Innovations in Systems and Software Engineering manuscript No.
(will be inserted by the editor)

BOWL - Augmenting the Semantic Web with Beliefs

Jin Song Dong · Yuzhang Feng · Yuan-Fang Li · Colin Keng-Yan Tan · Bimlesh Wadhwa · Hai H. Wang

Received: date / Accepted: date

Abstract As the Semantic Web is an open, complex and constantly evolving medium, it is the norm, but not exception that information at different sites is incomplete or inconsistent. This poses challenges for the engineering and development of agent systems on the Semantic Web since autonomous software agents need to understand, process and aggregate this information. Ontology language OWL provides core language constructs to semantically markup resources on the Semantic Web, on which software agents interact and cooperate to accomplish complex tasks. However, as OWL was designed on top of (a subset of) classic predicate logic, it lacks the ability to reason about inconsistent or incomplete information. Belief Augmented Frames (BAF) is a frame-based logic system that associates with each frame a supporting and a refuting belief value. In this paper, we propose a new ontology language BOWL (Belief-augmented OWL) by integrating OWL DL and BAF to incorporate the notion of confidence. BOWL is paraconsistent, hence it can perform useful reasoning services in the presence of inconsistencies and incompleteness. We define the abstract syntax and semantics of BOWL by extending those of OWL. We have proposed reasoning algorithms for various reasoning tasks in the BOWL framework and we have implemented the algorithms using the constraint logic programming framework. One example in the sensor fusion domain is presented to demonstrate the application of BOWL.

Keywords Semantic Web, OWL, Probabilistic Ontology Language

Author for correspondence: Yuan-Fang Li.

Jin Song Dong, Colin Keng-Yan Tan and Bimlesh Wadhwa
School of Computing, National University of Singapore
Computing 1, 13 Computing Drive, Singapore 117417
Tel.: +65 6516 2727
Fax: +65 6779 4580
E-mail: {dongjs, ctank, bimlesh}@comp.nus.edu.sg

Yuzhang Feng
SAS Institute, Singapore
20 Anson Rd, Singapore 079912
E-mail: yuzhang.feng@sas.com

Yuan-Fang Li
Faculty of IT, Monash University
Wellington Road Clayton VIC 3800
Tel: +61 3 9905 9688
Fax: +61 3 9905 5159
E-mail: yuanfang.li@monash.edu

Hai H. Wang
School of Engineering and Applied Science, Aston University
Aston Triangle, Birmingham B4 7ET, United Kingdom, UK
E-mail: H.WANG10@aston.ac.uk

1 Introduction

The Semantic Web [4] is a vision to make resources on the web readily available not only to humans, but also to software agents so that they can communicate and cooperate to accomplish complex tasks autonomously. As the web is an extremely complex, globally distributed and constantly evolving medium, it is often the case that information on different sites is incomplete or inconsistent with respect to each other. Hence, software agents need to cope with this situation.

The cornerstone language in the Semantic Web is OWL [7], which provides core language constructs to semantically markup Web resources. OWL is based on description logic, a subset of first-order predicate logic. This dictates that any formula can be inferred from an inconsistent knowledge base (aggregate information). This is neither practical nor desirable as no useful reasoning services are available even in the presence of very

slight inconsistency. Hence, a mechanism of representing confidence and ignorance is very desirable.

The Belief-augmented Frames (BAF) [27] is an extension to the Minsky frame knowledge representation system [18]. Its unique feature is that it adds a belief and a disbelief value to each frame in a frame system. In BAF, belief and disbelief values are independent from each other, allowing for greater flexibility in modeling arguments for and against a fact. Consequently, ignorance and confidence can be incorporated.

In this paper, we propose to integrate BAF with OWL DL to form a new ontology language BOWL (Belief-augmented OWL) that can easily express beliefs.

The rest of the paper is organized as follows. In Section 2, we briefly present works related to this paper. In Sections 3 and 4, we give a brief introduction to BAF and OWL languages. In Section 5, we present the Belief-augmented OWL, and its semantics for the various language constructs. Section 6 discusses how we can reason about BOWL ontologies and how confidence factors can be computed. In Section 7, we present an example in the sensor fusion domain to demonstrate the reasoning process in BOWL. Conclusion and future work directions are discussed in Section 8.

2 Related Works

Expert systems are used to assist decision making in individual narrow application domains, such as the medical domain. Historically, uncertainty has been an research subject in expert systems. A comprehensive survey can be found at [19]. Probabilistic measures were used in expert systems to deal with uncertainty. For example, the Pathfinder project [11] uses subjective probability theory, belief networks, for decision-support system for hematopathology diagnoses. It uses influence diagrams to track dependencies among observed features. Possibility theory was also incorporated in some expert systems for dealing with uncertain information. The Cadiag-2 [2] system was developed to diagnose rheumatic, hepatic and pancreatic diseases. It is based on fuzzy theory and fuzzy set [29] and uses fuzzy inference to propagate and track belief. In general, expert systems are very limited in scope and specifically designed for particular application domains. On the contrary, the Semantic Web is envisioned to be an open environment, encompassing more complex and more unreliable resources and data.

There have been many proposals [17, 25, 28] on probabilistic/fuzzy extensions to description logics such as $\mathcal{ALC}$ and Classic, which are less expressive than the description logic ($\mathcal{SHOIN}(D)$) on which is OWL based. Ding and Peng proposed a Bayesian network-based probabilistic extension to OWL [9]. The main focus of their works is the modeling of a priori and conditional probabilities of OWL classes and the reasoning tasks are concept satisfiability, overlapping and subsumption, which is different from ours. Nottelmann and Fuhr proposed a probabilistic Datalog-based extension to DAML+OIL [20]. Their approach is less general than ours in the sense that a fact with both true and false evidence present is considered inconsistent, whereas in our approach evidence for and against a fact are allowable and ignorance is computed based on these values. Straccia [25] proposed a fuzzy extension for the description logic $\mathcal{ALC}$, which is less expressive than $\mathcal{SHOIN}(D)$. Based on this work, Straccia proposed a fuzzy extension for OWL [26]. Again, based on fuzzy set, each assertion is associated with a single value representing its fuzziness. Therefore, our approach is more flexibility as software agents may very possibly receive both supporting and refuting values for a certain assertion. BOWL enables agents to make use of these values at their own discretion. The other approaches, however, hide this step of processing and is hence more rigid and less transparent.

3 The Web Ontology Language OWL

The Semantic Web [4] was originally proposed as an extension to the current Web in which resources, including static information and dynamic services, are semantically marked up so that they can be understood, processed and aggregated by software agents autonomously. The usefulness of the Web is greatly expanded by agents' ability to perform tasks on human's behalf. Web resources are marked up by ontologies, defining concepts of and relations between various resources. Ontology languages provide core vocabularies for expressing ontologies.

In 2003, based on DAML+OIL [10] and RDF Schema [5], the Web Ontology Language (OWL) [7] was published by W3C as a Recommendation. It contains three sublanguages: OWL Lite, DL and Full, with increasing expressiveness. The three sublanguages are designed for user groups with different expressiveness requirements. These constraints are relaxed in OWL Full to allow for greater expressiveness. OWL Lite and DL are decidable whereas OWL Full is generally not.

$\mathcal{C} ::= C$	[Class name]
$\mid \top$	[Top class]
$\mid \bot$	[Bottom class]
$\mid \mathcal{C} \sqcup \mathcal{C}$	[Class union]
$\mid \mathcal{C} \sqcap \mathcal{C}$	[Class intersection]
$\mid \neg \mathcal{C}$	[Class negation]
$\mid \forall P.\mathcal{C}$	[Universal quantification]
$\mid \exists P.\mathcal{C}$	[Existential quantification]
$\mid P{:}\, o$	[Value restriction]
$\mid \geq n\ P$	[At least number restriction]
$\mid \leq n\ P$	[At most number restriction]
$\mid \{a_1, \cdots, a_n\}$	[Enumeration]

Fig. 1 OWL class expressions

3.1 Abstract Syntax

With its origin from description logics [3], OWL describes knowledge in a particular domain. In OWL, abstract concepts are *Classes*, related by binary relations called *Properties* and are populated (instantiated) by concrete *Individuals*. In a compact form, OWL can be presented in the Description Logics (DL) syntax [12].

Classes in OWL are first-class citizens. Existing classes can be used in *class expressions* to define new complex ones using class constructors. Class expressions in OWL can be of the following form as shown below. Note that in Fig. 1 below, $\mathcal{C}$ represents (possible complex) class expressions; C is a class name; P stands for a property; n is a natural number and a_i's are individuals. In OWL, datatype properties and object properties are distinguished. Without loss of generality and for brevity reasons, we omit the discussion related to datatypes. They can be treated in a similar manner.

