A Probabilistic OWL Reasoner for Intelligent Environments

David Ausín[1], Diego López-de-Ipiña[1], and Federico Castanedo[2]

[1] Deusto Institute of Technology, DeustoTech. University of Deusto, Avda. de las Universidades, 24, 48007 Bilbao, Spain. {david.ausin, dipina}@deusto.es
[2] Wise Athena. 71 Stevenson Street, San Francisco, USA.
fcastanedo@wiseathena.com *

Abstract. OWL ontologies have gained great popularity as a context modelling tool for intelligent environments due to their expressivity. However, they present some disadvantages when it is necessary to deal with uncertainty, which is common in our daily life and affects the decisions that we take. To overcome this drawback, we have developed a novel framework to compute fact probabilities from the axioms in an OWL ontology. This proposal comprises the definition and description of our probabilistic ontology. Our probabilistic ontology extends OWL 2 DL with a new layer to model uncertainty. With this work we aim to overcome OWL limitations to reason with uncertainty, developing a novel framework that can be used in intelligent environments.

Keywords: OWL, Bayesian networks, probability, probabilistic ontology

1 Introduction

In Ambient Intelligence applications, context can be defined as any data which can be employed to describe the state of an entity (a user, a relevant object, the location, etc.) [6]. How this information is modelled and reasoned over time is a key component of an intelligent environment in order to assist users in their daily activities or execute the corresponding actions. An intelligent environment is any space in which daily activities are enhanced by computation [4].

One of the most popular techniques for context modelling is OWL ontologies [18]. They have been employed in several Ambient Intelligence projects such as SOUPA[3], CONON[20] or CoDAMoS [15], to name a few.

OWL is the common way to encode description logics in real world. However, when the domain information contains uncertainty, the employment of OWL ontologies is less suitable [11]. The need to handle uncertainty has created a growing interest in the development of solutions to deal with it.

As in other domains, uncertainty is also present in Ambient Intelligence [16] and affects to the decision making process. This task requires context information

* This work is supported by the Spanish MICINN project FRASEWARE (TIN2013-47152-C3-3-R)

in order to respond to the users' needs. Data in Ambient Intelligence applications are provided by several sensors and services in real time. Unfortunately, these sensors can fail, run out of battery or be forgotten by the user, in the case of wearable devices. On the other hand, the services can also be inaccessible due to network connectivity problems or technical difficulties on the remote server. Nonetheless, that unavailable information may be essential to answer correctly user's requirements.

For this reason, we present a novel approach to deal with uncertainty in intelligent environments. This work proposes a method to model uncertainty, that combines OWL ontologies with Bayesian networks. The rest of this article is organized as follows. The next section describes the problem that we address. Section 3 explains the semantics and syntax of our proposal. Section 4 gives an exemplary use case where our proposal is applied and describes how to model it. Finally, section 5 summarizes this work and addresses the future work.

2 Description of the Problem

In intelligent environments, the lack of information causes incomplete context information and it may be produced by several causes:

- Sensors that have run out of batteries. Several sensors, such as wearable devices, depend on batteries to work.
- Network problems. Sensors, actuators and computers involved in the environment sensing and monitoring are connected to local networks that can suffer network failures. In these cases, the context information may be lost, although the sensors, actuators and computers are working properly.
- Remote services' failures. Some systems rely on remote services to provide a functionality or to gather context information.
- A system device stops working. Computer, sensors and actuators can suffer software and hardware failures that hamper their proper operation.

When one of these issues occurs, the OWL reasoner will infer conclusions that are insufficient to attend the user's needs. Besides, taking into account that factors can improve several tasks carried in intelligent environments, such as ontology-based activity recognition. For instance, *WatchingTVActivity* can be defined as an activity performed by a *Person* who is watching the television in a room:

$$\begin{aligned} WatchingTVActivity \equiv \exists isDoneBy.(Person \sqcap \exists isIn(Room \sqcap \\ \exists containsAppliance.(TV \sqcap \exists isSwitched.(_{true})))) \end{aligned} \quad (1)$$

If the user is watching the television and the system receives the values of all the sensors, then it is able to conclude that the user's current activity is of the type *WatchingTVActivity*. In contrast, if the value of the presence sensor is not available, then it is not possible to infer that the user is watching the television.

