Towards a Deontic Cognitive Event Ontology

Miroslav VACURA [a,1] and Vojtěch SVÁTEK [b]
[a] *Department of Philosophy, University of Economics, Prague.*
[b] *Department of Information and Knowledge Engineering, University of Economics, Prague.*

Abstract. Although there have been efforts to integrate Semantic Web technologies and Cognitive Science approaches, they are still relatively isolated. Herein we introduce a new ontology framework supporting representation and reasoning on complex cognitive information using Semantic Web tools. The framework consists of four parts: an event ontology for information about actions and events, an epistemic ontology containing facts about knowledge, belief, perception and communication, an ontology about future intentions and desires, and, finally, a deontic ontology for modeling obligations. The structure of the ontology framework is derived from the syntax of the Deontic Cognitive Event Calculus.

Keywords. electronic camera-ready manuscript, IOS Press, LaTeX, book, layout

1. Introduction

One of the most difficult problems of artificial intelligence is the autonomous action of artificial agents. One of possible approaches to this problem is inspired from common-sense psychology (CSP) and mentalistic models of human behavior. Knowledge representation models are called *mentalistic* if they are related to mental notions like knowledge, belief, goals etc. [23]. Mentalistic models have been used for some time for user modeling in traditional knowledge representation and in some cases combined with machine learning techniques [24]. Contrasting to that, in the Semantic Web context the use of mentalistic models has been mostly limited to utilizing the concept of belief when taking care of changing knowledge [11]. It has been also argued that application of such models is likely to be particularly apt for artificial agents operating in environments where they have to communicate or even influence the behavior of other artificial or human agents [1].

When an artificial agent in such an environment communicates some information, either publicly or privately to another agent, this information cannot be taken at face value because it only represents a belief or can be even intention-

[1]Corresponding Author: vacuram@vse.cz

ally misleading. Such an account of communication requires mentalistic models distinguishing belief from knowledge.

An artificial agent also needs to reason about the consequences of his own intended actions. On the top of that, reasoning about actions of other agents and predicted consequences of such actions is also required. In other words, models enabling such reasoning have to deal with past, current and future actions and generally with events, so flexible handling of time is necessary.

Dealing with future events caused by actions of other agents requires mentalistic models of their internal mental representations of future – their desires and intentions. We call these future-involving mental states *protential*, based on the term 'protention', meaning the consciousness of future and coined by the philosopher E. Husserl [21]. These internal mental states directly influence the external behavior of agents – intended actions materialize and became real actions performed in an external environment.

The last level of mentalistic model contains the representation of obligations. These are limitations of behavior of the reasoning agent himself and of other agents, which can be used for predicting their actions. Obligations require an agent to act (or suspend action) when some defined conditions of the environment are observed. We call the model involving obligations *deontic*, based on similar use of the term in the context of deontic logic [18].

Note that even if we use metalistic models and discus an agent's behavior in mentalist terms, this does not mean 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 above defined context:

- Model of events and actions
- Epistemic mentalistic model
- Protential mentalistic model
- Deontic mentalistic model

As we have said above, there have only been very limited efforts to use mentalistic approach in the Semantic Web context. However there has been a number of approaches to model events in general, on the semantic web; for a most recent overview see [9]. Outside the Semantic Web, a mentalistic approach has been successfully deployed by means of the Deontic Cognitive Event Calculus ($\mathcal{DCEC}^*$), which provided inspiration and a starting point for our approach. However, transforming a calculus to an ontology is not an easy or straightforward task. One such effort to provide an ontology representation of an event calculus was the *Discrete Event Ontology*, for which an OWL[2] ontology, SWRL[3] rules and a resolver have been provided. However, only a simple version of event calculus has been covered, and the resulting ontology only comprised three classes (Events, Fluents, Timepoints) and a couple of rules [19, 20].

