Applied Ontology 15 (2020) 493–527
DOI 10.3233/AO-200236
IOS Press

Modeling artificial agents' actions in context – a deontic cognitive event ontology

Miroslav Vacura
University of Economics, Prague, W. Churchill Sq. 4, 130 67 Prague 3, Czech Republic
E-mail: vacuram@vse.cz

Abstract. Although there have been efforts to integrate Semantic Web technologies and artificial agents related AI research approaches, they remain relatively isolated from each other. Herein, we introduce a new ontology framework designed to support the knowledge representation of artificial agents' actions within the context of the actions of other autonomous agents and inspired by standard cognitive architectures. The framework consists of four parts: 1) an event ontology for information pertaining to actions and events; 2) an epistemic ontology containing facts about knowledge, beliefs, perceptions and communication; 3) an ontology concerning future intentions, desires, and aversions; and, finally, 4) a deontic ontology for modeling obligations and prohibitions which limit agents' actions. The architecture of the ontology framework is inspired by deontic cognitive event calculus as well as epistemic and deontic logic. We also describe a case study in which the proposed DCEO ontology supports autonomous vehicle navigation.

Keywords: Artificial, agent, action, context, ontology, cognitive

Accepted by: Roberta Ferrario

1. Introduction

Modeling the activity of artificial agents in the context of actions by other autonomous agents is one of the more difficult problems in artificial intelligence. Most implemented or projected systems use a single-agent scenario and differ only in the way in which the agent approaches its environment (Asprino et al., 2016). In real conditions artificial agents often interact with other artificial agents or with humans, rendering single-agent scenarios unusable. An artificial agent may be an autonomous vehicle on the road that must interact with other vehicles that are either also autonomous or controlled by a human driver or it may be an industrial transportation robot that must maneuver an environment it shares with other more or less sophisticated robots and/or human workers.

An autonomous vehicle on the road should be able to receive information from other autonomous vehicles (or human drivers), to model their intentions, and to compare such information with its own intended actions framed by obligations based on laws and other traffic regulations. Notably, these features are at once part of the vehicle's modeling of planned actions that advance it towards the desired final destination.

In this paper, we abstract some of the features of the complex environment in which an agent's behavior takes place and focus on modeling the information context and the actions of other agents that may influence the agent's own reasoning and actions. Effective control of the agent's activity in the context of the actions of other agents requires some understanding of the "inner context" of these actions – each action on the part of any agent is based on information that a) the agent considers to be true (e.g., that a traffic light is green) and b) involves a purpose or motive, i.e., what the agent intends to accomplish with such an action. The purpose of the action may be viewed as the state that the agent "desires" to realize (e.g., reach the destination, move cargo to a designated location). Accordingly, the state may be described by a statement and the agent "desires" the statement to be true. For sure, the actions of an agent cannot be understood (or predicted) without taking such context into account.

One of the possible approaches to address the problem of understanding other agents' actions is to model their mental states. To model the mental states of agents, one can utilize mentalistic models of human behavior based on Hobbesian or common-sense psychology (Gert, 1996). *Mentalistic* language is a language which includes terms describing mental phenomena, such as "beliefs", "aversions" or "desires" (Spiker, 1989). Knowledge representation models are called *mentalistic* if they are related to such mental notions (Pohl, 1997). Mentalistic models have been used to provide a stable framework for knowledge representation systems which can be combined with other content using machine learning techniques (Pohl and Achim, 1999).

In contrast, the use of mentalistic models has been very limited in the Semantic Web context. A rudimentary mentalistic language, consisting of the "belief" concept, was used to model changing knowledge (Kang and Lau, 2004). While Semantic Web technologies have their drawbacks, they excel when used as highly flexible knowledge representation techniques and tools for extracting meaning from large amounts of unstructured web content, doing so through the use of standards with high interoperability, such as XML, RDF, and OWL. Integration of traditional AI techniques with Semantic Web technologies should therefore open up a wealth of semantically structured information on, e.g., Linked Data sources for the artificial agents.

The rest of the paper is organized as follows: the second section of the paper describes the methodology and requirements for an ontological model for the artificial agent's actions. Section 3 provides a general overview of the architecture of the proposed framework. The next section describes deontic cognitive event calculus briefly. Section 5 describes the proposed ontology framework DCEO itself along with the description of state of the art of modeling in respective areas. Section 6 provides evaluation of the ontology and compares the presented formalism with previously defined requirements. Section 7 describes a case study – enhancing autonomous vehicle navigation. Then Section 8 discusses possible future extensions of the framework. Finally, the last section provides the conclusions.

2. The methodology used for DCEO development

To develop a sound ontology one needs to use a formal methodology that provides structured guidelines and well-defined ontology life cycle management. We base the development of the DCEO ontology on the On-To-Knowledge methodology developed by Sure et al. (2009) and Staab et al. (2001). We also take into account some methodological principles introduced by the METHONTOLOGY methodology by Fernández-López et al. (1997) (see also the work by Gómez-Pérez et al. (2003)) and NeOn Methodology by Suárez-Figueroa (2012).

The development of our ontology began with a *feasibility study* that sought to identify the problem and opportunity areas of a proposed ontology as well as the potential solutions it might provide. It was found that descriptions of the interactions between artificial agents and some aspects of their behavior planning could benefit from the application of Semantic Web technologies. Existing research in this field was limited and thus there was an opportunity to improve existing solutions used in AI communities by situating them as a starting point and integrating them with Semantic Web approaches.

The second phase of the DCEO ontology development was the *kickoff phase*, the phase in which the development of ontology actually began. In this phase the requirements for the ontology were gathered. The outcomes of this phase were the ontology requirements specification document (ORSD) and a semi-formal description of our ontology. The overview of gathered requirements is informally presented in Section 2.1, and the first semi-formal description of the DCEO ontology was presented in (Vacura and Svátek, 2016).

In the *refinement phase* we fully formalized our ontology. This phase was cyclical and required several iterations of enhancing and fine-tuning the ontology based on feedback from domain (AI) experts and comparisons with initial requirements. The outcome of this phase was v. 1.0 of the DCEO ontology.

In the *evaluation phase* we first performed a technology-focused evaluation that judged language conformity (syntax) and that involved consistency (semantics) tests and expert evaluations of interoperability, scalability, and other important characteristics. The user-focused evaluation was performed in cooperation with domain experts and we collected their recommendations and comments. These evaluations were followed by more iterations of the refinement phase. After two iterations, the outcome of this phase was an evaluated ontology – v. 1.2 of the DCEO ontology. Evaluation is discussed in more detail in Section 6.

The *application and evolution phases* followed, consisting of several new updates to the DCEO ontology, followed by refinement-evaluation cycles that produced v. 2.0 (2018) and v. 2.1 (2019) of DCEO. Several implementation and integration projects using the DCEO ontology were started; one case study is briefly described in Section 7.

2.1. Requirements for the ontological modeling of an artificial agent's actions

An elementary set of requirements can be devised based on current and prospective usage scenarios for the ontology such as those involving an interaction between autonomous vehicles on the road or an interaction between robots in an industrial environment. Experiences with similar systems based on calculus paradigms have also been helpful in formulating the set of design requirements presented below. However, this paper does not specify any requirements with reference to specific applications.

When an artificial agent operates in the context of other agents' actions, it must communicate with or even influence the behavior of these other artificial or human agents. Thus, an elementary requirement is the ability to model the acquisition and provision of information by means of communication with other agents.

The artificial agent has to communicate and interact with other agents, and it can also receive publicly available information, but this information cannot be taken at face value because it only represents the belief of another agent, and such a belief can be wrong. In the case of some real world scenarios an agent can even intentionally provide misleading information. Pieces of information received from different agents (or perceived) may also be mutually inconsistent (a human driver may signal intention to turn right, however, he/she may start to turn left). Such an account of communication requires *epistemic mentalistic models* in order to distinguish belief from knowledge, as argued by Arkoudas and Bringsjord (2009).

An artificial agent also needs to reason about the context of its own intended actions, i.e., their background, motivation, and predicted consequences, and those of all other agents presenting the external context which is important for planning of his own actions. In other words, models enabling such reasoning must deal with past, current, and future actions and generally with *events*, so flexible handling of *time* is necessary.

Dealing with future events caused by the actions of other agents requires mentalistic models for their internal mental representations of the evaluated future – their *desires* and *aversions* – from Hobbesian psychology (Gert, 1996). T. Hobbes considered desires and aversions to be fundamental emotions which provide background for any agent's actions. Mental representations of these emotions then enable deriving the *intentions* of an agent. However, it is worth noting that the Hobbesian theory of desires and aversions as being future oriented can be criticized as limited, if one understands desires and aversions as something that can also relate to present or past events.

These future-involving mental states can be called *protential*, based on the term "protention", meaning the consciousness of future, which was coined by the philosopher E. Husserl (McInerney, 1988). These internal mental states influence directly the external behavior of agents, i.e., intended actions materialize and became real actions performed in an external environment.

Another component of the mentalistic model is the inner representation of the context of the agent's actions which consists of the external norms of his behavior, which can be called *obligations*, *permissions* and *prohibitions*. Different agents may accept different obligations and different prohibitions. An obligation is a requirement on the part of the agent to act when some defined condition of his inner context is present. Prohibition is the requirement to suspend an action or abstain from it. Knowledge of these limitations in behavior on the part of other agents can be used to predict their actions. A model involving obligations and prohibitions is referred to as *deontic*, which is based on a similar use of the term in the context of deontic logic (Gabbay et al., 2013).

Note that even if we use mentalistic models and discuss an agent's behavior in mentalist terms, this does not imply that such an agent would be required to have genuine mental states; accepting a thoroughly instrumentalist view of mental states of artificial agents is sufficient for our purposes.

We can now summarize the requirements for (or, directly, components of) a minimal model enabling artificial agents' reasoning in the contexts defined above:

(1) A comprehensive model of artificial agents' interactions and communications requires:

  (1a) a model of events and actions,
  (1b) an epistemic mentalistic model,
  (1c) a protential mentalistic model, and
  (1d) a deontic mentalistic model.

There are also other requirements for an ontology to be able to handle all of the necessary aspects for modeling the actions of autonomous agents. It is not enough to model that an agent believes or desires something. Phenomenological philosophy (F. Brentano, E. Husserl) asserts that *intentionality* is a fundamental feature of any mental act (Smith and McIntyre, 1982). Intentionality means that every mental act has *content*, i.e., it is "about" something (Smith and Ceusters, 2015; Barton et al., 2018). The ontology has to provide a way to model the contents of mental states.

The contents of epistemic states, i.e., descriptions of what an agent believes or desires, may be *complex*. They are possible *states of affairs* in the sense similar to the one described by the philosopher Wittgenstein (1922). However, the discussion of states of affairs and their relation to facts is a complex issue in contemporary philosophy – see e.g. Texor's work (2016) for an introduction.

Different agents may believe different statements and these beliefs may be *inconsistent*. An ontology enabling the representation of the epistemic states of these agents should be able to include such mutually inconsistent beliefs and still enable reasoning at some level.

We can now summarize additional requirements for an ontology to be able to handle all these aspects of modeling the actions of autonomous agents:

(2) The ontology should be able to model:
  (2a) the content of mental states,
  (2b) complex epistemic states,
  (2c) epistemic inconsistency.

Artificial agents can be differentiated into several classes, and these classes influence the character of their interactions. The necessity of modeling different types of artificial agents' interactions will be discussed in detail in the following subsection.

2.2. *Different types of artificial agent interaction scenarios*

Another requirement is related to the multiplicity of possible scenarios that involve artificial agents. While we have already highlighted scenarios involving autonomous vehicles and autonomous industrial transport robots, there are, for sure, as many other scenarios as there are different types of artificial agent interactions. There are different types of artificial agent interactions and it would be beneficial to be able to model all of them. As a first step, we propose several distinctions that can be used to classify different scenarios.

First, artificial agents can be either *cooperative* or *non-cooperative*. We note above a situation in which agents are non-cooperative, even to the extent that they provide misleading information. These agents are usually controlled by different parties and may sometimes demonstrate competitive attitudes toward each other. Meanwhile, cooperative agents are usually controlled or deployed by a single party and work toward a single goal. However, even agents who are not working toward a single goal may exhibit cooperative behavior depending on the context. In some contexts, an agent's individual goal may require it to cooperate with other agents. If the context changes cooperative behavior may be diminished or replaced by non-cooperative behavior. We therefore term a group of agents "cooperative" only when they continuously work toward a single given goal.

Artificial agents can be either *heterogeneous* or *homogenous* when related to other agents. Homogenous agents are agents which are similar in terms of what types of information they accept, process, and provide. Homogenous agents may, however, use different internal architecture and seek different goals. They may sometimes also produce different "behavior". Heterogeneous agents produce different kinds of information or process information in different domains. Heterogeneous characteristics of agents may be used to clearly differentiate agents into a few classes or groups.

Cooperative agent interactions are more commonly used and implemented. An example of a cooperative model is the well-known *pandemonium architecture* for object recognition (Selfridge, 1959; Lindsay and Norman, 1977). This architecture consists of a group of feature agents, a group of cognitive agents and one lone decision agent. In this architecture, all these agents (demons) work together toward the common goal of the recognition of an object – that's why we consider these agents and overall architecture cooperative.

Each agent is specialized: feature agents excel in the detection of individual features. Each of them may detect different feature, however, they are the same with regard to types of information they accept, process, and provide – that's why we describe them as homogenous group of agents. Similarly, cognitive agents use information provided by feature agents and recognize individual patterns or objects, and, finally, a decision agent decides which object was recognized utilizing the information provided by the cognitive agents. In this architecture, group of feature agents is homogenous, group of cognitive agents is also homogenous, and the lone decision agent is heterogeneous in relation to the others.

We can now summarize requirements for the proposed ontology architecture related to modeling different types of artificial agent interaction scenarios. It would be beneficial for the ontology to enable the user to model different types of scenarios for artificial agent interactions:

(3) cooperative and non-cooperative,
(4) heterogeneous, and homogenous.

The most challenging are usually scenarios involving non-cooperative, heterogeneous agents because they require the most complex modeling of the epistemic states described above.

2.3. Other general requirements of ontology design

The proposed ontology is aimed partially at the AI community and this audience necessitates some additional requirements for our design. Most notably, the designed ontology should use paradigms known to the AI community – using completely new terminology and conceptual structures alien to community experts makes a proposed ontology difficult to use and less likely to be adopted. To the contrary, if ontology designers align the proposed design of an ontology with a well-known existing technique or approach that is widely understood in the community, then accessibility and usability increas. This approach can be also understood as, in a sense, "reusing" existing knowledge and as such is recommended by a number of methodologies (Suárez-Figueroa, 2012).

The purpose of our work is not to create an abstract academic model which would be perfect in theory but not usable by anyone but its author because of its steep learning curve. It is worthy, of course, to provide a firm theoretical background justifying the design choices made in developing the ontology; however, this theoretical complexity should not stand in the way of the average user of the ontology, who may be an expert in a different area. The complexity of ontology should be adequate for its purpose. The resulting ontology may be complex in some aspects if the domain it describes is also complex. The possible complexity of the ontology can be alleviated by providing examples, case-studies and documentation that let users understand the model. The resulting ontology may be also relatively simple and not consider all intricate complexities of notions it formalizes if it can fulfill some important applied purpose. Also well-defined focus of the ontology and clearly described internal structure based on separation of concerns principle may help to keep the complexity of the ontology manageable.