In addition, sometimes there is not a rule of thumb to classify an individual as a member of a class. For instance, we can classify the action that the system has to perform regarding the current activity of the user. Thus, we can define that the system should turn off the television, when the user is not watching it:

$$TurnOffTV \equiv \exists requiredBy.(Person \sqcap \forall isDoing.\neg WatchingTVActivity \sqcap \exists hasAppliance.(TV \sqcap \exists isSwitched.(_{true}))) \quad (2)$$

However, this concept definition does not accurately model the reality. The action can fulfil every condition expressed in the *TurnOffTV* definition, but the television should not be turned off. This situation may occur when the user goes to the toilet or answers a call phone in another room, among others.

In these cases in which the information of the domain comes with quantitative uncertainty or vagueness, ontology languages are less suitable [11]. Uncertainty is usually considered as the different aspects of the imperfect knowledge, such as vagueness or incompleteness. In addition, the uncertainty reasoning is defined as the collection of methods to model and reason with knowledge in which boolean truth values are unknown, unknowable or inapplicable [19]. Other authors [1] [11] consider that there are enough differences to distinguish between uncertainty and vague knowledge. According to them, uncertainty knowledge is comprised by statements that are either true or false, but we are not certain about them due to our lack of knowledge. In contrast, vagueness knowledge is composed of statements that are true to certain degree due to vague notions.

In our work, we are more interested in the uncertainty caused by the lack of information rather than the vague knowledge. For this reason, probabilistic approaches are more suitable to solve our problem.

3 Turambar Solution

Our proposal, called Turambar, combines a Bayesian network model with an OWL 2 DL ontology in order to handle uncertainty. A Bayesian network [13] 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.

Turambar is able to calculate the probability associated to a class, object property or data property assertions. These probabilistic assertions have only two constraints:

- The class expression employed in the class assertion should be a class.
- For positive and negative object property assertions, the object property expression should be an object property.

However, these limitations can be solved declaring a class equivalent to a class expression or an object property as the equivalent of an inverse object property. Examples of probabilistic assertions that can be calculated with Turambar are:

$$isIn(John, Bedroom1)\ 0.7 \tag{3}$$
$$WatchingTVActivity(Action1)\ 0.8 \tag{4}$$
$$isSwitched(TV1, true)\ 1 \tag{5}$$

The probabilistic object property assertion expressed in (3) states that John is in Bedroom1 with a probability of 0.7. On the other hand the probabilistic class assertion (4) describes that the Action1 is member of the class *WatchingTVActivity* with a probability of 0.2. Finally, the probabilistic data property assertion (5) defines that the television, *TV1*, is on with a probability of 1.0. The probability associated to these assertions is calculated through Bayesian networks that describe how other property and class assertions influence each other. In Turambar, the probabilistic relationships should be defined by an expert. In other words, the Bayesian networks must be generated by hand, since learning a Bayesian network is out of the scope of this paper and it is not the goal of this work.

3.1 Turambar Functionality Definition

The classes, object properties and data properties of the OWL 2 DL ontology involved in the probabilistic knowledge are connected to the random variables defined in the Bayesian network model. For example, the OWL class *WatchingTVActivity* is connected to at least one random variable, in order to be able to calculate probabilistic assertions about that class. The set of data properties, object properties and classes that are linked to random variables is called $\mathcal{V}_{prob}$ and a member of $\mathcal{V}_{prob}$, $vprob_i$; such that $vprob_i \in \mathcal{V}_{prob}$.

In Turambar, every random variable (RV) is associated to a $\mathcal{V}_{prob}$ and every RV's domain is composed of a set of functions that determine the values that a random variable can take, such as $Val(RV) = \{f_1, f_2 ... f_n\}$ and $f_i \in Val(RV)$. These functions require a property or class and individual to calculate the probabilistic assertion, such as $f_i : a_1, ex \rightarrow result$ where a_1 is an OWL individual, ex, a class, data property or object property; result, a class assertion, object property assertion, data property assertion or void (no assertion). In the case, that every function in the domain of a random variable returns void, it means that the random variable is auxiliary. In contrast, if any f_i in the domain of a random variable returns a probability associated to an assertion, then the random variable is called final.