The Section 2 of the paper provides a general overview of the architecture of the proposed framework. Section 3 describes the Deontic Cognitive Event Calculus

[2]`https://www.w3.org/TR/owl2-syntax/`
[3]`https://www.w3.org/Submission/SWRL/`

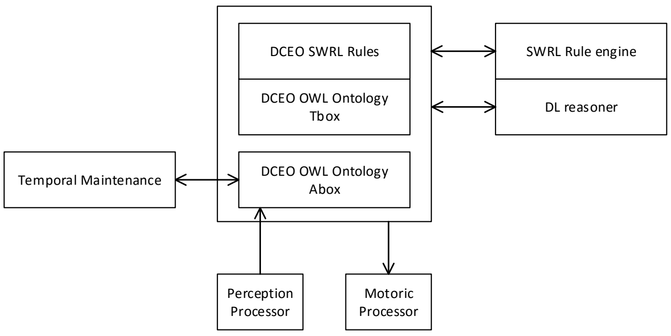

Figure 1. Prospective Application Architecture based on the $\mathcal{DCEO}$

($\mathcal{DCEC}^*$). Section 4 describes the proposed ontology framework itself. Section 5 discusses open problems and possible future extensions of the framework. Finally, the last section provides some conclusions.

2. Architecture and use of the Deontic Cognitive Event Ontology

The overview of a hypothetical application architecture making use of the proposed Deontic Cognitive Event Ontology ($\mathcal{DCEO}$) is depicted in Fig. 1. We have been inspired by the Soar cognitive architecture [15] and the classical concept of the Model Human Processor [6,22], which is in turn based on the Standard Model of human cognition [12,26].

There are however some differences. The core of our architecture consists in the Tbox of an OWL ontology (described in this paper) and a set of SWRL rules. These represent stable knowledge about the world. Information obtained from the environment is transformed by a component that is traditionally called Perception processor and, in our case, produces OWL statements to be included in the Abox of the ontology. On the top of the whole ontology, consisting of a Tbox, Abox and SWRL rules, there is a SWRL rule engine and a DL reasoner, which are responsible for inferential tasks. Another component traditionally called Motoric Processor monitors the current time and retrieves the statements representing actions that are to be performed at the given time point. The last component, which we call Temporal Maintenance, 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 continuous addition of new statements to the Abox combined with axiom production by the SWRL rule engine might be enormous; clearing of old data representing no-longer-useful knowledge would thus be necessary.

While the overall architecture is inspired by the standard Model Human Processor, 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 [17], control of robot behavior [4] or even simulation of some features of human consciousness [5].

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

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

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

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

Figure 2. $\mathcal{DCEC}^*$ Syntax

3. 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 M. Manzano [16]. $\mathcal{DCEC}^*$ 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 formal syntax of $\mathcal{DCEC}^*$ can be found in the original works [2,3].

The proof calculus is based on natural deduction [10] and includes all the introduction and elimination rules of first-order logic as well as rules for modal operators. In this paper we only focus on the ontological representation of $\mathcal{DCEC}^*$, which covers its syntax using the apparatus of OWL, and omit the inferential rules [2], which would mostly have to be expressed using SWRL.

$\mathcal{DCEC}^*$ is based on the Event Calculus (EC), which was first introduced by Kowalski and Sergot as a logic programming formalism for representing events and their effects [13], and later also presented in a simplified version [13]. A detailed presentation of EC can by found in the work of Shanahan [25].

$\mathcal{DCEC}^*$ 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 orders 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 to 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 $\mathit{prior}(t_1, t_2)$ 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 $\mathit{holds}(f, t)$ says that fluent f holds at a given time t (e.g., $f =$ "Peter is student" holds at $t = 01/01/2016$). The expression $\mathit{initially}(f)$ indicates that the fluent f holds from time 0.

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 $\mathit{happens}(e, t)$ thus informs that event e happened in time t (e.g., $e =$ "Peter concluded his studies" at $t = 05/03/2016$). The expression $\mathit{terminates}(e, f, t)$ states that following event e fluent f ceased to hold at 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 $\mathit{initiates}(e, f, t)$ states that after event e fluent f started to hold at time t (e.g., after $e =$ "Peter was inaugurated" at $t = 02/03/2011$, proposition $f =$ "Peter is a student" started to be true). The expression $\mathit{clipped}(t_1, f, t_2)$ 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. 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. 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 work by Goble [8] and McNamara [18].

