humans do and perform informed actions, such as buying commercial goods online and negotiating prices. To that end, preliminary work on Web ontologies and machine understanding started, to eventually be standardized by the World Wide Web Consortium (W3C) as Semantic Web technologies: RDF, SPARQL and OWL, the Web Ontology Language. In 2010, James Hendler and Tim Berners-Lee underline the importance of Semantic Web technologies to build “social machines” on the Web [11].

Meanwhile, Semantic Web technologies have evolved and deviated from the original vision of autonomous Web agents. However, the Semantic Web is now entering novel domains of applications that revive the need for agent-oriented programming. The Web of Things<sup>5</sup> (WoT) and the Social Web<sup>6</sup>, as standardized by the W3C, are two such domains. The Web of Things allows for new forms of industrial control that tend towards self-organization, a characteristic that is often associated with agent-based modeling [21]. The Social Web allows for uniform human-to-human and human-to-machine interactions, e.g. with chatbots. Most W3C standards for WoT and the Social Web reuse and extend Semantic Web technologies, narrowing the gap between the original vision of autonomous agents on the Web and available technologies.

In the MAS literature, various research prototypes with WoT and social Web applications have been recently proposed, in particular among the workshop series on Engineering MASs (EMAS) [18,4,5,6]. However, most of these proposals do not use Semantic Web technologies (only one proposal does [5], although another includes Linked Data—a subfield of the Semantic Web—as future work) [18].

Not using Semantic Web technologies such as RDF and OWL<sup>7</sup>, has a direct consequence: agents can neither be developed independently from each other, nor can their environment (which includes e.g. WoT ‘things’ or a social networking platform) be developed independently from them. Such coupling between agents and their environment go against the promise of hypermedia of unifying information management. Yet, as we will see in Sec. 2, if one strictly applies hypermedia principles to MAS architectures, certain architectures with direct agent-to-agent interactions may not be realizable on the Web.

In this paper, we introduce a unifying framework, rooted in RDF, for agents situated on the Web. This framework applies the usual Semantic Web abstrac-

<sup>5</sup> <https://www.w3.org/WoT/>

<sup>6</sup> <https://www.w3.org/Social/>

<sup>7</sup> or, more generally, not providing guarantees as to shared message semantics across agents in MAS prototypes.

tions (RDF triples, resources and datasets) to MAS architectures, to maximize decoupling between system components. We make no assumption as to agent architectures but rather characterize interactions between agents and their (hypermedia) environment in abstract terms. To that end, we base our framework on a formalism first introduced by Genesereth and Nilsson [8], which is, to the best of our knowledge, the most commonly accepted formalism of the sort. We show how to reconcile hypermedia principles, a subset of the Representational State Transfer (REST) principles [7], and MAS architectures, such that any classical MAS (as defined by Genesereth and Nilsson) has an equivalent hypermedia MAS.

In the next section (Sec. 2), we analyse recent EMAS prototypes with respect to the REST principles and identify potential limitations to address in our formalism. We then move on to the main contribution of the paper: a MAS formalism based on RDF (Sec. 3) and evaluate the feasibility of implementing MASs according to our framework, by providing examples of RDF and OWL ontologies that the reviewed EMAS prototypes could leverage (Sec. 4). We conclude the paper in Sec. 5.

## 2 Related Work

### 2.1 Cyber-Physical Systems on the Web & Hypermedia

Recent research initiatives demonstrate renewed interest for topics at the intersection of autonomous agents and the Web. A workshop on hypermedia MASs took place at TheWebConf in 2019<sup>8</sup>, followed by a Dagstuhl-Seminar on the same topic in 2021<sup>9</sup>.

Papers emanating from the EMAS series of workshops confirm this trend. The Web appears in three papers, either as a scalable distributed system made of Web services [18,4] or as a uniform interface to cyber-physical systems [5]. In all three papers, WoT is invoked as a new domain of application for autonomous agents. A fourth paper even makes use of WoT principles without naming them: its prototype indeed involves controlling physical devices via a Web API [6].

Two of these EMAS prototypes insist on hypermedia as the main distinctive feature of their approach [18,5]. Ciortea et al. insist on the fact that hypermedia helps agents “discover at runtime other entities in a MAS and the means to interact with those entities”. Runtime discovery is made possible by the interlinking of Web resources (via hyperlinks) such that agents can navigate from one resource to the other. Web resources should further include pointers to potential actions (via Web forms). These hypermedia design principles are part of the REST architectural principles, which have conditioned much of the architecture of the Web itself [7]. The recent WoT standards published by the W3C [16,13] acknowledge the importance of hypermedia on the Web and define interactions between a ‘thing’ (a Web server) and a WoT ‘consumer’ (a Web client) in terms of links and forms.

<sup>8</sup> <https://www.hyperagents.org/>

<sup>9</sup> <https://www.dagstuhl.de/21072>

## 2.2 Autonomous Agents and Representational State Transfer

The benefits of hypermedia as a mechanism unifying agent interactions only hold if the links and forms embedded in Web resources have shared semantics among agents, as underlined in introduction. RDF, the Resource Description Framework, allows agents to discover the semantics of hyperlinks at runtime as well, by making link relation types themselves dereferenceable resources. As a result, every statement about a Web resource is a hyperlink (an RDF triple), which creates an interdependency between the representation of a resource (as a set of RDF triples) and its location on the Web (via a URI).

This interdependency between representation and location on the Web makes REST an important requirement in Semantic Web applications. In particular, the REST constraint that a hypermedia system has to be layered ensures the proper addressing of semantic resources by so called ‘origin servers’. This constraint, known as the ‘layered system’ constraint is one of the six architectural constraints of REST. It implies that “the large-grain data flows of hypermedia interaction can each be processed like a data-flow network, with filter components selectively applied to the data stream in order to transform the content as it passes” [7]. In REST, there is a sharp distinction between origin servers, which provide data at one end of the data stream, and ‘user agents’, which collect data at the other end of the stream.