For instance, the data property *lieOnBedTime* is linked to a random variable named *SleepTime* whose domain is composed of two functions f_1 that check if the user has been sleeping for less than 8 hours and f_2 function that checks if the user has been sleeping for more than 8 hours. Both functions are not able to generate assertions, so the random variable *SleepTime* is auxiliary. By contrast, *WatchingTVActivity* class is linked to a random variable called *WatchingTV* whose domain comprises f_3 function that checks if an individual is member of the class *WatchingTVActivity* (e.g. $WatchingTVActivity(Activity1)$ 0.8) and

the f_4 function which checks if an individual is a member of the complement of *WatchingTVActivity*.

It is also important to remark that a $vprob_i$ can be referenced from several random variables. For example, the *TurnOffTV* depends on the user's impairments, so if the blind user is deaf, it is more likely that the television needs to be turned off. Additionally, having an impairment also affects to the probability of having another impairment: deaf people have a higher probability of also being mute. In this case, we can link *hasImpairment* object property with two random variables in order to model it.

Apart from the conditional probability distribution, nodes connected between them may have an associated node context. The context defines how different random variables are related between them and the condition that must fulfil. This context establishes an unequivocal relationship in which every individual involved in that relationship should be gatherer before calculating the probability of an assertion. If the relationship is not fulfilled then the probabilistic query cannot be answered. For example, to estimate the probability for the *TurnOffTV*, the reasoner needs to know who is the user and in which room he is. For this case the relationship may be the following one $isIn(?user, ?room) \wedge requiredBy(?action, ?user) \wedge hasAppliance(?user, ?tv)$, being $?user, ?action, ?tv$ and $?room$ variables. So, if we ask for the probability that *Action1* is member of *TurnOffTV*, such as Pr(TurnOffTV(Action1)), then the first step to calculate it is to check its context. If everything is right the evaluation of this relationship should return that the *Action1* is required only by one user who is only in one room and has only one television. Otherwise, the probability cannot be calculated.

Our proposal can be viewed as a $\mathcal{SROIQ}(\mathcal{D})$ extension that includes a probabilistic function Pr which maps role assertions and concept assertions to a value between 0 and 1. The sum of the probabilities obtained for a random variable is equal to 1. In contrast, the sum of probabilities for the set of assertions obtained for a $vprob_i$ may be different from 1. For instance, the object property *hasImpairment* is related to two random variables one to calculate the probability of being deaf and another one to calculate the probability of being mute. If both random variables have a domain with two functions, we can get four probabilistic assertions that sums 2 instead of 1, but the sum of probabilities obtained in one random variable is 1:

- Random variable deaf's assertions: $hasImpairment(John, Deaf)0.8$ and $\neg hasImpairment(John, Deaf)0.2$.
- Random variable mute's assertions: $hasImpairment(John, Mute)0.7$ and $\neg hasImpairment(John, Mute)0.3$.

The probability of an assertion that exists in the OWL 2 DL ontology is always 1 although the data property, object properties or class is not member of V_{prob}. For example, if an assertion states that John is a *Person* ($Person(John)$) and we ask for the probability of this assertion, then its probability is 1, such as $Person(John)1$. However, if the data property, object properties or class is not member of V_{prob} and there is not an assertion in the OWL 2 DL ontology

that states it, then the probability for that assertion is unknown. We consider that the probabilistic ontology is satisfied if the OWL 2 DL ontology is satisfied and the Bayesian network model is not in contradiction with the OWL ontology knowledge.

3.2 Turambar Ontology Specification

In Turambar, a probabilistic ontology comprises an ordinary OWL 2 DL ontology and the dependency description ontology that defines the Bayesian network model.

The ordinary OWL ontology imports the Turambar annotations ontology, which defines the following annotations:

- *turambarOntology* annotation defines the URI of the dependency description ontology.
- *turambarClass* annotation links OWL classes in the ordinary ontology to random variables in the dependency description ontology.
- *turambarProperty* annotation connects OWL data properties and object properties in the ordinary ontology to random variables in the dependency description ontology.

We choose to separate the Bayesian network definition from the ordinary ontology in order to isolate the probabilistic knowledge definition from the OWL knowledge. We define isolation as the ability of exposing an ontology with an unique URI that locates the traditional ontology and the probabilistic one. So, given the URI of a probabilistic ontology, a compatible reasoner loads the ordinary ontology and the dependency description ontology it. In contrast, a traditional reasoner only loads the ordinary ontology. So, if the Turambar probabilistic ontology is loaded by a traditional reasoner, the traditional reasoner does not have access to the knowledge encoded in the dependency description ontology. In this way, we also want to promote the re-utilization of probabilistic ontologies as simple OWL 2 DL ontologies by traditional systems and the interoperability between our proposal and them.

