TURAMBAR: An Approach to Deal with Uncertainty in Semantic Environments *

David Ausín, Federico Castanedo and Diego López-de-Ipiña

Deusto Institute of Technology, DeustoTech. University of Deusto.
Avda. Universidades 24. 48007 Bilbao, Spain.
{david.ausin, fcastanedo, dipina}@deusto.es
http://www.morelab.deusto.es/

Abstract. Research community has shown a great interest in OWL ontologies as a context modeling tool for semantic environments. OWL ontologies are characterized by its expressive power and are based on description logics. However, they have limitations when dealing with uncertainty and vagueness knowledge. To overcome these caveats, some approaches have been proposed. This work presents a novel approach to deal with uncertainty in semantic environments, called TURAMBAR.

1 Introduction

In Ambient Intelligence applications (AmI), context, could be defined as any data which can be employed to describe the state of an entity (an user, a relevant object, the location, etc) [1]. OWL ontologies are one of the most popular, employed and recommended ways to model context [2]. Several AmI projects model the context with ontologies, such as SOUPA[3], CONON[4] or CoDAMoS [5], to name a few.

However, previous approaches are not flexible enough to deal with inaccurate or incomplete information. Vagueness and uncertainty management has become a hot topic in AmI research. Several approaches have been tried to cope with this kind of uncertainties. Different approaches can be classified into probabilistic theory, possibilistic theory or fuzzy theory. But none of them has obtained enough popularity to turn them into an standard or common approach. So, it seems to be enough room for research and advance in this direction.

In this work, we present a novel approach that will try to combine OWL 2 expressibility and reasoning power with uncertainty through the use of Bayesian Networks with the goal to overcome well-known limitations of OWL. The aim of this work is to determine the main features that the framework will have and to explain some of the implementations options.

The structure of this article is as follows. The next section provides an overview of OWL Web Ontology Language and is relationship to Description

* This work is supported by the Spanish MICINN project TALIS+ENGINE (TIN2010-20510-C04-03).

logics. Besides, it includes a brief description of some OWL reasoners. Then section 3 gives a brief description of the OWL limitations and some of the proposed solutions. Section 4 describes the TURAMBAR proposal in the current initial state and some benefits. Finally, section 5 summarizes this work.

2 A brief introduction to OWL

The OWL Web Ontology Language, or simply OWL, was designed for representing knowledge by the World Wide Web Consortium (W3C). The current version of OWL, OWL 2 [6] was published in 2004. OWL 2 allows ontology engineers to describe the domain of an application. An advantage of modelling the knowledge with OWL 2 is that new knowledge can be inferred from the explicit knowledge, thanks to the use of semantic reasoners. However, OWL 2 does not specify how the inference is realized and only the correct answer is predetermined by the formal semantics. There are two versions of formal semantics: the Direct Semantics [7] and the RDF-Based Semantics [8]. The latter is a semantic extension of "D-Entailment". The former is based on Description Logics (DL) and it is in which we are interested on.

DL provides a formal logic-based semantics and the ability of reasoning as its most important features [9]. The more expressive a logic is, the more complex the reasoning process is. Due to this fact, there are several languages which try to balance the complexity of the inference and the expressive power. Under the direct semantics, OWL 2 can be viewed as an extension to the semantics of $\mathcal{SROIQ}$ [10]. The most remarkable features which are included in OWL and not in $\mathcal{SROIQ}$ are support for data types and punning.

However, there are some cases in which the efficiency of reasoning is more important than expressiveness. For those cases, OWL 2 *profiles* have been created. OWL 2 *profiles* establish expressive restrictions in order to guarantee an scalable reasoning. OWL 2 specification defines the following three *profiles* [11]: OWL 2 EL, OWL 2 QL and OWL 2 RL.

Ontology statements are called axioms and they are divided in the following three groups: (i) *ABox* are axioms which encode information about named individuals, (ii) *TBox* consists on axioms which define the terminology and (iii) *RBox* are axioms which describe relationship between roles.

An important characteristic of OWL, is that it follows the open world assumption. This means that if a fact is not explicitly stated, it is impossible to determine its truthfulness, in contrast to the closed world assumption in which every fact which are not explicitly defined as true are false.

2.1 OWL reasoners

This subsection describes two of the most famous and widely employed OWL reasoners: HermiT and Pellet.