OWL also defines *class axioms* that inter-relate classes. These class axioms, shown in Fig. 2 below, include class subsumption, equivalence, disjointness, etc. Axioms are also defined for describing properties. These include property subsumption, equivalence, domain, range, etc. Note that the symbol $\mathcal{AX}$ represents axioms.

$\mathcal{AX} ::= C \sqsubseteq \mathcal{C}$	[Class subsumption]
$\mid C = \mathcal{C}$	[Class equivalence]
$\mid C \sqcap \mathcal{C} = \bot$	[Class disjointness]
$\mid P \sqsubseteq P$	[Property subsumption]
$\mid P = P$	[Property equivalence]
$\mid \geq 1\ P \sqsubseteq \mathcal{C}$	[Property domain]
$\mid \top \sqsubseteq \forall P.\mathcal{C}$	[Property range]
$\mid \top \sqsubseteq \leq 1\ P$	[Functional property]
$\mid P = ({}^{-}P)$	[Inverse property]

Fig. 2 OWL class axioms

The following example helps to illustrate some of the modeling ideas of OWL.

Example 1 Suppose that we want to model the animal domain and we have identified the following names that we want to model as classes: Animal and Carnivore, where Carnivore is a sub class of Animal. We want to define the concept (class) carnivores, which only eats animals. For this reason, we need an object property Eats. The above classes and properties can be modeled as follows.

$$\mathit{Animal} \sqsubseteq \top \qquad \mathit{Carnivore} \sqsubseteq \forall\, \mathit{Eats}.\mathit{Animal}$$
$$\mathit{Carnivore} \sqsubseteq \mathit{Animal} \qquad \mathit{Carnivore} \sqsubseteq \exists\, \mathit{Eats}.\mathit{Animal}$$

The above definitions define *Animal* as an OWL class and *Carnivore* as its sub class. It also uses universal quantification restriction to model the fact that a *Carnivore* only *Eats*, which is an object property, *Animal*. The existential quantification restriction is used to ensure that a *Carnivore* does *Eats* at least one *Animal*.

Assertions in OWL are used to model individuals. An assertion can model the fact that an individual is a instance of a class, or a pair of individuals is a member of a property. In addition, assertions can be used to state the (in)equality of two individuals. Fig. 3 shows the assertions available in OWL. Note that the symbol $\mathcal{AS}$ represents assertions.

$\mathcal{AS} ::= a \in \mathcal{C}$	[Class membership]
$\mid \langle a, b \rangle \in P$	[Property membership]
$\mid a = b$	[Individual equality]
$\mid a \neq b$	[Individual inequality]

Fig. 3 OWL assertions

3.2 Semantics

OWL is based on formal logic, hence it has a formal semantics, an *abstract interpretation* $\mathcal{I}$ [12], which essentially is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the *domain of interpretation* and $\cdot^{\mathcal{I}}$ is the interpretation function. The interpretation function $\cdot^{\mathcal{I}}$ maps an individual name into a member of the domain $\Delta^{\mathcal{I}}$; a class name into a set of elements in the domain and a property name (note that we are talking about object properties) into a set of pairs of domain elements ($\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$).

The class expressions, axioms, assertions and their interpretation can be summarized in Table 1 below.

4 Belief-augmented Frames

In belief models the possibility of an event occurring is modeled as a range of values rather than as a single point probability. This range allows us to express

Table 1 OWL class expressions/axioms/assertions & their interpretations

OWL class expression	**Interpretation**
C	$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\top$	$\top^{\mathcal{I}} = \Delta$
$\bot$	$\bot^{\mathcal{I}} = \varnothing$
$\mathcal{C}_1 \sqcup \mathcal{C}_2$	$\mathcal{C}_1^{\mathcal{I}} \cup \mathcal{C}_2^{\mathcal{I}}$
$\mathcal{C}_1 \sqcap \mathcal{C}_2$	$\mathcal{C}_1^{\mathcal{I}} \cap \mathcal{C}_2^{\mathcal{I}}$
$\neg\, \mathcal{C}$	$\Delta^{\mathcal{I}} \setminus \mathcal{C}^{\mathcal{I}}$
$\forall\, P.\mathcal{C}$	$\{x \mid \forall\, y.\langle x, y\rangle \in P^{\mathcal{I}} \rightarrow y \in \mathcal{C}^{\mathcal{I}}\}$
$\exists\, P.\mathcal{C}$	$\{x \mid \exists\, y.\langle x, y\rangle \in P^{\mathcal{I}} \wedge y \in \mathcal{C}^{\mathcal{I}}\}$
$P{:}\, o$	$\{x \mid \langle x, o^{\mathcal{I}}\rangle \in P^{\mathcal{I}}\}$
$\geq n\ P$	$\{x \mid \#\{y \mid \langle x, y\rangle \in P^{\mathcal{I}}\} \geq n\}$
$\leq n\ P$	$\{x \mid \#\{y \mid \langle x, y\rangle \in P^{\mathcal{I}}\} \leq n\}$
$\{a_1, \cdots, a_n\}$	$\{a_1^{\mathcal{I}}, \cdots, a_n^{\mathcal{I}}\}$
OWL axiom	**Interpretation**
$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
$C_1 = C_2$	$C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$
$C_1 \sqcap C_2 = \bot$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} = \varnothing$
OWL assertion	**Interpretation**
$a \in \mathcal{C}$	$a^{\mathcal{I}} \in \mathcal{C}^{\mathcal{I}}$
$\langle a, b\rangle \in P$	$\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in P^{\mathcal{I}}$
$a = b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
$a \neq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

ignorance, which standard statistical measures do not accommodate. Statistical measures only provide limited ability to express ignorance, e.g., as given in [1], where doctors, who predicted with a certainty of $x\%$ that a patient was suffering from a particular illness, were reluctant to predict with a certainty of $(100-x)\%$ that the patient was not suffering from that illness. This apparent contradiction is a reflection of classical statistics' inability to cater to ignorance. Various models of beliefs have been proposed, including the seminal Dempster-Shafer theory [8, 23]. Smets generalized the model in [24] to form the Transferable Belief Model. Picard [22] proposed the Probabilistic Argumentation System, which combines propositional logic with probability measures to perform reasoning.

4.1 Belief Augmented Systems

In classical AI a frame represents an object in the world, and slots within the frame indicate the possible relations that this object can have with other objects. A value (or set of values) in a slot indicates the other objects that are related to this object through the relation represented by the slot. The existence of a slot-value pair indicates a relation.

- In BAFs each slot-value pair is augmented by a pair of belief/disbelief masses. We define this pair as `BAF`, with `BAF_range` as the value range from 0 to 1, *i.e.,* $[0, 1]$. Hence, a BAF b is a pair $\langle a_t, a_f\rangle$, where a_t and a_f are from `BAF_range`.
- Furthermore, we define two functions $\phi^{\mathtt{T}}$ and $\phi^{\mathtt{F}}$ to project out the belief/disbelief values of a `BAF` value pair: $\phi^{\mathtt{T}}(b) = a_t$ (the first element of the pair $x.1$) and $\phi^{\mathtt{F}}(b) = a_f$ (the second element $x.2$)[1].

  $$\phi^{\mathtt{T}}, \phi^{\mathtt{F}} : \mathtt{BAF} \rightarrow \mathtt{BAF\_range},\ \text{where}$$
  $$\forall\, x : BAF \bullet \phi^{\mathtt{T}}(x) = x.1 \wedge \phi^{\mathtt{F}}(x) = x.2$$

  Note that $\phi^{\mathtt{T}}$ and $\phi^{\mathtt{F}}$ of a particular `BAF` may not necessarily sum to 1. This frees us from the classical statistical assumption that $\phi^{\mathtt{T}}(rel) = 1 - \phi^{\mathtt{F}}(rel)$ and it allows us to model ignorance. It is also possible that $\phi^{\mathtt{T}}(rel) + \phi^{\mathtt{F}}(rel) > 1$. More information on this can be found in [27] where ignorance is discussed in detail. Both $\phi^{\mathtt{T}}$ and $\phi^{\mathtt{F}}$ may be derived from various independent sources, or may be computed by using a system of logic called "BAF-Logic" which will be presented in the next subsection. Effectively this allows us to model the belief in a problem as a set of arguments for the belief, and a set of arguments against it.
- While $\phi^{\mathtt{T}}$ and $\phi^{\mathtt{F}}$ represent the degree of belief for and against a claim, the overall truth is given by the Degree of Inclination `DI`, a function from `BAF`, pairs of BAF values, to `DI_range`, an interval from -1 to 1 ([-1, 1]).

  $$\mathtt{DI} : \mathtt{BAF} \rightarrow \mathtt{DI\_range},\ \text{where}$$
  $$\forall\, x : \mathtt{BAF} \bullet \mathtt{DI}(x) = \phi^{\mathtt{T}}(x) - \phi^{\mathtt{F}}(x)$$

  $\mathtt{DI}(rel)$ measures the overall degree of truth of the relationship *rel*, with -1 representing falsehood, 1 representing truth, and values in between representing various degrees of truth and falsehood. As an example, we could take -0.25 to mean "possibly false", -0.5 to mean "probably false", etc.
- The Utility Function `U` is defined as follows:

  $$\mathtt{U} : \mathtt{BAF} \rightarrow \mathtt{BAF\_range},$$
  $$\text{such that}\, \forall\, x : \mathtt{BAF} \bullet \mathtt{U}(x) = \frac{1 + \mathtt{DI}(x)}{2}$$

  `U` maps the Degree of Inclination to a $[0, 1]$ range. If we normalize the Utility Functions for all relations so that they sum to 1, we can use these normalized values as statistical measures representing the probability of a relation being true.
  It can be easily verified that for any BAF value b, $\mathtt{U}(b) = 1 - \mathtt{U}(\neg^{\mathcal{B}} b)$ and if $\mathtt{U}(b) \geq n$, then $\mathtt{U}(\neg^{\mathcal{B}} b) \leq 1 - n$, for $n \in [0, 1]$. Note that the BAF *not* operator $\neg^{\mathcal{B}}$ will be introduced in the next subsection.