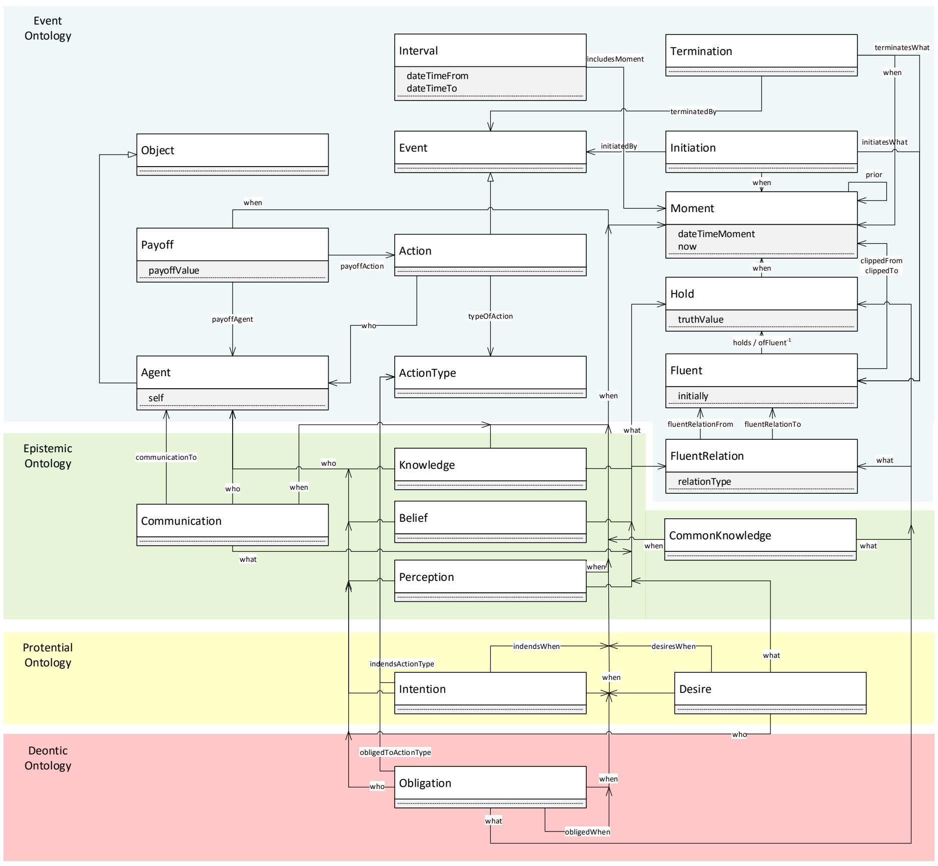

Figure 3. Deontic Cognitive Event Ontology (DCEO)

Finally, the deontic predicate $\mathbf{O}(a, t, \phi, happens(action(a, \alpha), t'))$ 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 [7].

These predicates also require the introduction of the sort $\mathsf{ActionTypes}$ covering the general types of action, and of the function $action(a, \alpha) = b$, 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 $payoff(a, \alpha, t)$ 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 $\mathsf{Numerical}$ sort.

4. Deontic Cognitive Event Ontology

The proposed Deontic Cognitive Event Ontology (DCEO) will be described in detail in this section. Fig. 3 depicts the general overview of the ontology, while Table 1 summarizes the object properties of the ontology and their features (domain,