HermiT [12] is a description reasoner for $\mathcal{SROIQ}$, it is distributed under the terms of LGPL which can be accessed through OWLlink or OWL API. The

main difference against other reasoners is the employed reasoning algorithm, which is based on the hypertableau calculus algorithm[13].

HermiT supports DL Safe SWRL rules, but it does not include built-in implementations. Therefore, rules that call built-in atoms can not be executed under HermiT.

Pellet [14] is an OWL 2 reasoner for Java under dual license terms: AGPLv3 for open source projects and a special one for closed source applications. Pellet can be accessed through Jena, OWL API or OWLlink.

It uses a a tableaux algorithm which is able to handle $\mathcal{SROIQ}(D)$ logic and includes several optimizations such as back-jumping, simplification or absorption. Pellet has also techniques for incremental reasoning, such as incremental consistency checking and incremental classification. Besides, it implements DL-Safe SWRL rules support. All the built-ins for SWRL rules described in [15] are provided, with the exceptions of built-ins for *List* and some built-ins for *date*, *time* and *duration*. Anyway, custom built-ins can be developed and registered into the reasoner in order to be called from SWRL rules.

3 Main limitations of OWL

However, OWL is not perfect and has some limitations such as the management of uncertainty and vagueness [16] which have been tried to overcome by several approaches as we show below.

An uncertain statement is a statement which is true or false, but due to the lack of information it is no possible to ensure its value. In contrast to uncertainty, vagueness defines a statement which is true to a certain grade.

In general, modelled application domains contains uncertainty, vagueness or both at the same time. To cope with uncertainty, there are two popular approaches: the possibilistic theory and the probabilistic theory. *Possibilistic theory* define the possibility of an event as the most likely case in which the event occurs. In contrast, in *probabilistic theory* the probability of an event is the amount of favourable case in which an event occurs. Typical reasons which enforce developers to use uncertainty comes by: the requirement to determine the overlapping between classes of different ontologies, information retrieval tasks, handling inconsistency in ontologies or imprecise information about the context. On the other hand, vagueness describes the statement degree of truth. For scenarios in which vagueness is presented, fuzzy logic is usually applied. It has been applied for information retrieval, in medical domain or describing subjective context, to name a few.

Some approaches have appeared to solve these drawbacks, being the most important (grouped by category) the following ones:

- Probability based approaches to overcome OWL limitations
  BayesOWL [17]: translates OWL ontologies into a Bayesian Network acyclic graph. The translation process has two steps. First, a set of rules are applied to create the Bayesian Network. Then the conditional probability tables are

calculated, applying the decomposed iterative proportional fitting procedure (D-IPFP) algorithm. BayesOWL can represent two different types of probabilities: (i) prior or marginal probability and (ii) conditional probability.
Pronto [18]: is a probabilistic OWL reasoner with P-$\mathcal{SHIQ}(\mathcal{D})$ logic support. Pronto allows to encode conditional constrains about TBox and ABox and they are expressed as OWL 2 annotations. One of its goals is to be an alternative to Bayesian approaches. Currently, Pronto's development is not completed and its performance is not quite good.

- Possibility based approaches to overcome OWL limitations
  PossDL [19]: is a possibilistic description logic reasoner. PossDL extends Pellet adding uncertainty reasoning. This reasoner relies on Pellet and OWL API [20].
- Fuzzy logics based approaches to overcome OWL limitations
  fuzzyDL [21]: is a description logic reasoner which supports fuzzy logic and fuzzy rough set reasoning. It includes support for "Zadeh semantics" and Lukasiewicz Logic.
  DeLorean [22]: is a fuzzy rough description logic reasoner, which is able to convert a fuzzy rough ontology into a OWL or OWL 2 ontology. Pellet reasoner, HermiT reasoner or any reasoner which supports OWLlink protocol can be employed with DeLorean.

4 TURAMBAR approach

OWL follows the open world assumption, which defines that a non asserted fact is unknown. However, we believe that sometimes is possible to determine the probability that a fact were true via the relationships and influences that other facts have on this. So, TURAMBAR's main goal is to offer a mechanism to predict the probability that an unknown fact was true under certain conditions.