[1] Note that the symbols $\bullet$ represents "such that".

- The plausibility Pl is, in the ideal case, the upper bound that $\phi^{\mathtt{T}}$ can take, and it is defined by:

  $$\mathtt{Pl} : \mathtt{BAF} \rightarrow \mathtt{BAF\_range}, \quad \text{such that } \forall x : \mathtt{BAF} \bullet \mathtt{Pl}(x) = 1 - \phi^{\mathtt{F}}(x)$$

  For any given BAF value b, if $\phi^{\mathtt{T}}(b) > \mathtt{Pl}(b)$, then either the data generating $\phi^{\mathtt{T}}$ is overly optimistic ($\phi^{\mathtt{T}}$ is too large), or overly pessimistic ($\phi^{\mathtt{F}}$ is too large, resulting in Pl being too small). In either case the condition $\phi^{\mathtt{T}}(b) > \mathtt{Pl}(b)$ indicates that the data generating $\phi^{\mathtt{T}}$ and $\phi^{\mathtt{F}}$ is conflicting.
- The ignorance in our system is given by Ig, and is defined as:

  $$\mathtt{Ig} : \mathtt{BAF} \rightarrow \mathtt{DI\_range}, \text{ such that } \mathtt{Ig}(x) = \mathtt{Pl}(x) - \phi^{\mathtt{T}}(x)$$

  Note that $\mathtt{Ig}(b)$ is a negative number when $\phi^{\mathtt{T}}(b) > \mathtt{Pl}(b)$. As discussed earlier this is indicative that the data supporting and refuting b is conflicting. In such cases $\phi^{\mathtt{T}}(b) + \phi^{\mathtt{F}}(b) > 1$.

There are many other features of BAFs like inheritance of relationships, generalization of concepts and daemons that are beyond the scope of this paper. Interested readers are referred to [27].

4.1.1 Predefined Beliefs

We define three predefined BAFs $\phi_{\mathtt{def}}$, $\phi_{\mathtt{one}}$ and $\phi_{\mathtt{zero}}$ as follows. They are convenient shorthand for frequently used BAF values.

$$\phi_{\mathtt{def}}, \phi_{\mathtt{one}}, \phi_{\mathtt{zero}} : \mathtt{BAF}, \text{ such that } \phi_{\mathtt{def}} = \langle 0, 0\rangle \quad \phi_{\mathtt{one}} = \langle 1, 0\rangle \quad \phi_{\mathtt{zero}} = \langle 0, 1\rangle$$

For example, following $\phi_{\mathtt{def}}$, we obtain:

$$\mathtt{DI}(\phi_{\mathtt{def}}) = 0.0, \quad \mathtt{U}(\phi_{\mathtt{def}}) = 0.5, \quad \mathtt{Pl}(\phi_{\mathtt{def}}) = 1.0, \quad \mathtt{Ig}(\phi_{\mathtt{def}}) = 1$$

Thus by choosing 0 for both supporting and refuting belief values, we get a DI representing lack of knowledge of evidence for and against the statement, a Utility Function that is 50% true (and therefore 50% untrue), a plausibility that represents that there is no reason why the relationship cannot be true, and complete ignorance about the truth of the relationship.

4.2 Belief Augmented Frames Logic

Belief Augmented Frame Logic (BAF-Logic) is a system designed to reason over the $\phi^{\mathtt{T}}$ and $\phi^{\mathtt{F}}$ values in the frame.

- We define the BAF *conjunction* $\cap^{\mathcal{B}}$ as a function from two BAFs to a BAF. Hence, given two BAFs P and Q, their conjunction $P \cap^{\mathcal{B}} Q$ is defined as follows.

  $$\_ \cap^{\mathcal{B}} \_ : \mathtt{BAF} \times \mathtt{BAF} \rightarrow \mathtt{BAF}, \text{ such that } \forall P, Q : \mathtt{BAF} \bullet \phi^{\mathtt{T}}(P \cap^{\mathcal{B}} Q) = min(\phi^{\mathtt{T}}(P), \phi^{\mathtt{T}}(Q)) \wedge \phi^{\mathtt{F}}(P \cap^{\mathcal{B}} Q) = max(\phi^{\mathtt{F}}(P), \phi^{\mathtt{F}}(Q))$$

  This definition is based on the intuitive idea that the strength of $P \cap^{\mathcal{B}} Q$ being true rests on the strength of the weakest proposition P or Q. Likewise, if either P or Q were false, then $P \cap^{\mathcal{B}} Q$ would be false, and we can base our degree of belief in $P \cap^{\mathcal{B}} Q$ being false on the strongest proposition that either P or Q is false.
- Similarly we define the BAF *disjunction* $\cup^{\mathcal{B}}$ between P and Q as:

  $$\_ \cup^{\mathcal{B}} \_ : \mathtt{BAF} \times \mathtt{BAF} \rightarrow \mathtt{BAF}, \text{ such that } \forall P, Q : \mathtt{BAF} \bullet \phi^{\mathtt{T}}(P \cup^{\mathcal{B}} Q) = max(\phi^{\mathtt{T}}(P), \phi^{\mathtt{T}}(Q)) \wedge \phi^{\mathtt{F}}(P \cup^{\mathcal{B}} Q) = min(\phi^{\mathtt{F}}(P), \phi^{\mathtt{F}}(Q))$$

- Finally, we define the BAF *not* operator $\neg^{\mathcal{B}}$ as:

  $$\neg^{\mathcal{B}}\_ : \mathtt{BAF} \rightarrow \mathtt{BAF}, \text{ such that } \forall P : \mathtt{BAF} \bullet \phi^{\mathtt{T}}(\neg^{\mathcal{B}} P) = \phi^{\mathtt{F}}(P) \wedge \phi^{\mathtt{F}}(\neg^{\mathcal{B}} P) = \phi^{\mathtt{T}}(P)$$

  This means that the degree that we believe that our data support $\neg^{\mathcal{B}} P$ is equal to the degree that they refute P. Likewise the degree that our data refute $\neg^{\mathcal{B}} P$ is equal to the degree that they support P.

5 Belief-augmented OWL (BOWL)

The Belief-augmented OWL incorporates belief/disbelief values defined in BAF into OWL to enable the representation & reasoning of incomplete, subjective and sometimes conflicting resources on the Semantic Web. In this section, we introduce the BOWL and present its semantics by defining interpretations of the various language constructs of BOWL.

5.1 BAF

Each fact about individuals in OWL is augmented with a BAF, a pair consisting of a belief and a disbelief measure of the type BAF_range. Hence, a BAF value is of the form $\langle b_t, b_f\rangle$, where b_t and b_f are the belief/disbelief values, respectively.

$$\mathtt{BAF} ::= \langle \mathtt{BAF\_range},\ \mathtt{BAF\_range}\rangle$$

BAF_range can be viewed as a data type derived from **float** defined in XML Schema.

