

# Towards an Ontology for Uniform Representations of Agent State for Heterogeneous Inter-Agent Explanations

Katharine Beaumont<sup>1,\*</sup>, Elena Yan<sup>2,†</sup> and Rem Collier<sup>1</sup>

<sup>1</sup>UCD School of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland

<sup>2</sup>Mines Saint-Etienne, Univ Clermont Auvergne, INP Clermont Auvergne, CNRS, UMR 6158 LIMOS, F-42023 Saint-Etienne France

## Abstract

Inter-agent explanations are an emerging approach to agent communication that enable agents to share their cognitive processes in order to reach mutual understanding. A key challenge is that agents are often heterogeneous, built on different paradigms and architectures, which makes their internal representations difficult to exchange directly. The Semantic Web offers key technologies, in the form of ontologies, that can facilitate interoperability and shared understanding between hypermedia agents operating on the Web. We present work towards providing *Agent Abstraction ontologies* that would allow hypermedia agents to abstract and exchange information about their cognitive processes as part of inter-agent explanations.

## Keywords

Explainable Agency, Belief Desire Intention, Hypermedia Agents, Ontology

## 1. Introduction

A key challenge for multi-agent systems is enabling heterogeneous agents to understand and explain each other's reasoning and behaviour. While the emerging field of *explainable agency* has focused on agents explaining their own decisions to humans [1], less attention has been given to *inter-agent explanations* [2]. For example, an autonomous vehicle detects a pedestrian stepping out onto a crosswalk and takes the action to brake. It may communicate this decision to autonomous vehicles behind it which have not detected this event, and explain that the decision to take the action to brake over continuing is due to the pedestrian on the road ahead. However, in real scenarios, agents are highly heterogeneous. They can be built on different paradigms (e.g., Belief-Desire-Intention (BDI) [3], reactive, LLM-based), operate at different levels of granularity, and use diverse internal representations. Without a common way to abstract and share their internal state, explanations remain close to the single architecture.

We argue that inter-agent explainability demands a uniform abstraction of agent state, enabling explanations to be exchanged across heterogeneous agents. [2] proposes the use of symbolic techniques for agents to represent and manipulate their cognitive processes and results, and share these representations with other agents. The Multi-Agent-Oriented Programming (MAOP) paradigm provides abstractions for structuring Multi-Agent Systems (MAS) along different dimensions [4], similarly in heterogeneous agents, a mapping from their agent state a uniform or shared abstraction is necessary in order to exchange inter-agent explanations.

For example, the granular brake action of the autonomous vehicle may need to be represented at a higher level of abstraction in order to be meaningful for a different type of agent which does not share the same agent architecture. Similarly to the *cognitive hourglass* described in [5], we propose a hierarchy of abstractions which allow heterogeneous agents (and humans) to understand agents and multi-agent systems regardless of the implementation and application context.

---

*The Second International Workshop on Hypermedia Multi-Agent Systems (HyperAgents 2025), in conjunction with the 28th European Conference on Artificial Intelligence (ECAI 2025); October 26, 2025, Bologna, Italy*

\*Corresponding author.

†These authors contributed equally.

✉ katharine.beaumont@ucdconnect.ie (K. Beaumont); elena.yan@emse.fr (E. Yan); rem.collier@ucd.ie (R. Collier)

ORCID 0009-0001-9250-0090 (K. Beaumont); 0009-0000-6660-9378 (E. Yan); 0000-0003-0319-0797 (R. Collier)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Hypermedia agents can offer particularly valuable insight into how mutual understanding can be shared: a core feature is the use Semantic Web technologies and standards to process knowledge embedded in hypermedia [6]. Specifically, Semantic Web technology in the form of ontologies provides semantically rich distributed data models and vocabularies which give shared meaning for a domain. They can be reasoned over, enhanced, and extended to facilitate knowledge discovery as well as information exchange [7]. The BDI agent paradigm has been used as the underlying technology in various hypermedia agent systems; for example *Jason* [8] agents, which are based on the AgentSpeak(L) formalism [9] are frequently used for example as part of the JaCaMo framework [4] (e.g. [10, 11, 12, 13]). In addition implementations of the Multi-Agent Microservices (MAMS) framework [14, 15] have used ASTRA [16] agents (which are also based on the AgentSpeak(L) formalism), for example [17, 18, 19, 20].

In advancing the path to a full technical solution for inter-agent explanations, we present work towards an ontology for uniform representations of agent state, which can be shared between heterogeneous agents operating in hypermedia environments, *Agent Abstraction ontologies*. The goal is to provide mappings of the inner mechanisms of agent architectures to higher levels of abstraction and back again. We use BDI agents as motivating examples. Section 2 presents related work. Section 3 presents initial analysis towards the creation of *Agent Abstraction ontologies*. Section provides a detailed example of how such ontologies could facilitate inter-agent explanations between hypermedia agents. Section 5 discusses the limitations of the approach and future work. Finally Section 6 concludes the paper.

## 2. Background

### 2.1. Explainable Agency

Explainable agency refers to the capacity for intelligent agents and robots to explain their decisions to a human audience, and is an important challenge for academia and industry as such agents become more pervasive [1]. There is a wealth of research into explainable agency; [21] provide a review of explainable agency for robots and intelligent agents. [22] review prototypes in research which demonstrate explainability in BDI agents. [23] propose an abstract framework for explainable multi-agent systems, distinguishing between different explanation types and between interpretation and explanations. [24] propose an explainability design pattern for agents, TriQPAN. On inter-agent explainability, [2] presents the case for explainability, including between agents, as a central notion in engineering intelligent system. [25] provide a prototype of inter-agent explainability, modelling inter-agent explanations as a protocol which is geared towards agents providing recommendations using inter-agent explanations.