TURAMBAR proposes to enrich OWL 2 reasoning with a probabilistic extension to deal with uncertainty. To achieve this we pretend to extend the OWL API, adding new methods which allow developers to perform queries about facts which are uncertain. Developers could choose between using traditional reasoning services or extended probabilistic services to answer queries. In other words, TURAMBAR adds a new layer to current description reasoners, such as Pellet, enhancing them with new capabilities to cope with uncertainty via Bayesian Networks. A Bayesian Network is a graphical model that is defined as a directed acyclic graph. The nodes in the model represent the random variables and the edges define the dependencies between the random variables. Each variable is conditionally independent of its non descendants given the value of its parents. They offer a simple solution to describe the relationships between facts and how they influence each other.

In our approach, probability axioms are added via OWL 2 annotations. OWL 2 annotations offers an straightforward, cleaned way to add uncertainty to the ontology. The main advantage of annotations is that they offer an standard mechanism to add extra information about an axiom. Also, Bayesian Networks will be described by annotations.

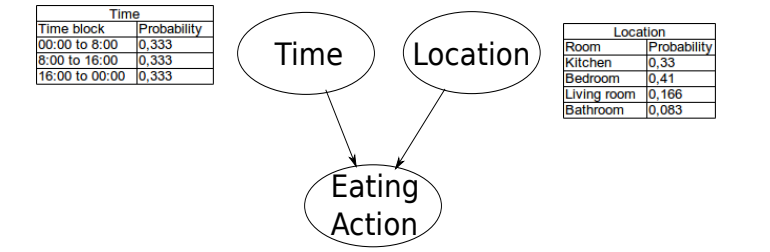
The next example (1.1) aims to illustrate the advantages of this approach. It models a smart home which contains several sensors which detect user's activity and the location of the activity. In this example, we have defined the existing relationship between the probability of an individual classified as *EatingAction* which can be classified also as a subclass of it. As shown the figure 1, an individual which is a subclass of *EatingAction* is influenced by the time and place in which the action took place. The complete ontology of this example can be seen below.

Listing 1.1. Example of an ontology with probabilistic knowledge employed in TURAMBAR

```
Declaration(Class(p0:Action))
Declaration(Class(p0:Bathroom))
SubClassOf(p0:Bathroom p0:Location)
Declaration(Class(p0:Bedroom))
SubClassOf(p0:Bedroom p0:Location)
Declaration(Class(p0:BreakfastAction))
SubClassOf(p0:BreakfastAction p0:EatingAction)
Declaration(Class(p0:DinnerAction))
SubClassOf(p0:DinnerAction p0:EatingAction)
Declaration(Class(p0:EatingAction))
AnnotationAssertion(p0:talismanClassProbability
p0:EatingAction
"ID: EatingAction
InitGraph
Time -> this;
Location -> this;
EndGraph
p0.BreakfastAction :
0.125, 0.1, 0, 0, 0.9, 0.9, 0.6, 0, 0, 0, 0, 0;
p0.LunchAction :
0, 0, 0, 0, 0, 0, 0, 0, 0.27, 0.2, 0.1, 0 ;
p0.DinnerAction :
0.125, 0.1, 0, 0, 0, 0, 0, 0, 0.27, 0.2, 0.1, 0;
p0.SnackingAction:
0.75, 0.8, 1, 1, 0.1, 0.1, 0.4, 1, 0.45, 0.6, 0.8, 1;")
SubClassOf(p0:EatingAction p0:Action)
Declaration(Class(p0:Kitchen))
SubClassOf(p0:Kitchen p0:Location)
Declaration(Class(p0:LaunchAction))
SubClassOf(p0:LaunchAction p0:EatingAction)
Declaration(Class(p0:LivingRoom))
SubClassOf(p0:LivingRoom p0:Location)
Declaration(Class(p0:Location))
Declaration(Class(p0:SnackingAction))
SubClassOf(p0:SnackingAction p0:EatingAction)
Declaration(ObjectProperty(p0:atLocation))
AnnotationAssertion(p0:talismanPropertyProbability
p0:atLocation
"ID: Location
p0.Kitchen : 0.33;
p0.Bedroom : 0.41;
p0.LivingRoom : 0.166;
p0.Bathroom : 0.083;")
FunctionalObjectProperty(p0:atLocation)
ObjectPropertyDomain(p0:atLocation p0:Action)
ObjectPropertyRange(p0:atLocation p0:Location)
Declaration(DataProperty(p0:time))
AnnotationAssertion(p0:talismanPropertyProbability
p0:time "ID: Time
>= 0 && <= 28800 : 0.333;
>= 28800 && <=57600 : 0.333;
>= 57600 && <= 86400 : 0.333;")
FunctionalDataProperty(p0:time)
DataPropertyDomain(p0:time p0:Action)
```

```
DataPropertyRange(p0:time xsd:double)
Declaration(AnnotationProperty(
p0:talismanClassProbability))
Declaration(AnnotationProperty(
p0:talismanPropertyProbability))
```