In the following, we will use angle brackets "⟨⟩" to denote the association of an OWL language construct and its BAF value.

$$
\begin{array}{lll}
\mathcal{AX} ::= & \langle C \sqsubseteq \mathcal{C}, \texttt{BAF} \rangle & \text{[Class subsumption]} \\
| & \langle C = \mathcal{C}, \texttt{BAF} \rangle & \text{[Class equivalence]} \\
| & \langle C \sqcap \mathcal{C} = \bot, \texttt{BAF} \rangle & \text{[Class disjunction]} \\
| & \langle P \sqsubseteq P, \texttt{BAF} \rangle & \text{[Property subsumption]} \\
| & \langle P = P, \texttt{BAF} \rangle & \text{[Property equivalence]} \\
| & \langle P = (^{-}P), \texttt{BAF} \rangle & \text{[Inverse property]}
\end{array}
$$

Fig. 4 BOWL axioms

$$
\begin{array}{lll}
\mathcal{AS} ::= & \langle a \in \mathcal{C}, \texttt{BAF} \rangle & \text{[Class membership]} \\
| & \langle \langle a, b \rangle \in P, \texttt{BAF} \rangle & \text{[Property membership]} \\
| & \langle a = b, \phi_{\texttt{one}} \rangle & \text{[Individual equality]} \\
| & \langle a \neq b, \phi_{\texttt{one}} \rangle & \text{[Individual inequality]}
\end{array}
$$

Fig. 5 BOWL assertions

5.2 BOWL Axioms & Assertions

As in OWL, BOWL axioms are about classes and properties and BOWL assertions are facts about ground knowledge entities such as individuals and data values. BOWL augments OWL axioms and assertions with BAF values, as follows. Both class and property membership assertions are treated alike. For any OWL axiom/assertion, its BOWL extension is summarized in Figs 4 and 5 below.

The second kind of facts asserts the relationship between individuals. BOWL attaches $\phi_{\texttt{one}}$ to each of these assertions hence the (in)equality between individuals will be treated in the same way as in OWL.

$\forall\, x \in \Delta^{\mathcal{I}_b} \bullet \top^{\mathcal{I}_b}(x) = \phi_{one}$

5.3 Semantics of BOWL

We construct the semantics of BOWL by extending the model-theoretic semantics of OWL [21].

Firstly, we assume that the **datatype map** `D` (as in OWL) is extended to include a mapping from the (abbreviated) URI `bowl:BAF` to the datatype `BAF`, as defined in Section 4. For brevity reasons and without loss of generality, we leave out the discussion related to data types, such as datatype properties, etc. They can be treated similarly as object properties.

5.3.1 The BOWL Interpretation

A BAF extended interpretation $\mathcal{I}_b$ is a pair $(\Delta^{\mathcal{I}_b}, \cdot^{\mathcal{I}_b})$, where $\Delta^{\mathcal{I}_b}$ is, as in the OWL case, the *domain of interpretation* and $\cdot^{\mathcal{I}_b}$ is the interpretation function. In OWL, the interpretation function maps an individual name into a member of the domain; a class name into a set of elements in the domain and a property name into a set of pairs of domain elements $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The BOWL interpretation still maps into members of the domain $\Delta^{\mathcal{I}_b}$. However, it maps a class (*resp.* a property) into a function from members of $\Delta^{\mathcal{I}_b}$ (*resp.* pairs of members of $\Delta^{\mathcal{I}_b}$) into a BAF value.

$$
\begin{array}{l}
\mathcal{C}^{\mathcal{I}_b} \colon \Delta^{\mathcal{I}_b} \rightarrow \texttt{BAF} \\
P^{\mathcal{I}_b} \colon \Delta^{\mathcal{I}_b} \times \Delta^{\mathcal{I}_b} \rightarrow \texttt{BAF}
\end{array}
$$

As defined above, each of the BOWL classes and properties is a function returning a BAF value. Intuitively, the BAF interpretation returns this value as the belief/ disbelief value of an individual (*resp.* a pair of individuals) being a member of the class (*resp.* property).

In addition, the following conditions must be satisfied by the interpretation $\mathcal{I}_b$. For any $a \colon \Delta^{\mathcal{I}_b}$,

- $\top^{\mathcal{I}_b}(a) = \phi_{\texttt{one}}$. This condition states that any individual of the domain is a member of the top class with the highest possible belief.
- $\bot^{\mathcal{I}_b}(a) = \phi_{\texttt{zero}}$. This condition states that no individual is a member of the bottom class.
  The above two conditions reiterate the facts that the top class is still the super class of all classes and that the bottom class is still the sub class of all classes.
- $(C_1 \sqcup C_2)^{\mathcal{I}_b}(a) = C_1^{\mathcal{I}_b}(a) \cup^{\mathcal{B}} C_2^{\mathcal{I}_b}(a)$. This condition states that the interpretation of disjunction of two BOWL class expressions is the BAF disjunction of the BAF values associated with the two classes.
- $(C_1 \sqcap C_2)^{\mathcal{I}_b}(a) = C_1^{\mathcal{I}_b}(a) \cap^{\mathcal{B}} C_2^{\mathcal{I}_b}(a)$. Similar to the disjunction case, this condition states that the interpretation of conjunction of two BOWL class expressions is the BAF conjunction of the BAF values associated with the two classes.
- $(\neg\ C)^{\mathcal{I}_b}(a) = \neg^{\mathcal{B}} C^{\mathcal{I}_b}(a)$. The interpretation of a class negation is the BAF negation of the BAF value of the original class C.
- $(\forall\, P.C)^{\mathcal{I}_b}(a) = \bigcap^{\mathcal{B}}_{b \in \Delta^{\mathcal{I}_b}} \left(\neg^{\mathcal{B}} P^{\mathcal{I}_b}(a,b) \cup^{\mathcal{B}} C^{\mathcal{I}_b}(b)\right)$
  According to the semantics in Table 1, the restriction $\forall\, P.C(a)$ in OWL can be seen as an open first-order formula $\forall\, b.\langle a, b\rangle \in P^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}$, and consequently, $\forall\, b.(\neg\ \langle a, b\rangle \in P^{\mathcal{I}} \vee b \in C^{\mathcal{I}})$. Therefore, the universal quantification restriction in BOWL is interpreted as a distributed conjunction (universal quantification) of the union (logical or) of the two components.
- $(\exists\, P.C)^{\mathcal{I}_b}(a) = \bigcup^{\mathcal{B}}_{b \in \Delta^{\mathcal{I}_b}} \left(P^{\mathcal{I}}(a,b) \cap^{\mathcal{B}} C(b)\right)$
  Similar to the universal quantification restriction case above, the restriction $\exists\, P.C(a)$ can be seen as $\exists\, b.(\langle a, b\rangle \in P^{\mathcal{I}} \wedge b \in C^{\mathcal{I}})$.
- $(P\colon o)^{\mathcal{I}_b}(a) = P^{\mathcal{I}_b}(a, o^{\mathcal{I}_b})$.
- $(\geq n\ P)^{\mathcal{I}_b}(a) = \bigcup^{\mathcal{B}} \left(\bigcap^{\mathcal{B}\ n}_{i=1} P^{\mathcal{I}}(a, b_i)\right)$, for all $\{b_1, b_2, \cdots, b_n\} \subseteq \Delta^{\mathcal{I}_b}$ and $\#\{b_1, \cdots, b_n\} = n$.

The interpretation of the maximum number restriction is modeled by finding at least n distinct individuals b_is in the domain $\Delta^{\mathcal{I}_b}$, taking the conjunction of each of the BAF value $P^{\mathcal{I}_b}(a, b_i)$, and taking the distributed disjunction over all such sets of n individuals.

- $(\leq\ n\ P)^{\mathcal{I}_b}(a) = \neg^{\mathcal{B}}\big(\bigcup^{\mathcal{B}}\big(\bigcap^{\mathcal{B}}_{i=1}{}^{n+1} P^{\mathcal{I}}(a, b_i)\big)\big)$, for all $\{b_1, \cdots, b_{n+1}\} \subseteq \Delta^{\mathcal{I}_b}$.
  The interpretation of the minimum number restriction of n can be taken by that of the *negation* of the maximum number restriction of $n+1$, *i.e.,* $(\leq\ n\ P)^{\mathcal{I}}(a) = (\neg\ \geq (n+1)P)^{\mathcal{I}}(a)$.

The duality of the concept relationships in the BOWL interpretation are worth discussing. In the following, the symbol $\cong$ denotes concept equivalence. We have in BOWL $\neg\top \cong \bot$, $C \sqcap \top \cong C$, $C \sqcup \top \cong \top$, $C \sqcap \bot \cong \bot$, $C \sqcup \bot \cong C$, $\neg\neg C \cong C$, $\neg\,(C \sqcup D) \cong (\neg\, C) \sqcap (\neg\, D)$, $\neg\,(C \sqcap D) \cong (\neg\, C) \sqcup (\neg\, D)$, $C_1 \sqcap (C_2 \sqcup C_3) \cong (C_1 \sqcap C_2) \sqcup (C_1 \sqcap C_3)$, $C_1 \sqcup (C_2 \sqcap C_3) \cong (C_1 \sqcup C_2) \sqcap (C_1 \sqcup C_3)$. For concepts involving roles, we have $\neg\ (\forall\, P.C) \cong \exists\, P.(\neg\ C)$, $\forall\, P.\top \cong \top$, $\exists\, P.\bot \cong \bot$ and $(\forall\, P.C) \sqcap (\forall\, P.D) \cong \forall\, P.(C \sqcap D)$. The proof of these equivalence relationships are obvious to see and omitted.