On the other hand, the dependency description ontology defines the probabilistic model employed to estimate the probabilistic assertions. To model that knowledge, it imports the Turambar ontology, which defines the vocabulary to describe the probabilistic model. As the figure 1 shows, the main classes and properties in the Turambar ontology are the following ones:

- *Node* class represents the nodes in Bayesian networks. *Node* instances are defined as auxiliary random variables through the property *isAuxiliar*. The *hasProbabilityDistibution* object property links *Node* instances with their corresponding probability distributions and *hasState* object property associates *Node* instances with their domains. Furthermore, *hasChildren* object property and its inverse *hasParent* set the dependencies between *Node* instances. Finally, *hasContext* object property defines the context for a node and *hasVariable* object property, the value of the variable that the node requires.

- *MetaNode* is a special type of *Node* that is employed with non functional object properties and data properties. Its main functionality is to group several nodes that share a context and are related to the same property. For instance, in the case of the *hasImpairment* object property we can model a *MetaNode* with two nodes: *Deaf* and *Mute*. Both nodes share the same context but have different parents and states. The object property *compriseNode* identifies the nodes that share a context.
- *State* class defines the values of random variables' domain. In other words, it describes the functions which generate probabilistic assertions. These functions are expressed as a string through the data property *stateCondition*.
- *ProbabilityDistribution* class represents a probability distribution. *Probability* distributions are given in form of conditional probability tables. Cells of the conditional probability table are associated to the instances of *ProbabilityDistribution* through *hasProbability* object property.
- *Probability* class represents a cell of a conditional probability table, such $P(x_1 \mid x_2, x_3) = value$, where x_1, x_2 and x_3 are *State* individuals and *value* is the probability value. x_1 *State* is assigned to an instance of *Probability* class through the *hasValue* object property and x_2 and x_3 conditions through *hasCondition* object property. Finally, the data property *hasProbabilityValue* sets the probability value for that cell.
- *Context* class establishes the relationships between the nodes of a Bayesian network. Relationships between nodes are expressed as a SPARQL-DL query through the data property *relationship*.
- *Variable* class represents the variables of the context. Their instances identify the SPARQL-DL variables defined in the context SPARQL-DL query. The *variableName* data property establishes the name of the variable. For example, if the context has been defined with the following SPARQL-DL expression: *select ?a ?b where* { *PropertyValue(p:livesIn, ?a, ?b)*} , then we should create two instances of *Variable* with the *variableName* property value of *a* and *b*, respectively.
- *Plugin* class defines a library that provides some functions that are referenced by *State* class instances and are not included as member of the Turambar core. The core functions are the following ones: (i) numbers and strings comparison, (ii) ranges of number and string comparison, (iii) individual instances comparison, (iv) boolean comparison, (v) class memberships checking and (vi) the void assertion to define the probability that no assertion involves an individual. Only i, iii and iv are able to generate probabilistic assertions. Every function, except the void function, has their inverse function to check if that value has been asserted as false.

4 Related Works

We can classify probabilistic approaches to deal with uncertainty in two groups: probabilistic description logics approaches and probabilistic web ontology languages [11].