In the previous example, if a developer was interested to find out the kind of meal that a user is having, he would query the reasoner about the specific action. Then, two different options could take place. One of them is when the activity is correctly classified as a subclass of *EatingAction*. In this case, the developer does not have to employ the probabilistic extension because traditional reasoning (using first-order logic) is enough. However, if the activity is not classified as a subclass due to a lack of information (second case), the developer can be able to query TURAMBAR probabilistic extension to obtain the odds that the action was a $DinnerAction$, $BreakfastAction$, $SnackingAction$ or $LaunchAction$. To answer the probabilistic query, a Bayesian Network and their corresponding conditional probability distributions must be defined (as shown in figure 1). In other words, TURAMBAR can run as an OWL reasoner or as uncertainty reasoner depending on the developer needs.



Time	
Time block	Probability
00:00 to 8:00	0,333
8:00 to 16:00	0,333
16:00 to 00:00	0,333

Location	
Room	Probability
Kitchen	0,33
Bedroom	0,41
Living room	0,166
Bathroom	0,083

Eating Action												
Time	00:00 to 8:00				8:00 to 16:00				16:00 to 00:00			
Location	Kitchen	Bedroom	Living room	Bathroom	Kitchen	Bedroom	Living room	Bathroom	Kitchen	Bedroom	Living room	Bathroom
Meal												
Breakfast	0,125	0,1	0	0	0,9	0,9	0,6	0	0	0	0	0
Lunch	0	0	0	0	0	0	0	0	0,27	0,2	0,1	0
Dinner	0,125	0,1	0	0	0	0	0	0	0,27	0,2	0,1	0
Snacking	0,75	0,8	1	1	0,1	0,1	0,4	1	0,45	0,6	0,8	1

Fig. 1. An example of TURAMBAR Bayesian Network and their corresponding conditional probabilities tables.

Constraints probabilities can be given in the ontology by the ontologist engineer or by the domain expert and will be updated according to historical data.

Two OWL annotations has been defined to describe an axiom probability: (i) *talismanPropertyProbability* and (ii) *talismanClassProbability*. The first one states the probability that an OWL individual with a property x has the value y. On the other hand, *talismanClassProbability* states the probability of an OWL individual which is classified as a member of a class x was also a member of its subclasses. Both annotations can be used to build a probabilistic graph which define the probabilities of an axiom. The annotation content is divided in three parts: (a) the identification, (b) the probabilistic graph definition and (c) the conditional probability distribution. The identification is the first line of a annotation and is mandatory. The probabilistic graph is defined between *InitGraph*

and *EndGraph* tags and is optional. Each line between these tags describes a dependency between two variables. As an example, the *EatingAction* dependencies shown in figure 1 are modeled in this way:

```
Time -> this;
Location -> this;
```

In the example, the keyword *this* is used instead of *EatingAction* id to identify an *EatingAction* variable and its definition must appear before the conditional probability distributions. Lastly, the conditional probability distribution section is mandatory. It is possible to define a probability for a specific value or for a value range (only available for data properties whose ranges are numerics). When the variable depends on several variables, probabilities are expressed regarding to their graph description order. For example, in the previous graph definition, the probability of being an *EatingAction* and also a *BreakfastAction* is described as "*p0.BreakfastAction : 0.125, 0.1, 0, 0, 0.9, 0.9, 0.6, 0, 0, 0, 0, 0;*".

Another feature that TURAMBAR will offer is the creation of rules combining the probabilistic knowledge with the description logic knowledge. The idea behind this feature is to create a set of custom built-ins which can be called from SWRL rules. These features are not available in the previous probabilistic reasoners described. Among the different OWL reasoners, we honestly though that Pellet is the best one to be extended with a probabilistic layer. Besides, it offer a friendly own API interface to developers, the OWL API is chosen. This decision is based on experimental results from previous works [23] [24], where Pellet and OWL API combination has demonstrated to be a good choice to perform a fast reasoning.