We can now summarize other general requirements of ontology design:

(5) The ontology should reuse existing knowledge where it is possible and meaningful.
(6) The complexity of ontology should be adequate for its purpose.
(7) The internal structure of the ontology should be based on the separation of concerns principle.

There are other methodological ontology design requirements which should be taken into account by any ontology developer, as summarized by Sure et al. (2009). Others, such as Oberle et al. (2006), describe the characteristics of badly modeled ontologies. We have tried to follow the suggestions provided in these works during the course of the development of our ontology.

3. Architecture of the deontic cognitive event ontology

To facilitate the representation of an artificial agent's knowledge in the context of activity of other agents, we developed the architecture depicted on Fig. 1, inspired by the Soar cognitive architecture

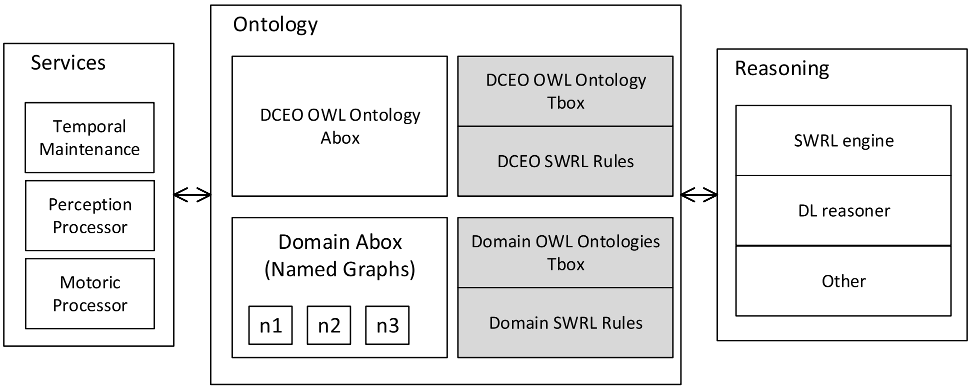

Fig. 1. Application architecture based on the DCEO.

(Laird et al., 1987) and the classical concept of the Model Human Processor (Newell et al., 1998; Card et al., 1983), which is, in turn, based on the Standard Model of human cognition (Simon and Kaplan, 1998; Klahr and MacWhinney, 1998). The central part of the architecture is the Deontic Cognitive Event Ontology (DCEO), inspired by $\mathcal{DCEC}^*$ (Bringsjord and Govindarajulu, 2013; Bringsjord et al., 2014a), that will be described in the next section.

The core of our architecture consists of the Tbox of an OWL ontology grayed in Fig. 1. The axioms of this Tbox represent stable knowledge about the ontological structure of the world. The content of the agent's own mental states, the mental states of other agents, and the context of the agent's actions are represented by axioms of the Abox of the ontology. Different agents may have mutually inconsistent mental states, and these may be complex and consist of several axioms. We decided to use named graphs (Carroll et al., 2005a) to model these mental states. The Abox is, therefore, split into a main graph that describes agents' mental states, events, and actions, and other named graphs that represent the content of these entities. These named graphs are identified by a unique URI.

There are several services that update the content of the Abox. Information obtained about the external context of the agent's actions (its environment) is processed by a component that is traditionally called the *Perception processor* (Newell et al., 1998; Card et al., 1983). It produces facts that are inserted into the Abox of the ontology classified as perceptions or communications from other agents.

Another service traditionally called the *Motoric Processor* (Newell et al., 1998; Card et al., 1983) monitors the current time and retrieves the statements representing actions that are to be performed at the given time point, facilitating the actual performance of these actions. Action can also include communication of content to other agents, sending a data message or starting a process in a virtual environment.

Temporal Maintenance is another service that carries out auxiliary operations such as removing old Abox axioms from the data store thus enabling the whole system to function efficiently. Namely, the growth of the number of axioms caused by the continuous addition of new statements to the Abox combined with axiom production by the SWRL (Semantic Web Rule Language) engine could be enormous; clearing of old data representing no-longer-useful knowledge could thus be necessary. The use of this service depends on the implementation; in some cases, such maintenance may not be necessary, or the complete history of operations may be valuable. Maintenance service may not delete old axioms, but instead move them to an archive KR where the complete history is stored, and that may set up an artificial agent as a kind of "long-term memory" that is queried only when specifically required.

There are also some stable axioms of the Abox that, in some scenarios, change rarely, for example, axioms about agent's obligations and prohibitions. A finite set of an agent's final goals (desires) that do not change may also exist. However, agent's representations of obligations, prohibitions, and desires of the other agents are susceptible to change because agents may change, or the agent recognizes that the original assumption about other agents' desires might have been mistaken.

The overall architecture is inspired by the standard Model Human Processor, and the internal structure of the ontology is based on the Deontic Cognitive Event Calculus ($\mathcal{DCEC}^*$), which has been successfully used in a number of real-world scenarios such as reasoning over a scene description (Marton et al., 2015), control of robot behavior (Bringsjord et al., 2014b), or even simulation of some features of human consciousness (Bringsjord et al., 2015). Following section introduces basic features of $\mathcal{DCEC}^*$.

4. Deontic cognitive event calculus ($\mathcal{DCEC}^*$)

Deontic Cognitive Event Calculus ($\mathcal{DCEC}^*$) is a multi-sorted quantified modal logic developed at Rensselaer Polytechnic Institute, which has a well-defined syntax and a proof calculus. Detailed information about multi-sorted first order logic (MSL) can be found in a book by Manzano (1996). $\mathcal{DCEC}^*$ syntax includes a system of sorts S, a signature f, a grammar for terms t, and a grammar for sentences ϕ; these are shown in Fig. 2. An overview of the whole formal syntax of $\mathcal{DCEC}^*$ can be found in the original works (Bringsjord and Govindarajulu, 2013; Bringsjord et al., 2014a).

The proof calculus is based on natural deduction (Jaśkowski, 1934) and includes all the introduction and elimination rules of first-order logic as well as rules for modal operators. In this paper we use the

$$S ::= \begin{array}{l}\mathsf{Object} \mid \mathsf{Agent} \mid \mathsf{Self} \sqsubset \mathsf{Agent} \mid \mathsf{ActionType} \mid \mathsf{Action} \sqsubseteq \mathsf{Event} \mid \\ \mathsf{Moment} \mid \mathsf{Boolean} \mid \mathsf{Fluent} \mid \mathsf{Numeric}\end{array}$$

$$f ::= \begin{array}{l} \mathit{action} : \mathsf{Agent} \times \mathsf{ActionType} \rightarrow \mathsf{Action} \\ \mathit{initially} : \mathsf{Fluent} \rightarrow \mathsf{Boolean} \\ \mathit{holds} : \mathsf{Fluent} \times \mathsf{Moment} \rightarrow \mathsf{Boolean} \\ \mathit{happens} : \mathsf{Event} \times \mathsf{Moment} \rightarrow \mathsf{Boolean} \\ \mathit{clipped} : \mathsf{Moment} \times \mathsf{Fluent} \times \mathsf{Moment} \rightarrow \mathsf{Boolean} \\ \mathit{initiates} : \mathsf{Event} \times \mathsf{Fluent} \times \mathsf{Moment} \rightarrow \mathsf{Boolean} \\ \mathit{terminates} : \mathsf{Event} \times \mathsf{Fluent} \times \mathsf{Moment} \rightarrow \mathsf{Boolean} \\ \mathit{prior} : \mathsf{Moment} \times \mathsf{Moment} \rightarrow \mathsf{Boolean} \\ \mathit{interval} : \mathsf{Moment} \times \mathsf{Boolean} \\ * : \mathsf{Agent} \rightarrow \mathsf{Self} \\ \mathit{payoff} : \mathsf{Agent} \times \mathsf{ActionType} \times \mathsf{Moment} \rightarrow \mathsf{Numeric} \end{array}$$

$$t ::= x : S \mid c : S \mid f(t_1, \ldots, t_n)$$

$$\phi ::= \begin{array}{l} p : \mathsf{Boolean} \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \phi \leftrightarrow \psi \mid \forall x : S.\phi \mid \exists x : S.\phi \mid \\ \mathbf{P}(a, t, \phi) \mid \mathbf{K}(a, t, \phi) \mid \mathbf{C}(t, \phi) \mid \mathbf{S}(a, b, t, \phi) \mid \mathbf{S}(a, t, \phi) \mid \\ \mathbf{B}(a, t, \phi) \mid \mathbf{D}(a, t, \mathit{holds}(f, t')) \mid \mathbf{I}(a, t, \mathit{happens}(\mathit{action}(a^*, \alpha), t')) \mid \\ \mathbf{O}(a, t, \phi, \mathit{happens}(\mathit{action}(a^*, \alpha), t')) \end{array}$$

Fig. 2. $\mathcal{DCEC}^*$ syntax.

syntax of $\mathcal{DCEC}^*$ as a starting point for development of the ontology to describe artificial agents' interaction. The calculus uses also a number of inferential rules described by Bringsjord and Govindarajulu (2013). However, to keep the ontology lightweight we consider these rules as optional. They may be included (after transformation to SWRL) if an application requires them. In this section we provide a brief overview of $\mathcal{DCEC}^*$ calculus to provide a background for the task of modeling artificial agents' action and preliminary introduction to reasons behind the design decision made during development of the ontology.

$\mathcal{DCEC}^*$ is based on the Event Calculus (EC), which was first introduced by Kowalski and Sergot (1986) as a logic programming formalism for representing events and their effects, and later also presented in a simplified version (Kowalski, 1992). A detailed presentation of EC can be found in the work of Shanahan (2001).

$\mathcal{DCEC}^*$ adapts three sorts from EC: Event, Moment and Fluent. The sort Boolean is only used to capture truth values. The following elements of the signature f of the syntax of $\mathcal{DCEC}^*$ are adapted from EC: *initially*, *holds*, *happens*, *clipped*, *initiates*, *terminates* and *prior*. For the relation *prior*, which introduces an order over time points, EC sometimes uses the simple symbol $<$.

We will now briefly describe the components of $\mathcal{DCEC}^*$. A Fluent is anything the value of which can change over time. Typically it is a truth value of a proposition (e.g., "Peter is student"). A fluent can be also a numerical value of a property that is subject to variation, e.g., temperature, but such a value can be easily transformed into a proposition (e.g., "The temperature is between 5 and 10 degrees Celsius.") so EC usually confines its focus to propositional fluents.

A Moment is a point in time. Points in time are ordered by the relation *prior*. The expression $prior(t_1, t_2)$ means that the time point t_1 precedes the time point t_2 (e.g., $t_1 = 01/01/2015$ precedes $t_2 = 01/01/2016$). The term $holds(f, t)$ says that fluent f holds at a given time t (e.g., $f =$ "Peter is student" holds at $t = 01/01/2016$). The expression $initially(f)$ indicates that the fluent f is true unless it was made false at some previous point in time.

Several signature members describe relations between events and fluents. The general idea of EC is that events cause changes of truth values of fluents. The expression $happens(e, t)$ thus informs that event e happened in time t (e.g., $e =$ "Peter concluded his studies" at $t = 05/03/2016$). The expression $terminates(e, f, t)$ states that following the event e, the fluent f ceased to hold at the time t (e.g., after $e =$ "Peter concluded his studies" at $t = 05/03/2016$, proposition $f =$ "Peter is a student" was no longer true). Similarly, the expression $initiates(e, f, t)$ states that after the event e, the fluent f started to hold at the time t (e.g., after $e =$ "Peter was inaugurated" at $t = 02/03/2011$, proposition $f =$ "Peter is a student" started to be true). That also means that both $terminates(e, f, t)$ and $initiates(e, f, t)$ imply $happens(e, t)$.

The expression $clipped(t_1, f, t_2)$ says that fluent f is terminated between time t_1 and time t_2 (e.g., $f =$ "Peter is a student" is terminated between $t_1 = 01/01/2016$ and $t_2 = 01/01/2017$).

$\mathcal{DCEC}^*$ introduces a mechanism to deal with epistemic information on the top of the event conceptualization of EC. $\mathcal{DCEC}^*$ has a classical monotonic view of the agents' knowledge of the world. The knowledge possessed by agents is considered to be unchanging, so if an agent knows ϕ at some time t, then the agent will continue to know ϕ for all time after t during which the agent is operational. On the other hand, an agents' *beliefs* can change as time passes. This marks a fundamental difference in understanding knowledge vs. belief in $\mathcal{DCEC}^*$.

The epistemic predicate $\mathbf{C}(t, \phi)$ indicates *common knowledge* (possessed by all agents) of ϕ at time t. The predicate $\mathbf{K}(a, t, \phi)$ says that agent *a knows* ϕ at time t. The predicate $\mathbf{B}(a, t, \phi)$ says that agent *a believes* in ϕ at time t. Finally, the predicate $\mathbf{P}(a, t, \phi)$ says that agent *a perceives* ϕ at time t.

$\mathcal{DCEC}^*$ also introduces tools for capturing the communication of agents. The predicate $\mathbf{S}(a, b, t, \phi)$ describes the *communication* of information ϕ from agent a to agent b at time t. A *public communication* of information ϕ at time t by agent a is denoted as $\mathbf{S}(a, t, \phi)$.

There is another set of predicates, which we may call behavioral: the predicate $\mathbf{D}(a, t, \mathit{holds}(f, t'))$ says that agent a *desires* that fluent f would hold at time t'. Similarly, to say that agent a at time t *intends* to perform an action of type α at time t', we use the predicate $\mathbf{I}(a, t, \mathit{happens}(\mathit{action}(a, \alpha), t'))$. These predicates are based on works by Goble (2003) and McNamara (2010).

Finally, the deontic predicate $\mathbf{O}(a, t, \phi, \mathit{happens}(\mathit{action}(a, \alpha), t'))$ should be interpreted according to authors of $\mathcal{DCEC}^*$ as: "If it is the case that a at time t believes ϕ, then that α is obligatory for a and this [obligation] is known by a." The semantics of this predicate is based on a study by Castañeda (1999).

These predicates also require the introduction of the sort ActionType covering the general types of action, and of the function $\mathit{action}(a, \alpha) = b$, expressing that for a given agent a, an action type α produces a specific action b. The operator $*$ is used to point out the reasoning agent himself among other agents in the universe of discourse. The operator $\mathit{payoff}(a, \alpha, t)$ is used to evaluate an action of type α performed by agent a at time t; the result of such an evaluation is of the Numerical sort.

There are some differences between DCEO ontology introduced in following sections and $\mathcal{DCEC}^*$ calculus. To name few of them: alongside the *desire* we formalize also *aversion* as its opposite. The modeling of desire and aversion is based on Hobbesian psychology. Hobbes considered desire and aversion to be fundamental emotions – all other emotions are based on these two. These two emotions are also opposites – desire means an inclination to move in some direction, to reach for some object, to acquire some object etc. Aversion means an inclination to move away from some object, to avoid any contact with some objects etc.

Alongside the *obligation* we define also *prohibition* and *permission*. We introduce operator $\mathit{results}(a, \alpha, t)$ that links a fluent with an action that produced it. Expressions initiates, terminates, clipped, initially and happens are modeled indirectly so they are not explicitly present in the ontology (details in the following sections). Chosen modeling approach enables modeling of epistemic meta-knowledge (x knows that y believes that f holds) in the ontology. There are also many other differences in the design of the ontology that were required because an ontology models reality in a very different way than a calculus. We will discuss details of the ontology design in the next section.

