A Unifying Framework for Agency in Hypermedia Environments

Victor Charpenay[1][0000−0002−9210−1583], Tobias Käfer[2][0000−0003−0576−7457], and Andreas Harth[3][0000−0002−0702−510X]

[1] Laboratoire d'informatique, de modélisation et d'optimisation des systèmes (LIMOS), Saint-Étienne, France
victor.charpenay@emse.fr
[2] Institute AIFB, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
tobias.kaefer@kit.edu
[3] 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 allows agents to have a single uniform interface to their environment, be it real or simulated. Most recent proposals for hypermedia MASs tend, however, to use heterogeneous data models for their hypermedia environments, preventing interoperability across MASs.
This paper introduces a framework based on the latest 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. We give examples of ontologies to use in hypermedia MASs in the paper.
Our framework formalizes the guiding principle of agent-environment interactions in hypermedia, state transfer, 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[4] emphasizes the fact that the Internet or, rather, the Web is merely an information system, organized as a set of resources that agents produce and consume. Servers have no means to distinguish between human agents and autonomous agents (or 'bots') that perform request according to a predefined plan. Conversely, agents

[4] originally about dogs, from a Peter Steiner cartoon published in 1993.

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.

The Web is standardizing interactions between agents [12]. As such, it has been seen as a good candidate architecture for building multi-agent systems (MASs) since at least the 2000s [11]. 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.

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[5] (WoT) and the Social Web[6], 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 [19]. 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 literature, various research prototypes closely connected to WoT and social Web applications have been recently proposed, in particular among the workshop series on Engineering MASs (EMAS) [16,4,5,7]. However, these proposals are difficult to compare, for lack of common architectural principles and a common data model, despite W3C standards being available. Most importantly, although the architecture of the Web decouples client and server implementations, it is not possible to reuse the agent implementation of one prototype on the environment of another prototype.

In order to further realize the vision of autonomous agents on WoT or the Social Web, we introduce in this paper a unifying framework, rooted in RDF, for agents situated on the Web. This framework, whose objective is to guarantee interoperability across MAS implementations, is centered on the core feature of the Web: hypermedia. It makes no assumption as to agent architectures but rather characterizes 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 [9], which is, to the best of our knowledge, the most widely used formalism of the sort.

In the next section (Sec. 2), we give a short overview of recent contributions in engineering hypermedia MASs. We then move on to the main contribution of the paper: a formalism that applies MAS principles to the usual Semantic Web abstractions: RDF triples, resources and datasets (Sec. 3). On that basis, we

[5] `https://www.w3.org/WoT/`
[6] `https://www.w3.org/Social/`

then provide examples of RDF and OWL ontologies that hypermedia agents are likely to have to deal with (Sec. 4). In this section, we also discuss the notion of 'artifact', which is commonly used in modeling MAS environments. We conclude the paper in Sec. 5.

2 Related Work

2.1 Cyber-Physical Systems on the Web

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[7], followed by a Dagstuhl-Seminar on the same topic in 2021[8].

Papers emanating from the EMAS series of workshops confirm this trend. The Web appears in various MAS configurations, either as a scalable distributed system made of Web services [16,4] or as a uniform interface to cyber-physical systems [5]. In all recent EMAS papers mentioning the Web, WoT is invoked as a new domain of application for autonomous agents. Other lines of work even make use of WoT principles without naming them, by controlling physical devices via a Web AP [7].

Two of these EMAS prototypes insist on hypermedia as the main distinctive feature of their approach [16,5]. 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 to perform on resources (via Web forms). These hypermedia design principles are part of the Representational State Transfer (REST) architecture, which has conditioned much of the architecture of the Web itself [8].

2.2 Reprensentation State Transfer and Autonomous Agents

These EMAS prototypes project agents onto distinct architectural elements of the Web. To those adopting a service-oriented view on the Web, an agent is a mixed software component that includes both a server (for perception and agent-to-agent communication) and a client (for action). To the other two papers including cyber-physical equipment in their use case, an agent has a pure client role while servers are purely reactive components translating remote control into physical phenomena. The work by Ciortea et al. [5] follows that separation up to one exception: their architecture uses WebSub [10] for agents to perceive the environment. WebSub requires their agent platform to manage a Web server to receive notifications from a WebSub hub.