5.3.2 Semantics of BOWL Axioms

Subsumption and equivalence relationships among classes and properties are the essential constructs in OWL. They are considered in this subsection for the BOWL.

- Class subsumption.
  $(C\ \sqsubseteq\ D)^{\mathcal{I}_b} \equiv \bigcap^{\mathcal{B}}\big(\neg^{\mathcal{B}} C^{\mathcal{I}_b}(a) \cup^{\mathcal{B}} D^{\mathcal{I}_b}(a)\big)$, for all a: $\Delta^{\mathcal{I}_b}$
  In OWL, class (and property) subsumption relationships are crisp, meaning that two classes are either of, or not of, subsumption relationship. In BOWL, the subsumption relationships are interpreted as a BAF value.
  The above interpretation is made by viewing $C \sqsubseteq D$ as a first-order logic formula $\forall\, c.c \in C \rightarrow c \in D \equiv \forall\, c.(\neg\ c \in C) \vee c \in D$. Intuitively, the universal quantifier is translated to the distributed BAF conjunction, the logical not is translated to a BAF negation and the logical or is translated to a BAF or.
- Class equivalence.
  $(C = D)^{\mathcal{I}_b} \equiv (C \sqsubseteq D \wedge D \sqsubseteq C)^{\mathcal{I}_b} \equiv (C \sqsubseteq D)^{\mathcal{I}_b} \cap^{\mathcal{B}} (D \sqsubseteq C)^{\mathcal{I}_b}$
  As can be seen, a class equivalence is viewed as two class subsumption relationships, which can subsequently interpreted by BOWL accordingly. As a result, class equivalence is also interpreted as a BAF value.
- Property subsumption.
  $(P \sqsubseteq Q)^{\mathcal{I}_b} \equiv \bigcap^{\mathcal{B}}\big(\neg^{\mathcal{B}} P^{\mathcal{I}_b}(a, b) \cup^{\mathcal{B}} Q^{\mathcal{I}_b}(a, b)\big)$, for all a, b: $\Delta^{\mathcal{I}_b}$
  For the same reason as class subsumption, a property subsumption $P \sqsubseteq Q$ is viewed as the first-order formula $\forall\, a_1, a_2.(a_1, a_2) \in P \rightarrow (a_1, a_2) \in Q \equiv \forall\, a_1, a_2.(\neg\,(a_1, a_2) \in P) \vee (a_1, a_2) \in Q$.
- Property equivalence.
  $(P = Q)^{\mathcal{I}_b} \equiv (P \sqsubseteq Q \wedge Q \sqsubseteq P)^{\mathcal{I}_b} \equiv (P \sqsubseteq Q)^{\mathcal{I}_b} \cap^{\mathcal{B}} (Q \sqsubseteq P)^{\mathcal{I}_b}$
  The interpretation of property equivalence in BOWL can be obtained like that of class equivalence.

5.3.3 Semantics of BOWL Assertions

As shown in Section 5.2, the BOWL interpretation is of the form $\langle \alpha, b \rangle$, where α is an assertion in OWL and b is a value in BAF. For assertion $\langle \alpha, b \rangle$, its interpretation is just the BAF value b.

5.4 Knowledge Base, Satisfiability and Entailment

As for the OWL case, the BOWL knowledge base consists of a finite number of BOWL axioms and assertions. We denote the knowledge base by Σ, the TBox by Σ_T and the ABox by Σ_A.

A BOWL interpretation $\mathcal{I}_b$ satisfies a knowledge base Σ iff it satisfies all of its elements (axioms and/or assertions). Then the interpretation is a *model* of the knowledge base.

The satisfiability of an axiom/assertion of the form $\langle \alpha, b \rangle$ in Fig. 4 by a BOWL interpretation is denoted by $\mathcal{I}_b \models \langle \alpha, b \rangle$.

Since BAF is a pair of values in the range of $[0, 1]$, a single value needs to be derived from this pair of values to determine the satisfiability of the axiom by the interpretation.

The Utility function $\mathtt{U}$ defined in BAF (Section 4) is a normalized version of the Degree of Inclination, which is the difference between the belief and disbelief values. Given a BAF value, its utility gives the overall truth value, ideally suited for the above purpose.

Therefore, $\mathcal{I}_b \models \langle \alpha, b \rangle$ iff $\mathtt{U}(\alpha^{\mathcal{I}_b}) \geq \mathtt{U}(b)$, where α can be an axiom, $C_1 \sqsubseteq C_2$, $C_1 = C_2$ or $C_1 \sqcap C_2 = \bot$; or an assertion $\langle a \rangle \in C$ or $\langle a_1, a_2 \rangle \in P$, etc. The interpretation of α is, as given in the previous two subsections, a BAF value. Hence, the utility function $\mathtt{U}$ gives a single value for comparison. If the utility of the interpretation of α is greater than or equal to that of b, then we conclude that the axiom is satisfied by the interpretation.

A knowledge base Σ entails an axiom $\langle \alpha, b \rangle$, denoted by $\Sigma \models \langle \alpha, b \rangle$, iff it is satisfied by each of the models

(interpretation) of Σ. Similarly, a knowledge base Σ entails an assertion $\langle \alpha, b \rangle$, denoted by $\Sigma \models \langle \alpha, b \rangle$, iff all its models satisfies $\langle \alpha, b \rangle$. A BOWL interpretation $\mathcal{I}_b$ satisfies an assertion $\langle a \in C, b \rangle$ iff $\mathtt{U}(C^{\mathcal{I}_b}(a^{\mathcal{I}_b})) \geq \mathtt{U}(b)$. The same applies to the case of property assertions of the form $\langle \langle a_1, a_2 \rangle \in P, b \rangle$, *i.e.,* $\mathtt{U}(P^{\mathcal{I}_b}(a_1^{\mathcal{I}_b}, a_2^{\mathcal{I}_b}) \geq \mathtt{U}(b)$.

Modus ponens on classes and properties are supported in BOWL. Assume b, d are BAF values such that $\mathtt{U}(b) > \mathtt{U}(\neg^{\mathcal{B}} d)$. For classes, $\{\langle a \in C, b \rangle, \langle a \in \neg\, C \sqcup D, d \rangle\} \models \langle a \in D, d \rangle$ holds. For properties, $\{\langle \langle a, b \rangle \in P, b \rangle, \langle a \in \forall\, P.C \rangle, d \rangle\} \models \langle a \in C, d \rangle$ holds.

Other functions, such as the degree of inclination `DI`, can also be used in the place of `U` according to the user's needs and the source of the belief/disbelief values.

6 Reasoning about BOWL

As we have discussed earlier, each fact in OWL is augmented with a BAF, a pair consisting of a belief value and a disbelief value. In this section, we present some algorithms for BOWL class membership and property membership entailment. More specifically, given an ontology and a BOWL class or property membership assertion, the algorithm determines if the ontology entails the BOWL assertion.

6.1 Class Membership

We explain the algorithm for class membership. Algorithm 1 is simply the outer algorithm which calls Algorithm 2 to compute the BAF values of a class membership assertion in an ontology.

Data: Ontology $\mathcal{O}$, and BOWL assertion $\langle a \in \mathcal{C}, \beta \rangle$ where a is an individual, $\mathcal{C}$ is a class description and β is a BAF
Result: Returns true if $\mathcal{O} \models \langle a \in \mathcal{C}, \beta \rangle$ and false otherwise
compute the belief values, γ, of $a \in \mathcal{C}$ in $\mathcal{O}$ by Algorithm 2;
return $\gamma \geq^{\mathcal{B}} \beta$;