In a hypermedia MAS, however, agents play both roles: they may in turn be origin servers and user agents. As a result, the hypermedia system would not be layered anymore, wherever data flows give form to cycles among components. Fig. 1 shows the different data flows that have been implemented in the four EMAS prototypes previously mentioned. In the two prototypes featuring Web services [18,4], an agent is a mixed component that includes both a server connector (for perception and agent-to-agent interactions) and a client connector (for action), enabling direct interactions with other agents (Fig. 1a). In the prototype involving a Web API to physical devices [6], agents have a pure client role while servers are purely reactive components translating remote control into physical phenomena. In this application, servers are thus genuine origin servers (Fig. 1b). Finally, the prototype by Ciordea et al. [5] does include origin servers but it also uses WebSub [9] for agents to perceive their environment. WebSub requires their agent platform to manage a Web server to receive notifications from a WebSub hub (Fig. 1c).

Among the three patterns observed here, only one meets the layered system constraint (Fig. 1b). In the two other patterns, there is no obvious distinction between an agent space and an environment space among system components. Indeed, if one considers an entire system component as an agent, then some hypermedia agents are not strictly situated in some environment (due to the absence of origin servers). Conversely, if one considers any RDF data as being part of the environment, system components mix both agent behavior and environmental resources. The word ‘servient’ emerged during standardization work on WoT, as the contraction for ‘server and client’, to characterize such system components [15].

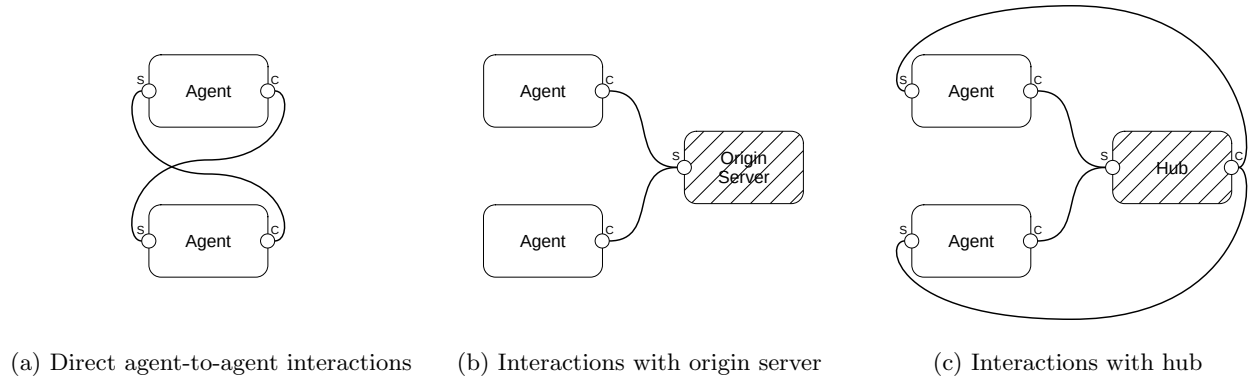


Fig. 1: Graphical representation of agent interactions on the Web; rounded rectangles are components, circles are component connectors (S: server, C: client) [7]; hatched components are purely reactive components

Because of this ambiguity, there is no obvious definitions for perception and action on RDF data in the presence of interaction cycles. In this paper, we intend to bridge that gap through a single abstract formalism that defines perception and actions of agents with respect to a hypermedia environment defined as an RDF dataset. We consider the following requirement: while preserving the usual RDF abstractions used in the Semantic Web literature, our formalism shall capture all MAS architectures, including those involving servients.

### 3 Formalism

The main contribution of the paper is a hypermedia MAS formalism, in which the environment and agent spaces are strictly separated, to maximize uncoupling. The formalism we now present will be evaluated in Sec. 4 by showing that existing EMAS prototypes could be re-implemented with equivalent functionalities but with existing W3C standards for representing agent resources (including Linked Data Platforms and ActivityStream, both relying on Semantic Web technologies).

#### 3.1 Preliminaries

We start from a classical representation of agency (the ability of agents to act on their environment) as functions on environmental states and actions, borrowed from Genesereth and Nilsson (chapter 13) [8] and Wooldridge (chapter 2.5) [22]. To the best of our knowledge, this representation has remained the most widely known reference to study generic interactions between agents and their environment, without making assumptions on agent architectures.

In the following, we define abstract structures for an environment, an agent and a multi-agent system.

**Definition 1 (environment).** *An environment definition  $\mathcal{E}$  is a tuple*

$$\mathcal{E} = \langle E, e_0, A, do \rangle$$

where

- $E$  is a set of states
- $e_0 \in E$  is an initial state
- $A$  is a set of actions
- $do : A \times E \rightarrow 2^E$  is an effectory function

Note that function  $do$  maps to subsets of  $E$  rather than to elements of  $E$ . This choice allows for non-deterministic actions on the environment, as per Wooldridge’s definition of  $\mathcal{E}$ .

**Definition 2 (agent).** *A stateful (or hysteretic) agent definition  $\mathcal{A}$  is a tuple*

$$\mathcal{A} = \langle P, I, i_0, see, internalize, act \rangle$$

where

- $P$  is a set of percepts
- $I$  is a set of internal states
- $i_0 \in I$  is an initial internal state
- $see : E \rightarrow P$  is a sensory function
- $internalize : I \times P \rightarrow I$  is a memory function
- $act : I \rightarrow A$  is a decision-making function

The definition above includes the basic components of an agent’s cognitive loop: the agent perceives its environment, changes its internal state of mind accordingly and then acts. When multiple agent are situated in the same environment, they form a MAS.

**Definition 3 (system).** *A multi-agent system definition  $\mathcal{S}$  is a tuple*

$$\mathcal{S} = \langle \mathcal{E}, \mathcal{A}_1, \mathcal{A}_2, \dots \rangle$$

where

- $\mathcal{E}$  is an environment definition
- every  $\mathcal{A}_i$  is an agent definition

Together, agents change the state of their environment over time. We model a MAS run as a sequence of environmental states obtained through agent actions. Agent actions are themselves conditioned by what agents perceive and by their internal state of mind. See Fig. 2a for an overview of how functions are chained during a MAS run.