However, it is obvious that due to differences between approach utilizing a calculus and Semantic web techniques the alignment between these two is a bit vague. Still it is beneficial to use the $\mathcal{DCEC}^*$ calculus as a starting point for development of the ontology, as it provides a verified vocabulary and reasoning structures familiar to AI community.

5. Deontic cognitive event ontology

The proposed Deontic Cognitive Event Ontology (DCEO) is designed to satisfy the requirements defined in Sections 2.1, 2.2, 2.3 and utilizes some principles derived from the $\mathcal{DCEC}^*$ calculus. The development of DCEO was initiated in 2015, and a preliminary version was developed in 2016 (Vacura and Svátek, 2016). The first version was closer to $\mathcal{DCEC}^*$; however, during testing of the ontology on several scenarios, some design concepts had to be changed to make the ontology more understandable for practical use. The resulting ontology is presented in this paper; all necessary conceptual changes that may be important for the user acquainted with $\mathcal{DCEC}^*$ are mentioned in the following subsections. Figure 3 depicts the general overview of the DCEO ontology (plain arrows mark relations, arrows with

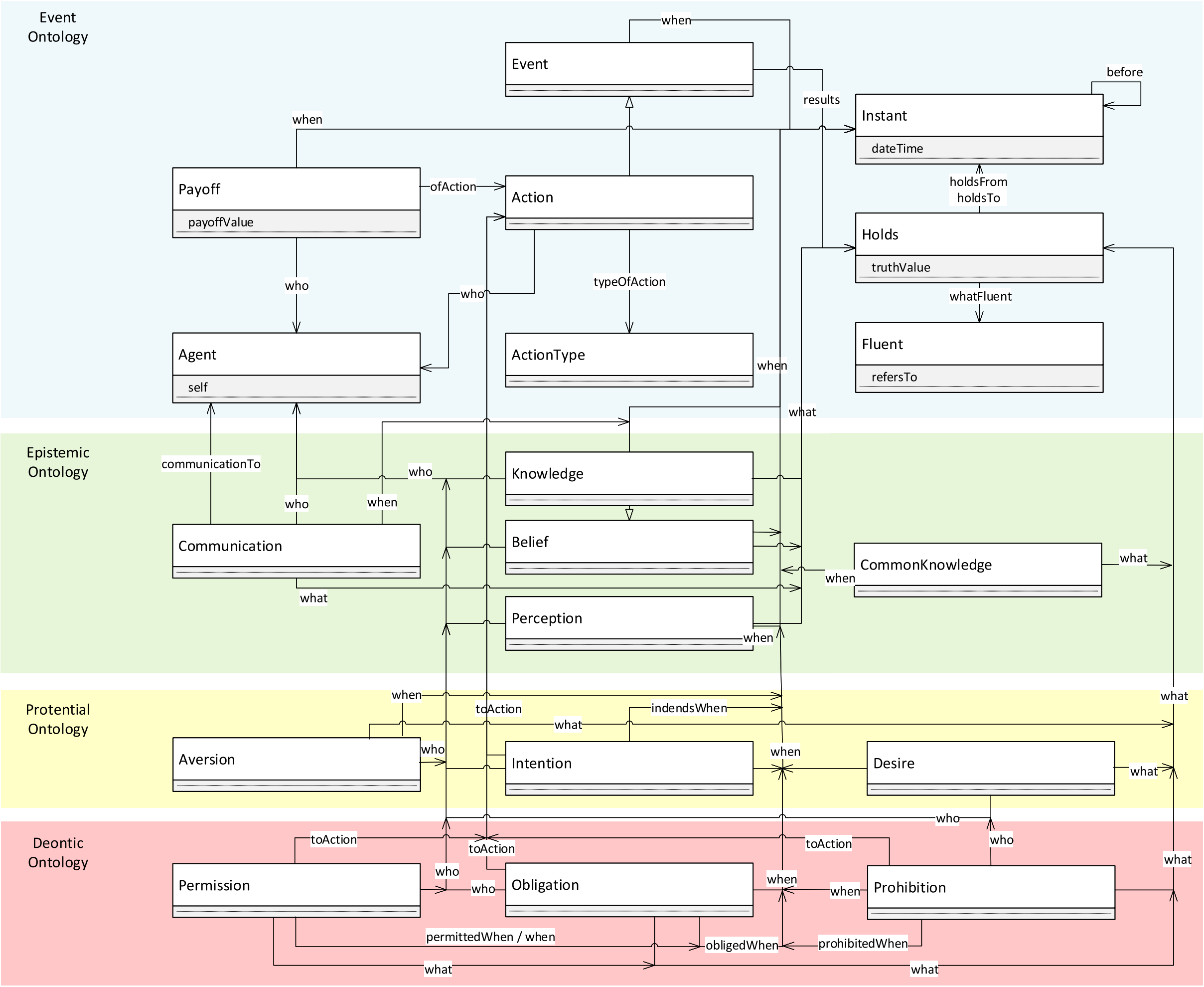

Fig. 3. Deontic cognitive event ontology (DCEO).

full arrowhead mark subsumptions). Specific axioms for each part of the ontology will be listed at the end of each respective section of the following text. All classes of the ontology are disjoint, and we omitted disjunction axioms for brevity. The exceptions are that we consider the class action to be a subclass of the class event: `Event` $\sqsupseteq$ `Action`, and with some reservations (see Section 5.2), the class knowledge is a subclass of class belief: `Belief` $\sqsupseteq$ `Knowledge`. The ontology can be downloaded from our website.[1]

5.1. The event section of the ontology

The ontology engineering community has been discussing the problem of modeling events for some time and has proposed several different ways of handling events (Hanzal et al., 2016). Most of these

---

[1] https://name.vse.cz/vacuram/ontologies/dceo/

approaches have been developed strictly within the context of the Semantic Web and were not related to any research based on different techniques or paradigms. One exception is the effort to provide an ontologic representation of the Discrete Event Calculus (DEC; Mepham and Gardner, 2009; Mepham, 2010), an alternative to Event Calculus (EC) (see Section 4). The Discrete Event Ontology (DEO) consisted of OWL ontology, several SWRL rules, and a resolver. This ontology only partially covered DEC; the resulting ontology was comprised of three classes (Events, Fluents, Timepoints) and a couple of rules.

The design of the event section of our ontology was influenced by DEC, which is closely related to $\mathcal{DCEC}^*$. While we also learned some lessons from the observations provided by the authors of DEO and DEO aimed to yield as precise as possible a reconstruction of the inferential rules of DEC using SWRL, we were primarily concerned with developing more effective knowledge representation in the ontology design as well as with its general usability and practicality (as stated in the overview of the requirements in Section 2.1).

The core of the event section of DCEO constitutes of classes `Event`, `Holds`, `Fluent` and `Instant`. The class `Fluent` represents any state of affairs, the truth-value of which can change over time. These states of affairs can be the result of actions, such as actual states of the world, or they may be the contents of mental states of agents. Because these mental states (e.g. what an agent believes or desires) may be complex, they could not be represented by a single RDF triplet. A description of these complex states of affairs may require a complex RDF graph, which is why there is the object property `refersTo` that links `Fluent` to a URI of the named graph that describes such state of affairs in its entirety, using the concepts from the appropriate domain ontology and/or from DCEO in the case of more complex mentalistic models.

The events are understood in this ontology in an abstract way that may differ from the way in which foundational ontologies understand events. For example Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) by Masolo et al. (2002, 2003) distinguishes between *endurants* and *perdurants* (eventive occurrences) that actually correspond to *continuants* and *occurrents* as defined, for example, in the KR Ontology theorized by Sowa (2000). Different kinds of perdurants are in DOLCE distinguished by notions of *homeomericity* and *cumulativity*. A detailed discussion of these terms can be found in (Jarrar and Ceusters, 2017). Meanwhile, a perdurant is either *stative* or *eventive* (it is an *event*) according to whether or not it is *cumulative*. In *stative* occurrences, DOLCE distinguishes between states and processes according to *homeomericity*. *Events* are called *achievements* if they are atomic, otherwise they are termed *accomplishments* (Masolo et al., 2003, 24). Time locations are in DOLCE considered individual qualities such as colors, weights, etc. Their corresponding qualia are called temporal regions – the temporal location of an occurrence is its quality and this corresponds to a quale that is a region in the temporal space (Masolo et al., 2003, 18).

There are also foundational ontologies such as Unified Foundational Ontology (UFO), which, because it focuses on structural (as opposed to dynamic) aspects of the world and accepts a descriptive commonsensical view of reality, was originally conceived as an *ontology of endurants* rather than one of perdurants (Guizzardi, 2005, 211). And yet even this ontology was later extended to handle temporal entities, ultimately yielding the foundational ontology of UFO-B (Guizzardi et al., 2013) which uses the term *event* for all perduring entities. Notably, events may be composed of other events and may be complex or atomic (having no proper parts). Moreover, events are constituted by transformations from one portion of reality (situation/fact) to another. While the notion of the situation is similar to the philosophical notion of the state of affairs, situations are notably bound to specific time points. Event calculus provides a foundation for our model: we model our events similarly, i.e., as entities bound to specific time points and that refer to changes in the world.

In our ontology, we locate events primarily only in time. Other approaches, like the Core Pattern for Events by Krisnadhi and Hitzler (2016) and the Simple Event Model Ontology,[2] assign a spatiotemporal location to each event. Many ontologies and vocabularies define events only with regard to their specific domains, such as music-oriented Event Ontology[3] or LODE (Linking Open Descriptions of Events)[4] that explicitly aim at historical events (for more information, please see the discussion on this topic in Hanzal et al. (2016)).

The events that are members of the class `Event` (or its subclass `Action`) happen at some time `Instant` (property `when`). The event `results` in some change in the world, such as that some `Fluent` `Holds` from some time `Instant` to another.

To specify time, we utilize W3C OWL Time ontology[5] and use class `time:Instant` (in $\mathcal{DCEC}^*$, it is Moment). Instants are point-like time entities that have no interior points. They can be identified with intervals with zero length, where the beginning and end are the same. For simplicity, we usually use only the `dateTime` property of this class.

At the base level, we prefer not to work with intervals. The class `Holds` enables us to specify the starting and ending instant. To represent intervals OWL Time class `time:ProperInterval` can be used, with properties `hasBeginning` and `hasEnd` referring to instants that define limits of an interval. However, according to our experience, this makes using the ontology a bit more cumbersome.

The following code is an example of the action `a1` and its result: fluent `f1` that holds from instant `i1` to the instant `i2` (in code we use `dceo:` as the prefix associated with DCEO concepts).

```
ex:a1  rdf:type  dceo:Action ;
     dceo:when          ex:i1 ;
     dceo:results       ex:h1 .

ex:h1  rdf:type  dceo:Holds ;
     dceo:holdsFrom     ex:i1 ;
     dceo:holdsTo       ex:i2 ;
     dceo:whatFluent    ex:f1 .
```

```
ex:i1  rdf:type  time:Instant  .
ex:i2  rdf:type  time:Instant  .
ex:f1  rdf:type  dceo:Fluent   .
```

The class `Action` can be linked to the class `Agent` by the relation `who` to describe the acting agent. We used a relatively straightforward modeling approach; however, we are aware of the fact that modeling actions is a controversial issue (Seddig-Raufie et al., 2018). We use the same relation to link an agent to such ontologically different entities as actions or epistemic states, however this relation may be later differentiated to more specific subrelations if needed.

Along with the object property `who`, we also introduced two other general object properties: `what` and `when`. These reflect various properties of $\mathcal{DCEC}^*$ calculus with common ranges. The range of property `who` is the class `Agent`, and it is used to link not only actions, but also mental states like beliefs or desires to their subject. The range of property `when` is class `Instant` and is similarly used to localize not only actions but also mental states in time. The range of property `what` is class `Holds` and is used

---

[2] http://semanticweb.cs.vu.nl/2009/11/sem/

[3] http://purl.org/NET/c4dm/event.owl

[4] http://linkedevents.org/ontology/

[5] https://www.w3.org/TR/owl-time/

to refer to changes in the world (`Fluent`) resulting from an event, but also states of affairs that describe the content of mental states.

Other approaches, such as Affordance Ontology Design Pattern (Asprino et al., 2016), prefer to utilize the concept of the "task" as something that is executed by `Action` and performed by an `Agent`. Such a model is useful when it is necessary to model *affordances* (a concept introduced by cognitive psychologist Gibson (1977)), that is opportunities or possible actions offered by the environment in which the agent operates. Tasks are then provided by these affordances, framed by situations dependent on spatio-temporal location of the agent (Toyoshima and Barton, 2018).

The class `Agent` has one specific Boolean data property: `self`. This corresponds to the function $*$ and the sort Self of $\mathcal{DCEC}^*$. The ontology may model a large number of different agents – only one of which is the reasoning agent, and this property is used to represent this knowledge. Usually this property can hold the value `true` for only one agent.

There are also several additional classes that have auxiliary functions. The class `ActionType` represents types of actions and provides an elementary classification for individual actions of the class `Action`, which is also used in conjunction with descriptions of future actions. Obligations, permissions, prohibitions and intentions are generally characterized with regards to `Action` classified by some `ActionType`, which makes it possible to evaluate, for instance, whether the real action performed by the agent is in accordance with his obligation.

The class `Payoff` is an auxiliary class that explicitly represents the payoff (using a numeric data property `payoffValue`) of an `Action` of the type represented by the class `ActionType` at given time `Instant`. It can be used to represent knowledge about the evaluation of the profitability of alternative actions by an agent. The payoff of action may fluctuate in time; the same action performed at different times may have different payoffs. Likewise, a similar action done at the same moment but performed by a different agent may also have a different payoff value. Explicitly defining payoff values is an alternative to defining desires (see Section 5.3). These techniques may be used independently.

The axioms of the event section of the ontology are the following:

$$\mathtt{Instant} \sqsubseteq (\exists \mathtt{before.Instant}) \tag{1}$$

$$\begin{aligned}\mathtt{Holds} \sqsubseteq\ & (\exists \mathtt{whatFluent.Fluent}) \sqcap (\exists \mathtt{truthValue}.\top) \\ & \sqcap (\exists \mathtt{holdsFrom.Instant} \sqcup \exists \mathtt{holdsTo.Instant})\end{aligned} \tag{2}$$

$$\mathtt{Payoff} \sqsubseteq (\exists \mathtt{who.Agent}) \sqcap (\exists \mathtt{ofAction.Action}) \sqcap (\exists \mathtt{when.Instant}) \tag{3}$$

$$\mathtt{Event} \sqsubseteq (\exists \mathtt{when.Instant}) \sqcap (\exists \mathtt{results.Holds}) \tag{4}$$

$$\mathtt{Action} \sqsubseteq (\exists \mathtt{who.Agent}) \tag{5}$$

Note that class `Holds` requires only one of the object time specifying properties – either `holdsFrom` or `holdsTo`. This enables, for example, action to result in initiating (or terminating) a state of affairs without specifying how long such a state will hold.

We integrated some concepts of $\mathcal{DCEC}^*$ calculus into the design of the ontology so they are not expressed explicitly as concepts or properties but implicitly using its different features. Concepts Initiation and Termination that can be associated with fluents in $\mathcal{DCEC}^*$ calculus are in the ontology represented by relating concept `Action` (or generally `Event`) to concept `Fluent` by property `results`. So if an action or event initiates a fluent, then in the ontology the result of the `Event` is that `Fluent Holds`, while not specifying the time limit (`holdsTo`) of this state of affairs (see the code on page 505).