Table 1. Object properties of the Deontic Cognitive Event Ontology

Relation	Domain	Range	Symm.	Refl.	Trans.	Func.
`when`	D_1[a]	`Moment`	No	No	No	Yes
`what`	D_1	R_1[b]	No	No	No	No[c]
`who`	D_2[d]	`Agent`	No	No	No	No
`includesMoment`	`Interval`	`Moment`	No	No	No	No
`terminatedBy`	`Termination`	`Event`	No	No	No	Yes
`initiatedBy`	`Initiation`	`Event`	No	No	No	Yes
`payoffAction`	`Payoff`	`Action`	No	No	No	Yes
`payoffAgent`	`Payoff`	`Agent`	No	No	No	Yes[e]
`typeOfAction`	`Action`	`ActionType`	No	No	No	Yes[f]
`prior`	`Moment`	`Moment`	No	No	Yes	No
`clippedFrom`	`Fluent`	`Moment`	No	No	No	No[g]
`clippedTo`	`Fluent`	`Moment`	No	No	No	No[h]
`fluentRelationFrom`	`FluentRelation`	`Fluent`	No	No	No	Yes
`fluentRelationTo`	`FluentRelation`	`Fluent`	No	No	No	Yes
`communicationTo`	`Communication`	`Agent`	No	No	No	No[i]
`desiresWhen`	`Desire`	`Moment`	No	No	No	No
`intendsWhen`	`Intention`	`Moment`	No	No	No	No
`obligedWhen`	`Obligation`	`Moment`	No	No	No	No
`intendsActionType`	`Intension`	`ActionType`	No	No	No	No
`obligedToActionType`	`Obligation`	`ActionType`	No	No	No	No

[a]D_1 = `EpistemicState` $\sqcup$ `CommonKnowledge` $\sqcup$ `ProtentialState` $\sqcup$ `Communication` $\sqcup$ `Obligation`
[b]R_1 = `Holds` $\sqcup$ `FluentRelation`
[c]E.g. one event can terminate several fluents.
[d]D_2 = `EpistemicState` $\sqcup$ `ProtentialState` $\sqcup$ `Communication` $\sqcup$ `Obligation`
[e]Actions are generally considered to be performed by individual agents.
[f]We suppose that every action has only one ActionType.
[g]We may have several clipping information items for a single fluent.
[h]Dtto.
[i]A communication can be aimed to several receiving agents.

range, symmetry, reflexivity, transitivity and functionality). Specific axioms for each part of the ontology will be listed at the end of each respective section of the following text. The ontology can be downloaded from our website.[4]

4.1. The event section of the ontology

The foundation of the ontology provides the class `Moment`, which consists of time points ordered by the object property `prior`. The property `prior` has the class `Moment` as both its domain and its range. We have introduced an additional data property `dateTimeMoment`, which explicitly specifies precise the date and time for those moments where it is known.

The function *interval* of the calculus has been reified to the class `Interval` of the ontology, and we introduced two similar data properties as for previous

[4]`https://webhosting.vse.cz/vacuram/ontologies/dceo/`

class, `dateTimeFrom` and `dateTimeTo`, which specify the precise date and time boundaries of an interval. The object property `includesMoment` connects intervals (members of class `Interval`) with moments (members of class `Moment`) that belong to it.

The class `Fluent` represents propositional entities that are subject to change. It has a boolean data property `initially`, which corresponds to the function *initially* of $\mathcal{DCEC}^*$. The class `Hold` represents the reified function *holds* of the calculus. The class `Fluent` is connected to the class `Hold` by the object property `holds`. The class `Hold` is in turn connected to the class `Moment` by the object property `when`, and it also has a data property `truthValue`. This whole structure allows us to represent that a fluent holds a truth value at a given moment. The same fluent can hold truth value `true` at one moment and the truth value `false` at another.

This structure then can become the content of an epistemic state, for example, an agent can believe that a fluent (proposition) has the truth value *false* in the given moment. The data property `truthValue` is assumed to be boolean (truth values represented by 0 or 1) as in $\mathcal{DCEC}^*$; however, using this ontology design it may be easily adapted to dealing with various types of uncertainty.

The class `Event` and its subclass `Action` do not have any data type properties, and together with the class `Agent` they constitute the core of the event-focused section of the ontology. The class `Agent` has one specific boolean data property, `self`. This corresponds to the function * and the sort Self of $\mathcal{DCEC}^*$. The ontology in use may contain and describe number of different agents – only one of them is the reasoning agent himself and this property is used to represent this knowledge. Obviously only one agent can have this property `true`.