We now formally define MAS runs. In the following definition, we choose to model time in the most abstract possible way, as a fully ordered set  $T$  of time positions—a timeline—with lower bound  $t_{\min}$ . We denote  $t^-$  and  $t^+$  the (unique) predecessor and successor of any point in time  $t \in T$ .

**Definition 4 (system run).** Let  $T$  be a timeline. A sequence  $\langle e_t \rangle_{t \in T}$  of environmental states is a system run for  $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}_1, \mathcal{A}_2, \dots \rangle$  if for all  $t \in T$

$$e_t = \begin{cases} e_0 & \text{if } t = t_{min} \\ e \in do(act(i_t), e_{t-}) & \text{for some } i_t, \text{ otherwise} \end{cases}$$

and if for all  $t \in T$  and all  $\mathcal{A}_i = \langle P, I, i_0, see, internalize, act \rangle$

$$i_t = \begin{cases} i_0 & \text{if } t = t_{min} \\ internalize(i_{t-}, see(e_t)) & \text{otherwise} \end{cases}$$

With this definition, we choose to deal with potentially conflicting actions between agents by assuming that no two actions can be executed at the same time. A MAS can however be defined such that an agent keeps choosing the same action as long as it does not see its effects. As a result, concurrent actions are in fact *serialized* in an arbitrary order by the environment.

Note that in our definition,  $i_t$  may be equal to  $i_{t-}$  if the two successive environment states are indistinguishable by the agent. In this modeling, perception is instantaneous. An agent always internalizes a state as soon as an action occurs (as soon as function  $do()$  is applied). We will see how a hypermedia MAS differs in that respect.

### 3.2 Dataset, Operations

As discussed in Sec. 2, the Web can be seen as a single open environment, which agents browse through hypermedia. A common abstraction for the Web is to see it as an RDF dataset, i.e. as a set of labeled graphs, each identified with a URI [3]. If nodes of these graphs are themselves URIs, an edge can then be seen as a hyperlink, which agents can follow to discover more data.

This abstraction (which is a formalization of the Linked Data principles) slightly alters the nature of perception by autonomous agents. If everything on the Web is made of URIs, an agent may universally interpret Web resources. URIs are indeed unambiguous. As a result, agent situatedness in a hypermedia environment does not depend on the individual sensory capabilities of agents but rather on the fact they may only retrieve a finite set of resources at a time. In the following, we formally introduce the RDF abstraction for the Web and redefine the function  $see()$  in the context of a hypermedia environment.

We first briefly introduce the RDF data model<sup>10</sup>:  $U$  and  $L$  are respectively the set of (internationalized) URI resources and literals.  $UL = U \cup L$  is the set of Web resources.  $T = U \times U \times UL$  is the set of RDF triples. The elements of an RDF triple  $\langle s, p, o \rangle \in T$  are respectively called its subject, its predicate and its object. The set of RDF graphs is  $G = 2^T$ . Finally,  $U \times G$  is the set of named graphs. The first element of a named graph is the name of the RDF graph given as the second element.

<sup>10</sup> we leave out ‘blank nodes’ in our definitions, for the sake of clarity.

An RDF dataset is a set of named graphs. The Web is thus (at a given time) an infinite set of named graphs  $d = \{\langle u_1, g_1 \rangle, \langle u_2, g_2 \rangle, \dots\}$ . When an agent performs a **GET** request on resource  $u_1$ , what it gets as a response is  $g_1$ . In the definitions to come next, we will use the shorthand notation  $\sigma_{u_1}(d)$  to denote  $\{\langle u_1, g_1 \rangle\}$ . Moreover, we will denote  $O$  the set of *operations* to perform on Web resources.  $O$  is defined as  $\{\text{GET}, \text{PUT}, \text{POST}, \text{DELETE}\} \times U \times G$ .

As discussed above, perception in a hypermedia environment consists in retrieving a subset of the Web. We define the set of environmental states as  $D$ , the set of all RDF datasets and the set of percepts as the set  $D' \subset D$  of all finite datasets. On this basis, we can now define a standard sensory function on RDF datasets.

In a hypermedia environment, we only consider perception as resulting from link traversal. On the Web, link traversal is initiated by the agent, not by the server, via operations of the form  $\langle \text{GET}, u, \emptyset \rangle$ . We denote  $O_{\text{GET}}$  the set of such operations and define the function transfer :  $O_{\text{GET}} \times D \rightarrow D'$  such that

$$\text{transfer}(\langle \text{GET}, u, \emptyset \rangle, d) = \sigma_u(d)$$

The name ‘transfer’ gets its name from the REST architectural principles, which are oriented towards a ‘state transfer’ from servers to clients.

Similarly to the transfer() function, we can define a standard effectory function based on operations. We define update :  $(O \setminus O_{\text{GET}}) \times D \rightarrow D$  such that

$$\begin{aligned} \text{update}(\langle \text{PUT}, u, g \rangle, d) &= d \setminus \sigma_u(d) \cup \{\langle u, g \rangle\} \\ \text{update}(\langle \text{POST}, u, g \rangle, d) &= d \cup \{\langle u, g \rangle\} \\ \text{update}(\langle \text{DELETE}, u, g \rangle, d) &= d \setminus \sigma_u(d) \end{aligned}$$

This definition, along with that of transfer(), is aligned with the HTTP Graph Store protocol, a W3C standard to manipulate RDF datasets over a REST interface [17]. Operations with **GET** are said to be *safe* because they never lead to any update in the environment.

While we’ve considered hyperlinks in the partitioning of  $D$ , there is another important aspect of hypermedia that must be properly modeled as well: Web forms. The environment should include forms, i.e. request templates to indicate what operations are permitted in the environment. We can define another function to map the state perceived by the agent to *potential* actions that the environment offers or, in other words, *affords*. We define it as afford :  $D \rightarrow 2^O$ .

We now have everything at hand to redefine environments, agents and multi-agent systems in a hypermedia context.

**Definition 5 (hypermedia environment).** *A hypermedia environment  $\mathcal{E}_h$  is a tuple*

$$\mathcal{E}_h = \langle D, d_0, O, \text{transfer}, \text{update}, \text{afford} \rangle$$

Note that the definition above defines a singleton, in the sense that there exists only a single set  $D$ , a single set  $O$ , a single function transfer, *etc.* Only the definition of  $d_0$  could arguably be defined on an application basis.