[7] `https://www.hyperagents.org/`
[8] `https://www.dagstuhl.de/21072`

The major distinction between these works and REST as it was originally conceived is the presence of interdependencies between system components. One of the six architectural constraints of REST is that of a 'layered system'. That constraint 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" [8]. In REST, there is a sharp distinction between 'origin servers', which provide data, and 'user agents' (or simply 'clients'), which collect the data, possibly after several transformations through 'proxies' and 'gateways'.

In a hypermedia MAS, however, agents play both roles: they may in turn be origin servers and user agents. As a result, the information system would not be layered anymore, wherever data flows form cycles among components. The word 'servient' emerged during standardization work on WoT, as the contraction for 'server and client' [14]. The presence of true servients in a hypermedia system is an indication that the system should be considered as a MAS. It is interesting to note that not all WoT systems are to be implemented as a MAS. When connected devices such as vacuum cleaner robots only offer a REST API to the physical world, they are pure 'origin servers' without goals and/or intentions. As such, they are part of the environment and not autonomous agents.

2.3 Engineering Hypermedia Multi-Agent Systems

The example of vacuum cleaning robots, similar to another contribution by Ciortea et al. in the industrial domain [6], highlights a major property of hypermedia MASs: an environment may not be designed by the same stakeholders as the agents. Connected devices, as part of the environment, have fixed APIs provided by manufacturer, such as Neato [7] or Universal Robotics [6]. This property of hypermedia MASs is a direct consequence of the fact that hypermedia allows for large-scale *open* environments without bounds. Agent-oriented programmers should thus assume that the agents they program might communicate with unknown origin servers. One could even consider the Web as a single environment shared by all hypermedia agents, themselves potentially designed by different stakeholders.

To deal with potentially unknown environments, O'Neill et al. [16] and Ciortea et al. [5] make use of the CArtAgO meta-model [17]: 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.

While discussing their work, O'Neill et al. express their will to rely on Semantic Web technologies for resource representations, especially JSON-LD. New important standards published after 2015, in particular for WoT and the Social Web, could indeed be reused to improve interoperability and thus reduce the implementation effort of engineering hypermedia environments. For instance, the HAL representation of resources lists as used by O'Neill et al. is semantically equivalent to an RDF representation that would use the Linked Data Platform

vocabulary[9]. Our contribution in this paper is a unification of hypermedia MAS architectures with W3C standards through a single abstract formalism. Our formalism shall also clarify the distinction between pure Web clients and servients. We present this formalism next.

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 interoperability. The formalism we now present will be evaluated in section 4 by showing that it both follows W3C standards and is consistent with existing MAS prototypes.

3.1 Preliminaries

We start from a classical representation of agency (the ability of agents to act on ther environment) as functions on environmental states and actions, borrowed from Genesereth and Nilsson (chapter 13) [9] and Wooldridge (chapter 2.5) [20]. 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*
- *$do : A \times E \to 2^E$ 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 $\mathcal{E}$.

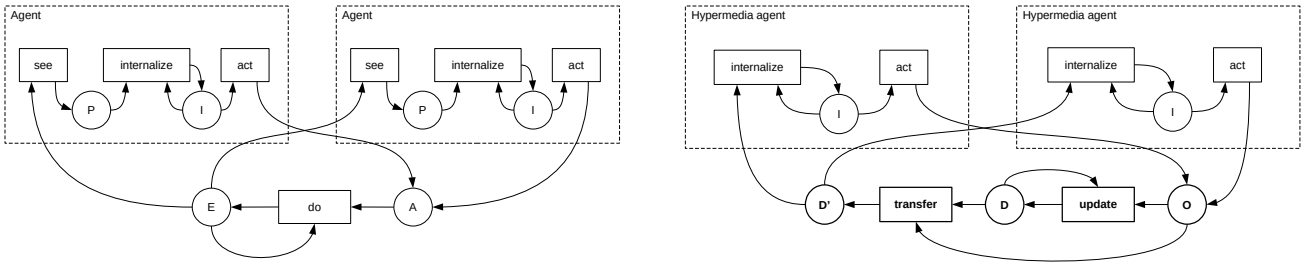
Definition 2 (agent). *A stateful (or hysteretic) agent definition $\mathcal{A}$ is a tuple*