In [22] the explanation lifecycle for a BDI explainer agent is discussed, beginning with the generation of explanation events during the agent’s deliberation cycle. These events trigger a *unit generation algorithm* which produces structured explanation units which are added to an explanation store; these are then used to generate the explanation (using an *explanation generation algorithm*) which is communicated with an explaine. In addition, [22] present five core requirements for engineering inter-agent explainability in BDI multi-agent systems to address requesting explanations, generating explanatory content, storing it, retrieving it, and using it to generate explanations, and then communicating the explanation. The proposed *Implementation Strategies (IS)* are: *IS1* An inter-agent explanation protocol to exchange requests and guide communication; *IS2* Implicit and explicit addition mechanisms to an explanation store; *IS3* Runtime state identifiers and state inspection mechanisms; *IS4* Configurable retrieval and explanation generation mechanisms; *IS5* Uniform representations for machine-understandable explanations.

As mentioned in the introduction, ASTRA and *Jason* agents have been used as hypermedia agents in the MAMS and JaCaMo frameworks. As such we focus on state-of-the-art research into explainability in ASTRA and *Jason* agents. In [22] a simple prototype in the ASTRA agent programming language is presented, which explicitly and implicitly adds content to the explanation store (*IS2*), provides runtime state identifiers and inspection mechanisms (*IS3*), and configurable retrieval and explanation generation mechanisms (*IS4*). A protocol (*IS1*) and uniform representations for machine-understandable explanations (*IS5*) is not provided; for *IS5* the authors suggest that this is a route to inter-agent

explanations in heterogeneous systems, and that a practical way to implement it is the definition of ontologies that can encode the semantics of the building blocks of an explanation.

[26] present a framework for multi-level explainability which distinguishes between the *implementation* level (the technical, engineering level), the *design level* (the agent's knowledge about the world and themselves, desires, intentions and such) and the *domain level*, which is the highest level of abstraction which allows experts in the domain to understand the behaviour of the system, without knowledge of the underlying processes. They present an implementation in *Jason*, which maps explanatory content stored in an explanation store from the implementation to the design level, based on the concept of the *cognitive neck* detailed in [5]; a conceptual framework which proposes the abstraction of the inner workings of different agents to a level that humans can understand, without knowledge of the underlying implementation or application.

## 2.2. Ontologies for Explainable Agency

[27] provide a set of guidelines for ontology development: they stress the importance of using existing ontologies where possible, and generating competency questions to determine questions any knowledge base based on the ontology should be able to answer. Ontologies consist of classes and properties, plus instances, and in this sense can act like a Web-based knowledge base. On the use of ontologies in communicating explanations, [28] provide a generic explanation ontology which provides a simple explanation interaction protocol, explanation profile and explanation strategy. [29] provide an ontology to formally define an explanation by analysing understandings of explanations from different disciplines. [30] build on this to present a general *Explanation Ontology*<sup>1</sup> and associated resources to provide human-oriented system explanations of varying types such as contrastive, contextual, case-based, data, and which facilitates explanations in knowledge-enabled systems, however the ontology does not provide a full model of the different explanation types identified. The authors also detail an *Explanations Pattern Ontology*<sup>2</sup>. These introduce classes and properties to construct explanations that support associations between explanations, and recommendations, facts, or contextual knowledge on which they are based. They also model that explanations answer questions, and may be the results of AI tasks generating recommendations.

[31] propose the *Explanation Interchange Format* ontology, which aims to describe the communication act of explanations and enable reasoning based on explanatory structures. [32] provide a description of the IEEE 7007-2021 Ontological Standard for ethically driven Robotic & Autonomous systems (ERASs), which includes a top-level ontology (TLO) and various sub level ontologies. The TLO models concepts including agents and elements of agent state such as action events, agent communication, plans and intentions, and also processes and physical objects and situations. The sub level Transparency and Accountability ontology (TA) models concepts to provide agent explanations to address transparency concerns. On the use of ontologies to represent agent state, [33] present an ontological analysis of belief, desire as foundational work towards an ontology for BDI agents. The *Hypermedia MAS Core Ontology*<sup>3</sup> describes concepts which describe MAS, such as agent, signifiers, and artifacts, hypermedia MAS platforms, workspaces, and organisations.

## 3. Uniform Representations for Agent State

The vision of this research is a set of ontologies which allows heterogeneous agents to generate explanations and reach mutual understanding by sharing their internal state. In order to share information about internal processes, the ontologies need to describe and define elements of agent state (meaning symbolic representations of an agent's cognitive processes). We approach this challenge by analysing

---

<sup>1</sup><https://raw.githubusercontent.com/tetherless-world/explanation-ontology/master/Ontologies/v2/explanation-ontology.owl>

<sup>2</sup><https://raw.githubusercontent.com/tetherless-world/explanation-ontology/master/Ontologies/explanations-pattern-ontology.owl>

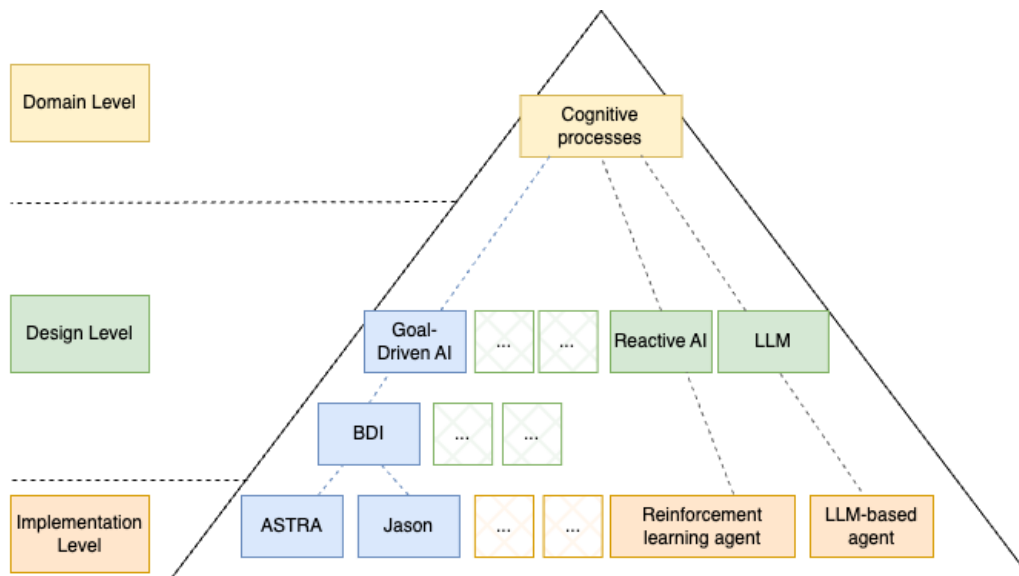
<sup>3</sup><https://ci.mines-stetienne.fr/hmas/core>

the abstractions that can be suitable for different agent technologies, to propose a design for *Agent Abstraction* ontologies which would allow these concepts to be shared between hypermedia agents.