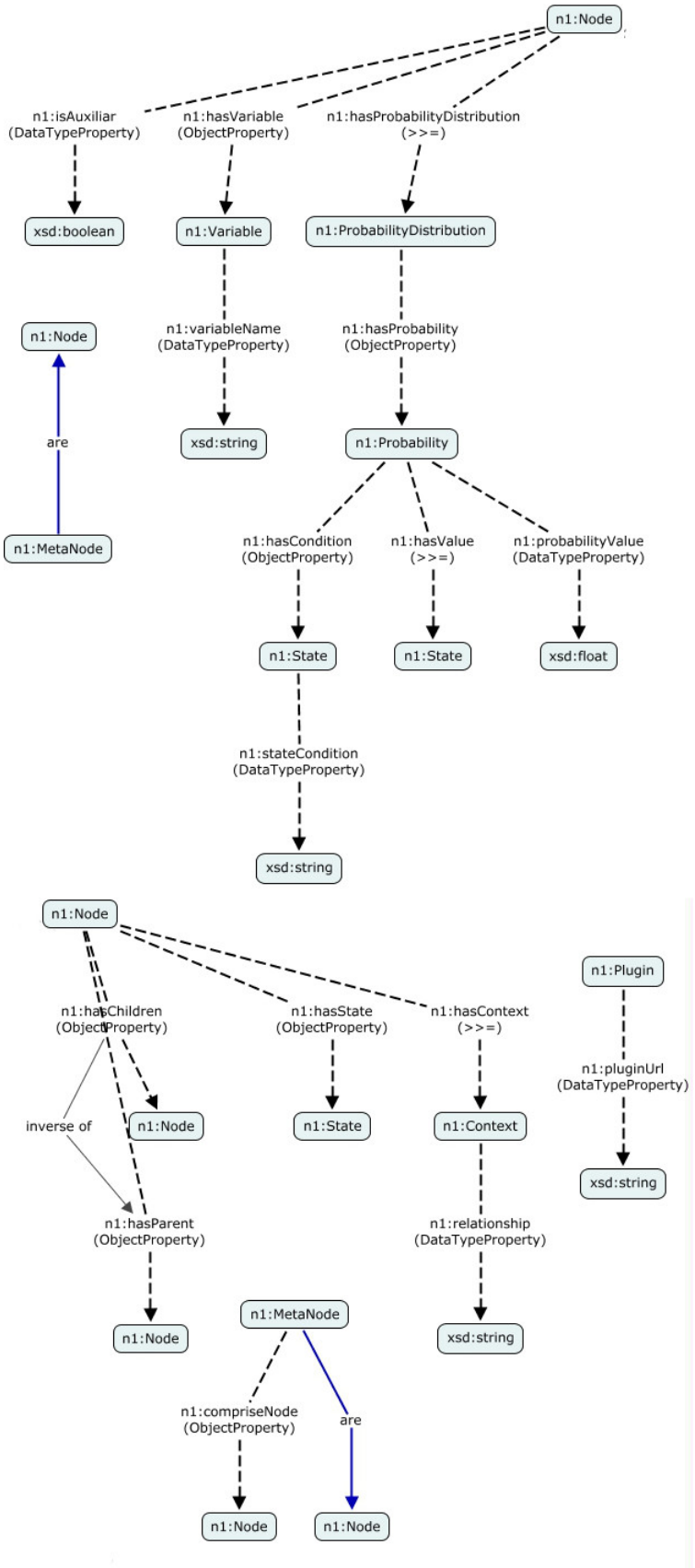

Fig. 1. Classes and properties defined by the Turambar ontology

In the first group, P-CLASSIC [9] extends description logic CLASSIC to add probability. In contrast, Pronto [8] is a probabilistic reasoner for P-$SROIQ$, a probabilistic extension of $SROIQ$. Pronto models probability intervals with its custom OWL annotation *pronto#certainty*. Apart from the previously described works, there are several other approaches that have been explained in different surveys such as [14].

In contrast, probabilistic web ontology languages combine OWL with probabilistic formalisms based on Bayesian networks. Since our proposal falls under this group, we will review in depth the most important works in this category: BayesOWL, OntoBayes and PR-OWL. The BayesOWL [7] framework extends OWL capacities for modelling and reasoning with uncertainty. It applies a set of rules to transform the class hierarchy defined in an OWL ontology into a Bayesian network. In the generated network there are two types of nodes: concept nodes and L-Nodes. The former one represents OWL classes and the latter one is a special kind of node that is employed to model the relationships defined by *owl:intersectionOf*, *owl:unionOf*, *owl:complementOf*, *owl:equivalentClass* and *owl:disjointWith* constructors. Concept nodes are connected between them by directed arcs that link superclasses with their classes. On the other hand, L-Nodes and concept nodes involved in a relationship are linked following the rules established for each constructor. The probabilities are defined with the classes *PriorProb*, for prior probabilities, and *CondProb*, for conditional probabilities. For instance, BayesOWL [22] recognizes some limitations: (i) variables should be binaries, (ii) probabilities should contain only one prior variable, (iii) probabilities should be complete and (iv) in case of inconsistency the result may not satisfy the constraints offered. BayesOWL approach is not valid for our purpose, because it only supports uncertainty to determine the class membership of an individual and this may not be enough for context modelling. For example, sensors' values may be represented as data and object properties values and knowing the probability that a sensor has certain value may be very useful for answering user's needs.