**Definition 6 (hypermedia agent).** A hypermedia agent  $\mathcal{A}_h$  is a tuple

$$\mathcal{A}_h = \langle I, i_0, \text{internalize}, \text{act} \rangle$$

where

- $I, i_0$  are as per Def. 2
- $\text{internalize} : I \times D' \rightarrow I$  is a memory function (on RDF datasets)
- $\text{act} : I \rightarrow O$  is a decision-making function (with respect to operations)

**Definition 7.** A hypermedia multi-agent system definition  $\mathcal{S}_h$  is a tuple

$$\mathcal{S}_h = \langle \mathcal{E}_h, \mathcal{A}_{h,1}, \mathcal{A}_{h,2}, \dots \rangle$$

Fig. 2b gives a comparison with generic MAS in terms of function chaining. The main difference is in the position of the sensory functions  $\text{transfer}()$  vs.  $\text{see}()$ . By defining a shared function for perception, agents can all be situated in the same open environment. The downside of the approach is that perception becomes an action on its own: the decision-making function  $\text{act}()$  outputs both safe operations (for state transfer) and unsafe update operations. Transfer results from a (GET) request/response exchange between an agent and a server.

**Definition 8 (hypermedia system run).** Let  $T$  be a timeline. A sequence  $\langle d_t \rangle_{t \in T}$  of datasets is a hypermedia system run for  $\mathcal{S}_h = \langle \mathcal{E}_h, \mathcal{A}_{h,1}, \mathcal{A}_{h,2}, \dots \rangle$  if for all  $t \in T$

$$d_t = \begin{cases} d_0 & \text{if } t = t_{\min} \\ \text{update}(\text{act}(i_t), d_{t-}) & \text{if } \text{act}(i_t) \in \text{afford}(d_{t-}), \text{ for some } i_t \end{cases}$$

and if for all  $t \in T$  and all  $\mathcal{A}_{h,i} = \langle I, i_0, \text{internalize}, \text{act} \rangle$

$$i_t = \begin{cases} i_0 & \text{if } t = t_{\min} \\ \text{internalize}(i_{t-}, \text{transfer}(\text{act}(i_{t-}), d_t)) & \text{if } \text{act}(i_{t-}) \in O_{\text{GET}} \\ \text{internalize}(i_{t-}, \emptyset) & \text{otherwise} \end{cases}$$

In this modeling, we assume that  $\text{update}()$  and  $\text{transfer}()$  are instantaneous. Yet, one cannot build all MAS variants as hypermedia MASs. As discussed in Sec 2, the notion of servant is however sufficient to have an equivalence between classically defined MASs and hypermedia MASs.

### 3.3 Servients

In hypermedia systems, the situatedness of an agent is primarily conditioned by the hypermedia controls (links and forms) it finds in the environment. Hypermedia controls do constrain the perception and action range of the agent. Yet, in the various prototypes we have reviewed in Sec. 2, the perception of agents also depends on another factor: the resources it owns as a Web server. As a

servient, an agent has full access to the resources it owns and, in particular, it gets immediately notified whenever these resources are updated (by another agent).

We now incorporate the notion of resource ownership to Def. 6. In the following definition, we use the shorthand notations  $\delta_t$  as the difference  $d_t \setminus d_{t-}$  and  $\sigma_R(d)$  as the union  $\bigcup_{u \in R} \sigma_u(d)$ .

**Definition 9 (hypermedia servient).** *A hypermedia servient  $\mathcal{A}_{hs}$  is a tuple*

$$\mathcal{A}_{hs} = \langle I, i_0, R, \text{internalize}, \text{act} \rangle$$

where

- $I, i_0$ , transfer, internalize and act are as per Def. 6
- $R \subset U$  is a set of resources owned by the agent

We also modify Def. 8 accordingly.

**Definition 10 (hypermedia system run bis).** *A sequence  $\langle d_t \rangle_{t \in T}$  of datasets is a hypermedia system run for  $\mathcal{S}_{hs} = \langle \mathcal{E}_h, \mathcal{A}_{h,1}, \mathcal{A}_{h,2}, \dots, \mathcal{A}_{hs,1}, \mathcal{A}_{hs,2}, \dots \rangle$  if, in addition to constraints of Def. 8, for all  $\mathcal{A}_{hs,i} = \langle I, i_0, R, \text{internalize}, \text{act} \rangle$*

$$i_t = \begin{cases} i_0 & \text{if } t = t_{\min} \\ \text{internalize}(i_{t-}, \text{transfer}(\text{act}(i_{t-}), d_t) \cup \sigma_R(\delta_t)) & \text{if } \text{act}(i_{t-}) \in O_{GET} \\ \text{internalize}(i_{t-}, \sigma_R(\delta_t)) & \text{otherwise} \end{cases}$$

The modification allows us to assert an equivalence between classical MASs and hypermedia MASs, as formally expressed below.

**Theorem 1.** *Let  $\tau : E \rightarrow D$  be a bidirectional transformation that maps every arbitrary environmental state (Def. 1) to some RDF dataset (Def 5).*

*For every multi-agent system  $\mathcal{S}$ , there is an equivalent hypermedia system  $\mathcal{S}_{hs}$ . That is, for every run  $\langle e_t \rangle_{t \in T}$  of  $\mathcal{S}$ , there is an equivalent run  $\langle \tau(e_t) \rangle_{t \in T}$  of  $\mathcal{S}_{hs}$ .*

This equivalence only holds if servients are allowed in the hypermedia MAS.

### 3.4 Artifacts

Our formalism for hypermedia MASs is based on a generic abstraction for the Web: the RDF data model. In practice, agents are likely not to recognize all URIs they find in the environment. Rather, they would be programmed to recognize specific Web ontologies, which specify the structure of resources through a vocabulary and the potential actions available on these resources. To that end, the two EMAS prototypes dealing with hypermedia environments [18,5] make use of the CArtAgO meta-model [19]: resources e.g. with a certain content type, such as the Hypermedia Application Language (HAL), or a certain data structure, such as RDF triples with a specific vocabulary (EVE), are turned into software

objects called ‘artifacts’, that agents use as proxies to manipulate the origin Web resources.