### 3.1. Levels of Abstractions for Representing Agent State

Agents can be built on diverse technologies and paradigms, resulting in different ways of representing their cognitive processes. For instance, we can distinguish between goal-driven agents, reactive agents, and LLM-based agents [34]. Goal-driven agents include BDI agents, which are further instantiated in systems such as ASTRA, Jason, or other agent technologies.

To represent the agent state across these different technologies, we define three broad levels, inspired from [26], to represent the agent abstractions: (i) the *implementation level* concerns agent concepts specific to the agent technologies (e.g., *Jason*, or ASTRA agents); (ii) the *design level* concerns concepts used in the agent paradigm, for example, in BDI agent, this includes agent's beliefs, intentions, plans, goals and desires; (iii) the *domain level*, which is the highest level of abstractions, concerns concepts common to all agents, e.g., cognitive concepts proposed in [5] such as constraints, wishes, hows, whys, and whats.



**Figure 1:** Pyramid of abstractions

The need for interoperability across different types of agent, which may conform to different level of abstractions, could be approached using these concepts. To frame the consideration of inter-agent explanations using the levels of abstractions, an ontology could capture these different levels and be used by agents to explain to other agents (or humans) at the appropriate level of mutual understanding.

Agents can communicate using abstractions from a common level. For example two ASTRA agents could communicate at the implementation level to share implementation specific details such as custom events, represented through the ontology, as part of an inter-agent explanation exchange. An ASTRA agent and a *Jason* agent (both BDI agents based on AgentSpeak(L)) could share common elements at the design level via a mapping which refers to subclasses which represent BDI concepts such as beliefs and desires. Two very unrelated agents, such as a BDI agent and an LLM agent, may have to use higher level or even top level classes in the ontology, representing higher level abstractions, using the domain level to reach mutual understanding in an inter-agent explanation exchange.

### 3.2. Towards Agent Abstraction Ontologies

Existing ontologies and standards can be used to facilitate sharing explanations over the Web, which could be reused or extended for inter-agent explanations, for example the *Explanation Patterns Ontology*

and *Explanation Ontology* [30]. In addition, the Hypermedia MAS Core Ontology<sup>4</sup> provides core concepts such as *Agent*, *Artifact*, *HMAS platform*, etc. that can be used to communicate information about the global multi-agent system. We propose that a core *Agent Abstraction ontology* could be used in conjunction with these to allow agents to communicate explanations based on the agent’s internal state.

For example, the following is an extract from an example in [30] of the Turtle representation of the process a system would undergo to generate a contrastive explanation requested in natural language, *Why drug B over drug A?*<sup>5</sup>:

```

1 :ContrastiveQuestion
2   a sio:'question';
3   rdfs:label 'Why Drug B over Drug A?' .
4
5 :ContrastiveExpInstance
6   a eo:ContrastiveExplanation;
7   ep:isBasedOn :SystemRecExampleA, :SystemRecExampleB;
8   rdfs:label 'Guidelines recommend Drug B';
9   :addresses :ContrastiveQuestion .

```

We expect that the *Agent Abstraction ontologies* will be used to allow agents to expose their cognitive processes and elements of their internal state. For the example above, the inter-agent request for an explanation would replace line 3, and instead of using a natural language question could ask a question by referring to observed actions described in a shared domain ontology. Similarly, the generated explanation could detail the system recommendations (line 7) in machine-understandable terms at a level of abstraction that both agents can understand.

For this, a core set of concepts is required. We start by considering the top level of the pyramid and the *cognitive processes* detailed in [5], namely *wishes*, *hows*, *constraints*, and *whats*. However we omit *whys*: we envisage that these take the form of a protocol for inter-agent explanations as per [22]. We term the class to which these *cognitive processes* extend as *Domain*.

To represent the different levels of abstraction within the ontology we also include the top level classes *Design* and *Implementation*. We envisage that the three top level classes would sit under an umbrella concept *Abstraction*. *Design* and *Implementation* would allow subclasses of the *Domain* to be categorised, for example to indicate that a subclass of *how*, such as a *plan*, is represented in the BDI paradigm (a subclass of *Design*), and is implemented in *ASTRA* (a subclass of *Implementation*).

Domain Level Class	Design Level Class	Implementation Level Class
Constraint	Rule	
What	Action	External Action Internal Action
	Event	
	Belief	Belief From Source Internal Belief
	Intention	
How	Plan	
Wish	Desire	
	Goal	

**Table 1**

Mapping of the core classes of the Abstract Agent ontology in the Domain, Design, and Implementation Levels to represent BDI agents

<sup>4</sup><https://ci.mines-stetienne.fr/hmas/core>

<sup>5</sup>In the latest version of the ontology (the link is provided in Section 2), only contrastive questions are modelled which is why in Section 4 this format is not used.



For example, to show how the core *Agent Abstraction ontology* might represent elements of agent state for BDI agents we present Table 1. Subclasses of the core cognitive processes (*Constraint*, *What*, *How*, *Wish*) represent core concepts in BDI agents: rules, actions, events, beliefs, plans, desires, and goals. Relationships between classes would indicate the different levels that the concepts relate to: for example *Action* is modelled in *AgentSpeak(L)* and both ASTRA and *Jason*. For implementation specific concepts such as *External Action*, subclasses of cognitive processes may be linked using relationships which indicate that the concept is represented in the specific implementation *Jason*, and the lack of a relationship with a higher abstraction (such as the *BeliefDesireIntention* design and the formalism *AgentSpeak(L)*) indicates that it has an implementation level-only scope. For different agent architectures, we expect that it may be necessary to create additional ontologies that extend a core *Agent Abstraction Ontology* in order to fully describe their implementations.