We also do not use the property Initially to define whether a fluent holds at time 0. We concluded that there is no epistemic justification for its use in the ontology. If it is generally known that a fluent has held from the beginning of the time interval that we are interested in modeling, then it should be represented using the `CommonKnowledge` concept. Otherwise, it is a knowledge available only to some agents, and it should be represented using the `Knowledge` concept. Both of these concepts are described in the next section.

5.2. The epistemic section of the ontology

The epistemic section of the ontology is based on $\mathcal{DCEC}^*$ calculus and epistemic logic (Ditmarsch et al., 2015; Rescher, 2005; Hintikka, 2005). We use standard notation of epistemic logic and interpret Kxp as "x knows that p" (similarly, Bxp in the case of belief). In most cases, knowledge and belief have the same formal structure (therefore, we abstain from repeating the discussion). There is one exception: common knowledge does not have its doxastic (i.e. related to belief) counterpart. The question whether knowledge entails belief, i.e. $Kxp \rightarrow Bxp$, is a philosophical problem – it is normal to define knowledge as a kind of belief (e.g. justified belief); however, some authors consider the problem more complex, such as Armstrong (1969). The discussion gets more complicated when we consider non-human agents: does the fact that the bear knows that water is to the north imply that he believes that? We decided to go with the prevailing view that knowledge entails belief, but we will provide an alternative version of ontology (DCEO^{kb}) that lacks these axioms in future.

The epistemic section of the ontology consists of several classes that describe the epistemic aspects of an artificial agent, starting with the acquisition of new knowledge. Classes `Perception` and `Communication` describe how an agent obtains new information: either directly from the environment or by communication from other agents.

There may also be public information available to an agent, described by the class `CommonKnowledge`. Epistemic logic (Ditmarsch et al., 2015) also uses the term *common knowledge*; however, Rescher (2005), for one, speaks about *obvious knowledge*. In both cases, the meaning is the same; p is common knowledge iff $(\forall x)Kxp$. We decided not to use the concept of "patent knowledge" described by the same author; that would require including a specific individual called, for example, s (as for "system") and asserting that p is obvious knowledge iff Ksp. For practical reasons, we preferred to model common knowledge as a class.

Information received by communication with other agents or by perception from the environment may produce belief or knowledge, discussed, for example, by Millar (2015). In the case of simple artificial agents, perceiving may even be something as elementary as measuring the temperature of the environment. Two classes form the epistemic core of the ontology: `Knowledge` and `Belief`, which represent two fundamental epistemic attitudes. These classes are connected to the knowing/believing/perceiving/communicating `Agent` using the property `who`. The epistemic content of the agent's attitude is determined by the object property `what` connected to the classes `Holds`. The time when it happens is determined by the property `when` connected to class `Instant`. The class `Communication` has one more property `communicationTo` that connects the reified communication with the information receiving agent.

The following code is an example of the agent `a1` believing at time instant `i1` that in the interval from `i2` to `i3`, the fluent `f1` will hold.

```
ex:b1  rdf:type  dceo:Belief ;
    dceo:who          ex:a1 ;
    dceo:when         ex:i1 ;
    dceo:what         ex:h1 .

ex:h1  rdf:type  dceo:Holds ;
    dceo:holdsFrom    ex:i2 ;
    dceo:holdsTo      ex:i3 ;
    dceo:whatFluent   ex:f1 .
```

```
ex:a1  rdf:type  dceo:Agent   .
ex:i1  rdf:type  time:Instant  .
ex:i2  rdf:type  time:Instant  .
ex:i3  rdf:type  time:Instant  .
ex:f1  rdf:type  dceo:Fluent   .
```

The class `CommonKnowledge` represents knowledge, defined by the property `what`, available at a given time, specified by the property `when` to all agents and considered to be true. It is similar to the epistemic states described above, but it lacks the object property `who`, because it is known by all agents.

We have considered an intuitive modeling approach where `CommonKnowledge` is formally a kind of `Knowledge`. All members of this class then have to be known by all agents. This could be achieved by following SWRL rule. The antecedent of this rule is the conjunction of two atoms, variables are treated as universally quantified, with their scope limited to a given rule.

$$\texttt{CommonKnowledge}(?c1) \land \texttt{Agent}(?a1) \rightarrow \texttt{Knowledge}(?c1) \land \texttt{who}(?c1, ?a1) \tag{6}$$

In some application scenarios, there may be no difference between perception and belief or even knowledge; i.e. everything that is perceived is believed by the agent or is considered to be a known fact. However, as described above, although everything that is perceived by the agent may be believed, it does not imply that everything that is believed may have its origin in perception. Some statements believed by the agent may originate in communication with other agents (Dretske, 1981).

The class `Communication` represents the cases of information transfer between agents. It describes the communication of some content given by the property `what` at some time defined by the property `when` by an `Agent` described by the property `who` to another `Agent` – specified by the object property `communicationTo`.

We consider it to be a matter of the application to handle incoming communication similarly, as we do not prescribe the processing of data in the case of perception. Whether perceived (or communicated) data are believed automatically by an agent[6] depends on, from our point of view, the epistemic approach chosen by the application (it can be handled by adding the appropriate SWRL rules).

Note that our approach enables the expression of some more advanced constructs available in epistemic logic. It is possible to express *metaknowledge*: $Kx(Kyp)$ – "x knows that y knows that p". In this case an agent x knows that the fluent f holds and this fluent refers to a named graph expressing that an agent y knows that the fluent p holds. Usually, a model uses only "depth-two" knowledge, although it is also possible to model deeper levels. Ditmarsch et al. (2015) assumes that common knowledge is characterized by general metaknowledge. If p is commonly known, it not only means that $(\forall x)Kxp$ but also that $(\forall x)(\forall y)KxKyp$ – "everybody knows that everybody knows that p". While our approach enables us to model such general metaknowledge, it is rarely needed by real applications, so we consider it optional.[7]

---

[6] Dretske (1969) disagrees with this, because perceived data may be product of e.g., optical illusion.

[7] It is also possible to express what Rescher (2005) calls *secret*: $Kxp \land Kx(\neg Kyp)$ – "x knows that p and also knows that y does not know that p".

The axioms of the epistemic section of the ontology are as follows:

$$\begin{aligned}\mathtt{Communication} \sqsubseteq\ & (\exists \mathtt{communicationTo.Agent}) \sqcap (\exists \mathtt{who.Agent}) \\ & \sqcap (\exists \mathtt{when.Instant}) \sqcap (\exists \mathtt{what.Holds})\end{aligned} \tag{7}$$

$$\mathtt{Belief} \sqsubseteq (\exists \mathtt{who.Agent}) \sqcap (\exists \mathtt{when.Instant}) \sqcap (\exists \mathtt{what.Holds}) \tag{8}$$

$$\mathtt{Perception} \sqsubseteq (\exists \mathtt{who.Agent}) \sqcap (\exists \mathtt{when.Instant}) \sqcap (\exists \mathtt{what.Holds}) \tag{9}$$

$$\mathtt{CommonKnowledge} \sqsubseteq (\exists \mathtt{when.Instant}) \sqcap (\exists \mathtt{what.Holds}) \tag{10}$$

5.3. The protential section of the ontology

The protential section of the ontology focuses on modeling the future. The protential model is based on a $\mathcal{DCEC}^*$ extended in accordance with Hobbesian psychology, a theory that understands all motives of actions, including emotions, as reducible to desires and aversions. If there is no desire or aversion, there is no reason for any action (Gert, 1996).

These concepts are modeled by classes `Desire` and `Aversion`, representing more general future-oriented stances of an agent. Objects of desires and aversions are states of affairs. An agent may desire or have aversion to some state of affairs that may occur in the future (e.g. the temperature in the room will be higher than 40°C). The concept of aversion may be considered a syntactic sugar; it depends on whether the statement having aversion towards ϕ means desiring $\neg\phi$ and vice versa are accepted.

There is also the class `Intention` that represents a more specific concept of intending an action in the future. The object of the intention is an `Action` that is to be done in some foreseeable future. There can be a connection between desires and aversions on the one hand and intentions on the other. If an agent desires some state of affairs to be obtained at time `t2`, it may be meaningful to intend to make some type of action before that, at the time `t1` (e.g. executing some process). Our ontology, however, only provides a means to represent such knowledge; its content has to be created and reflected by the agent.

The class `Intention` represents an intention of an `Agent` (connected by the property `who`) to perform an `Action` (connected by property `toAction`) of the type determined by the class `ActionType`. There are two time specifications related to the intention: the object property `when` determines the time when the intention itself takes place, and the object property `intendsWhen` that determines the time when the intended action should take place. We do not currently allow modeling cases when an agent intends to do something without intending to do it at a specific time. Modeling such cases can be allowed by modifying axiom (11) – removing its last section and thus the requirement to state the time when the intended action should take place by specifying property `intendsWhen`.

While classes `Desire` and `Aversion` are in some sense similar to `Intention`, they do not directly comprise any action of the agent. They relate to an agent `who` desires or has aversion. They also relate to some state of affairs that is desired or aversed through the object property `what` and class `Holds`. There is again time specification related to these mental states: the object property `when` determines the time when they themselves take place. Some states of affairs may be aversed at specific times, while at other times they may be desirable (e.g. sun shades should be extended during day but not during the night). This is described by the temporal localization of the class `Holds`.

The axioms of the protential section of the ontology are as following:

$$\begin{aligned}\mathtt{Intention} \sqsubseteq\ & (\exists \mathtt{toAction.Action}) \sqcap (\exists \mathtt{who.Agent}) \sqcap (\exists \mathtt{when.Instant}) \\ & \sqcap (\exists \mathtt{intendsWhen.Instant})\end{aligned} \tag{11}$$

$$\mathtt{Desire} \sqsubseteq (\exists \mathtt{what.Holds}) \sqcap (\exists \mathtt{who.Agent}) \sqcap (\exists \mathtt{when.Instant}) \tag{12}$$

$$\mathtt{Aversion} \sqsubseteq (\exists \mathtt{what.Holds}) \sqcap (\exists \mathtt{who.Agent}) \sqcap (\exists \mathtt{when.Instant}) \tag{13}$$

5.4. *The deontic section of the ontology*

The deontic section of the ontology aims to model the deontic normative knowledge of an agent and therefore deals with an agent's knowledge of his obligations and prohibitions. A few other studies have tried to model similar conceptual contexts. For example, the Document Acts Ontology (d-acts) is an OWL ontology by Almeida et al. (2012) and Brochhausen et al. (2013) that represents social acts that create new entities relevant to social life. D-acts ontology uses the term *declaration* to signify a social act that creates rights and obligations. Declarations that use documents that are signed or stamped are called *document acts*. The ontology is based on Documents Acts Theory by Smith (2012) and inspired by the philosophical work of Reinach (2013) that captures long-lasting responsibilities within an institution. D-acts ontology is linked to Basic Formal Ontology (BFO) by Arp et al. (2015) and reuses some concepts from Information Artefact Ontology (IAO). For our purposes, what is important to note here is that the d-acts ontology represents those acts that document acts theory delineates as Social Generically Dependent Continuants (SGDCs). Accordingly, deontic entities are modeled as socio-legal SGDCs (see also Almeida and Brochhausen (2017); Almeida et al. (2018)).

Ontology developed by Donohue (2017) focuses on formally representing deontic entities and their relationships in the biomedical context such as a health-care professional's obligations to her patients, a patient's claim to information requisite for consent, etc. However, it is argued that such an ontology would also be useful in other domains of interest (e.g., legal knowledge bases or military doctrines and intelligence). Donohue's work is based on the above-mentioned work by Almeida et al. (2012) and his ontology is also based on BFO foundational ontology. Deontic concepts like obligation are categorized as a species of the class Directive Information Entity of BFO-based IAO.

Another related BFO-aligned ontology – Informed Consent Ontology (ICO) by Lin et al. (2014) – also originates in the biomedical context and also reuses some concepts from IAO and describes document acts. However, this ontology is very specific and focuses only on concepts related to informed consent. Rudimentary Requirement Ontology[8] was developed by Nowara as part of Decision Ontology (see Blomqvist et al. (2012) for more information). While there were efforts to integrate it with the BFO foundational ontology, this project was not completed. All these works, however, focus primarily on modeling deontic concepts as legal constructs, usually in the biomedical context.

The deontic model in DCEO is based on $\mathcal{DCEC}^*$ extended in accordance with deontic logic (Gabbay et al., 2013). We use O for obligatory and I for impermissible/prohibited and interpret Oxp as "p is obligatory for x" (similarly for I). In some deontic systems, obligations are linked only to propositions, not to subjects, because they are considered to be universal: Op as "p is obligatory". However, we consider scenarios with heterogeneous agents that may have different obligations and prohibitions so we understand these deontic operators as binary.

Deontic logic also defines the relation between obligation, permission and prohibition. Prohibition is equivalent to an obligation to *abstain* from an action and may be formally defined using obligation: $Ixp \leftrightarrow Ox\neg p$. However, modeling obligation to abstain from an action directly would be difficult because the obligation is linked to class `Action` and it would be necessary to create some auxiliary

---

[8] https://code.google.com/archive/p/requirement-ontology/

passive `ActionType`, e.g. "waiting", to achieve such goal. One can abstain from an action also by doing something else, so such solution would not be proper. Similarly a permission may be also defined as not having an obligation to not perform the action. A permission may be therefore formally defined using an obligation: $Pxp \leftrightarrow \neg Ox\neg p$. These concepts may be considered a syntactic sugar, but due to the nature of the ontology, we will see that modeling all these constructs directly is useful.

The obligation is modeled by the class `Obligation`, prohibition using the class `Prohibition` and permission using the class `Permission`. These classes are similar to class `Intention` since we consider obligations, prohibitions and permissions structurally similar to intentions. The difference is that in the case of intention, the reason for an action is the internal motive of the agent, while in the case of obligation and prohibition, the restrictions are usually based on some given deontic limits external to the agent or inherent to the cognitive makeup of the agent. However, it is worth noting that there are other differences between intention and deontic concepts (an intention does not need a social context, whereas deontic entities usually do; intentions have a different relation to beliefs and desires than deontic concepts, etc.) that are not captured by this ontology and may require a different modeling approach and a complex discussion. The modeling approach presented was chosen because it is adequate for the intended use of the ontology, as it facilitates the process for applications to check whether the agent's intended actions are under the obligations and prohibitions of the agent.

Formally these classes – `Obligation`, `Prohibition` and `Permission` – relate, at some point of time (`when`), to an individual (`who`) is obliged/prohibited to do some `Action` of an `ActionType` in a current or future time point (`obligedWhen`, `permittedWhen`, `prohibitedWhen`). These deontic concepts may be alternatively related to some (obligatory, prohibited, permitted) state of affairs by relation `what`.

The deontic concepts (e.g., obligations) can be modelled in different ways. One way of modeling obligations is to suppose that we know all relevant obligations. Then we can represent agents' actions and monitor whether they are in compliance with their obligations. We make no assumptions whether agents know or do not know their obligations. The other way of modeling obligations is to model them with reference to an individual agent, who is mentally aware of them as specific behavioral limits, applicable to his actions – $(\forall x)Oxp \rightarrow KxOxp$). The obligation is always an obligation to act somehow. The presented approach enables modeling obligations in both ways.