In the context of a hypermedia environment, it is however not clear whether artifacts should be part of the environment itself (i.e. modeled as resources) or added to the formalism as their own kind of entity. In the former case, the notion would be redundant with that of a Web resource, introducing again a coupling between the environment and agent spaces.

Rather, we make the assumption here that artifacts are ‘translators’ between datasets and operations, on the one hand, and more idiomatic representations of states and actions, on the other hand. Artifacts would allow any existing agent architecture to be used against a hypermedia environment. Formally, an artifact can be modeled as a function that maps  $D$  to a higher-level state space (e.g. a set of predefined beliefs) and a function that maps arbitrary actions (e.g. WoT forms or social actions) to  $O$ . To be consistent with how CArtAgO is used in practice, we also make artifacts stateful entities, as follows.

**Definition 11 (artifact).** *An artifact (or proxy) definition is a tuple*

$$\mathcal{P} = \langle E', A', I', transfer', update', internalize' \rangle$$

where

- $E'$  is a set of proxy states
- $A'$  is a set of proxy actions
- $I'$  is a set of proxy internal states
- $transfer' : D' \rightarrow E'$  is a proxy transfer function
- $update' : A' \times I' \rightarrow O$  is a proxy update function
- $internalize' : E' \times I' \rightarrow I'$  is a proxy memory function

An illustration of the above definition is given in Fig. 2c. Our modeling is consistent with the fact that artifacts are not autonomous agents. As proxies, they do not include any `act()` function. Moreover, artifact definitions are not tied to specific agents, they can apply to all agents sharing the same abstraction of states and actions.

We do not model actual communication channels in our formalism. In practice, the HTTP communication channel is often between an artifact and the environment rather than between the agent and its environment. The simplest artifact for hypermedia agents is a Web client that turns local actions to HTTP requests.

## 4 Ontologies for a Hypermedia Environment

In the following, we briefly introduce ontologies relevant for engineering hypermedia MASs (see Table 1 for an overview). This review shows how to re-implement the four EMAS prototypes in a framework including Semantic Web technologies, as per our formalism. To illustrate how artifacts can help integrate existing agent architectures with a hypermedia MAS, we introduce artifact definitions where  $E'$  and  $A'$  are sets of AgentSpeak beliefs and actions [2].

Table 1: Ontologies relevant for hypermedia MASs

Name	Namespace URL	Prefix
Brick schema	<a href="http://brickschema.org/">http://brickschema.org/</a>	brick:
Hypermedia Controls	<a href="https://www.w3.org/2019/wot/hypermedia#">https://www.w3.org/2019/wot/hypermedia#</a>	hctl:
Thing Description (TD)	<a href="https://www.w3.org/2019/wot/td#">https://www.w3.org/2019/wot/td#</a>	td:
Schema.org	<a href="http://schema.org/">http://schema.org/</a>	schema:
Linked Data Platform (LDP)	<a href="https://www.w3.org/ns/ldp#">https://www.w3.org/ns/ldp#</a>	ldp:
ActivityStream	<a href="https://www.w3.org/ns/activitystreams#">https://www.w3.org/ns/activitystreams#</a>	as:

#### 4.1 Reasoning with Web Ontologies

All Web ontologies (should) follow the RDF Schema and OWL specifications. These specifications provide means to declare a certain vocabulary to use in other RDF graphs, as well as axioms associated with that vocabulary. An OWL artifact could process all ontological definitions for the vocabulary found in an RDF graph, materialize implicit triples stated through axioms and turn the original RDF graph into a set of Prolog/AgentSpeak predicates.

For example, we assume the existence of resource `<room>` in an environment  $d$  such that

$$\text{transfer}(\langle \text{GET}, \langle \text{room} \rangle, \emptyset \rangle, d) = \{ \langle \langle \text{room} \rangle, g_1 \rangle \}$$

where

$$g_1 = \{ \langle \langle \text{room} \rangle, \text{rdf:type}, \text{brick:Room} \rangle, \\ \langle \langle \text{room} \rangle, \text{brick:partOf}, \langle \text{floor} \rangle \rangle, \\ \langle \langle \text{floor} \rangle, \text{brick:partOf}, \langle \text{building} \rangle \rangle \}$$

If an agent chooses to look up schema axioms defined in the Brick schema, at location `brick:`<sup>11</sup>, it gets graph  $g_2$  defined as

$$g_2 = \{ \langle \text{brick:Room}, \text{rdf:type}, \text{owl:Class} \rangle, \\ \langle \text{brick:Zone}, \text{rdf:type}, \text{owl:Class} \rangle, \\ \langle \text{brick:Room}, \text{rdfs:subClassOf}, \text{brick:Zone} \rangle, \\ \langle \text{brick:partOf}, \text{rdf:type}, \text{owl:ObjectProperty} \rangle, \\ \langle \text{brick:partOf}, \text{rdf:type}, \text{owl:TransitiveProperty} \rangle \}$$

After internalizing  $g_1$  and  $g_2$ , an OWL artifact should take into account OWL class and property definitions, as well as the sub-class and transitivity axioms.

<sup>11</sup> we represent URIs either as relative URIs or as ‘compact URIs’ (prefix followed by local name).

It could e.g. return the following predicates for `<room>` (assuming the artifact's internal state  $i'$  has already internalized  $g_2$ ):

$$\text{transfer}'(\{\langle \text{room} \rangle\}, g_1, i') = \{\text{room}(\text{'room'}), \\ \text{partOf}(\text{'floor'}, \text{'building'}), \\ \text{zone}(\text{'room'}), \\ \text{partOf}(\text{'room'}, \text{'floor'}), \\ \text{partOf}(\text{'room'}, \text{'building'})\}$$

Brick schema is an ontology for the domain of building automation. A Brick representation of a building is e.g. relevant for autonomous agents controlling vacuum cleaning robots navigating in the building, as in the case of the Neato API [6]. It is also relevant for building automation systems to locate sensors and actuators in the building. The Building on Linked Data (BOLD) benchmark<sup>12</sup> includes various tasks to perform on a simulated building. The BOLD server, exposing the simulation as RDF, closely follows the formalism we introduce in this paper.