Designing the properties of different classes, and the relationships between classes and concepts will be a key challenge. In addition to properties to describe the relationships between core concepts, for properties to describe the design or runtime state of an agent we identify four primary dimensions:

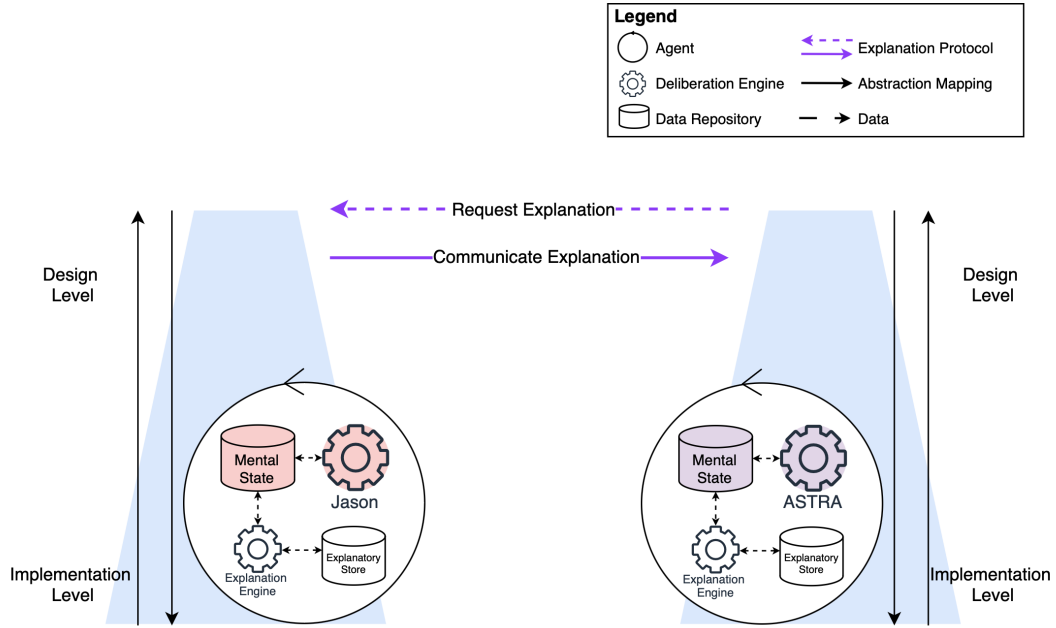
- Logical dimension: for example conjunctions, disjunctions, and more complex operations over cognitive processes, in order to be able to express more complex facets of agent reasoning such as inferences and context conditions.
- Hierarchical dimension: representing the programmatic or procedural relationships between different cognitive processes, such as hierarchical plans and to define the bodies of plans, such as sequences of events and actions.
- Causal dimension: representing the consequences of the execution of agent state, for example that an event triggers a plan, or a belief update causes an action.
- Temporal dimension: representing the temporal dimension of the execution of the agent program.

A challenge for representing the logical dimension of agent state will be the need for high degrees of expressive power to represent sophisticated cognitive processes and reasoning techniques [35]. Additional challenges include how agents handle inconsistencies in internal models, and how to ensure that when mapping between levels of abstractions, information does not get lost in translation between different agent systems, that meaning isn't distorted, and how to evaluate the proposed ontologies.

## 4. Exchanging Explanations in Heterogeneous Agents

In order to illustrate how some *Agent Abstraction Ontologies* could allow heterogeneous agents to exchange explanations which contain mutually understandable representations of their cognitive processes, we refer to the implementation strategies (IS1-5) from [22] which are detailed in Section 2.1. We use an illustrative example of a scenario in which multiple autonomous vehicles navigate and exchange explanations within an urban environment. These autonomous cars are controlled by autonomous agents that may be developed using different architectures, hosted on the Web. The mapping of levels of abstractions to reach the high cognitive level, used as a uniform representation in a shared ontology, facilitates explanations and mutual understanding across heterogeneous agents.

Consider a scenario in there are two autonomous vehicles, *Car A* implemented in a *Jason* agent, and *Car B* implemented in an ASTRA agent. Figure 2 illustrates the exchange of explanations between two such agents. *Car A* is accessible at [www.example/simulation#carA](http://www.example/simulation#carA) and *Car B* at [www.example/simulation#carB](http://www.example/simulation#carB). As *Jason* and ASTRA agents have different concepts, but are both BDI agents, they may find mutual understanding at the design level. However, at the start of the scenario, the agents may not be knowledgeable about the implementation of other agents. In the scenario, *Car A* has detected a pedestrian crossing the road and takes the action to brake. Another autonomous car, *Car B*, that has not detected the pedestrian, does not understand the brake action of *Car A*. *Car B* requests that *Car A* explain the action. In the following, we describe how the explanation can be exchanged by extending the strategies introduced in [22].



**Figure 2:** The process of exchanging explanations in heterogeneous agents involves mapping their abstractions across the design and implementation levels (blue pyramid).

**Generating Explanatory Content at the Implementation Level** In order to enable sharing explanations, a core part is the generation of explanatory content. The generation of the explanatory content is performed at the implementation level to align with the events that are generated during the agent deliberation cycle. Two type of mechanisms for generating explanatory content can be used (as suggested in IS2): *implicit* automatic mechanisms during the agent execution and *explicit* manually configurable mechanisms at the agent language level.