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

where

- *P is a set of percepts*

[9] `https://www.w3.org/TR/ldp/`



(a) Multi-agent system

(b) Hypermedia multi-agent system

Fig. 1: Graphical representation of abstract (hypermedia) multi-agent systems; rectangles contain function names, circles contain set names

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

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

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. 1a 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* $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}_1, \mathcal{A}_2, \ldots \rangle$ *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* $\mathcal{A}_i = \langle P, I, i_0, see, internalize, act \rangle$

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

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[10]: 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.

An RDF dataset is a set of named graphs. The Web is thus (at a given time) an infinite set of named graphs $d = \{\langle u_1, g_1 \rangle, \langle u_2, g_2 \rangle, \ldots\}$. 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 $\{\texttt{GET}, \texttt{PUT}, \texttt{POST}, \texttt{DELETE}\} \times U \times G$.

[10] we leave out 'blank nodes' in our definitions, for the sake of clarity.

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 $\langle \texttt{GET}, u, \emptyset \rangle$. We denote $O_{\texttt{GET}}$ the set of such operations and define the function transfer : $O_{\texttt{GET}} \times D \to D'$ such that

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

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 : $(O \setminus O_{\texttt{GET}}) \times D \to D$ such that

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

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 [15]. 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 : $D \to 2^O$.

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*

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

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*

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

where

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

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

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

Fig. 1b gives a comparison with generic MAS in terms of function chaining. The main difference is in the position of the sensory functions transfer() vs. 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 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* $\mathcal{S}_h = \langle \mathcal{E}_h, \mathcal{A}_{h,1}, \mathcal{A}_{h,2}, \ldots \rangle$ *if for all* $t \in T$

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

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

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

In this modeling, we assume that update() and transfer() are instantaneous. Yet, one cannot build all MAS variants as hypermedia MASs. As discussed in Sec 2, the notion of servient 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 δ_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*

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

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 $\mathcal{S}_{hs} = \langle \mathcal{E}_h, \mathcal{A}_{h,1}, \mathcal{A}_{h,2}, \ldots, \mathcal{A}_{hs,1}, \mathcal{A}_{hs,2}, \ldots \rangle$ if, in addition to constraints of Def. 8, for all $\mathcal{A}_{hs,i} = \langle I, i_0, R, internalize, act \rangle$*

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

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

Theorem 1. *Let $\tau : E \to D$ 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.

4 Ontologies for a Hypermedia Environment

Our formalism for hypermedia multi-agent systems 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. In the following, we briefly introduce ontologies relevant for engineering hypermedia MASs (see Table 1 for an overview).

Throughout the section, we draw parallels with existing MAS engineering concepts. One such concept is that of 'artifact', as defined by the CArtAgO engineering approach [17]. Artifacts are shared objects and tools that agents may use to perceive and act on the environment. 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*.

Table 1: Ontologies relevant for hypermedia MASs

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

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. TD actions or social actions) to O. To be consistent with how CArtAgO is used in practice, we also make artifacts stateful entities. An artifact (or proxy) definition is then a tuple

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

That 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 'user agent' that turns local actions to HTTP requests. In the examples we provide in the following, we include artifact definitions where E' and A' are sets of AgentSpeak beliefs and actions [2].

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

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

where

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

If an agent chooses to look up schema axioms defined in the Brick schema, at location `brick:`[11], it gets graph g_2 defined as

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

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

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

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 [7]. It is also relevant for building automation systems to locate sensors and actuators in the building. The Building on Linked Data (BOLD) benchmark[12] 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 [16]. Linked Data Platforms (LDPs) are a recent W3C standard to

[11] we represent URIs either as relative URIs or as 'compact URIs' (prefix followed by local name).

[12] `https://github.com/bold-benchmark/`

implement the resource collection pattern. In LDPs, resource collections are called 'containers', as in the following example:

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

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

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

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:

$$g_3' = \{\langle \texttt{<coll>}, \texttt{rdf:type}, \texttt{as:Collection}\rangle,$$
$$\langle \texttt{<coll>}, \texttt{as:items}, \texttt{<member1>}\rangle,$$
$$\langle \texttt{<coll>}, \texttt{as:items}, \texttt{<member2>}\rangle\}$$

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[13], 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 environmnent.