Algorithm 1: BOWL class membership assertion entailment

Data: Ontology $\mathcal{O}$ and OWL assertion $a \in C$ where a is an individual and C is a class description
Result: Computes the BAF values of $a \in C$ in $\mathcal{O}$, or compute($\mathcal{O}$, a, $\mathcal{C}$)
if $\langle a \in \mathcal{C}, \beta \rangle$ *is given in* O **then**
  return β;
else
  if $\mathcal{C}$ *is a class name* **then**
    $\beta_1 \leftarrow \phi_{\mathtt{zero}}$;
    for $D_i \sqsubseteq \mathcal{C}$ **do**
      $\beta_1 \leftarrow \beta_1 \cup^{\mathcal{B}}$ compute($\mathcal{O}$, a, $\mathcal{D}_i$);
    end
    $\beta_2 \leftarrow \phi_{\mathtt{zero}}$;
    for $D_i = \mathcal{C}$ *or* $\mathcal{C} = D_i$ **do**
      $\beta_2 \leftarrow \beta_2 \cup^{\mathcal{B}}$ compute($\mathcal{O}$, a, $\mathcal{D}_i$);
    end
    $\beta_3 \leftarrow \phi_{\mathtt{zero}}$;
    for ≥ 1 $P_i \sqsubseteq \mathcal{C}$ **do**
      $\gamma \leftarrow \phi_{\mathtt{zero}}$;
      for $(a, a_j) \in P$ **do**
        $\gamma \leftarrow \gamma \cup^{\mathcal{B}}$ compute($\mathcal{O}$, a, a_j, P_i);
      end
      $\beta_3 \leftarrow \beta_3 \cup^{\mathcal{B}} \gamma$;
    end
    $\beta_4 \leftarrow \phi_{\mathtt{zero}}$;
    for $\top \sqsubseteq \forall\, P.\mathcal{C}$ **do**
      $\gamma \leftarrow \phi_{\mathtt{zero}}$;
      for $(a_j, a) \in P$ **do**
        $\gamma \leftarrow \gamma \cup^{\mathcal{B}}$ compute($\mathcal{O}$, a, a_j, P_i);
      end
      $\beta_4 \leftarrow \beta_4 \cup^{\mathcal{B}} \gamma$;
    end
    return $\beta_1 \cup^{\mathcal{B}} \beta_2 \cup^{\mathcal{B}} \beta_3 \cup^{\mathcal{B}} \beta_4$;
  endif
  else if $\mathcal{C}$ *is* $\top$ **then**
    return $\phi_{\mathtt{one}}$;
  endif
  else if $\mathcal{C}$ *is* $\bot$ **then**
    return $\phi_{\mathtt{zero}}$;
  endif
  else if $\mathcal{C}$ *is* $\mathcal{C}_1 \sqcup \mathcal{C}_2$ **then**
    return compute($\mathcal{O}$, a, $\mathcal{C}_1$) $\cup^{\mathcal{B}}$ compute($\mathcal{O}$, a, $\mathcal{C}_2$);
  endif
  else
    omitted...
  endif
endif

Algorithm 2: Computing class membership assertion belief values

Algorithm 2 works as follows. It first checks if the requested BAF values are already given in the ontology. If so, it halts and returns it directly. Otherwise it computes the BAF values according to the types of the class description. If it is the top or bottom class, it simply returns $\phi_{\mathtt{one}}$ and $\phi_{\mathtt{zero}}$ respectively. If it is a class name, it computes the BAF values by four axioms, namely class subsumption, class equivalence, property domain and property range, doing a BAF disjunction of the four results which are obtained by recursively calling Algorithm 2 and 4. If the class description is a class union, we return the BAF union of the BAF values for the individual to be the instance of the two classes. The other cases are closely related to the semantics of class descriptions introduced in Section 5.3 and are left out for brevity.

6.2 Property Membership

The reasoning algorithm for property membership is simpler than that for class membership, because neither OWL nor BOWL allows the notion of property description; we can describe a property only by referencing its name. Thus the reasoning algorithms are described in Algorithm 3 and 4, where Algorithm 3 is the outer algorithm for property membership entailment and Algorithm 4 is for computing the belief values of a given property membership assertion. In Algorithm 4, we only consider sub property, equivalent property, inverse property and transitive property relationships and take the BAF disjunction when computing the belief values.

Data: Ontology $\mathcal{O}$, and BOWL assertion $\langle(a_1, a_2) \in P, \beta\rangle$ where a_1 and a_2 are two individual, P is a property name and β is a BAF
Result: Returns true if $\mathcal{O} \models \langle(a_1, a_2) \in P, \beta\rangle$ and false otherwise
compute the belief values, γ, of $(a_1, a_2) \in P$ in $\mathcal{O}$ by Algorithm 4;
return $\gamma \geq^{\mathcal{B}} \beta$;

Algorithm 3: BOWL property membership assertion entailment

Data: Ontology $\mathcal{O}$ and OWL assertion $(a_1, a_2) \in P$ where a_1, a_2 are individuals and P is a property name
Result: Computes the BAF values of $(a_1, a_2) \in P$ in $\mathcal{O}$, or compute($\mathcal{O}$, a_1, a_2, P)
if $\langle(a_1, a_2) \in P, \beta\rangle$ *is given in* O **then**
  return β;
else
  $\beta_1 \leftarrow \phi_{\texttt{zero}}$;
  for $Q_i \sqsubseteq P$ **do**
    $\beta_1 \leftarrow \beta_1 \cup^{\mathcal{B}}$ compute($\mathcal{O}$, a_1, a_2, Q_i);
  end
  $\beta_2 \leftarrow \phi_{\texttt{zero}}$;
  for $Q_i = P$ *or* $P = Q_i$ **do**
    $\beta_2 \leftarrow \beta_2 \cup^{\mathcal{B}}$ compute($\mathcal{O}$, a_1, a_2, Q_i);
  end
  $\beta_3 \leftarrow \phi_{\texttt{zero}}$;
  for $P = (^- Q_i)$ *or* $(^- Q_i) = P$ **do**
    $\beta_3 \leftarrow \beta_3 \cup^{\mathcal{B}}$ compute($\mathcal{O}$, a_1, a_2, Q_i);
  end
  $\beta_4 \leftarrow \phi_{\texttt{zero}}$;
  if $Tr(P)$ **then**
    for $(a_1, b_i), (b_i, a_2) \in P$ **do**
      $\gamma =$ compute($\mathcal{O}$, a_1, b_i, Q_i) $\cap^{\mathcal{B}}$ compute($\mathcal{O}$, b_i, a_2, Q_i) $\beta_4 \leftarrow \beta_4 \cup^{\mathcal{B}} \gamma$;
    end
  else
  endif
  return $\beta_1 \cup^{\mathcal{B}} \beta_2 \cup^{\mathcal{B}} \beta_3 \cup^{\mathcal{B}} \beta_4$;
endif

Algorithm 4: Computing property membership assertion belief values

6.3 Simple Implementation in CLP($\mathcal{R}$)

Following the reasoning algorithms above, we have implemented a simple reasoner for BOWL in CLP($\mathcal{R}$). In this section, we present some preliminary definitions of CLP before we describe our implementation. Some CLP programme fragments are illustrated in the appendix.

Constraint Logic Programming (CLP) [14] began as a natural combination of two declarative paradigms: constraint solving and logic programming. The CLP scheme defines a class of languages based upon the paradigm of rule-based constraint programming, where CLP($\mathcal{R}$) [15] is an instance of this class with the special support of real numbers.

A CLP atom is of the form $p(t_l, \ldots, t_n)$ where p is a predicate symbol distinct from $=$, $<$, and $\leq$, and $t_l, \ldots, t_n$ are terms which can be predicates, variables or constants. A variable starts with a upper-case letter whereas a constant starts with a lower-case letter.

A CLP rule is of the form A_0 :- $\alpha_1, \ldots, \alpha_k$ where each α_i, is either a primitive constraint (such as an arithmetic comparison) or an atom. The atom A_0 is called the head of the rule while the remaining atoms and primitive constraints are known collectively as the body of the rule. In case there are no atoms in the body, we may call the rule a fact or a unit rule.

A CLP program is defined as a finite set of rules. Rules in CLP have much the same format as those in PROLOG except that primitive constraints may appear with atoms in the body. The same applies to a CLP goal which is of the form ? - $\alpha_1, \ldots, \alpha_k$. where each α_i, is either a primitive constraint or an atom.

For class membership computations, we define a CLP predicate `computeClass(I,C,A,B)` with `I` being an individual, `C` being a class (which can be both a class name or a complicated class description), `A` being the belief value and `B` being the disbelief value for the particular class membership assertion. For property membership computations, we similarly define a CLP predicate `computeClass(I1,P,I2,A,B)` with `I1` and `I2` being individuals, `P` being a property name, `A` being the belief value and `B` being the disbelief value for the particular property membership assertion. Then for the entailment of class membership assertions, we define a CLP predicate `entails(instance(I,C),A,B)` with `I` being an individual, `C` being a class, and A and B being the belief and disbelief values respectively. Similarly for the entailment of property membership assertions, we

define a predicate `entails(sub_val(I1,P,I2),A,B)` with `I1` and `I2` being individuals, `P` being a property name, `A` being the belief value and `B` being the disbelief value for the particular property membership assertion. Then the algorithms can be easily converted to CLP($\mathcal{R}$) programmes. A partial code listing for Algorithm 2 can be found in Appendix I. Because our CLP program is highly recursive, we apply the coinductive tabling [16] techniques to prevent infinite loops and reduce unnecessary invocations of the rules.

7 Case Study

In this section, we present an example in the sensor fusion domain to demonstrate the derivation of belief values using BAF-Logics. Sensor fusion [6] technologies aim at fusing information from different sensors (possibly of different types) to detect, recognize, identify or track a target. Sensor fusion has important applications in the defense domain where accurate sensor decisions minimize casualty and improve strike efficiency.