Axioms of the deontic section of the ontology follow:

$$\mathtt{Obligation} \sqsubseteq (\exists \mathtt{who.Agent}) \sqcap (\exists \mathtt{when.Instant}) \sqcap (\exists \mathtt{obligedWhen.Instant}) \tag{14}$$

$$\mathtt{Obligation} \sqsubseteq \big((\exists \mathtt{toAction.Action}) \sqcup (\exists \mathtt{what.Holds})\big) \tag{15}$$

$$\begin{aligned}\mathtt{Prohibition} \sqsubseteq{} & (\exists \mathtt{who.Agent}) \sqcap (\exists \mathtt{when.Instant}) \\ & \sqcap (\exists \mathtt{prohibitedWhen.Instant})\end{aligned} \tag{16}$$

$$\mathtt{Prohibition} \sqsubseteq \big((\exists \mathtt{toAction.Action}) \sqcup (\exists \mathtt{what.Holds})\big) \tag{17}$$

$$\begin{aligned}\mathtt{Permission} \sqsubseteq{} & (\exists \mathtt{who.Agent}) \sqcap (\exists \mathtt{when.Instant}) \\ & \sqcap (\exists \mathtt{permittedWhen.Instant})\end{aligned} \tag{18}$$

$$\mathtt{Permission} \sqsubseteq \big((\exists \mathtt{toAction.Action}) \sqcup (\exists \mathtt{what.Holds})\big) \tag{19}$$

6. Evaluation of the ontology

Formally-defined ontology evaluation that uses proper methodology is an important topic – ontology designers need a way to guide the *evaluation phase* of the process of ontology development and to evaluate the resulting ontology. Different ontology evaluation techniques are summarized by Brank et al. (2005). We base our evaluation of the ontology on the On-To-Knowledge methodology developed by Sure et al. (2009) and Staab et al. (2001) that we discussed in Section 2.

It is important to note that the DCEO ontology is not meant to describe some large thematic area using hundreds of concepts. Therefore, it is not possible to evaluate the ontology at a lexical or vocabulary level by measuring its similarity to a collection of similar ontologies or to some gold standard ontology (Maedche and Staab, 2002) Moreover, in our case, it was also impossible to use a body of natural-language text or those techniques proposed by Velardi et al. (2005) (on a lexical level) or by Brewser et al. (2004) (on a taxonomic level) to measure the degree of some kind of fit between an ontology and a corpus of documents.

Evaluation at the taxonomic level using a formal technique anchored in philosophically-important notions (e.g., essentiality, rigidity, or unity) such as OntoClean (Guarino and Welty, 2009) is also impossible because it primarily evaluates hierarchical structures of concept subsumptions, a feature that our ontology lacks. The lack of taxonomical structure can be considered a limitation of the ontology because a hierarchical taxonomy is frequently considered to be the backbone of an ontology.

In accordance with On-To-Knowledge methodology we started with a technology-focused evaluation that checked syntax and semantics. More specifically, we validated the ontology's language conformity (syntax) to ensure that the ontology was fully compliant with the OWL standard (the standard OWL syntax validator[9] was therefore used to evaluate the validity of the syntax). We also performed consistency (semantics) tests using Hermit and Pellet reasoner plug-ins in Protegé editor (Musen, 2015).[10]

The next stage of our evaluation consisted of checking whether or not the ontology satisfied the ontology requirement specifications and whether or not the ontology supported the solving of problems analyzed in the kickoff phase of the project (we discuss this in detail in the following Sections 6.1–6.3). In this stage we also tested the ontology in the environment of the case-study (discussed in detail in the Section 7). We then used the results of this testing as well as feedback from involved domain experts to refine our ontology. The outcome of this phase (after two iterations) was an evaluated ontology – v. 1.2 of the DCEO ontology.

The ontology was further refined when it was used (i.e., during the *application and evolution phase*) and these refinements were followed with similarly-structured evaluations. These refinement-evaluation cycles produced updated versions of the ontology – v. 2.0 (2018) and v. 2.1 (described in this paper). As part of evaluation a case study described in Section 7 was performed.

6.1. Comparison with requirements for the modeling of an artificial agents

We discuss now whether the requirements stated in Section 2.1 are satisfied with our proposed modeling of the artificial agent action.

The core of the ontology provides a representation of events, agents' actions and time instants. The design, based on named graphs, permits complex descriptions of the content of these events and agents'

---

[9] https://www.w3.org/2001/sw/wiki/OWL_Validator

[10] https://protege.stanford.edu/

actions using a chosen domain ontology. By providing these features the ontology satisfies Requirement 1a, defined in Section 2.1.

The proposed ontology enables modeling of communication between agents, such as sending or receiving data messages. It is also possible to model common knowledge available to all agents and considered to be generally trustworthy. Another way of acquiring information that can be represented using concepts available in the ontology is the perception of the environment by the agent, e.g., using his own sensory equipment.

The ontology also enables distinguishing between information that can be trusted and that has a lower level of epistemic value. Employing an epistemic mentalistic model together with some ideas based on epistemic logic produced a flexible representation of concepts of knowledge and belief understood as mental states of the agent. An alternative model for specific scenarios where knowledge does not entail belief is available. By providing an epistemic mentalistic model the ontology satisfies Requirement 1b, defined in Section 2.1. The ontology also makes it possible to represent the content of mental states of agents and thus satisfies Requirement 2a, defined in the same section. Because the content of mental states is represented in the form of named graphs, the content described may be very complex, so it satisfies also Requirement 2b. The proposed ontology also enables modeling of different agents believing different statements, satisfying Requirement 2c.

DCEO ontology makes it possible to represent knowledge related to future actions: the knowledge about desired states of affairs that are preferred by the artificial agent, but also of aversions related to states of affairs that are to be avoided. Agent's intentions represent actions that the agent plans to perform in the future. Desires and aversions are understood as mental states and complex representation of their content is possible. Intention on the contrary simply refers to an action that is planned for the future. By providing a protential model the ontology satisfies Requirement 1c, defined in Section 2.1.

Limitations of agents' interactions are captured by the deontic part of the ontology consisting of classes representing obligations, permissions and prohibitions. It is, therefore, also possible to model that the agent intends (or desires) to do something, however it is prohibited. Resolving these contradictions has to be done by the application. Obligations, permissions and prohibitions refer to types of actions to be performed in the future. By providing a deontic mentalistic model the ontology satisfies Requirement 1d, defined in Section 2.1.

6.2. Comparison with requirements for the modeling different types of agent interaction scenarios

An abundance of possible scenarios exist that include artificial agents, such as those involving autonomous vehicles (see Section 7), autonomous industrial transport robots and many others. These different scenarios involving different types of artificial agent interactions, can be formally classified using distinctions introduced in Section 2.2, alongside corresponding requirements. We discuss now whether our proposed modeling of the artificial agent action satisfies these ontology requirements.

The first distinction that was introduced divided agents' interactions into cooperative and non-cooperative. Cooperative agents work continuously toward a single given goal and are usually controlled or deployed by a single party. However, agents who are not controlled by a single party may also exhibit cooperative behavior, depending on the context. In other scenarios agents classified as non-cooperative have different and sometimes inconsistent goals. Some agents may be non-cooperative even to the extent that they provide intentionally misleading information or intentionally prevent other agents from attaining their goals. The epistemic mentalistic model used in the proposed ontology makes it possible to represent both cooperative and non-cooperative agents' interactions and satisfies Requirement 3, defined in Section 2.2.

The second distinction that was introduced, divided agents into heterogeneous or homogenous. Homogenous agents are agents which are similar in terms of which types of information they accept, process and provide. They may still use different internal architectures, be controlled by different parties and seek different goals. Heterogeneous agents process different kinds of information and while they may not be able to communicate directly, they may, e.g., perceive each other's behavior. The proposed model makes it possible to model scenarios involving both homogenous and heterogeneous agents and satisfies Requirement 4, defined in Section 2.2.

6.3. Comparison with general requirements of ontology design

We discuss now whether the general requirements of ontology design, stated in Section 2.3 correlate with our proposed ontology.

The designed ontology uses paradigms known to the AI community. The architecture of the ontology framework is inspired by the Soar cognitive architecture (Laird et al., 1987) and the classical concept of the Model Human Processor (Newell et al., 1998; Card et al., 1983), which, in turn, is based on the Standard Model of Human Cognition (Simon and Kaplan, 1998; Klahr and MacWhinney, 1998). The ontology itself was inspired by Deontic Cognitive Event Calculus – $\mathcal{DCEC}^*$ (Bringsjord and Govindarajulu, 2013; Bringsjord et al., 2014a) as well as epistemic and deontic logic. The proposed model reuses existing knowledge approaches of the AI community and therefore, satisfies Requirement 5, defined in Section 2.3.

While designing the model, we tried to abstain from using unnecessary complex constructs by leveraging common-sense intuition and scholarly discussions about epistemic and deontic logic. Using $\mathcal{DCEC}^*$ calculus as a starting point, should further reduce the steepness of the learning curve for some users in the AI community. Still, the ontology is very expressive and its architecture enables its use as a model for many different scenarios involving artificial agents, who occupy different roles. Formally, the current expressiveness of the ontology is $\mathcal{ALFC}(\mathcal{D})^*$. The proposed ontology, therefore, satisfies Requirement 6, defined in Section 2.3.

The focus of the ontology is well-defined and the internal structure of the ontology is clearly described. The separation of concerns principle helps keep the complexity of the ontology manageable and the structure of the ontology understandable. The event, epistemic, protential and deontic sections of the ontology are clearly separated and their relations defined. The proposed ontology, therefore, satisfies Requirement 7, defined in Section 2.3.

7. A case study – enhancing autonomous vehicle navigation

It has been argued that autonomous vehicles must have an internal representation of entities, events and situations in the world, as well as a mechanism for computing values and priorities that enable them to determine their next action (Albus et al., 2002, 196). There have already been some efforts to use Semantic Web technologies to enhance the performance of autonomous vehicles. Schlenoff et al. (2003) explored the possibility of using ontologies to improve route planning in autonomous vehicles in the context of the 4D/RCS system architecture developed at NIST. Follow-up research by Provine et al. (2004) used an ontology to support reasoning in relation to obstacles as well as to improve route planning.

It is generally recognized that, "in its full generality, the problem of automated vehicle navigation is extremely challenging" (Schlenoff et al., 2003). This makes it necessary to split the problem into different sub-problems and related components that may be researched independently. Architecture that deals with the problem of automated vehicle navigation using Semantic Web technologies must at least include the following components (Russell and Norvig, 2016, 1004):

(1) *Sensor interface*: captures the environment surrounding the autonomous vehicle.
(2) *Perception*: a low-level media analysis provides a base analysis, transformation, and description of the captured audio-visual data; object recognition provides information about objects surrounding the autonomous vehicle and produces a 3D world model of the current state of affairs.
(3) *Traffic environment ontology*: semantically describes the states of affairs (world models) in the 3D world model and the necessary services that produce and maintain this ontology based on the 3D world model.
(4) *Agent Ontology (DCEO)*: describes agents, their actions, and events as well as communication between agents, agents' mental states, and the deontic status of the states of affairs.
(5) *Future possible states of affairs generator*: generates possible future states of affairs from the current state of affairs using its knowledge of physics and of the characteristics of involved entities (e.g., people, vehicles, roads, etc.)
(6) *Value judgment component*: evaluates different states of affairs and assigns them deontic evaluations.
(7) *Decision component*: judges available information to determine the next course of action.
(8) *Vehicle interface*: transfers control commands back to the vehicle.
(9) *User interface*: handles communication with the human user or operator.

Low level media analysis (1, 2) is out of the scope of this research and, moreover, several research communities are already dealing with these problem areas (Rosique et al., 2019; Leonard et al., 2008). Similarly, we do not discuss the vehicle interface (8) or the user interface (9) layers (Russell and Norvig, 2016, 1004).

Meanwhile, at the level of semantic description (3, 4) it is possible to distinguish between ontology that describes the states of affairs in a traffic environment (3) and ontology that describes agents, agents' mental states, communication, events, and actions as well as the deontic status of these events (4).

It is necessary to use a specialized ontology to describe the state of affairs in a traffic environment. For our purposes, it is important to note here that because autonomous vehicles usually move continuously, the situation on the road that the ontology must evaluate continuously changes. To semantically describe the state of affairs we must fix descriptions of the separate snapshots captured during the vehicle's continuous stream. These snapshots must meet at an adequate level of granularity. Similarly, descriptions of future possible states of affairs (5) should be produced at the same level of granularity. Traffic ontologies already exist that describe traffic environments at such required levels of detail and granularity such as that developed by Bagschik et al. (2018) or that preliminary version of ontology developed by Zhao et al. (2015). These ontologies, however, only focus on describing the scene.

It is therefore useful to complement traffic ontologies with the DCEO ontology this paper presents. Agents (both artificial and human) participating in traffic have various intentions (e.g., an agent intends to turn right) and beliefs (e.g., one agent believes another agent will not turn right) and communicate with each other (e.g., one agent signals to another that they intend to turn right). The autonomous vehicles that are likely to be produced in accordance with common standards in the future will probably communicate with each other about their intentions using specialized protocols. However, this communication may be mistaken (especially in the case of human agent) or intentionally misleading (as in the case of the scenario described by Bello et al. (2015), in which an evil cyber-hacker infects an autonomous robot with a virus). Notably, separating the model into two ontologies allows for the separation of concerns: while traffic environment ontologies are purely descriptive, DCEO can model the normative characteristics of states of affairs.

The whole architecture must include some more sophisticated functions in addition to low-level services: it should also generate and describe possible future states of affairs (5). Such a capability requires a description of the current state of affairs as well as knowledge about physics and about the characteristics of entities involved in the decision (e.g., people, vehicles, roads, etc.). The component of value judgment (6) evaluates the given description of the state of affairs with regard to laws, traffic rules, ethical principles, etc. by categorizing these items into the deontic classes of "prohibited," "permitted," or "obligatory." Finally, there must be a decision component (7) that reasons using available information and decides the next course of action. Components 3 to 7 constitute what is usually called the "planning and control layer."

The following section zooms in on the details involved in integrating DCEO into the architecture of autonomous vehicles.

7.1. An example: A cross-road incident

Figure 4 depicts an agent/autonomous vehicle `a1` in a decision situation. Scene `s0` describes the current state of affairs provided by sensory arrays and associated services (in terms of a traffic environment ontology) to which the fluent `f0` refers. This scene is described in the concepts of traffic ontology in the named graph, referred to by its URI `s0-uri`. The vehicle `a1` arriving (previous action `act0`) at the crossroad intends to continue straight forward to reach its goal marked by star. However, the vehicle identified the presence of a pedestrian on the road ahead of it (agent `a2`).