We have also introduced three general object properties: `what`, `who` and `when`. These represent various properties of $\mathcal{DCEC}^*$ with common ranges. The range of the object property `what` are classes `Hold` and `FluentRelation`: aside of epistemic states directed towards simple propositional fluents (represented by the ontological structure around the class `Hold` described above) there may be epistemic states about relations between fluents. The class `FluentRelation` has two related object properties `fluentRelationFrom` and `fluentRelationTo`, both connected to fluents, and one data property `fluentRelationType` denoting the specific type of relation between fluents.

The range of the object property `who` is the class `Agent`. The property usually denotes the subject of an epistemic state or the active agent of a communication or of an action.

The range of the object property `when` is the class `Moment`, denoting the time of events, actions, epistemic or protential states throughout the ontology. Note that this property denotes the occasion of the actual occurrence of a protential state: the occasion when, e.g., the intention is mentally present. The protential state also has the second specification of time – the future time when something is intended (e.g., the agent now at time t_1 intends to move to point X later at time t_2). This second time specification is denoted by specific separate object properties (see Sec. 4.3).

The classes `Termination` and `Initiation` represent the functions *terminates* and *initiates* of $\mathcal{DCEC}^*$. They are connected by the object property `when` with the class `Moment`, and by the object properties `terminatesWhat` and `initiatesWhat` with the class `Fluent`, indicating what fluent was terminated/initiated and when. The object properties `terminatedBy` and `initiatedBy` connect these classes with the class `Event`, representing the event that caused initiation or termination of the fluent.

There are also four additional classes fulfilling auxiliary functions. The class `ActionType` represents types of actions and provides classification for individual actions of the class `Action`. It is to be mainly used by higher ontological levels, to theoretize about possible or future actions. The class `Object` (superclass of class `Agent`) is a general class for all objects. Any other objects modeled by the ontology belong to this class.

The class `Payoff` represents the payoff (using a numeric data property `payoffValue`) of an action represented by the class `Action` connected by the data property `payoffAction`. The reification of payoff property to the class `Payoff` was necessary because the payoff value is related to the `Agent`, `Action` and `Moment`. The same action of the same agent done at different moments may have different payoff value. Similarly, the same action done at the same moment but different agents may have also different payoff value.

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

$$\mathtt{Termination} \equiv (\exists \mathtt{terminatedBy.Event}) \sqcap (\exists \mathtt{when.Moment}) \sqcap (\exists \mathtt{what.Fluent}) \tag{1}$$

$$\mathtt{Initiation} \equiv (\exists \mathtt{initiatedBy.Event}) \sqcap (\exists \mathtt{when.Moment}) \sqcap (\exists \mathtt{what.Fluent}) \tag{2}$$

$$\mathtt{Moment} \equiv (\exists \mathtt{prior.Moment}) \tag{3}$$

$$\mathtt{Fluent} \equiv (\exists \mathtt{when.Moment}) \tag{4}$$

$$\mathtt{Payoff} \equiv (\exists \mathtt{payoffAgent.Agent}) \sqcap (\exists \mathtt{payoffAction.Action}) \sqcap (\exists \mathtt{when.Moment}) \tag{5}$$

$$\mathtt{Action} \equiv (\exists \mathtt{who.Agent}) \tag{6}$$

$$\mathsf{Disjoint}(\mathtt{Termination},\mathtt{Initiation},\mathtt{Moment},\mathtt{Fluent},\mathtt{Interval},\mathtt{Event},\mathtt{ActionType},\mathtt{Object},\mathtt{Payoff}) \tag{7}$$

4.2. The epistemic section of the ontology

The core of the epistemic section of the ontology consists of three mutually similar classes, `Knowledge`, `Belief` and `Perception`, representing three different epistemic attitudes (there is also their superclass `EpistemicState`, which is for brevity not depicted in Fig. 3). They are connected to the knowing/believing/perceiving `Agent` using the property `who`. The content of the epistemic attitude is determined by the object property `what` connected to the classes `Hold` and `FluentRelation`. The time of knowing/believing/perceiving is determined by the property `when` connected to class `Moment`.