Decision fusion [6] is a branch of sensor fusion technology where sensors are combined in various configurations (parallel, serial, etc.) and their decisions are given confidence factors by the decision fusion processor, which calculates the final decision after a number of iterations.

We believe that decision fusion can be a new application domain for the Semantic Web as sensors may reside at different geographical sites and communications between sensors and the decision fusion processor can be expressed in terms of ontologies to maximize portability and inter-operability of the sensor networks.

7.1 The Sensor Ontology

We developed a BOWL ontology which defines taxonomies of sensors, environmental conditions, targets, etc. It also defines an object-property *isAffectedBy*, capturing the fact that sensors are affected by environmental conditions. Furthermore we define a class *CurrentCondition* as a sub class of *Environment* and a BAF value is attached to every environment condition instance to capture how certain we are about the presence of the condition in the working environment of the sensors. Ontology fragments are presented in the DL syntax as in Section 3. Note that whenever the BAF value is omitted from the quadruple, it is assumed to be $\phi_{\texttt{one}}$, which is $\langle 1, 0\rangle$.

$Sensor \sqsubseteq \top$
$ActiveSensor \sqsubseteq Sensor$
$PassiveSensor \sqsubseteq Sensor$
$EMFrequencySensor \sqsubseteq Sensor$
$OtherSensor \sqsubseteq Sensor$
$\geq 1\ is\_affected\_by \sqsubseteq Sensor$
$\top \sqsubseteq \forall\, is\_affected\_by.CurrentCondition$
$Environment \sqsubseteq \top$
$CurrentCondition \sqsubseteq Environment$
$LightCondition \sqsubseteq Environment$
$TerrainCondition \sqsubseteq Environment$
$WeatherCondition \sqsubseteq Environment$
$RainCondition \sqsubseteq WeatherCondition$
$SmokeConditioin \sqsubseteq WeatherCondition$
$WindCondition \sqsubseteq WeatherCondition$

Sensors and sensor networks identify targets. Thus we define a class called *Target* and an object-property called *identifies* as follows. The object property *identifies* has as domain the union of the classes *Sensor* and *SensorNetwork* and has as range *Target*.

$Target \sqsubseteq \top$
$\geq 1\ identifies \sqsubseteq (Sensor \sqcup SensorNetwork)$
$(\top \sqsubseteq \forall\, identifies.Target)$

Certain kinds of sensors are more affected by certain environmental conditions than others. To capture this information in BOWL, we could have defined some sub properties of *isAffectedBy*, such as *isSlightlyAffectedBy* and *isSeverelyAffectedBy*. Unfortunately, such properties are by no means clear or meaningful to software agents, decision fusion processors in this case. Linguistic hedges like "very" and "quite" are impossible to represent in bi-valued logic systems such as classic description logics, but can be easily captured in belief systems such as BAF-Logic. In our example, different sensors are affected by various environmental conditions differently. We unify several properties into a single property *isAffectedBy*. Then its fuzzy set [29] is

$$\{completely, severely, moderately, slightly, not\}$$

Now, we can assign a BAF supporting value to each element in the fuzzy set to express the extent to which a certain sensor is affected by an environmental condition. Following the conventions used in fuzzy sets, the *isAffectedBy* set may be modeled as

$$\{1/completely, 0.75/severely, 0.5/moderately, 0.25/slightly, 0/not\}$$

Here we assume complete knowledge (no ignorance) of the sensors and compute the refuting masses by subtracting the supporting masses from 1. Then we get the BAF values $\langle 1, 0\rangle$, $\langle 0.75, 0.25\rangle$, $\langle 0.50, 0.50\rangle$, $\langle 0.25, 0.75\rangle$ and $\langle 0, 1\rangle$ respectively.

The following shows how BAFs are added to ground facts (instances). We know the existence of *Speed7Wind1*, a wind instance and a chemical sensor *ChmSensor1*. *ChmSensor1* is affected by *Speed7Wind1* with supporting and refuting measures of 0.75, 0.25 respectively.

$\langle Speed7\,Wind1 \in WindCondition, \langle 1.0, 0.0\rangle\rangle$
$\langle Speed7\,Wind1 \in CurrentCondition, \langle 0.8, 0.1\rangle\rangle$
$\langle ChmSensor1 \in ChemicalSensor, \langle 1.0, 0.0\rangle\rangle$
$\langle (ChmSensor1, Speed7\,Wind1) \in isAffectedBy, \langle 0.75, 0.25\rangle\rangle$

7.2 Computing Confidence Values of Sensors

As stated in [6], initial sensor confidence values are important for the overall system performance. In our examples, the initial sensor confidence values are an effect of the conjunction of several environment conditions. We define a class *TrustedSensor* to denote the belief that the decision fusion processor puts in a particular sensor.

$$TrustedSensor \equiv Sensor \sqcap \forall\, isAffectedBy.CurrentCondition$$

This axiom states that the degree of a sensor instance is trusted depends on how it is affected by current conditions. As defined in the semantics, if a certain sensor is affected by more than one environmental condition, the conjunction of all these conditions is taken into account. For example, night vision devices are severely affected by both rain conditions and smoke conditions. We first define the class *NightVisionDevice* as follows. All subsumption axioms have BAF value of $\phi_{\texttt{one}}$.

$EMFrequencySensor \sqsubseteq Sensor$
$ElectroOpticalSensor \sqsubseteq EMFrequencySensor$
$NightVisionDevice \sqsubseteq ElectroOpticalSensor$

Suppose that a night-vision device nvd_1 is currently deployed in an area where both rain $rain_1$ and smoke $smoke_1$ are present. The sensor fusion processor has certain belief value for each condition to be *current*. Since night vision devices are severely affected by these conditions, we associate a BAF value $\langle 0.75, 0.25\rangle$ to each of the property instances for isAffectedBy as follows.

$\langle nvd_1 \in NightVisionDevice, \langle 1.0, 0.0\rangle\rangle$
$\langle rain_1 \in RainCondition, \langle 1.0, 0.0\rangle\rangle$
$\langle smoke_1 \in SmokeCondition, \langle 1.0.0.0\rangle\rangle$
$\langle rain_1 \in CurrentCondition, \langle 0.7, 0.3\rangle\rangle$
$\langle smoke_1 \in CurrentCondition, \langle 0.5, 0.5\rangle\rangle$
$\langle (nvd_1, rain_1) \in isAffectedBy, \langle 0.75, 0.25\rangle\rangle$
$\langle (nvd_1, smoke_1) \in isAffectedBy, \langle 0.75, 0.25\rangle\rangle$

Our goal is to see whether the knowledge entails the assertion that nvd_1 is a trusted sensor with belief value $\langle 0.7, 0.3\rangle$. So we formulate a CLP goal `entails( instance(nvd1,trustedSensor),0.7,0.3)`. The programme terminates and returns a "No". This means that the BAF values of the assertion calculated from the BOWL ontology gives a smaller utility value than $\langle 0.7, 0.3\rangle$ does. If we are interested in finding the precise BAF values of the specific assertion we can fire the CLP goal `computeClass(nvd1,trustedSensor,A,B)`. Then the programme returns `A = 0.5` and `B = 0.5`. Further analysis shows $\texttt{U}(\langle 0.5, 0.5\rangle) = 0.5 < 0.7 = \texttt{U}(\langle 0.7, 0.3\rangle)$. Therefore, $\texttt{U}(\alpha) = 0.7 > \texttt{U}(\langle 0.5, 0.5\rangle) = 0.5$. Hence, sensor nvd_1 cannot be inferred to be a trusted sensor with that high confidence.

8 Conclusion

The Semantic Web has been designed as a ubiquitous information medium where autonomous software agents can carry out complex tasks. The ontology language OWL is a core language as it semantically marks up web resources. It is based on description logic, where any formula can be inferred from fallacy. Hence, OWL reasoning engines are not capable of performing useful reasoning services in the presence of incomplete or inconsistent information This presents a challenge for engineering complex software agent systems on the Semantic Web.

In this paper, we propose BOWL, Belief-augmented OWL, as an ontology language enriched with belief information. As an extension of OWL DL, the main contribution of BOWL is the ability to associate belief/disbelief factors directly with web resources, enabling software agents to perform more flexible and accurate reasoning. We define the abstract syntax of BOWL and augment the model-theoretic semantics of OWL to incorporate belief values. We also define the reasoning tasks and algorithms for BOWL and present a prototype implementation using the constraint logic programming technique. We also present an example in the sensor fusion domain to demonstrate the reasoning process in BOWL.

The Semantic Web Rules Language (SWRL) [13] enhances the expressivity of the Semantic Web by augmenting OWL with Horn-style rules. We are investigating the effects of adding belief values to such rules.