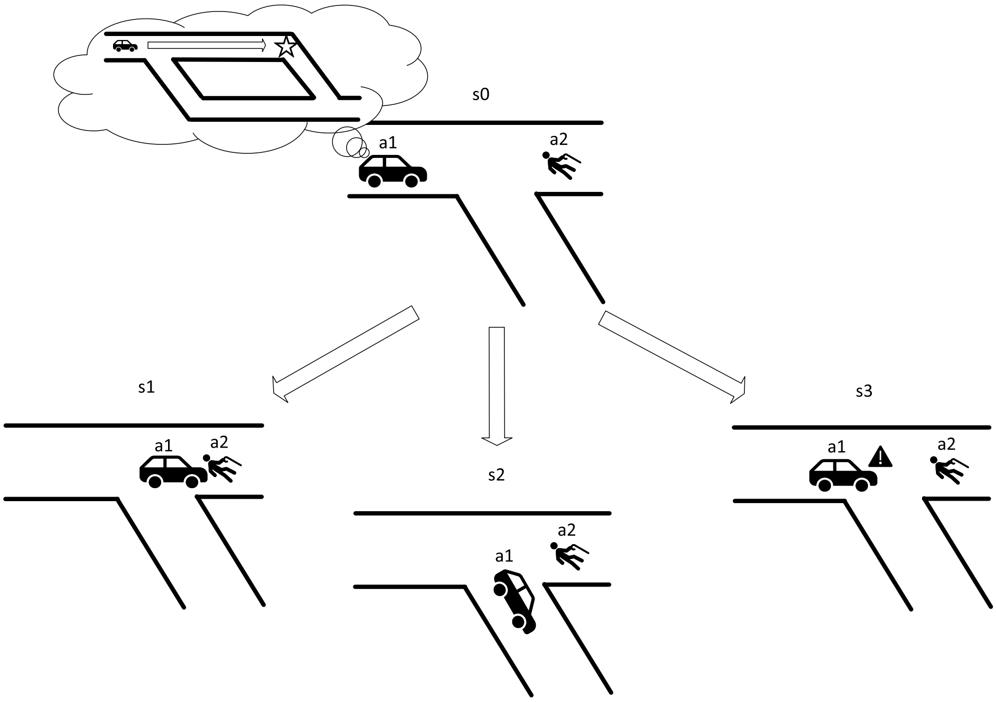

Fig. 4. An example of autonomous vehicle route planning.

```
ex:act0  rdf:type  dceo:Action ;
    dceo:results         ex:h0 ;
    dceo:when            ex:i1 ;
    dceo:who             ex:a1 .

ex:h0  rdf:type  dceo:Holds ;
    dceo:holdsFrom       ex:i2 ;
    dceo:holdsTo         ex:i3 ;
    dceo:whatFluent      ex:f0 .
```

```
ex:f0  rdf:type  dceo:Fluent ;
    dceo:refersTo        ex:s0-uri .

ex:a1  rdf:type  dceo:Agent  .
ex:a2  rdf:type  dceo:Agent  .
ex:i1  rdf:type  time:Instant  .
ex:i2  rdf:type  time:Instant  .
ex:i3  rdf:type  time:Instant  .
```

In this scene `s0` there are three possible actions (`act1, act2, act3`) and three possible resulting scenes (`s1, s2, s3`) determined by the *Future possible states of affairs generator* using its knowledge of physics and of the characteristics of involved the vehicle, road, etc. The vehicle could go straight ahead, turn right, or stop and wait.

7.1.1. Action 1

The first possible action `act1` of `ActionType goStraight` involves the vehicle continuing straight ahead without stopping or turning. This action has the highest `Payoff` value (100) because it is the shortest route to destination, i.e., to the final scene `sF` *desired* by the vehicle (represented in the traffic ontology named graph referred to by its URI `sF-uri`). However, because a pedestrian (agent `a2`) is standing in the middle of the road, this action would result in the vehicle hitting the pedestrian. Evaluated together with the resulting scene `s1`, this action was determined as *prohibited* by the *Value judgment component*. The resulting scene is represented in the traffic ontology named graph referred to by its URI `s1-uri`.

```
ex:act1  rdf:type  dceo:Action ;
    dceo:results         ex:h1 ;
    dceo:when            ex:i4 ;
    dceo:who             ex:a1 .

ex:h1  rdf:type  dceo:Holds ;
    dceo:holdsFrom       ex:i5 ;
    dceo:holdsTo         ex:i6 ;
    dceo:whatFluent      ex:f1 .

ex:f1  rdf:type  dceo:Fluent ;
    dceo:refersTo        ex:s1-uri .

ex:p1  rdf:type  dceo:Payoff ;
    dceo:ofAction        ex:act1 ;
    dceo:payoffValue     int:100 .

ex:proh1  rdf:type  dceo:Prohibition ;
    dceo:toAction        ex:act1 ;
    dceo:when            ex:i4 ;
    dceo:prohibitedWhen ex:i4 ;
    dceo:what            ex:h1 ;
    dceo:who             ex:a1 .
```

```
ex:d1  rdf:type  dceo:Desire ;
    dceo:when            ex:i1 ;
    dceo:what            ex:hF ;
    dceo:who             ex:a1 .

ex:hF  rdf:type  dceo:Holds ;
    dceo:holdsFrom       ex:if1 ;
    dceo:holdsTo         ex:if2 ;
    dceo:whatFluent      ex:fF .

ex:fF  rdf:type  dceo:Fluent ;
    dceo:refersTo        ex:sF-uri .

ex:i4  rdf:type  time:Instant  .
ex:i5  rdf:type  time:Instant  .
ex:i6  rdf:type  time:Instant  .
```

7.1.2. Action 2

The second possible action `act2` of `ActionType` `turnRight` involves the vehicle turning right. This action has a lower `Payoff` value (60) because this route to the destination is a bit longer than going straight. Because there are no known problems with taking this route, this action and its resulting scene `s2` were evaluated as *permitted* by the *Value judgment component*. The resulting scene is represented in the traffic ontology named graph referred to by its URI `s2-uri`.

```
ex:act2  rdf:type  dceo:Action ;
    dceo:results         ex:h2 ;
    dceo:when            ex:i4 ;
    dceo:who             ex:a1 .

ex:h2  rdf:type  dceo:Holds ;
    dceo:holdsFrom       ex:i5 ;
    dceo:holdsTo         ex:i6 ;
    dceo:whatFluent      ex:f2 .

ex:f2  rdf:type  dceo:Fluent ;
    dceo:refersTo        ex:s2-uri .
```

```
ex:p2  rdf:type  dceo:Payoff ;
    dceo:ofAction        ex:act2 ;
    dceo:payoffValue     int:60 .

ex:perm1  rdf:type  dceo:Permission ;
    dceo:toAction        ex:act2 ;
    dceo:when            ex:i4 ;
    dceo:permittedWhen ex:i4 ;
    dceo:what            ex:h2 ;
    dceo:who             ex:a1 .
```

7.1.3. Action 3

The third possible action `act3` of `ActionType` `stop` involves the vehicle stopping. This action has a lower `Payoff` value (40) because it causes some delay. Because there are no known problems with stopping, this action and its resulting scene `s3` were evaluated as *permitted* by the *Value judgment component*. The resulting scene is represented in the traffic ontology named graph referred to by its URI `s3-uri`.

```
ex:act3  rdf:type  dceo:Action ;
    dceo:results         ex:h3 ;
    dceo:when            ex:i4 ;
    dceo:who             ex:a1 .

ex:h3  rdf:type  dceo:Holds ;
    dceo:holdsFrom       ex:i5 ;
    dceo:holdsTo         ex:i6 ;
    dceo:whatFluent      ex:f3 .

ex:f3  rdf:type  dceo:Fluent ;
    dceo:refersTo        ex:s3-uri .
```

```
ex:p3  rdf:type  dceo:Payoff ;
    dceo:ofAction        ex:act3 ;
    dceo:payoffValue     int:40 .

ex:perm2  rdf:type  dceo:Permission ;
    dceo:toAction        ex:act3 ;
    dceo:when            ex:i4 ;
    dceo:permittedWhen ex:i4 ;
    dceo:what            ex:h3 ;
    dceo:who             ex:a1 .
```

The payoff value of stopping is, in this example, lower than that of turning right, because if the vehicle turns right it will reach the destination with some estimated (short) delay, while if it stops, the delay cannot be estimated precisely, so this is considered a worse option. In a real-world situation, the payoff values may be different.

7.1.4. Mentalistic considerations related to planning an action

What vehicle `a1` will do may depend on what it believes (i.e., predicts) the pedestrian `a2` will do. The pedestrian is likely to perform one of two actions: they can either move away from the road (action

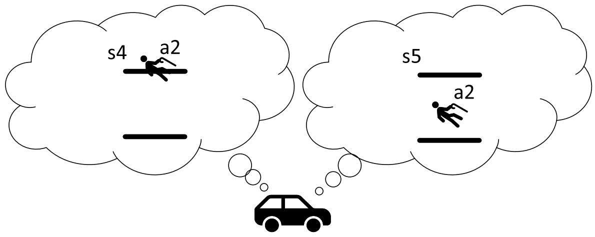

Fig. 5. Considerations related to planning an action.

`act4`, scene `s4`) or stay in middle of the road (action `act5`, scene `s5`, code is omitted) – see Fig. 5. Let's say that vehicle `a1` believes that pedestrian `a2` intends to move and therefore perform action `act4` and yield scene `s4` described by the named graph refered to by URI `s4-uri`:

```
ex:b1  rdf:type  dceo:Belief ;
    dceo:what           ex:h4 ;
    dceo:when           ex:i7 ;
    dceo:who            ex:a1 .
```

```
ex:h4  rdf:type  dceo:Holds ;
    dceo:holdsFrom      ex:i8 ;
    dceo:holdsTo        ex:i9 ;
    dceo:whatFluent     ex:f4 .

ex:f4  rdf:type  dceo:Fluent ;
    dceo:refersTo       ex:s4-uri .
```

Similarly, we can model that vehicle `a1` believes that pedestrian `a2` intends to stay in the middle of the road and therefore perform action `act5` and yield scene `s5` described by the named graph referred to by `s5-uri` (we omit the code here because it is similar to the previous one). In this situation where the pedestrian is not moving, the vehicle may identify them as a possibly injured person, potentially making the action of `callAmbulance` *obligatory*. These cases can also be modeled using the presented formalism.

7.1.5. Modeling communication between agents

Pedestrian `a2` may communicate (`c1`) to vehicle `a1` to make clear what his/her intentions are. The content of the communication is in the named graph referred to by URI `s6-uri`.

```
ex:c1  rdf:type  dceo:Communication ;
    dceo:what           ex:h6 ;
    dceo:when           ex:i9 ;
    dceo:who            ex:a2 ;
    dceo:communicationTo  ex:a1 .

ex:h6  rdf:type  dceo:Holds ;
    dceo:holdsFrom      ex:i10 ;
    dceo:holdsTo        ex:i11 ;
    dceo:whatFluent     ex:f6 .
```

```
ex:f6  rdf:type  dceo:Fluent ;
    dceo:refersTo       ex:s6-uri .

ex:i9  rdf:type  time:Instant  .
ex:i10  rdf:type  time:Instant  .
ex:i11  rdf:type  time:Instant  .
```

The content of the named graph `s6-uri` situates pedestrian `a2` as intending to move away from road, that is to enact `act4` and yield scene `s4` described by the named graph referred to by URI `s4-uri`. More specifically, the content of the named graph `s6-uri`:

```
ex:in1  rdf:type  dceo:Intention ;
    dceo:what           ex:h4 ;
    dceo:toAction       ex:act4 ;
    dceo:when           ex:i7 ;
    dceo:who            ex:a2 .
```

```
ex:h4  rdf:type  dceo:Holds ;
    dceo:holdsFrom      ex:i8 ;
    dceo:holdsTo        ex:i9 ;
    dceo:whatFluent     ex:f4 .

ex:f4  rdf:type  dceo:Fluent ;
    dceo:refersTo       ex:s4-uri .
```

More complex mentalistic models may be modeled using similar constructs. For example, we could model that the autonomous vehicle `a1` believes that pedestrian `a2` believes that the autonomous vehicle `a1` will turn right. Because this model is similar to those presented above, we omit it for sake of brevity.

This approach to modeling has some similarities to reification. Using named graphs has some advantages over simple reification, as described by Carroll et al. (2005b). There are, on the other hand, more complex approaches to reification, based on work by Davidson (1967), the advantages of which are demonstrated, e.g., by Robaldo and Sun (2017).

7.2. *Other applications of presented formalism*

DCEO is applicable in a number of different scenarios that involve interactions between artificial and human agents. Existing literature details a number of scenarios that might use $\mathcal{DCEC}^*$, similar scenarios might as well make use of DCEO as a modeling tool with the advantage of being integrated with Semantic Web technologies. For example, Bringsjord et al. (2014b) analyzes interactions between soldier robots in relation to more general ethical problems such as *akrasia* (lack of will) or vengefulness. Moreover, Bringsjord et al. (2014a) also discusses the use of $\mathcal{DCEC}^*$ in modeling agents' beliefs in a nuclear deterrence scenario.

We have already noted that DCEO can be used to model both artificial and human agents. We are currently investigating the possibility of using the presented formalism in a case study based on another project in real conditions to explore its useful features. The testing environment is provided by follow-up research based on a project that focused on the extraction of a structured knowledge from large amounts of multimedia content recorded over networks of cameras and microphones deployed in real sites such as the surveillance networks of the subways in Rome and Turin (Carincotte et al., 2008; Smrž et al., 2006). This case study is based on a task related to subway station monitoring: four cameras are installed in the station – two in the corridor and one on each platform. The system also involves a microphone array that records the primary level of ambient noise in different areas. The purpose of the project was to ease end-user missions (subway monitoring by safety/security operators). We believe that DCEO could be used to describe the actions and intentions of human agents in such a subway environment and to produce descriptions of their mental models. Such a use may help to identify non-standard behaviors by agents such as vandalism, theft, etc.

8. Possible future extensions

The presented ontology models obligations, permissions and prohibitions only as related to individual agents. Every obligation is an obligation of concrete individual artificial agent and it is not relevant to any other agent. Replication of obligations and prohibitions between agents is currently possible using a simple SWRL rule. However, we intend to investigate possible extension of the ontology involving general obligations and general prohibitions valid for all agents: $Op \rightarrow (\forall)x\,Oxp$.

Another possible research direction involves the relation between obligations, permissions, prohibitions and time. As Ajani et al. (2017) observes, many kinds of norms constantly evolve (e.g., legal norms) – the previous versions of the norms continue to be valid in the specific, previous time period, even if these old norms are currently no longer valid. Therefore an agent that guides his current actions generally with regard to his past actions and their results, should not take these old actions into account (or take them into account only with appropriate corrections), because old norms that limited past actions are no longer valid. For instance, in the past, the fastest way for an autonomous vehicle to go from point A to point B was to pass through street S. However, since a new prohibition (a strict speed limit in street S) has been introduced, it may now be quicker to use a different route.

The current version of the ontology also lacks a time-dependent representation of cooperativeness (it is not possible to represent agents being cooperative at time t1 and non-cooperative at time t2). This is a limit of the ontology and the future versions may provide a way to represent changes in cooperativeness of agents.

Other future research opportunities are linked to laws and other legally binding norms that involve a common range of prohibitions and obligations. Although modeling complex legal norms is a challenging task (Griffo et al., 2018), there are already legal knowledge management systems that use Semantic Web technologies, such as Eunomos (Boella et al., 2016). These systems, however, are only informative and the knowledge they provide has been prepared by human experts, who indexed the legal documents. Also, users of these systems are human lawyers, who consume this information in enriched hypertext form. We may imagine the evolution of systems like Eunomos that would contain formal representations of legal regulations in machine readable form. An artificial lawyer agent using the DCEO ontology could then check if the state of affairs (e.g., of his company) is compliant with these regulations. The user interface in this case can be realized in the form of a chatbot, enabling the artificial agent to interact with the human lawyer (Kluwer, 2011; MacTear et al., 2016). The human lawyer can address issues in his/her native language, the chatbot automatically formalizes these responses in DCEO items and may go back to the lawyer if the information is insufficient, until he obtains a consistent and complete DCEO representation, that can be checked against the formal representation of legal constraints. This scenario, however, requires further development and integration of natural language processing tools, such as those integrated in the Eunomos system (Boella et al., 2012, 2013).