The class `CommonKnowledge` represents knowledge – property `what` – available, at a given time specified by property `when`, to all agents. It is similar to the epistemic states described above, but it obviously lacks the object property `who`, because it is know by all agents.

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`.

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

$$\texttt{Communication}\equiv(\exists\texttt{communicationTo.Agent})\sqcap(\exists\texttt{who.Agent})\sqcap(\exists\texttt{when.Moment})\sqcap(\exists\texttt{what.Fluent}) \tag{8}$$

$$\texttt{Knowledge}\equiv(\exists\texttt{who.Agent})\sqcap(\exists\texttt{when.Moment})\sqcap(\exists\texttt{what.Fluent}) \tag{9}$$

$$\texttt{Belief}\equiv(\exists\texttt{who.Agent})\sqcap(\exists\texttt{when.Moment})\sqcap(\exists\texttt{what.Fluent}) \tag{10}$$

$$\texttt{Perception}\equiv(\exists\texttt{who.Agent})\sqcap(\exists\texttt{when.Moment})\sqcap(\exists\texttt{what.Fluent}) \tag{11}$$

$$\texttt{CommonKnowledge}\equiv(\exists\texttt{when.Moment})\sqcap(\exists\texttt{what.Fluent}) \tag{12}$$

4.3. The protential section of the ontology

The protential section of the ontology focuses on modeling the future. It currently only consists of two classes: `Intention` and `Desire`.

The class `Intention` represents an intention of an `Agent` (connected by the property `who`) to perform an action of the type determined by the class `ActionType` (connected by the object property `intentionActionType`). As mentioned above, there are two time specifications: the object property `when` determines the time when the act of intention takes place, and the object property `intendsWhen` determines the time when the intended action should take place.

The class `Desire` is similar to `Intention` but it does not comprise any action of an agent himself. While agent may intend to do something himself (e.g. to pickup friend at 6pm), he does desire something to happen to him (e.g. desires to be picked up at 6pm by friend) . So in the case of intention the agents role is active, while in the case of desire the agent is passive. The `Desire` therefore only uses the object properties `what`, `when`, `who` and `desiresWhen`.

Both `Intention` and `Desire` are subclasses of the class `ProtentialState` (for brevity not depicted on Fig. 3).

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

$$\begin{aligned}\texttt{Intention}\equiv&(\exists\texttt{intendsActionType.ActionType})\sqcap(\exists\texttt{who.Agent})\sqcap\\ &\sqcap(\exists\texttt{when.Moment})\sqcap(\exists\texttt{intendsWhen.Moment})\end{aligned} \tag{13}$$

$$\texttt{Desire}\equiv(\exists\texttt{what.Fluent})\sqcap(\exists\texttt{who.Agent})\sqcap(\exists\texttt{when.Moment})\sqcap(\exists\texttt{desiresWhen.Moment}) \tag{14}$$

4.4. The deontic section of the ontology

The deontic section of the ontology consists of a single class, `Obligation`. It is similar to class `Intention`, since obligation in this sense is structurally similar to intention. There is an individual (`who`) that at some point of time (`when`) thinks that his obligation is to do some action of an (`ActionType`), in a current or future time point (obligedWhen). Note that there are again two time specifications similarly to the case of protential concepts.

The obligations are modeled 'subjectively', so they are always related to an individual agent who is mentally aware of them as of a specific behavioral binding applicable to his actions. The obligation is always an obligation to act somehow. Another important kind of obligation, that to *abstain* from an action, may be only modeled indirectly as obligation to a specific passive action, e.g., "waiting at a given location".

The only axiom of the deontic section of the ontology is the following:

$$\begin{aligned}\texttt{Obligation}\equiv(\exists\texttt{obligedToActionType.ActionType})\sqcap(\exists\texttt{what.Fluent})\sqcap\\ \sqcap(\exists\texttt{who.Agent})\sqcap(\exists\texttt{when.Moment})\sqcap(\exists\texttt{obligedWhen.Moment})\end{aligned} \tag{15}$$

5. Open Problems and Possible Future Extensions

The $\mathcal{DCEC}^*$ calculus is based on the concept of a fluent that has propositional character. Most foundational ontologies do not model propositional entities. It might be necessary to adapt the ontology so that it could somehow model propositional entities or even their internal structure. The relation between individual fluents is now modeled using a FluentRelation, class but it might be insufficient.