Implicit mechanisms are embedded in the agent’s reasoning cycle, and the functions for generating explanatory content are implemented in the agent’s architecture. Explicit mechanisms allow the agent developer to configure the relevant events to be recorded in the agent code. As per IS3, we need to identify language-level identifiers for the core elements of an *agent runtime state* (e.g., beliefs, plans, actions, etc.) and their *causal relationships* (e.g., an action is explained by a goal). The detailed structure of these elements is platform-dependent. For instance, the explanatory content in *Jason*, as identified in [26], include events related to: (i) perception, (ii) speech act message, (iii) plan, (iv) belief, (v) goal, (vi) intention, (vii) internal action, and (viii) external action. In *ASTRA*, additional events related to plan acquisition [36] and module events can form part of explanatory content, as well as custom content added at the language level (for example Java objects or data values) [22].

**Storing Explanatory Content at the Implementation Level** Having generated the required events of the agent state, through either implicit and explicit mechanisms, these are added and stored in the explanatory content store. An example of the explanatory content store at the *Jason* Implementation Level for the *Car A* is illustrated in Table 2.

**Generating the Explanation Request at the Design Level** *Car B*, since it has not perceived the pedestrian, does not understand the action brake of *Car A*, but we assume that *Car B* interprets the action of *Car A*, using a shared understanding of the available actions they can take (for example, we may imagine an ontology for the observable behaviour of vehicles, car). In order to respond appropriately, *Car B* requests an explanation of the action brake from *Car A*. In order to determine the correct level of abstraction at which to communicate, a strategy may be to send a preliminary message to confirm the **implementation** class of the agent. In a real-time driving scenario however this may be impractical (or agents may have strategies to introduce themselves to surrounding agents). Instead, *Car B* could

Id	Type	State	Detail
$e_1^{impl}$	Percept	Added	pedestrian[source(crosswalk)]
$e_2^{impl}$	Belief From Src	Added	pedestrian[source(crosswalk)]
$e_3^{impl}$	Goal	Created	slow_down
$e_4^{impl}$	Plan	Selected	slow_down
$e_5^{impl}$	Intention	Created	int-1 slow_down
$e_6^{impl}$	External Action	Started	brake

**Table 2**

Events of the agent *Car A* state generated and stored at *Jason* Implementation Level.

include its own design and implementation classes in the communication, but keep the explanation request as high-level as possible (at the domain level). *Car A* can then select the appropriate level to respond at, or *Car B* could ask for more details if the resulting explanation is not satisfactory.

The content of the messages should contain ontologies to structure the message content. For example, the body of a request for an explanation could be formatted in JSON-LD<sup>6</sup>, with a `@context` section which defines the ontologies which will be used in the message; see below. We suggest that a strategy for the explanation request could be to include the explaine agent type and identifier, **design** and **implementation** details in addition to the body of the request, which is formatted at the **domain** level.

As a basic illustration, in order to frame the question, the Explanation Ontology (prefix `eo`) [30] could be used, for example using the relationship `eo:asks` to imply that it is a question expecting an explanation as an answer<sup>7</sup>. We may imagine that a core *Agent Abstraction Ontology* has the prefix `aa`, and the subclass of *Domain*, *What*, indicates that the agent is addressing cognitive processes addressing facts or beliefs:

```
CarB eo:asks [aa:what car:brake]
```

The request for the explanation may also use concepts from the Norms and Ethical Principles Ontology (NEP), but here the specific ontology to frame the explanation question is not intended to be prescriptive.

An existing ontology which is based on the FIPA ACL Message Structure Specification<sup>8</sup> ([20]) could be used to structure the content of the message (this is used below with the prefix `fipa`). The *Hypermedia MAS Core Ontology* could be used to provide additional information about the agent (prefix `hmas`). We may imagine that an ASTRA specific extension to the *Agent Abstraction Ontology* has the prefix `astra`, and a *Jason* specific extension has the prefix `jason`. We may imagine an ontology for the observable behaviour of vehicles with the prefix `car`. The request for the explanation from *Car B* may then take the form:

```

1  "@id": "www.example/simulation#carB",
2  "@type": "hmas:Agent",
3  "@context": {...},
4  "hmas:isMemberOf": "...",
5  "aa:design": "aa:AgentSpeakL",
6  "aa:implementation": "astra:mams",
7  "fipa:request": {
8    "fipa:sender": "www.example/simulation#carB",
9    "fipa:receiver": "www.example/simulation#carA",
10   ...
11   "fipa:content": {
12     "eo:asks": {"aa:what": "car:brake"}
13   }
14 }
```

<sup>6</sup><https://json-ld.org/>

<sup>7</sup><https://raw.githubusercontent.com/tetherless-world/explanation-ontology/master/Ontologies/v2/explanation-ontology.owl>

<sup>8</sup><http://www.fipa.org/specs/fipa00061/SC00061G.html>



Line 2 indicates that the sender of the message is an Agent as defined in the *Hypermedia MAS Core Ontology*. The core *Agent Abstraction Ontology* is used to indicate that *CarB* adheres to the BDI paradigm using AgentSpeak(L) (line 5) and at the implementation level, is an ASTRA agent using the MAMS framework (line 6). The FIPA-ACL ontology structures the message, which is framed as a request for an explanation (lines 13-14), concerning the action *brake* (line 15). This illustrates what we consider to be the most basic request for an explanation. In the future, we envisage this being more nuanced, for example asking for contrastive explanations.

**Sending and Receiving the Explanation Request** [25] propose a general content-agnostic protocol for inter-agent explanations, which allows agents to accept or reject given recommendations or request more details. The design of the protocol adheres to the REpresentational State Transfer (REST) architectural style [37]: that each message is designed to be self contained. Similarly, a protocol could be used to formalise the exchange of messages between *Car A* and *Car B*, which can be communicated using REST over HTTP. In order to request an explanation, *Car B* may send a POST request with the request for an explanation to *Car A*, which *Car A* receives.