There are also plans for releasing modules consisting of different sets of SWRL rules for different kinds of scenarios. These modules will, at first, be built manually and will be available to users of DCEO as starting points for developing an ontology-based infrastructure that suits their needs. Taking one of these modules, we plan to transform a set of SWRL rules from DEO that performs sophisticated event-related reasoning (Mepham, 2010). Another set of rules for applications involving legal agents may be based on LegalRuleML, a specific standard for representing content of legal text, that is built on RuleML and is fully compatible with SWRL (Athan et al., 2013, 2015).

Another planned extension concerns contents of mental states of agents. Currently, we model them independently and our ontology does not capture relations between them. That is sufficient for many typical applications because the content of mental states is usually handled by the internal logic of the application itself. However, in some cases, it may be useful to be able to represent some basic relations between these mental states: they may be mutually exclusive, one may be a subset of another, etc.

A related research opportunity concerns the question of how to achieve a high-level communication among agents and systems, that includes semantically meaningful content. Ferrario and Prévot (2007) identify several steps to fulfill this objective. It would require a closer look at agent communication languages, which understand speech acts as operators with preconditions and effects, discussed by Boella et al. (2007), who proposed an ontology of communication primitives, based on public mental attitudes, attributed to role instances. This ontology allows for the construction of artificial agents participating in a range of dialogues, without having to redefine existing communication protocols. An especially challenging scenario involves the communication of agents in heterogeneous multi-agent systems as described by Van Diggelen et al. (2007).

There is also an optional extension involving deontic and epistemic categories. Accordingly, it may be useful for some applications to introduce a level of fuzziness or specificity into these categories – different obligations may be obligatory in different ways or to different degrees. In legal contexts, there may be different legal interpretations – it is common that norms in legislation may be interpreted in different ways, some of them with inconsistencies. Interpretations from some authoritative sources (such as high courts) may clarify any contradictions, however, that may take time (Bartolini et al., 2016). To handle interpretative uncertainty a pattern-based approach can be used (Vacura et al., 2008).

The related question is also how to encompass defeasibility in the DCEO ontology. A framework supporting defeasible reasoning should be able to represent defeasible (non-strict) facts, i.e., facts that allow exceptions. This, however, introduces the problem of non-monotonicity, which is not encompassed in OWL-DL (Casini et al., 2015). To deal with defeasibility in legislation, reified Input/Output logic has been introduced by Robaldo and Sun (2017). For instance, a legal ontology for modeling GDPR concepts and norms, ascertained by Palmirani et al. (2018), was used to build a knowledge base described by Bartolini et al. (2016), that uses reified I/O logic and LegalRuleML.

We also plan to include a specific relation to capture the reasons behind intentions and actions. We usually say that an agent "performed" an action or that an agent "intends to perform" an action `because` of a particular belief, perception, desire, aversion, permission, obligation, or prohibition. Modeling this relation would make it possible to describe the reasons why an action was chosen and realized.

At last, it is also necessary to note that it is not only important to design ontology well, but also its performance has an impact on its adoption. In the real-world applications that need to perform in real time, the speed of reasoning is of prime importance. The current expressiveness of the ontology is $\mathcal{ALFC}(\mathcal{D})^*$. It is sufficiently fast for most of the reasoning tasks performed. However, in our tests the most of scenarios did not require any complex reasoning. The complexity of the ontology may rise, especially after adding proposed extensions. Such an increase in complexity could lead to significant reasoning-time overhead. For us, it is an important task to watch the performance of the ontology and keep it at a usable level.

9. Conclusions

Solving the problem of an autonomous action of artificial agents is indispensable to progress in many areas of artificial intelligence. The research dealing with this issue, in the context of the Semantic Web is very limited and our project presents an effort to bridge the gap between Semantic Web technologies and AI research. While many existing systems focus on a single-agent scenario and differ only in the way in which the agent approaches its environment, our approach makes it possible to model how artificial agents interact with other artificial agents, or with humans. An artificial agent may be an autonomous vehicle on the road, interacting with other vehicles, an industrial transportation robot that must maneuver a complex environment, or a legal chatbot. All these artificial agents may interact with other artificial or human agents, some of which may have the same goals. Others, however, may have different goals or may even try to obstruct or harm other agents.

We have presented Deontic Cognitive Event Ontology (DCEO) inspired by the $\mathcal{DCEC}^*$ calculus, epistemic and deontic logics. We also presented a brief overview of our cognitive architecture that is inspired by the Soar cognitive architecture and several others and includes the proposed ontology as its core, enabling effective cognitive knowledge representation on top of it.

The DCEO ontology described in this paper consists of four parts: event ontology, epistemic ontology, protential ontology and deontic ontology. Event ontology allows modeling of actions of artificial agents, occurring at specific times or intervals. Epistemic ontology describes the mental states of these agents: belief, knowledge and perception and their content. It also enables the modeling of communication between agents and that of common knowledge available to all agents. The proposed protential ontology models the agents' attitudes to the future – their desires, aversions and intentions. This, in turn, influences their autonomous actions. The deontic ontology aims at modeling obligations and prohibitions – the limits of artificial agents' actions.

We have also described a case study in which the proposed DCEO ontology supports autonomous vehicle navigation. We have argued that the DCEO ontology enables autonomous vehicles to have an internal representation of entities, events and situations in the world, as well as representation of knowledge and beliefs of different agents and limitations, based on various obligations and prohibitions, therefore, it is the DCEO ontology, that makes it possible to determine their next action.

Finally, we proposed some possible enhancements and extensions of the ontology. We also noted the number of future research opportunities. We believe that the DCEO ontology provides a modeling framework that can be successfully used in many different areas involving artificial agents, from industrial robots to artificial legal advisers.

Acknowledgement

This work has been supported by CSF 18-23964S.

References

Ajani, G., Boella, G., Di Caro, L., Robaldo, L., Humphreys, L., Praduroux, S., Rossi, P. & Violato, A. (2017). The European legal taxonomy syllabus: A multi-lingual, multi-level ontology framework to untangle the web of European legal terminology. *Applied Ontology*, *11*(4), 1–51. doi:10.3233/AO-170174.

Albus, J.S., et al. (2002). 4D/RCS Version 2.0: A reference model architecture for unmanned vehicle systems. NIST Interagency/Internal Report (NISTIR) – 6910, National Institute of Standards and Technology, Gaithersburg, MD.

Almeida, M.B. & Brochhausen, M. (2017). An ontological approach to the normative dimension of organizations: An application of documents acts ontology. *Ciência da Informação*, *2017*(1). doi:10.18225/ci.inf.v46i1.4024.

Almeida, M.B., Pessanha, C.P. & Barcelos, R. (2018). Information architecture for organizations: An ontological approach. In C. Thomas (Ed.), *Ontology in Information Science*, Rijeka: IntechOpen. doi:10.5772/intechopen.69161.

Almeida, M.B., Slaughter, L. & Brochhausen, M. (2012). Towards an ontology of document acts: Introducing a document act template for healthcare. In P. Herrero, H. Panetto, R. Meersman and T. Dillon (Eds.), *On the Move to Meaningful Internet Systems: OTM 2012 Workshops*. Lecture Notes in Computer Science (Vol. 7567). Berlin, Heidelberg: Springer.

Arkoudas, K. & Bringsjord, S. (2009). Propositional attitudes and causation. *Int. J. Software and Informatics*, *3*(1), 47–65. http://kryten.mm.rpi.edu/PRICAI_w_sequentcalc_041709.pdf.

Armstrong, D.M. (1969). Does knowledge entail belief? *Proceedings of the Aristotelian Society, New Series*, *70*(1969–1970), 21–36).

Arp, R., Smith, B. & Spear, A.D. (2015). *Building Ontologies with Basic Formal Ontology*. Cambridge MA: MIT Press.

Asprino, L., Nuzzolese, A.G., Russo, A., Gangemi, A., Presutti, V. & Nolfi, S. (2016). An ontology design pattern for supporting behaviour arbitration in cognitive agents. In *Proceedings of the 7th Workshop on Ontology and Semantic Web Patterns Co-Located with ISWC 2016 (WOP 2016)*, Kobe, Japan.

Athan, T., Boley, H., Governatori, G., Palmirani, M., Paschke, A. & Wyner, A. (2013). LegalRuleML: From metamodel to use cases. In L. Morgenstern et al. (Eds.), *Theory, Practice, and Applications of Rules on the Web (RuleML 2013)*, Berlin, Heidelberg: Springer.

Athan, T., Governatori, G., Palmirani, M., Paschke, A. & Wyner, A. (2015). LegalRuleML: Design principles and foundations. In W. Faber and A. Paschke (Eds.), *Reasoning Web. Web Logic Rules*, Berlin: Springer. doi:10.1007/978-3-319-21768-0-6.

Bagschik, G., Menzel, T. & Maurer, M. (2018). Ontology based scene creation for the development of automated vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)* (pp. 1813–1820). IEEE. doi:10.1109/IVS.2018.8500632.

Bartolini, C., Lenzini, G. & Robaldo, L. (2016). Towards legal compliance by correlating standards and laws with a semi-automated methodology. In *Proceedings of the 28th Annual Benelux Conference on Artificial Intelligence (BNAIC2016)*, Amsterdam, The Netherlands.

Barton, A., Duncan, W., Toyoshima, F. & Ethier, J.-F. (2018). First steps towards an ontology of belief. In L. Jansen, D.P. Radicioni and D. Gromann (Eds.), *Proceedings of the Joint Ontology Workshops 2018. Episode IV: The South African Spring, Co-Located with the 10th International Conference on Formal Ontology in Information Systems (FOIS 2018)*, Cape Town, South Africa, September 17–18, 2018. Aachen: CEUR-WS. ISSN 1613-0073. http://ceur-ws.org/Vol-2205/paper17_epinon1.pdf.

Bello, P., Licato, J. & Bringsjord, S. (2015). Constraints on freely chosen action for moral robots: Consciousness and control. In *24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)* (pp. 505–510). IEEE.

Blomqvist, E., Waters, J., McGarry, D., Wheeler, A. & Nowara, P. (2012). Towards a semantic decision representation format. Final report of the Decisions and Decision-Making Incubator Group. W3C Incubator Group Report 17 April 2012. https://www.w3.org/2005/Incubator/decision/XGR-decision-20120417/.

Boella, G., Damiano, R., Hulstijn, J. & van der Torre, L.W.N. (2007). A common ontology of agent communication languages: Modeling mental attitudes and social commitments using roles. *Applied Ontology*, *2*(3–4), 217–265.

Boella, G., Di Caro, L., Humphreys, L., Robaldo, L., Rossi, P. & van der Torre, L. (2016). Eunomos, a legal document and knowledge management system for the Web to provide relevant, reliable and up-to-date information on the law. *Artificial Intelligence and Law*, *24*(3).

Boella, G., Di Caro, L., Humphreys, L., Robaldo, L. & van der Torre, L. (2012). NLP challenges for Eunomos a tool to build and manage legal knowledge. In *Proceedings of Lexical Resources and Evaluation Conference (LREC 2012)* (pp. 3672–3678). European Language Resources Association (ELRA).

Boella, G., Di Caro, L., Rispoli, D. & Robaldo, L. (2013). A system for classifying multi-label text into EuroVoc. In *Proceedings of 14th International Conference on Artificial Intelligence and Law (ICAIL2013)*, Rome, Italy.

Brank, J., Grobelnik, M. & Mladenic, D. (2005). A survey of ontology evaluation techniques. In *Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005)*, Ljubljana, Slovenia (pp. 166–170).

Brewser, C., Alani, H., Dasmahapatra, S. & Wilks, Y. (2004). Data driven ontology evaluation. In *Proceedings of International Conference on Language Resources and Evaluation*, Lisbon.

Bringsjord, S. & Govindarajulu, N.S. (2013). Deontic cognitive event calculus (formal specification). Rensselaer Artificial Intelligence and Reasoning Laboratory. http://www.cs.rpi.edu/~govinn/dcec.pdf.

Bringsjord, S., Govindarajulu, N.S., Ellis, S., McCarty, E. & Licato, J. (2014a). Nuclear deterrence and the logic of deliberative mindreading. *Cognitive Systems Research*, 28, 20–43). http://www.sciencedirect.com/science/article/pii/S1389041713000491.

Bringsjord, S., Govindarajulu, N.S., Thero, D. & Si, M. (2014b). Akratic robots and the computational logic thereof. In *Ethics in Science, Technology and Engineering, 2014 IEEE International Symposium on* (pp. 1–8). IEEE.

Bringsjord, S., Licato, J., Govindarajulu, N.S., Ghosh, R. & Sen, A. (2015). Real robots that pass human tests of self-consciousness. In *Robot and Human Interactive Communication (RO-MAN 2015), 24th IEEE International Symposium on* (pp. 498–504). IEEE.

Brochhausen, M., Almeida, M.A. & Slaughter, L. (2013). Towards a formal representation of document acts and the resulting legal entities. In *Johanssonian Investigations* (pp. 120–139). Frankfurt: Ontos.

Card, S., Moran, T.P. & Newell, A. (1983). *The Psychology of Human–Computer Interaction*. Hillsdale, New Jersey: Erlbaum.

Carincotte, C., Desurmont, X. & Bastide, A. (2008). Adaptive metadata management system for distributed video content analysis. In *Proceedings of the 10th International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS'08)*, Juan-les-Pins, France (pp. 20–24).

Carroll, J.J., et al. (2005a). Named graphs, provenance and trust. In *Proceedings of the 14th International Conference on World Wide Web*. ACM.

Carroll, J.J., Bizer, Ch., Hayes, P. & Stickler, P. (2005b). Named graphs. *Journal of Web Semantics*, *3*(4), 247–267. doi:10.1016/j.websem.2005.09.001.

Casini, G., Meyer, T.A., Moodley, K., Sattler, U. & Varzinczak, I.J. (2015). Introducing defeasibility into OWL ontologies. In M. Arenas et al. (Eds.), *International Semantic Web Conference (ISWC 2015)*, Bethlehem, PA, USA, October 11–15, 2015 (pp. 409–426).

Castañeda, H.-N. (1999). "He": A study in the logic of self-consciousness. In J.G. Hart and T. Kapitan (Eds.), *The Phenomeno-Logic of the I: Essays on Self-Consciousness*, Bloomington, Indiana, USA: Indiana University Press.

Davidson, D. (1967). The logical form of action sentences. In N. Rescher (Ed.), *The Logic of Decision and Action*. University of Pittsburgh Press.

Ditmarsch, H., Halpern, J.Y., Hoek, W. & Kooi, B. (2015). An introduction to logics of knowledge and belief. In H. van Ditmarsch et al. (Eds.), *Handbook of Epistemic Logic*, London: College Publications.

Donohue, B. (2017). Toward a BFO-based deontic ontology. In *The 2nd Workshop on Representing Social and Legal Entities in the Biomedical Domain (SoLe-BD 2017). Proceedings of the 8th International Conference on Biomedical Ontology (ICBO 2017)*, Newcastle-upon-Tyne, United Kingdom, September 13th–15th, 2017.