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 $\langle \texttt{<agenda>}, g_5\rangle$ and individual events belonging to the agenda are each a resource, for instance $\langle \texttt{<event>}, g_5'\rangle$, where g_5 and g_5' are defined as

$$g_5 = \{\langle \texttt{<agenda>}, \texttt{as:items}, \texttt{<event>}\rangle, \ldots\}$$

and

$$g_5' = \{\langle \texttt{<event>}, \texttt{rdf:type}, \texttt{as:Event}\rangle,$$
$$\langle \texttt{<event>}, \texttt{as:name}, \texttt{"Some agenda event"}\rangle,$$
$$\langle \texttt{<event>}, \texttt{as:startTime}, \texttt{"2021-03-05T00:09:00Z"}\rangle,$$
$$\langle \texttt{<event>}, \texttt{as:endTime}, \texttt{"2021-03-05T00:10:00Z"}\rangle\}$$

[13] `https://www.w3.org/TR/social-web-protocols/`

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 $\langle$`DELETE`, `<event>`, $\emptyset\rangle$, to then add the rescheduled event to the agenda.

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

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

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

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

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[14]. In the JaCaMo implementation, a `ThingArtifact` object would expose the following action for g_4: `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 g_4:

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

This graph uses the schema.org vocabulary for actions. The approach is being used in another research project on agents in manufacturing [18]. Moreover, schema.org actions are used by the Alexa virtual assistant, as a target representation of natural language commands [13].

[14] `https://gitlab.emse.fr/ai4industry/hackathon/`

Table 2: Mapping from ActivityStream to FIPA ACL

Activity type (ActivityStream)	**Comunicative 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

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 $\langle$`POST`, `<bidder/inbox>`, $g_6\rangle$, where

$$
\begin{aligned}
g_6 = \{&\langle\texttt{<announce>}, \texttt{rdf:type}, \texttt{as:Announce}\rangle,\\
&\langle\texttt{<announce>}, \texttt{as:name}, \texttt{"Some announcement"}\rangle,\\
&\langle\texttt{<announce>}, \texttt{as:actor}, \texttt{<auctioneer>}\rangle,\\
&\langle\texttt{<announce>}, \texttt{as:target}, \texttt{<bidder>}\rangle,\\
&\langle\texttt{<announce>}, \texttt{as:object}, \texttt{<auction>}\rangle\}
\end{aligned}
$$

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

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

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[15], 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 multi-agent systems based on the abstraction that the Web is equivalent to an RDF dataset. We were able to 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)

[15] `http://www.w3.org/ns/prov#`

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. Ciortea, 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. Ciortea, A., Mayer, S., Michahelles, F.: Repurposing manufacturing lines on the fly with multi-agent systems for the web of things. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018. pp. 813–822 (2018)
7. 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
8. Fielding, R.: Architectural Styles and the Design of Network-based Software Architectures. phdthesis, University of California, Irvine (2000)
9. Genesereth, M.R., Nilsson, N.J.: Logical Foundations of Artificial Intelligence. Morgan Kaufmann Publishers Inc. (1987)
10. Genestoux, J., Parecki, A.: Websub. W3C Recommendation (2018), `https://www.w3.org/TR/websub/`
11. Hendler, J.: Agents and the semantic web. IEEE Intelligent systems **16**(2), 30–37 (2001)
12. Jacobs, I., Walsh, N.: Architecture of the world wide web, volume one. W3C Recommendation (2004), `https://www.w3.org/TR/webarch/`
13. 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)
14. 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/`
15. Ogbuji, C.: Sparql 1.1 graph store http protocol. Tech. rep. (2013), `http://www.w3.org/TR/sparql11-http-rdf-update/`
16. 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
17. 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
18. 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
19. Wilensky, U., Rand, W.: An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo. The MIT Press (2015)
20. Wooldridge, M.J.: An introduction to multiagent systems. John Wiley & Sons, 2nd ed edn. (2009), OCLC: ocn246887666