In the real-world applications performing in the real time the speed of reasoning is of prime importance. The current expressiveness of the $\mathcal{DCEO}$ ontology is $\mathcal{ALFC}(\mathcal{D})^*$. It is sufficiently fast for the most of the reasoning tasks performed. However the complexity of the ontology may raise especially after adding the handling of internal structure of propositions. Such an increase of complexity could lead to significant reasoning-time overhead.

The protential section of the ontology currently contains the classes Intention and Desire. Both these classes concern future events and it might be possible to model those more precisely. Desires refer to foreseen future events that are positively evaluated but at the same time are not agent's own actions. Such events may be divided to those caused by the agent's actions (directly or indirectly) and those events that are desired but are out of the scope of being influenced by the agent's actions.

Foreseen future events may also differ in terms of evaluation. There are positively evaluated events – desires – but there are also neutral events that are just foreseen as part of the probable future course of events – we may call them expectations – and also foreseen events that are negatively evaluated – fears. Such distinctions may be very useful for agent's understanding of his environment and for planing and realization of his actions.

This line of thoughts is connected to another possible extension of the ontology, one allowing to more precisely represent the plans of agent and courses of events that happen in his environment.

In the future the deontic section of the ontology may also be extended so as to enable the handling of more general concepts of obligation than the current "subjective" obligation adapted form $\mathcal{DCEC}^*$ does. It might be useful to think of "universal" obligations applicable to all agents, or of obligations applicable to an agent but unknown to him. Another improvement may be more direct handling of obligations to abstain from an action. This type of obligation is in some context more appropriate and important.

6. Conclusions

Solving the problem of an autonomous action of artificial agents is indispensable in order to progress in many areas of artificial intelligence. The research dealing with this problem 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 Cognitive Science.

We have presented an ontology framework based on the Deontic Cognitive Event Calculus ($\mathcal{DCEC}^*$) and inspired by other general cognitive architectures. We also presented a brief overview of our cognitive architecture that includes proposed ontology as its core and enables effective cognitive reasoning on top of it.

The Deontic Cognitive Event Ontology ($\mathcal{DCEO}$) described in this paper consists of four parts: event ontology, epistemic ontology, protential ontology and deontic ontology. The event ontology allows modeling of actions of artificial agents occurring at specific times or intervals. The epistemic ontology describes the mental states of these agents: belief, knowledge and perception, and their propositional content. It also enables the modeling of communication between agents and of common knowledge available to all agents. The proposed protential ontology models the agents' attitudes to future – their desires and intentions. This in turn influences their autonomous actions. Finally, the deontic ontology aims at modeling obligations – the actions that the agent knows he is obliged to perform at a given situation.

The paper finally discusses some problems and limitations of the presented framework and proposes some possible future enhancements and extensions.

6.0.1. Acknowledgments.

This work has been supported by the long-term institutional support of research activities by the Faculty of Informatics and Statistics, University of Economics, Prague.

References

[1] Arkoudas, K., Bringsjord, S. Propositional Attitudes and Causation. In: Int. J. Software and Informatics 3.1 (2009): 47–65. (http://kryten.mm.rpi.edu/PRICAI_w_sequentcalc_041709.pdf)

[2] Bringsjord, S., Govindarajulu, N. S. (2013). Deontic cognitive event calculus (formal specification). (URL: http://www.cs.rpi.edu/~govinn/dcec.pdf)

[3] Bringsjord, S., Govindarajulu, N.S., Ellis, S., McCarty, E., Licato, J., Nuclear deterrence and the logic of deliberative mindreading, Cognitive Systems Research, Volume 28, June 2014, Pages 20-43, ISSN 1389-0417. (URL: http://www.sciencedirect.com/science/article/pii/S1389041713000491)

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

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

[6] Card, S., Moran, T. P., Newell, A. (1983). The Psychology of Human-Computer Interaction. Hillsdale, New Jersey: Erlbaum.