## 4.2 Resource Collections

A recurring pattern in hypermedia systems is to use resource *collections*. This pattern is e.g. used by O'Neill et al. in their Multi-Agent Microservices (MAMS) scenario [18]. Linked Data Platforms (LDPs) are a recent W3C standard to implement the resource collection pattern. In LDPs, resource collections are called 'containers', as in the following example:

$$g_3 = \{\langle \text{coll} \rangle, \text{rdf:type}, \text{ldp:BasicContainer}\}, \\ \langle \text{coll} \rangle, \text{ldp:contains}, \langle \text{member1} \rangle, \\ \langle \text{coll} \rangle, \text{ldp:contains}, \langle \text{member2} \rangle\}$$

LDP containers come with implicit affordances, e.g. to add a new item to the collection:

$$\text{afford}(\langle \text{coll} \rangle, g_3) = \{\langle \text{POST}, \langle \text{coll} \rangle, g \rangle \mid g \in G\}$$

An LDP artifact could implement the specification and provide an action to AgentSpeak agents of the form `add('coll', Item, ItemId)` for all instances of `ldp:BasicContainer` it would have internalized.

LDP is not the only standard to model resource collections. ActivityStream (also part of a Social Web standard) can also be used, for the same result. The following graph is semantically equivalent to  $g_3$ :

$$g'_3 = \{\langle \text{coll} \rangle, \text{rdf:type}, \text{as:Collection}\}, \\ \langle \text{coll} \rangle, \text{as:items}, \langle \text{member1} \rangle, \\ \langle \text{coll} \rangle, \text{as:items}, \langle \text{member2} \rangle\}$$

<sup>12</sup> <https://github.com/bold-benchmark/>

### 4.3 Social Activities

LDPs can be used for specific types of container, such as message inboxes. The Linked Data Notification (LDN) specification standardizes how to use inbox containers. LDN and ActivityStream are both part of a series of W3C standards meant for the Social Web<sup>13</sup>, which also includes WebSub. These standards allow for direct agent-to-agent communication without requiring a dedicated communication channel. Instead, messages are placed in and retrieved from the environment.

Another EMAS prototype based on JADE included a basic virtual assistant to manage one's agenda. We give below an example from the ActivityStream standard to represent agendas. The agenda itself is the named graph  $\langle \text{agenda}, g_5 \rangle$  and individual events belonging to the agenda are each a resource, for instance  $\langle \text{event}, g'_5 \rangle$ , where  $g_5$  and  $g'_5$  are defined as

$$g_5 = \{ \langle \text{agenda}, \text{as:items}, \text{event} \rangle, \dots \}$$

and

$$\begin{aligned} g'_5 = \{ & \langle \text{event}, \text{rdf:type}, \text{as:Event} \rangle, \\ & \langle \text{event}, \text{as:name}, \text{"Some agenda event"} \rangle, \\ & \langle \text{event}, \text{as:startTime}, \text{"2021-03-05T00:09:00Z"} \rangle, \\ & \langle \text{event}, \text{as:endTime}, \text{"2021-03-05T00:10:00Z"} \rangle \} \end{aligned}$$

In this example, the agenda, modeled as a collection of events, offers the same affordances as described in Sec. 4.2. Each event offers further affordances. For instance, an autonomous agent can reschedule an event by removing the original one from the collection with operation  $\langle \text{DELETE}, \text{event}, \emptyset \rangle$ , to then add the rescheduled event to the agenda with a POST operation.

### 4.4 Affordances

Our formalism enforces agents to follow 'affordances' provided by the environment via the function `afford()`. We now show how Web forms can be embedded in the environment through two ontologies: the Thing Description (TD) ontology (which includes a module for hypermedia controls) and `schema.org`.

The TD ontology makes affordances explicit by specifying HTTP request templates as RDF triples. For instance, graph  $g_4$  defined as the graph

$$\begin{aligned} g_4 = \{ & \langle \text{lamp}, \text{td:hasPropertyAffordance}, \text{status\_affordance} \rangle, \\ & \langle \text{status\_affordance}, \text{td:forProperty}, \text{status} \rangle, \\ & \langle \text{status\_affordance}, \text{td:hasForm}, \text{status\_form} \rangle \\ & \langle \text{status\_form}, \text{hctl:hasTarget}, \text{target} \rangle \\ & \langle \text{status\_form}, \text{hctl:forOperationType}, \text{td:readProperty} \rangle \} \end{aligned}$$

<sup>13</sup> <https://www.w3.org/TR/social-web-protocols/>

includes one affordance to retrieve the on/off status of a lamp via a GET request:

$$\text{afford}(\{\langle \text{<lamp>, } g_4 \rangle\}) = \{\langle \text{GET, <target>, } \emptyset \rangle\}$$

The TD ontology defines a small set of operations that are possible on ‘things’ (physical objects on WoT). A TD artifact could e.g. provide a high-level action for the `td:readProperty` operation type. This approach has been implemented with JaCaMo [1] for a summer school on Artificial Intelligence for industrial applications<sup>14</sup>. In the JaCaMo implementation, a `ThingArtifact` object would expose the following action for `g4: readProperty('lamp', 'status', Value)`.

As with resource collections, the TD ontology is not the only way to make affordances explicit. The following graph embeds the same affordance as `g4`:

$$\begin{aligned} g'_4 = \{ & \langle \text{<lamp>, schema:potentialAction, <status\_action>} \rangle, \\ & \langle \text{<status\_action>, schema:actionStatus, schema:PotentialActionStatus} \rangle, \\ & \langle \text{<status\_action>, schema:target, <status\_form>} \rangle \\ & \langle \text{<status\_form>, schema:httpMethod, "GET"} \rangle \\ & \langle \text{<status\_form>, schema:urlTemplate, "target"} \rangle \} \end{aligned}$$

This graph uses the schema.org vocabulary for actions. The approach is being used in another research project on agents in manufacturing [20]. Moreover, schema.org actions are used by the Alexa virtual assistant, as a target representation of natural language commands [14].