**Interpreting the Explanation Request from Design Level to Implementation Level** To interpret the explanation request, *Car A* needs a plan to react to the incoming message and parse the content of the message body. From this, *Car A* can extract the FIPA performative (request), and from the message content (fipa:content) that the sender is requesting an explanation (nep:Explanation of eo:Explanation), about car:brake at the domain level (aa:what). We assume that *Car A* has a mapping between the ontology description car:brake and its external action brake  $e_6^{impl}$ ; For example the *unit generation algorithm* that adds events to the explanatory content store could be extended to include Semantic data as suggested in [22]. Due to the additional information in the message *Car A* can also extract the **design** and **implementation** levels of *Car B*. *Car A* and *Car B* share the same **design**, and therefore *Car A* can generate an explanation at the design level.

It follows that inputs to an *explanation generation algorithm* (see [22]) would include information about the explaine, which the explainer can use to determine the level at which to generate the explanation, as well as the specific explanation request.

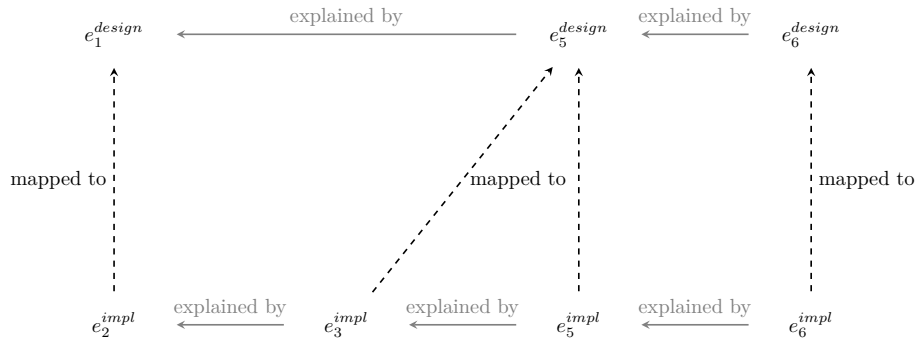
**Generating Explanation from Implementation Level to Design Level** In order to generate explanations, the agent needs first to select from the explanatory content store all the relevant events required to build the explanation. Second, the agent needs to select from a set of *explanation functions* the appropriate function according to the particular event, purpose, or level of abstraction of the explanation.

*Car A* received the request from *Car B* to explain at the domain level the external action brake, but it has inferred that they operate at the same design level and so can generate the explanation at this level. Looking at the causal relationships that lead the action brake  $e_6^{impl}$  we could have the following sequence at the implementation level:

$$e_2^{impl} \leftarrow e_3^{impl} \leftarrow e_5^{impl} \leftarrow e_6^{impl}$$

That is: “I brake because I have the intention int-1 created from the goal slow\_down, because I believe there is a pedestrian at the crosswalk”.

Together with the explanation functions, in order to communicate with a higher level of abstraction, within the agent we also need to define a set of *mapping functions* that are dependent on the technology used at the implementation level. These mapping functions map a set of explanatory contents at the lower level to one explanatory content at the higher level. An example of mapping is represented in Figure 3. The specific mappings and relationships could be informed by the *Agent Abstraction ontologies*: for example either within the core ontology, or an extension specific to *Jason* which extends the classes and relationships in the core *Agent Abstraction ontology*.



**Figure 3:** Example of mapping from the explanatory contents at the Implementation Level to the explanatory contents at the Design Level.

Id	Type	Detail
$e_1^{design}$	Belief	pedestrian at crosswalk
$e_5^{design}$	Intention	int1 slow down
	Goal	slow down
$e_6^{design}$	Action	brake

**Table 3**

Events of the agent *Car A* state generated and stored at *Jason* Design Level.

Table 3 shows an example of the outputs of the mapping: the implementation details are represented by design-level concepts, for inclusion in the explanation. To generate the explanation, an *explanation generation algorithm* is applied (as per [22]) which may select from a set of explanation functions, according to the intended audience which in this case is another agent. The explanation function may chain the design level details selected, represented using the core *Agent Abstraction Ontology*, using an ontology such as Explanation Ontology to frame the concepts, and the *Explanation Patterns Ontology* (prefix ep) ([30]) which allows an explanation to be linked to some prior knowledge.

For example, the explanation instance may be represented in JSON-LD as such:

```

1  "ContextualExplanationInstance": {
2    "rdf:type": "eo:ContextualExplanation",
3    "ep:isBasedOn":
4    [ "ContextualKnowledgeInstance1", "ContextualKnowledgeInstance2" ],
5    "eo:addresses": {"aa:what": "car:brake"}
6  }
7  "ContextualKnowledgeInstance1": {
8    "rdf:type": "eo:Contextual_Knowledge",
9    "aa:agentState": {
10     "rdf:type": "aa:Intention",
11     "aa:hasInformation": "int-1 slow down",
12     "aa:explainedBy": "ContextualKnowledgeInstance3"
13   }
14 }
15 "ContextualKnowledgeInstance2": {
16   "rdf:type": "eo:Contextual_Knowledge",
17   "aa:agentState": {
18     "rdf:type": "aa:Goal",
19     "aa:hasInformation": "slow down",
20     "aa:explainedBy": "ContextualKnowledgeInstance3"
21   }
22 }

```

```

23 "ContextualKnowledgeInstance3": {
24   "rdf:type": "eo:Contextual_Knowledge",
25   "aa:agentState": {
26     "rdf:type": "aa:Belief",
27     "aa:hasInformation": "pedestrian at crosswalk"
28   }
29 }

```

This defines a Contextual Explanation as per [30] which contains a collection of contextual knowledge. In this case, this is represented by the *Goal* and *Intention* of the agent, which are in turn based on the *Belief* of the agent that there is a pedestrian, which represents a fact about the world (being a subclass of *What* in the *Agent Abstraction Ontology*). For the sake of the example, we imagine some relationships in the *Agent Abstraction Ontology* such as *aa:agentState* to indicate that the content of the contextual knowledge relates to agent state, *aa:hasInformation* and *aa:explainedBy* to indicate the contents of the state and causal links respectively. In reality, the specific details (such as the belief *pedestrian at crosswalk*) may also need to be mapped to some external shared ontology such as the fictitious car ontology, in order for the details of agent state to be meaningful between agents.