[7] Castañeda, H.-N. (1999), "He": A Study in the Logic of Self-Consciousness, In: Hart, J. G., Kapitan, T., eds, The Phenomeno-Logic of the I: Essays on Self-Consciousness, Indiana University Press, 601 North Morton Street, Bloomington, Indiana 4704-3797 USA.

[8] Goble, L. (2003), Preference Semantics for Deontic Logic. Part I: Simple Models, In Logique et Analyse 46, 183–184.

[9] 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.

[10] Jaśkowski, S. (1934), On the Rules of Suppositions in Formal Logic, Studia Logica 1, 5–32.

[11] Kang, Seung Hwan and Lau, Sim Kim (2004), Ontology Revision Using the Concept of Belief Revision. In Knowledge-Based Intelligent Information and Engineering Systems: 8th International Conference, KES 2004, Wellington, New Zealand, September 20-25, 2004, Proceedings, Part III. Springer : Berlin Heidelberg, pp 8–15.

[12] Klahr, D. MacWhinney, B. (1998). Information Processing. In Handbook of Child Psychology. Ed. Deana Kuhn a Robert S. Siegler. Vol. 2: Cognition, Perception and Language. New York: John Wiley and Sons Inc.

[13] Kowalski, R.A., Sergot, M.J., A Logic-Based Calculus of Events, In: New Generation Computing, vol. 4 (1986), pp. 67–95.

[14] Kowalski, R.A. (1992), Database Updates in the Event Calculus, Journal of Logic Programming, vol. 12, pp. 121–146.

[15] Laird, J.E., Newell, A., Rosenbloom, P.S. (1987) Soar: An architecture for general intelligence. Artificial Intelligence 33(1), pp 1–64.

[16] Manzano, M. (1996), Extensions of First Order Logic, Cambridge University Press, Cambridge, UK.

[17] 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, Society for Computer Simulation International, San Diego, CA, USA.

[18] McNamara, P. (2010), Deontic logic, In: E. Zalta, ed., The Stanford Encyclopedia of Philosophy, Fall 2010 edn. (URL: http://plato.stanford.edu/entries/logic-deontic/chisholm.html)

[19] Mepham, W. and Gardner, S. (2009), Implementing Discrete Event Calculus with Semantic Web Technologies, In: Next Generation Web Services Practices, 2009. NWESP '09. Fifth International Conference on, Prague, 2009, pp. 90-93.

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

[21] Merlan, P. (1947). Time Consciousness in Husserl and Heidegger. In Philosophy and Phenomenological Research, 8(1), 23-54.

[22] Newell, A., Rosenbloom, P.S., Laird, J.E. Symbolic Architectures for Cognition. In Posner, M. I. (ed) Foundations of Cognitive Science, The MIT Press, Cambridge, Massachusetts, pp 93–132.

[23] Pohl, W. (1997). LaboUr machine learning for usermodeling. In Smith, M. J., Salvendy,G., and Koubek, R. J., eds., Design of Computing Systems: Social and Ergonomic Considerations (Proceedings of the Seventh International Conference on Human-Computer Interaction), volume B, 2730. Amsterdam: Elsevier Science.

[24] Pohl, W. and Achim, N. (1999) Machine learning and knowledge representation in the labour approach to user modeling. In UM99 User Modeling. Volume 407 of the series CISM International Centre for Mechanical Sciences pp 179–188

[25] Shanahan, M. (2001), The Event Calculus Explained, In: Artificial Intelligence Today, Volume 1600 of the series Lecture Notes in Computer Science, Springer, Berlin, pp. 409-430.

[26] Simon, H.A., Kaplan, C.A. (1998). Foundations of Cognitive Science. In Posner, M. I. (ed) Foundations of Cognitive Science, The MIT Press, Cambridge, Massachusetts, pp 1–48.