In contrast to BayesOWL, OntoBayes [21] focuses on properties. In OntoBayes, every random variable is a data or object property. Dependencies between them are described via the *rdfs:dependsOn* property. It supports to describe prior and conditional probabilities, besides it contains a property to specify the full disjoint probability distribution. Another improvement of OntoBayes over BayesOWL is that it supports multi-valued random variables. However, it is not possible to model relationships between classes in order to prevent errors when extracting Bayesian network structure from ontologies. OntoBayes offers us a solution for the limitation presented in BayesOWL regarding OWL properties, but its lack of OWL class support makes it unsuitable for our goal.

PR-OWL [5] is an OWL extension to describe complex bayesian models. It is based on the Multi-Entity Bayesian newtworks (MEBN) logic. MEBN [10] defines the probabilistic knowledge as a collection of MEBN fragments, named MFrags. A set of MFrags configures a MTheory and every PR-OWL ontology must contain at least one MTheory. To consider a MFrag set as a MTheory, it

must satisfy consistency constraints ensuring that it only exists a joint probability distribution over MFrags' random variables. In PR-OWL, probabilistic concepts can coexist with non probabilistic concepts, but these are only benefited by the advantages of the probabilistic ontology. Each MFrag is composed of a set of nodes which are classified in three groups: resident, input and context node. Resident nodes are random variables whose probability distribution is defined in the MFrag. Input nodes are random variables whose probability distribution is defined in a distinct MFrag than the one where is mentioned. In contrast, context nodes specify the constraints that must be satisfied by an entity to substitute an ordinary variable. Finally, node states are modelled with the object property named *hasPossibleValues*.

The last version of PR-OWL [2], PR-OWL 2, addresses the PR-OWL 1 limitations regarding to its compatibility with OWL: no mapping to properties of OWL and the lack of compatibility with existing types in OWL. Although, PR-OWL offers a good solution to deal with uncertainty, it does not provide some characteristics that we covet for our systems, such as isolation.

Our proposal is focused on computing the probability of data properties assertions, object properties assertions and class assertions. This issue is only covered by PR-OWL, because BayesOWL only takes into account class membership and OntoBayes, object and data properties.

In addition, we pretend to offer a way to keep the uncertainty information isolated as much as possible from the traditional ontology. With this policy, we want to ease the reutilization of our probabilistic ontologies by traditional systems that do not offer support for uncertainty and the interoperability between them. Furthermore, we aim to avoid that traditional reasoners load unnecessary information about the probabilistic knowledge that they do not need. Thus, if we load the Turambar probabilistic ontology located in http://www.example.org/ont.owl, traditional OWL reasoners load only the knowledge defined in the ordinary OWL ontology and do not have access to the probabilistic knowledge. In contrast, Turambar reasoner is able to load the ordinary OWL ontology and the dependency description ontology. The Turambar reasoner needs to access to the ordinary OWL ontology to answer traditional OWL queries and to find the evidences of the Bayesian networks defined in the dependency description ontology. It is also important to clarify that a class or property can have deterministic assertions and probabilistic assertions without duplicating them due to the links between Bayesian networks' nodes and OWL classes and properties through *turambarClass* and *turambarProperty*, respectively. Thanks to this feature, a Turambar ontology has a unique URI that allows it to be used as an ordinary OWL 2 DL ontology without loading the probabilistic knowledge. This characteristic is not offered by other approaches as far as we know.

Another difference with other approaches is that we have taken into account the extensibility of our approach through plug-ins to increase the basis functionalities. We believe that it is necessary to offer a straightforward, transparent and standard mechanism to extend reasoner functionality in order to cover heterogeneous domains' needs.

However, our approach has the shortcoming of assuming a simple attribute-value representation in comparison to PR-OWL. That means that each probabilistic query involves reasoning about the same fixed number of nodes, with only the evidence values changing from query to query. To solve this drawback, we can opt to employ situation specific Bayesian networks [12], as PR-OWL does. However, the development of custom plug-ins can overcome this limitation in some cases. Besides, thanks to this expressiveness restriction we are able to know the size of the Bayesian network and give a better estimation of the performance of the Turambar probabilistic ontology.

5 Conclusions and Future Work

In this work we have presented a proposal to deal with uncertainty in intelligent environments. Its main features are: a) it isolates the probabilistic information definition from traditional ontologies, b) it can be extended easily and c) it is oriented to intelligent environments.