**Communicate Explanation at the Design Level** As for above, the message is communicated by *Car A* to *Car B* over HTTP using REST. Per our discussion above, we suggest the messages be self-contained, i.e. in the generated explanation, the initial question posed by *Car B* is included (line 6 in the example above).

*Car B* can then receive and interpret the explanation, as it has been communicated at a level of shared understanding (the design level) using some *explanation interpretation algorithms* as per [22]. For example, *Car B* could determine whether the reason given for the action to brake (the belief that there is a pedestrian) corresponds with any plans it has, for example perhaps (hopefully) *Car B* has a plan to brake when it perceives a pedestrian and so takes this action too.

## 5. Limitations and Future Work

The above presented preliminary analysis towards *Agent Abstraction ontologies*, as such more analysis is required, in particular around the relationships between concepts across the dimensions identified in Section 3.2. We positioned the *Agent Abstraction ontologies* as a set; with a core *Agent Abstraction ontology* which is extended at the implementation levels. However, in designing the core ontology we may need to be realistic about how much meaning very disparate types of agents can share; extensions may also be required at the design level. Nonetheless, immediate future work is to continue analysis and create a prototype core *Agent Abstraction ontology* in order to demonstrate the example presented in Section 4. In the longer term, we hope to address the issue of sharing meaning between agents with a higher architectural dissimilarity.

Wider future work includes extending existing ontologies such as the *Explanation Ontology* [30] to include a greater range of question types, and concepts applicable to inter-agent explanations. What these inter-agent specific concepts are, and how inter-agent explanations can be evaluated and are distinguished from other inter-agent communication technologies, also requires further analysis.

Although the use of ontologies allows agents to share knowledge there are still programming overheads when designing agents which can use and interpret the information shared. In the above, agents may need to integrate domain-specific ontologies into the stage at which explanatory content is generated, for example instead of (or as well as) storing a detail such as *pedestrian at crosswalk*, a Semantic representation of the belief may need to be stored. This would be required so that agents can share domain understanding on the actions, beliefs, goals, etc. of other agents.

Once a core *Agent Abstraction ontology* is created, potential future work would be to expose the Semantically augmented explanation store of an agent as a Knowledge Graph which could be examined by other agents and reasoned over to generate new knowledge.

## 6. Conclusion

We presented preliminary analysis towards a core *Agent Abstraction ontology* which we propose can be extended for different agent architectures at the implementation level. We rely on existing ontologies for exchanging explanations, and position the *Agent Abstraction ontologies* as part of a suite of tools to facilitate inter-agent explanations between heterogeneous hypermedia agents. We propose that to facilitate such exchanges, the ontologies need to operate at different levels of abstraction: agents which have very dissimilar architectures interacting at the highest levels, and agents which are very similar interacting at lower levels. At the highest levels, we model classes based on the concept of the *cognitive neck* introduced in [5]; *constraints*, *wishes*, *hows* and *whats*. The analysis stops short of providing relationships between the classes and proposes four dimensions along which to design such relationships, the *logical*, *hierarchical*, *causal* and *temporal* dimensions. This is proposed as future work, along with an implementation which illustrates the example discussed.

We shared the example of two agents sharing understanding at the design level, generating explanatory content and exchanging messages containing representations of their internal state to reach mutual understanding. The agents demonstrated where ASTRA and *Jason* agents, which have been used in hypermedia agent frameworks. As we are aiming for interoperability and mutual understanding between agents with similar, and different architectures, the particular agents do not have to be hypermedia agents specifically. However they do need to be able to leverage Semantic Web technology, and beyond inter-agent explainability, we propose that uniform representations for agents state could also be valuable tools in information exchange and interoperability between heterogeneous agents on the Web generally.

## Acknowledgments

This study is partially funded by ANR-FAPESP NAIMAN project (ANR-22-CE23-0018-01, FAPESP 2022/03454-1).

## Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

## References

- [1] P. Langley, B. Meadows, M. Sridharan, D. Choi, Explainable agency for intelligent autonomous systems, in: Twenty-Ninth IAAI Conference, 2017.
- [2] A. Omicini, et al., Not just for humans: explanation for agent-to-agent communication, in: CEUR WORKSHOP PROCEEDINGS, volume 2776, Sun SITE Central Europe, RWTH Aachen University, 2020, pp. 1–11.
- [3] A. S. Rao, M. P. Georgeff, BDI agents: From theory to practice, in: V. R. Lesser, L. Gasser (Eds.), Proceedings of the First International Conference on Multiagent Systems, June 12–14, 1995, San Francisco, California, USA, The MIT Press, 1995, pp. 312–319.
- [4] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, A. Santi, Multi-agent oriented programming with jacamo, Science of Computer Programming 78 (2013) 747–761.
- [5] A. Ricci, S. Mariani, F. Zambonelli, S. Burattini, C. Castelfranchi, et al., The cognitive hourglass: Agent abstractions in the large models era., in: AAMAS, volume 24, 2024, pp. 2706–2711.
- [6] S. Mayer, A. Ciorrea, A. Ricci, M. I. Robles, M. Kovatsch, A. Croatti, Hypermedia to connect them all: Autonomous hypermedia agents and socio-technical interactions, Internet Technology Letters 1 (2018) e50.