References

1. Adams JB (1985) Probabilistic Reasoning and Certainty Factors. In: Buchanan BG, Shortliffe EH

(eds) Rule Based Expert Systems, pp 263–271
2. Adlassnig KP, Kolarz G, Scheithauer W, Effenberger H, Grabner G (1985) CADIAG: Approaches to Computer-assisted Medical Diagnosis. Computers in Biology and Medicine 15(5):315–335
3. Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider PF (eds) (2003) The Description Logic Handbook – Theory, Implementation and Applications. Cambridge University Press
4. Berners-Lee T, Hendler J, Lassila O (2001) The Semantic Web. Scientific American 284(5):35–43
5. Brickley D, (editors) RG (2004) Resource description framework (rdf) schema specification 1.0, http://www.w3.org/TR/rdf-schema/
6. Dasarathy BV (1994) Decision Fusion. IEEE Computer Society Press, Los Alamitos, CA, USA
7. Dean M, (editors) GS (2004) OWL Web Ontology Language Reference, http://www.w3.org/TR/2004/REC-owl-ref-20040210/
8. Dempster AP (1967) Upper and Lower Probabilities Induced by Multivalued Mapping. Annals of Mathematical Statistics 38
9. Ding Z, Peng Y (2004) A Probabilistic Extension to Ontology Language OWL. In: Proceedings of the 37th Hawaii International Conference On System Sciences (HICSS-37).
10. van Harmelen F, Patel-Schneider PF, (editors) IH (2001) Reference description of the DAML+OIL ontology markup language, http://www.daml.org/2000/12/reference.html
11. Heckerman D, Horvitz E, Nathwani B, Heckerman DE, Horvitz EJ, Nathwani BN (1989) Update on the Pathfinder Project. In: In Proceedings of the 13th Symposium on Computer Applications in Medical Care, pp 203–207
12. Horrocks I, Patel-Schneider PF, van Harmelen F (2003) From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. Journal of Web Semantics 1(1):7–26
13. Horrocks I, Patel-Schneider PF, Boley H, Tabet S, Grosof B, Dean M (2004) SWRL: A Semantic Web Rule Language Combining OWL and RuleML, http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/
14. Jaffar J, Lassez JL (1987) Constraint logic programming. In: Proceedings of the 14th Annual ACM Symposium on Principles of Programming Languages, pp 111–119
15. Jaffar J, Michaylov S, Stuckey PJ, Yap R (1992) The CLP($\mathcal{R}$) language and system. Transactions on Programming Languages and Systems 14(3):339–395
16. Jaffar J, Santosa A, Voicu R (2005) Modeling Systems in CLP. In: International Conference on Logic Programming, pp 412–413
17. Koller D, Levy A, Pfeffer A (1997) P-Classic: A Tractable Probabilistic Description Logic. In: Proceedings of the AAAI-97, pp 390–397
18. Minsky M (1981) A Framework for Representing Knowledge. In: Mind Design: Philosophy, Psychology, Artificial Intelligence, pp 95–128
19. Ng KC, Abramson B (1990) Uncertainty Management in Expert Systems. IEEE Intelligent Systems 5:29–48
20. Nottelmann H, Fuhr N (2004) pDAML+OIL: A probabilistic extension to DAML+OIL based on probabilistic Datalog. In: Proceedings of the 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-based systems (IPMU'04)
21. Patel-Schneider PF, Hayes P, (editors) IH (2004) OWL Web Ontology Semantics and Abstract Syntax, http://www.w3.org/TR/2004/REC-owl-semantics-20040210/
22. Picard J (2000) Probabilistic argumentation systems applied to information retrieval. PhD thesis, Universite de Neuchatel, Suisse
23. Shafer G (1976) A Mathematical Theory of Evidence. Princeton University Press
24. Smets P (2000) Belief functions and the transferrable belief model, http://ippserv.rug.ac.be/documentation/belief/belief.pdf
25. Straccia U (2001) Reasoning within Fuzzy Description Logics. Journal of Artificial Intelligence Research 14:137–166
26. Straccia U (2006) A fuzzy description logic for the Semantic Web. In: Sanchez E (ed) Fuzzy Logic and the Semantic Web, Capturing Intelligence, vol 1, Elsevier, chap 4, pp 73 – 90, DOI http://dx.doi.org/10.1016/S1574-9576(06)80006-7, URL http://www.sciencedirect.com/science/article/pii/S1574957606800067
27. Tan CKY (2003) Belief Augmented Frames. PhD thesis, National University of Singapore
28. Tresp C, Molitor R (1998) A Description Logic for Vague Knowledge. In: Proceedings of the 13th Biennial European Conference on Artificial Intelligence (ECAI'98), pp 361–365
29. Zadeh LA (1965) Fuzzy Sets. Information and Control 8:338–353

Appendix I: Partial CLP implementation for Algorithm 2

```
computeClass(_, top, 1, 0).
```

```
computeClass(_, bottom, 0, 1).

computeClass(I, oneOf(L), A, B) :-
  member(I, L) -> (A = 1, B = 0); (A = 0, B = 1).

computeClass(I, union(C1, C2), A, B) :-
  computeClass(I, C1, A1, B1),
  computeClass(I, C2, A2, B2),
  bafUnion([[A1, B1], [A2, B2]], [A, B]).
computeClass(I, intersection(C1,C2), A, B) :-
  computeClass(I, C1, A1, B1),
  computeClass(I, C2, A2, B2),
  bafIntersection([[A1, B1], [A2, B2]], [A, B]).
computeClass(I, negation(C), A, B) :-
  computeClass(I, C, A1, B1),
  bafNegation([A1, B1], [A, B]).

computeClass(I, allValuesFrom(P, C), A, B) :-
  getAllInstances(L),
  computeAVF(I, L, P, C, A, B).
computeClass(I, someValuesFrom(P, C), A, B) :-
  getAllInstances(L),
  computeSVF(I, L, P, C, A, B).
computeClass(I, hasValue(P, V), A, B) :-
  computeProperty(I, P, V, A, B).

computeClass(I, C, A, B) :-
  getAllSubclasses(C, CL1),
  computeSubclasses(I, CL1, A1, B1),
  getAllEquivalentClasses(C, CL2),
  computeEquivalentClasses(I, CL2, A2, B2),
  getAllDomains(C, PL1),
  computeDomains(I, PL1, A3, B3),
  getAllRanges(C, PL2),
  computeRanges(I, PL2, A4, B4),
  bafUnion([[A1, B1], [A2, B2], [A3, B3], [A4, B4]], [A, B]).

computeSubclasses(_, [], 0, 1).
computeSubclasses(I, [H|T], A, B) :-
  computeClass(I, H, A1, B1),
  computeSubsclasses(I, T, A2, B2),
  bafUnion([[A1, B1], [A2, B2]], [A, B]).

computeEquivalentClasses(_, [], 0, 1).
computeEquivalentClasses(I, [H|T], A, B) :-
  computeClass(I, H, A1, B1),
  computeEquivalentClasses(I, T, A2, B2),
  bafUnion([[A1, B1], [A2, B2]], [A, B]).

computeDomains(_, [], 0, 1).
computeDomains(I, [H|T], A, B) :-
  getAllImages(I, H, IL),
  computeImages(I, H, IL, A1, B1),
  computeDomains(I, T, A2, B2),
  bafUnion([[A1, B1], [A2, B2]], [A, B]).

computeImages(_, _, [], 0, 1).
computeImages(I, P, [H|T], A, B) :-
  computeProperty(I, P, H, A1, B1),
  computeImages(I, P, T, A2, B2),
  bafUnion([[A1, B1], [A2, B2]], [A, B]).

computeRanges(_, [], 0, 1).
computeRanges(I, [H|T], A, B) :-
  getAllSources(I, H, IL),
  computeSources(I, H, IL, A1, B1),
  computeRanges(I, T, A2, B2),
  bafUnion([[A1, B1], [A2, B2]], [A, B]).

computeSources(_, _, [], 0, 1).
computeSources(I, P, [H|T], A, B) :-
  computeProperty(H, P, I, A1, B1),
  computeSources(I, P, T, A2, B2),
  bafUnion([[A1, B1], [A2, B2]], [A, B]).

computeAVF(_, [], _, _, 1, 0).
computeAVF(I, [H|T], P, C, A, B) :-
  computeProperty(I, P, H, A1, B1),
  bafNegation([A1, B1], [A2, B2])
  computeClass(H, C, A3, B3),
  bafUnion([[A2, B2], [A3, B3]], [A4, B4]),
  computeAVF(I, T, P, C, A5, B5),
  bafIntersection([[A4, B4], [A5, B5]], [A, B]).

computeSVF(_, [], _, _, 1, 0).
computeSVF(I, [H|T], P, C, A, B) :-
  computeProperty(I, P, H, A1, B1),
  computeClass(H, C, A2, B2),
  bafIntersection([[A1, B1], [A2, B2]], [A3, B3]),
  computeSVF(I, T, P, C, A4, B4),
  bafUnion([[A3, B3], [A4, B4]], [A, B]).
```