As ongoing work, we are developing an extension to SPARQL-DL [17] in order to offer a simple mechanism to execute complex queries in a declarative way that abstracts developers from the reasoner implementation employed and its API. This extension proposes the addition of two new query atoms to query probabilistic knowledge: ProbType for probabilistic class assertions and ProbPropertyValue for probabilistic property assertions. We believe that this extension can ease the development of applications that employ Turambar.

As future work, we plan to create a graphical tool to ease the creation of probabilistic ontologies in order to promote its adoption. On the other hand, we plan to extend its expressivity and evaluate new and better ways to define the probabilistic description ontology in order to improve its maintainability. In addition, we are studying a formalism that allows us the definition of custom function for state evaluation that was independent of the programming language employed.

References

1. Baader, F., Küsters, R., Wolter, F.: The description logic handbook. chap. Extensions to description logics, pp. 219–261. Cambridge University Press, New York, NY, USA (2003), http://dl.acm.org/citation.cfm?id=885746.885753
2. Carvalho, R.N.: Probabilistic Ontology: Representation and Modeling Methodology. Ph.D. thesis, George Mason University (2011)
3. Chen, H., Perich, F., Finin, T., Joshi, A.: Soupa: standard ontology for ubiquitous and pervasive applications. In: Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on. pp. 258–267 (Aug 2004)
4. Coen, M.H.: Design principles for intelligent environments. In: Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence. pp. 547–554. AAAI '98/IAAI '98, American Association for Artificial Intelligence, Menlo Park, CA, USA (1998), http://dl.acm.org/citation.cfm?id=295240.295733

5. Costa, P.C.G.: Bayesian semantics for the Semantic Web. Ph.D. thesis, George Mason University (2005)
6. Dey, A.K.: Understanding and using context. Personal Ubiquitous Comput. 5(1), 4–7 (Jan 2001), http://dx.doi.org/10.1007/s007790170019
7. Ding, Z.: Bayesowl: A Probabilistic Framework for Uncertainty in Semantic Web. Ph.D. thesis, Catonsville, MD, USA (2005)
8. Klinov, P.: Practical reasoning in probabilistic description logic. Ph.D. thesis, University of Manchester (2011)
9. Koller, D., Levy, A., Pfeffer, A.: P-CLASSIC: A tractable probabilistic description logic. In: In Proceedings of AAAI-97. pp. 390–397 (1997)
10. Laskey, K.: MEBN: A language for first-order bayesian knowledge bases. Artificial Intelligence 172(2-3), 140–178 (2008)
11. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. Web Semantics: Science, Services and Agents on the World Wide Web 6(4), 291–308 (2008)
12. Mahoney, S.M., Laskey, K.B.: Constructing situation specific belief networks. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. pp. 370–378. UAI'98, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1998), http://dl.acm.org/citation.cfm?id=2074094.2074138
13. Pearl, J.: Fusion, propagation, and structuring in belief networks. Artificial Intelligence 29(3), 241 – 288 (1986), http://www.sciencedirect.com/science/article/pii/000437028690072X
14. Predoiu, L., Stuckenschmidt, H.: Probabilistic models for the semantic web. The Semantic Web for Knowledge and Data Management: Technologies and Practices pp. 74–105 (2009)
15. Preuveneers, D., Van den Bergh, J., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V., De Bosschere, K.: Towards an extensible context ontology for ambient intelligence. In: Ambient Intelligence, Lecture Notes in Computer Science, vol. 3295, pp. 148–159. Springer Berlin Heidelberg (2004)
16. Ramos, C., Augusto, J.C., Shapiro, D.: Ambient intelligencethe next step for artificial intelligence. Intelligent Systems, IEEE 23(2), 15–18 (March 2008)
17. Sirin, E., Parsia, B.: SPARQL-DL: SPARQL query for OWL-DL. In: OWLED. vol. 258 (2007)
18. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England (2004)
19. W3C: Uncertainty reasoning for the world wide web. Tech. rep., W3C (2008), http://www.w3.org/2005/Incubator/urw3/XGR-urw3/
20. 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. pp. 18–22 (2004)
21. Yang, Y., Calmet, J.: Ontobayes: An ontology-driven uncertainty model. In: Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on. vol. 1, pp. 457–463. IEEE (2005)
22. Yun Peng, Z.D.: Bayesowl: Reference manual (Feb 2013), http://www.csee.umbc.edu/ ypeng/BayesOWL/manual/index.html