- [7] M. Luck, P. McBurney, O. Shehory, S. Willmott, Agent technology, Computing as Interaction: A Roadmap for Agent Based Computing, University of Southampton on behalf of AgentLink III, 2005.
- [8] R. H. Bordini, J. F. Hübner, M. Wooldridge, Programming multi-agent systems in AgentSpeak using Jason, volume 15, John Wiley & Sons, 2007.
- [9] A. S. Rao, AgentSpeak (L): BDI agents speak out in a logical computable language, in: European workshop on modelling autonomous agents in a multi-agent world, Springer, 1996, pp. 42–55.
- [10] A. Ciorrea, S. Mayer, F. Michahelles, 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, 2018, pp. 813–822.
- [11] D. Vachtsevanou, P. Junker, A. Ciorrea, I. Mizutani, S. Mayer, Long-lived agents on the web: Continuous acquisition of behaviors in hypermedia environments, in: Companion Proceedings of the Web Conference 2020, 2020, pp. 185–189.
- [12] D. Vachtsevanou, B. de Lima, A. Ciorrea, J. F. Hübner, S. Mayer, J. Lemée, Enabling BDI agents to reason on a dynamic action repertoire in hypermedia environments, in: Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, 2024, pp. 1856–1864.
- [13] N. Hafiene, L. G. Nardin, O. Boissier, Knowledge level support for programming agents to interact in regulated online forums, in: International Workshop on Coordination, Organizations, Institutions, Norms and Ethics for Governance of Multi-Agent Systems, 2024.
- [14] R. Collier, E. O'Neill, D. Lillis, G. O'Hare, Mams: Multi-agent microservices, in: Companion Proceedings of The 2019 World Wide Web Conference, 2019, pp. 655–662.
- [15] E. Hobson-O'Neill, Multi-Agent Microservices: Integrating Multi-Agent Systems within Microservice-based Architectures, Ph.D. thesis, University College Dublin, 2024.
- [16] R. Collier, S. Russell, D. Lillis, Reflecting on agent programming with agentspeak(l), in: International Conference on Principles and Practice of Multi-Agent Systems, Springer, 2015, pp. 351–366.
- [17] E. O'Neill, K. Beaumont, N. V. Bermeo, R. Collier, Building management using the semantic web and hypermedia agents (2021) 32–37. URL: <http://ceur-ws.org/Vol-3111/short5.pdf>.
- [18] K. Beaumont, E. O'Neill, N. V. Bermeo, R. Collier, Collaborative route finding in semantic mazes, Proceedings of the All the Agents Challenge (ATAC 2021) co-located with the 20th International Semantic Web Conference (ISWC 2021) (2021).
- [19] E. O'Neill, R. Collier, Exploiting service-discovery and openapi in multi-agent microservices (mams) applications, 11th International Workshop on Engineering Multi-Agent Systems (EMAS2023) (2023).
- [20] K. Beaumont, R. Collier, Do you want to play a game? learning to play tic-tac-toe in hypermedia, in: Multi-Agent Systems: 21st European Conference, EUMAS 2024, Dublin, Ireland, August 26–28, 2024, Proceedings, volume 15685, Springer Nature, 2025, p. 441.
- [21] S. Anjomshoe, A. Najjar, D. Calvaresi, K. Främling, Explainable agents and robots: Results from a systematic literature review, in: 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019, International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1078–1088.
- [22] K. Beaumont, E. Yan, S. Burattini, R. Collier, Engineering inter-agent explainability in BDI agents, in: International Workshop on Explainable, Transparent Autonomous Agents and Multi-Agent Systems, 2025.
- [23] G. Ciatto, M. I. Schumacher, A. Omicini, D. Calvaresi, Agent-based explanations in ai: Towards an abstract framework, in: International workshop on explainable, transparent autonomous agents and multi-agent systems, Springer, 2020, pp. 3–20.
- [24] S. Rodriguez, J. Thangarajah, A. Davey, Design patterns for explainable agents (xag), in: Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, 2024, pp. 1621–1629.
- [25] G. Ciatto, M. Magnini, B. Buzcu, R. Aydoğan, A. Omicini, A general-purpose protocol for multi-agent based explanations, in: International Workshop on Explainable, Transparent Autonomous



Agents and Multi-Agent Systems, Springer, 2023, pp. 38–58.

- [26] E. Yan, S. Burattini, J. F. Hübner, A. Ricci, A multi-level explainability framework for engineering and understanding BDI agents, *Autonomous Agents and Multi-Agent Systems* 39 (2025) 9.
- [27] N. F. Noy, D. L. McGuinness, et al., *Ontology development 101: A guide to creating your first ontology*, 2001.
- [28] X. Su, M. Matskin, J. Rao, Implementing explanation ontology for agent system, in: *Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003)*, IEEE, 2003, pp. 330–336.
- [29] I. Tiddi, M. d’Aquin, E. Motta, An ontology design pattern to define explanations, in: *Proceedings of the 8th international conference on knowledge capture*, 2015, pp. 1–8.
- [30] S. Chari, O. Seneviratne, D. M. Gruen, M. A. Foreman, A. K. Das, D. L. McGuinness, Explanation ontology: a model of explanations for user-centered AI, in: *International semantic web conference*, Springer, 2020, pp. 228–243.
- [31] A. Groza, M. Pomarlan, Towards an ontology of explanations, in: *International Workshop on Measuring Ontologies in Value Environments*, Springer, 2020, pp. 73–85.
- [32] M. A. Houghtaling, S. R. Fiorini, N. Fabiano, P. J. Gonçalves, O. Ulgen, T. Haidegger, J. L. Carbonera, J. I. Olszewska, B. Page, Z. Murahwi, et al., Standardizing an ontology for ethically aligned robotic and autonomous systems, *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 54 (2023) 1791–1804.
- [33] F. Toyoshima, A. Barton, O. Grenier, Foundations for an ontology of belief, desire and intention, in: *Formal Ontology in Information Systems*, IOS Press, 2020, pp. 140–154.
- [34] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, et al., The rise and potential of large language model based agents: A survey, *Science China Information Sciences* 68 (2025) 121101.
- [35] V. Tamma, T. Bench-Capon, An ontology model to facilitate knowledge-sharing in multi-agent systems, *The Knowledge Engineering Review* 17 (2002) 41–60.
- [36] K. Beaumont, R. Collier, Fully embedded learning in BDI agents programmed in ASTRA, in: R. Collier, V. Mascardi, A. Ricci (Eds.), *Agents and Multi-Agent Systems Development – Platforms, Toolkits, Technologies*, Springer, 2025.
- [37] R. T. Fielding, *Rest: architectural styles and the design of network-based software architectures*, Doctoral dissertation, University of California (2000).