One difference between TURAMBAR and BayesOWL is that variables are not binary constrained. In contrast to BayesOWL, TURAMBAR needs that the Bayesian Networks are defined, because it does not rely on the class hierarchy and its restrictions to determine the network and classify the concept. Instead, TURAMBAR assumes that there are relationships between individuals and their properties whose values may influence the probability of the existence of an assumption which has not been able to infer using traditional reasoning service. So, TURAMBAR not only could be used to classify instances, but it could also be used to predict if an individual has a property with a determined value.

5 Conclusion

In this work we present the initial state of a new probabilistic approach to handle uncertainty in AmI environments. The proposed approach combines OWL 2 reasoning with Bayesian Networks to offer new reasoning capabilities to the developers of AmI applications. One of the main advantages comes from its complementary behaviour with OWL 2 reasoning instead of replacing it. Besides, it proposes to extend a very well known API for developers (OWL API) with Bayesian Networks to complement OWL reasoning with probabilistic capabilities.

References

1. Dey, A.: Understanding and using context. Per. and ub. comp. **5**(1) (2001) 4–7
2. Strang, T., Linnhoff-Popien, C.: A context modeling survey. (2004)
3. Chen, H., Perich, F., Finin, T., Joshi, A.: Soupa: Standard ontology for ubiquitous and pervasive applications. In: MOBIQUITOUS 2004, IEEE (2004) 258–267
4. Wang, X., Zhang, D., Gu, T., Pung, H.: Ontology based context modeling and reasoning using owl. In: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. (2004) 18–22
5. Preuveneers, D., et al: Towards an extensible context ontology for ambient intelligence. Ambient intelligence (2004) 148–159
6. Krötzsch, M., et al: OWL 2 web ontology language primer. Technical report, W3C (October 2009)
7. Patel-Schneider, P.F., Motik, B., Grau, B.C.: OWL 2 web ontology language direct semantics. W3C recommendation, W3C (October 2009)
8. Schneider, M.: OWL 2 web ontology language RDF-based semantics. W3C recommendation, W3C (October 2009)
9. Baader, F., Nutt, W.: Basic Description Logics. In the Description Logic Handbook. Cambridge University Press (2002)
10. Horrocks I., a.e.a.: The even more irresistible sroiq. In: Knowledge Reasoning. (2006) 57–67
11. Motik, B., et alu: OWL 2 web ontology language profiles. W3C recommendation, W3C (October 2009)
12. Shearer, R., Motik, B., Horrocks, I.: Hermit: A highly-efficient owl reasoner. In: Proc. of the 5th International Workshop on OWL: Experiences and Directions (OWLED). (2008) 26–27
13. Baumgartner, P., Furbach, U., Niemelä, I.: Hyper tableaux. Logics in Artificial Intelligence (1996) 1–17
14. Sirin, E., et al.: Pellet: A practical owl-dl reasoner. Web Semantics: science, services and agents on the WWW **5**(2) (2007) 51–53
15. Horrocks, I., et al: Swrl: A semantic web rule language combining owl and ruleml, 2004. W3C Submission (2006)
16. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. Web Semantics: Science, Services and Agents on the WWW **6**(4) (2008) 291–308
17. Ding, Z., Peng, Y., Pan, R.: Bayesowl: Uncertainty modeling in semantic web ontologies. Soft Computing in Ontologies and Semantic Web (2006) 3–29
18. Klinov, P., Parsia, B.: Pronto: Probabilistic ontological modeling in the semantic web. In: Proc. of International Semantic Web Conference (Posters Demos). (2008)
19. Qi, G., Ji, Q., Pan, J., Du, J.: Possdl - a possibilistic dl reasoner for uncertainty reasoning and inconsistency handling. The Semantic Web: Research and Applications (2010) 416–420
20. Horridge, M., Bechhofer, S.: The owl api: a java api for working with owl 2 ontologies. (2009)
21. Bobillo, F., Straccia, U.: fuzzydl: An expressive fuzzy description logic reasoner. In: IEEE Int. Conf. on Fuzzy Systems, IEEE (2008) 923–930
22. Bobillo, F., et al: Delorean: A reasoner for fuzzy owl 2. **423** (2008)
23. Ausín, D., Castanedo, F., López-de Ipiña, D.: Benchmarking results of semantic reasoners applied to an ambient assisted living environment. In: ICOST
24. Ausín, D., Castanedo, F., López-de Ipiña, D.: On the measurement of semantic reasoners in ambient assisted living environments. In: IS. (2012)