#### 4.5 Speech Acts

The MAMS scenario described by O'Neill et al. is based on FIPA's Agent Communication Language (ACL). We show here how to emulate ACL speech acts with ActivityStream.

While WebSub does not recommend a particular vocabulary for the exchanged messages, LDNs and ActivityPub (a third Social Web protocol) encourage using ActivityStream activities. Activities have properties such as actor, target, type, and object. By comparison, ACL messages include the analogous properties ‘sender’, ‘receiver’, ‘performative’ and ‘content fields’. Activities could therefore be a substitute for ACL messages on the Social Web. Table 2 gives a mapping from ActivityStream types to ACL communicative acts. Not all ACL speech acts have a correspondance in RDF but the list is enough to implement e.g. an auction, as in the MAMS scenario.

We give an illustration with the first two steps of an auction through LDNs: the auctioneer announces its auction to a bidder with  $\langle \text{POST, <bidder/inbox>, } g_6 \rangle$ ,

<sup>14</sup> <https://gitlab.emse.fr/ai4industry/hackathon/>

Table 2: Mapping from ActivityStream to FIPA ACL

Activity type (ActivityStream)	Communicative act (FIPA ACL)
as:Announce	Inform, Call for Proposal
as:Offer	Propose
as:Question	Request
as:Accept	Accept Proposal
as:Reject	Reject Proposal, Refuse
as:Follow	Subscribe
as:Undo	Cancel

where

$$g_6 = \{ \langle \text{<announce>, rdf:type, as:Announce}, \\ \langle \text{<announce>, as:name, "Some announcement"}, \\ \langle \text{<announce>, as:actor, <auctioneer>}, \\ \langle \text{<announce>, as:target, <bidder>}, \\ \langle \text{<announce>, as:object, <auction>} \}$$

to which the bidder submits the offer with  $\langle \text{POST, <auctioneer/inbox>, } g'_6 \rangle$ , where

$$g'_6 = \{ \langle \text{<bid>, rdf:type, as:Offer}, \\ \langle \text{<bid>, as:inReplyTo, <announce>}, \\ \langle \text{<bid>, as:name, "Some offer"}, \\ \langle \text{<bid>, as:actor, <bidder>}, \\ \langle \text{<bid>, as:target, <auctioneer>}, \\ \langle \text{<bid>, as:object, <offer>} \}$$

The auctioneer can then accept or reject the offer. Auctioneer and bidder discover each other's inbox through hypermedia, as specified in LDN. Multi-agent protocols can be further specified by using the W3C provenance ontology, PROV-O<sup>15</sup>, as suggested by the LDN specification. PROV-O provides a vocabulary to relate activities to entities used or produced by the activity and to agents involved in the activity.

## 5 Conclusion

In this paper, we introduced a formalism for hypermedia MASs based on the abstraction that the Web is equivalent to an RDF dataset. We were able to

<sup>15</sup> <http://www.w3.org/ns/prov#>



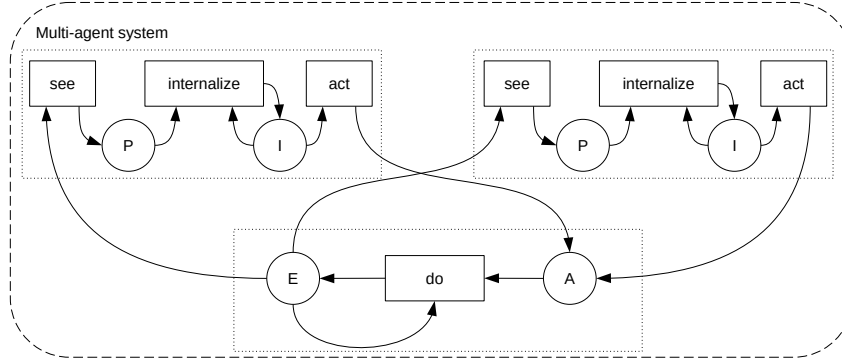
show how four different prototypes recently presented at the EMAS series of workshops could fit our formalism, although none of them uses RDF or other Semantic Web technologies. In addition, we showed how other implementations natively follow the formalism. With this paper, we have aimed at making MAS and Semantic Web technologies converge again, as per the original 2000 vision of autonomous agents on the Web.

Because it is based on Semantic Web technologies, our formalism should allow for a scalable hypermedia environment, hosting many (physical or simulated) resources and responding to many agents in parallel. Experimental proof of the scalability of such an environment is yet to be provided, though. Implementation effort could be targeted towards designing reusable artifacts for W3C standards, such as LDPs, ActivityStream and the TD ontology. More importantly, however, what remains to be proven is the ability of agents of different origins of interacting in the same (unknown) environment. The BOLD benchmark is an attempt to tend towards that goal. Other MAS competitions around hypermedia environments could be developed as well.