Dretske, F. (1969). *Seeing and Knowing*. Chicago: University Of Chicago Press.

Dretske, F. (1981). *Knowledge and the Flow of Information*. Boston: MIT Press.

Fernández-López, M., Gómez-Pérez, A. & Juristo, N. (1997). Methontology: From ontological art towards ontological engineering. AAAI Technical Report, SS-97-06, Association for the Advancement of Artificial Intelligence (AAAI).

Ferrario, R. & Prévot, L. (2007). Formal ontologies for communicating agents. *Applied Ontology*, *2*(3–4), 209–216.

Gabbay, D., Horty, J. & Parent, X. (Eds.) (2013). *Handbook of Deontic Logic and Normative Systems*. London: College Publications.

Gert, B. (1996). Hobbes's psychology. In T. Sorell (Ed.), *The Cambridge Companion to Hobbes* (pp. 157–174). Cambridge, Massachusetts: Cambridge University Press. doi:10.1017/CCOL0521410193.008.

Gibson, J. (1977). The theory of affordances. In R. Shaw and J. Bransford (Eds.), *Perceiving, Acting, and Knowing: Toward an Ecological Psychology* (pp. 67–82). Hillsdale, New Jersey: Lawrence Erlbaum Associates Publishers.

Goble, L. (2003). Preference semantics for deontic logic. Part I: Simple models. *Logique et Analyse*, *46*, 183–184.

Gómez-Pérez, A., Fernández-López, M. & Corcho, O. (2003). *Ontological Engineering*. Advanced Information and Knowledge Processing Series. Berlin: Springer. ISBN 1-85233-551-3.

Griffo, C., Almeida, J.P.A. & Guizzardi, G. (2018). Conceptual modeling of legal relations. In *International Conference on Conceptual Modeling* (pp. 169–183). Cham: Springer. doi:10.1007/978-3-030-00847-5_14.

Guarino, N. & Welty, C. (2009). An overview of OntoClean. In S. Staab and R. Studer (Eds.), *The Handbook on Ontologies* (pp. 151–172). Berlin: Springer.

Guizzardi, G. (2005). Ontological foundations for structural conceptual models. University of Twente, Telematica Instituut / CTIT.

Guizzardi, G., Wagner, G., De Almeida Falbo, R., Guizzardi, R. & Almeida, J.P.A. (2013). Towards ontological foundations for the conceptual modeling of events. In *Conceptual Modeling (ER2013)*. Lecture Notes in Computer Science (Vol. 8217). Berlin: Springer.

Hanzal, T., Svátek, V. & Vacura, M. (2016). Event categories on the semantic web and their relationship/object distinction. In *9th International Conference on Formal Ontology in Information Systems (FOIS 2016)*, Annecy, France, July 6–9, 2016.

Hintikka, J. (2005). *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. London: King's College Publications.

Jarrar, M. & Ceusters, W. (2017). Classifying processes and basic formal ontology. In *The 8th International Conference on Biomedical Ontology (ICBO 2017)*, Newcaste upon Tyune, UK.

Jaśkowski, S. (1934). On the rules of suppositions in formal logic. *Studia Logica*, *1*, 5–32.

Kang, S.H. & Lau, S.K. (2004). Ontology revision using the concept of belief revision. In *Proceedings, Part III*, Knowledge-Based Intelligent Information and Engineering Systems: 8th International Conference (KES 2004), Wellington, New Zealand, September 20–25, 2004 (pp. 8–15). Berlin: Springer.

Klahr, D. & MacWhinney, B. (1998). Information processing. In D. Kuhn and R.S. Siegler (Eds.), *Handbook of Child Psychology, Vol. 2: Cognition, Perception and Language*, New York: John Wiley and Sons Inc.

Kluwer, T. (2011). From chatbots to dialogue systems. In D. Perez-Marton and I. Pascual-Nieto (Eds.), *Conversational Agents and Natural Language Interaction: Techniques and Effective Practices* (7, pp. 1–22). IGI Global Publishing Group.

Kowalski, R.A. (1992). Database updates in the event calculus. *Journal of Logic Programming*, *12*, 121–146. doi:10.1016/0743-1066(92)90041-Z.

Kowalski, R.A. & Sergot, M.J. (1986). A logic-based calculus of events. *New Generation Computing*, *4*, 67–95).

Krisnadhi, A.A. & Hitzler, P. (2016). A core pattern for events. In *Proceedings of Workshop on Ontology and Semantic Web Patterns (WOP 2016), ISWC 2016*, Kobe, Japan.

Laird, J.E., Newell, A. & Rosenbloom, P.S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, *33*(1), 1–64. doi:10.1016/0004-3702(87)90050-6.

Leonard, J., et al. (2008). A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, *25*(10), 727–774. doi:10.1002/rob.20262.

Lin, Y., Harris, M.R., Manion, F.J., Eisenhauer, E., Zhao, B., Shi, W., Karnovsky, A. & He, Y. (2014). Development of a BFO-based informed consent ontology (ICO). In *Proceedings of the 5th International Conference on Biomedical Ontology*, Houston.

Lindsay, P.H. & Norman, D.A. (1977). *Human Information Processing* (2nd ed.). New York: Academic Press.

MacTear, M., Callejas, Z. & Griol, D. (2016). *The Conversational Interface: Talking to Smart Devices*. Berlin: Springer.

Maedche, A. & Staab, S. (2002). Measuring similarity between ontologies. In *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web (EKAW 2002)*. Lecture Notes in Computer Science Book Series (LNCS) (Vol. 2473, pp. 251–263). doi:10.1007/3-540-45810-7_24.

Manzano, M. (1996). *Extensions of First Order Logic*. Cambridge, UK: Cambridge University Press.

Marton, N., Licato, J. & Bringsjord, S. (2015). Creating and reasoning over scene descriptions in a physically realistic simulation. In *Proceedings of the Symposium on Agent-Directed Simulation (ADS 15)*, Alexandria, Virginia. San Diego, CA, USA: Society for Computer Simulation International.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N. & Oltramari, A. (2003). Ontology Library (final). WonderWeb Deliverable D18. The WonderWeb Library of Foundational Ontologies and the DOLCE ontology. Techreport, ISTC-CNR.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A. & Schneider, L. (2002). The WonderWeb Library of Foundational Ontologies and the DOLCE ontology. WonderWeb Deliverable D17. Techreport, ISTC-CNR.

McInerney, P. (1988). What is still valuable in Husserl's analyses of inner time-consciousness. *The Journal of Philosophy*, *85*(11), 605–616. doi:10.5840/jphil198885116.

McNamara, P. (2010). Deontic logic. In E. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Fall 2010th ed.). https://plato.stanford.edu/entries/logic-deontic/.

Mepham, W. (2010). Discrete event calculus using Semantic Web technologies. Unpublished PhD thesis, University of Glamorgan. http://hdl.handle.net/10265/541.

Mepham, W. & Gardner, S. (2009). Implementing discrete event calculus with semantic web technologies. In *Next Generation Web Services Practices (NWESP'09). Fifth International Conference*, Prague (pp. 90–93).

Millar, A. (2015). Reasons for belief, perception, and reflective knowledge. *Aristotelian Society Supplementary Volume*, *88*(1), 1 June 2014, 1–19. doi:10.1111/j.1467-8349.2014.00230.x.

Musen, M.A. (2015). The Protege project: A look back and a look forward. *AI Matters*, *1*(4). Association of Computing Machinery Specific Interest Group in Artificial Intelligence. doi:10.1145/2557001.25757003.

Newell, A., Rosenbloom, P.S. & Laird, J.E. (1998). Symbolic architectures for cognition. In M.I. Posner (Ed.), *Foundations of Cognitive Science* (pp. 93–132). Cambridge, Massachusetts: The MIT Press.

Oberle, D., Lamparter, S., Grimm, S., Vrandecic, D., Staab, S. & Gangemi, A. (2006). Towards ontologies for formalizing modularization and communication in large software systems. *Applied Ontology*, *1*(2), 163–202.

Palmirani, M., Bartolini, C., Martoni, M., Robaldo, L. & Rossi, A. (2018). Legal ontology for modelling GDPR concepts and norms. In *Proceedings of the 31th International Conference on Legal Knowledge and Information Systems (JURIX 2018)*, Groningen.

Pohl, W. (1997). LaboUr – machine learning for usermodeling. In M.J. Smith, G. Salvendy and R.J. Koubek (Eds.), *Design of Computing Systems: Social and Ergonomic Considerations*. Proceedings of the Seventh International Conference on Human–Computer Interaction (Vol. B, pp. 27–30). Amsterdam: Elsevier Science.

Pohl, W. & Achim, N. (1999). Machine learning and knowledge representation in the labour approach to user modeling. In *User Modeling (UM99)*. CISM International Centre for Mechanical Sciences (Vol. 407, pp. 179–188). doi:10.1007/978-3-7091-2490-1_18.

Provine, R., Schlenoff, C., Balakirsky, S., Smith, S. & Uschold, M. (2004). Ontology-based methods for enhancing autonomous vehicle path planning. *Robotics and Autonomous Systems*, *49*, 123–133). doi:10.1016/j.robot.2004.07.020.

Reinach, A. (2013). The apriori foundations of the civil law. Along with the lecture "concerning phenomenology". In J. Crosby (Ed.), *Realistische Phanomenologie / Realist Phenomenology, Book 8*. Ontos Verlag.

Rescher, N. (2005). *Epistemic Logic: A Survey of the Logic of Knowledge*. Pittsburgh, PA: University of Pittsburgh Press.

Robaldo, L. & Sun, X. (2017). Reified input/output logic: Combining input/output logic and reification to represent norms coming from existing legislation. *The Journal of Logic and Computation*, *27*(8).

Rosique, F., Navarro, P.J., Fernández, C. & Padilla, A. (2019). A systematic review of perception system and simulators for autonomous vehicles research. *Sensors*, *19*(3), 648. doi:10.3390/s19030648.

Russell, S.J. & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach*. Harlow, England: Pearson Education Limited.

Schlenoff, C., Balakirsky, S., Uschold, M., Provine, R. & Smith, S. (2003). Using ontologies to aid navigation planning in autonomous vehicles. *The Knowledge Engineering Review*, *18*(3), 243–255. doi:10.1017/S0269888904000050.

Seddig-Raufie, D., et al. (2018). Proposed actions are no actions: Re-modeling an ontology design pattern with a realist top-level ontology. *Journal of Biomedical Semantics 3*(Suppl 2).

Selfridge, O.G. (1959). Pandemonium: A paradigm for learning. In D.V. Blake and A.M. Uttley (Eds.), *Proceedings of the Symposium on Mechanisation of Thought Processes*, London (pp. 511–529).

Shanahan, M. (2001). The event calculus explained. In *Artificial Intelligence Today*. Lecture Notes in Computer Science (Vol. 1600, pp. 409–430). Berlin: Springer.

Simon, H.A. & Kaplan, C.A. (1998). Foundations of cognitive science. In M.I. Posner (Ed.), *Foundations of Cognitive Science* (pp. 1–48). Cambridge, Massachusetts: The MIT Press.

Smith, B. (2012). How to do things with documents. *Rivista di Estetica*, *50*(2), 179–198. doi:10.4000/estetica.1480.

Smith, B. & Ceusters, W. (2015). Aboutness: Towards foundations for the information artifact ontology. In F.M. Couto and J. Hastings (Eds.), *Proceedings of the International Conference on Biomedical Ontology*, Lisbon, Portugal, July 27–30, 2015 (pp. 27–30). Aachen: CEUR-WS. ISSN 1613-0073. http://ceur-ws.org/Vol-1515/regular10.pdf.

Smith, D.W. & McIntyre, R. (1982). *Husserl and Intentionality: A Study of Mind, Meaning, and Language* (Vol. 154). Berlin: Springer Science & Business Media, Springer.

Smrž, P., Vacura, M. & Šváb, O. (2006). Uncertainty extensions to ontologies as a tool for semantic interpretation in audiovisual systems. In Y. Avrithis et al. (Eds.), *Semantic Multimedia, First International Conference on Semantics and Digital Media Technologies (SAMT 2006)* (pp. 27–28). Aachen, Germany: CEUR-WS.org. ISSN 1613-0073, http://ceur-ws.org/Vol-233/p27.pdf.

Sowa, J.F. (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove, CA: Brooks Cole Publishing Co.

Spiker, C.C. (1989). Cognitive psychology: Mentalistic or behavioristic? *Advances in Child Development and Behavior*, *21*, 73–90). Academic Press Inc.

Staab, S., Schnurr, H.P., Studer, R. & Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent Systems*, *16*(1), 26–34).

Suárez-Figueroa, M.C. (2012). *NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse*. IOS Press.

Sure, Y., Staab, S. & Studer, R. (2009). On-to-knowledge methodology (OTKM). In S. Staab and S. Studer (Eds.), *Handbook on Ontologies*, Dordrecht: Springer.

Texor, M. (2016). States of affairs. In E.N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Winter 2016th ed.). https://plato.stanford.edu/archives/win2016/entries/states-of-affairs.

Toyoshima, F. & Barton, A. (2018). A formal representation of affordances as reciprocal dispositions. In O. Kutz and M.M. Hedblom (Eds.), *TriCoLore 2018. Creativity – Cognition – Computation, Joint Proceedings of the Workshops*, Bozen-Bolzano, Italy, December 13–15, 2018. Aachen, Germany: CEUR-WS.org. ISSN 1613-0073. http://ceur-ws.org/Vol-2347/paper9.pdf.

Vacura, M. & Svátek, V. (2016). Towards a deontic cognitive event ontology. In O. Kutz et al. (Eds.), *Proceedings of the Joint Ontology Workshops 2016 Episode 2: The French Summer of Ontology, Co-Located with the 9th International Conference on Formal Ontology in Information Systems (FOIS 2016)*, Annecy, France, July 6–9, 2016. Aachen, Germany: CEUR-WS.org. ISSN 1613-0073. http://ceur-ws.org/Vol-1660/caos-paper3.pdf.

Vacura, M., Svátek, V. & Smrž, P. (2008). A pattern-based framework for uncertainty representation in ontologies. In P. Sojka, A. Horak, I. Kopecek and K. Pala (Eds.), *Text, Speech and Dialogue*. Proceedings of 11th International Conference on Text, Speech and Dialogue (TSD 2008), Brno, Czech Republic, Sep 08–12, 2008 (pp. 227–234). Berlin: Springer. doi:10.1007/978-3-540-87391-4_30.

Van Diggelen, J., Beun, R.-J., Dignum, F., Van Eijk, R.M. & Meyer, J.-J. (2007). Ontology negotiation in heterogeneous multi-agent systems: The ANEMONE system. *Applied Ontology*, *2*(3–4), 267–303.

Velardi, P., Navigli, R., Cucchiarelli, A. & Neri, F. (2005). Evaluation of OntoLearn, a methodology for automatic learning of domain ontologies. In P. Buitelaar et al. (Eds.), *Onttology Learning from Text: Methods, Evaluation and Applications*. IOS Press.

Wittgenstein, L. (1922). *Tractatus Logico-Philosophicus*. C.K. Ogden (Ed.). London: Routledge & Kegan Paul.

Zhao, L., Ichise, R., Mita, S. & Sasaki, Y. (2015). Core ontologies for safe autonomous driving. In *The 14th International Semantic Web Conference, Poster and Demonstrations Track*, Bethlehem, PA, USA, October 11–15, 2015.