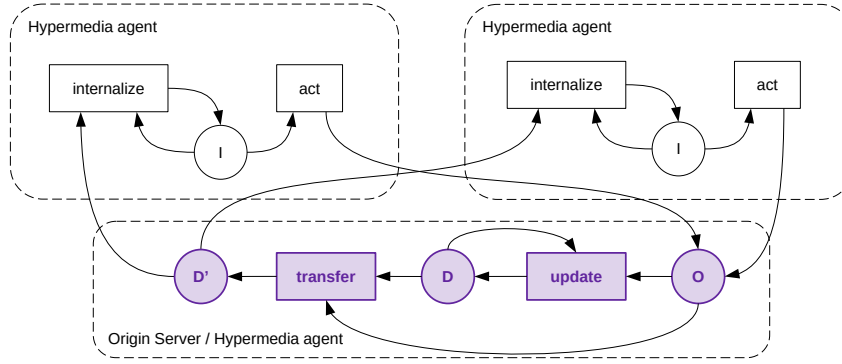
## References

1. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A.: Multi-agent oriented programming: programming multi-agent systems using JaCaMo. Intelligent robotics and autonomous agents series, The MIT Press (2020)
2. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming multi-agent systems in AgentSpeak using Jason, vol. 8. John Wiley & Sons (2007)
3. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named graphs. *Journal of Web Semantics* **3**(4), 247–267 (2005)
4. Casals, A., El Fallah-Seghrouchni, A., Negroni, O., Othmani, A.: Exposing agents as web services in JADE. In: Engineering Multi-Agent Systems. Springer International Publishing (2019), OCLC: 1144197734
5. Ciorrea, A., Boissier, O., Ricci, A.: Engineering world-wide multi-agent systems with hypermedia. In: Weyns, D., Mascardi, V., Ricci, A. (eds.) Engineering Multi-Agent Systems, vol. 11375, pp. 285–301. Springer International Publishing (2019), series Title: Lecture Notes in Computer Science
6. Collenette, J., Logan, B.: Multi-agent control of industrial robot vacuum cleaners. In: Baroglio, C., Hubner, J.F., Winikoff, M. (eds.) Engineering Multi-Agent Systems, vol. 12589, pp. 87–99. Springer International Publishing (2020), series Title: Lecture Notes in Computer Science
7. Fielding, R.: Architectural Styles and the Design of Network-based Software Architectures. phdthesis, University of California, Irvine (2000)
8. Genesereth, M.R., Nilsson, N.J.: Logical Foundations of Artificial Intelligence. Morgan Kaufmann Publishers Inc. (1987)
9. Genestoux, J., Parecki, A.: Websub. W3C Recommendation (2018), <https://www.w3.org/TR/websub/>
10. Hendler, J.: Agents and the semantic web. *IEEE Intelligent systems* **16**(2), 30–37 (2001)
11. Hendler, J., Berners-Lee, T.: From the Semantic Web to social machines: A research challenge for AI on the World Wide Web. *Artificial Intelligence* **174**(2), 156–161 (Feb 2010). <https://doi.org/10.1016/j.artint.2009.11.010>

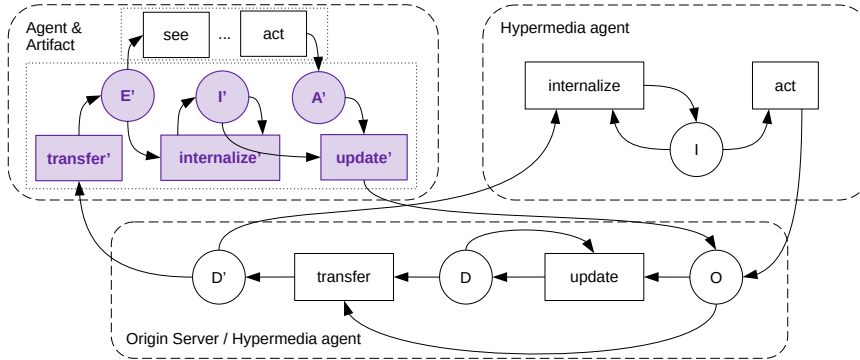
12. Jacobs, I., Walsh, N.: Architecture of the world wide web, volume one. W3C Recommendation (2004), <https://www.w3.org/TR/webarch/>
13. Kaebisch, S., Kamiya, T., McCool, M., Charpenay, V., Kovatsch, M.: Web of Things (WoT) thing description. W3C Recommendation (2020), <https://www.w3.org/TR/wot-thing-description/>
14. Kollar, T., Berry, D., Stuart, L., Owczarzak, K., Chung, T., Mathias, L., Kayser, M., Snow, B., Matsoukas, S.: The alexa meaning representation language. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers). pp. 177–184 (2018)
15. Kovatsch, M., Matsukura, R., Lagally, M., Kawaguchi, T., Toumura, K., Kajimoto, K.: Web of Things (WoT) Architecture. W3C Recommendation (2019), <https://www.w3.org/TR/wot-architecture/>
16. Kovatsch, M., Matsukura, R., Lagally, M., Kawaguchi, T., Toumura, K., Kajimoto, K.: Web of Things (WoT) architecture. W3C Recommendation (2020), <https://www.w3.org/TR/wot-architecture/>
17. Ogbuji, C.: Sparql 1.1 graph store http protocol. Tech. rep. (2013), <http://www.w3.org/TR/sparql11-http-rdf-update/>
18. O’Neill, E., Lillis, D., O’Hare, G.M.P., Collier, R.W.: Delivering multi-agent MicroServices using CArtAgO. In: Baroglio, C., Hubner, J.F., Winikoff, M. (eds.) Engineering Multi-Agent Systems, vol. 12589, pp. 1–20. Springer International Publishing (2020), series Title: Lecture Notes in Computer Science
19. Ricci, A., Viroli, M., Omicini, A.: CArtAgO: A framework for prototyping artifact-based environments in MAS. In: Weyns, D., Parunak, H.V.D., Michel, F. (eds.) Environments for Multi-Agent Systems III, vol. 4389, pp. 67–86. Springer Berlin Heidelberg (2007), series Title: Lecture Notes in Computer Science
20. Schraudner, D., Charpenay, V.: An HTTP/RDF-based agent infrastructure for manufacturing using stigmergy. In: Harth, A., Presutti, V., Troncy, R., Acosta, M., Polleres, A., Fernández, J.D., Xavier Parreira, J., Hartig, O., Hose, K., Cochez, M. (eds.) The Semantic Web: ESWC 2020 Satellite Events, vol. 12124, pp. 197–202. Springer International Publishing (2020), series Title: Lecture Notes in Computer Science
21. Wilensky, U., Rand, W.: An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo. The MIT Press (2015)
22. Wooldridge, M.J.: An introduction to multiagent systems. John Wiley & Sons, 2nd edn. (2009), OCLC: ocn246887666



(a) MAS



(b) Hypermedia MAS



(c) Integration of a non-hypermedia agent into a hypermedia MAS via an artifact

Fig. 2: Graphical representation of abstract (hypermedia) multi-agent systems; rectangles contain function names, circles contain set names, dashed rectangles are REST components (as presented in Fig. 1) and dotted rectangles are reusable software modules