PR-OWL 2.0 - Bridging the gap to OWL semantics

Rommel N. Carvalho, Kathryn B. Laskey, and Paulo C.G. Costa

Center of Excellence in C4I,
George Mason University, USA
rommel.carvalho@gmail.com, {klaskey,pcosta}@gmu.edu
http://www.gmu.edu

Abstract. The past few years have witnessed an increasingly mature body of research on the Semantic Web, with new standards being developed and more complex use cases being proposed and explored. As complexity increases in SW applications, so does the need for principled means to cope with uncertainty inherent to real world SW applications. Not surprisingly, several approaches addressing uncertainty representation and reasoning on the Semantic Web have emerged [3, 4, 6, 7, 10, 11, 13, 14]. For example, PR-OWL [3] provides OWL constructs for representing Multi-Entity Bayesian Network (MEBN) [8] theories. This paper reviews some shortcomings of PR-OWL 1 [2] and describes how they will be addressed in PR-OWL 2. A method is presented for mapping back and forth from triples into random variables (RV). The method applies to triples representing both predicates and functions. A complex example is given for mapping an n-ary relation using the proposed schematic.

Keywords: uncertainty reasoning, OWL, PR-OWL, MEBN, probabilistic ontology, Semantic Web, compatibility.

1 Introduction

Appreciation is growing within the Semantic Web community of the need to represent and reason with uncertainty. In recognition of this need, the World Wide Web Consortium (W3C) created the Uncertainty Reasoning for the World Wide Web Incubator Group (URW3-XG) in 2007 to identify requirements for reasoning with and representing uncertain information in the World Wide Web. The URW3-XG concluded that standardized representations were needed to express uncertainty in Web-based information [9]. A candidate representation for uncertainty reasoning in the Semantic Web is Probabilistic OWL (PR-OWL) [3], an OWL upper ontology for representing probabilistic ontologies based on Multi-Entity Bayesian Networks (MEBN) [8].

Compatibility with OWL was a major design goal for PR-OWL [3]. However, there are several ways in which the initial release of PR-OWL falls short of complete compatibility. First, there is no mapping in PR-OWL to properties of OWL. Second, although PR-OWL has the concept of meta-entities, which allows

the definition of complex types, it lacks compatibility with existing types already present in OWL.

These problems have been noted in the literature [12]:

> PR-OWL does not provide a proper integration of the formalism of MEBN and the logical basis of OWL on the meta level. More specifically, as the connection between a statement in PR-OWL and a statement in OWL is not formalized, it is unclear how to perform the integration of ontologies that contain statements of both formalisms.

This paper justifies the need for a formal mapping between random variables defined in PR-OWL and concepts defined in OWL, and proposes an approach to such a mapping. We first present a solution that is sufficient for binary relations. Next, we present a more robust solution that allows the user to define PR-OWL random variables with arbitrarily many arguments, while maintaining a 2-way mapping to OWL concepts. Finally, we present a schematic for the mapping back and forth from triples into random variables.

2 Why map PR-OWL Random Variables to OWL Concepts?

PR-OWL was proposed as an extension to the OWL language based on MEBN, which can express a probability distribution on interpretations of any first-order theory. In PR-OWL, a probabilistic ontology (PO) has to have at least one individual of class `MTheory`, which is basically a label linking a group of MFrags that collectively form a valid MTheory. In actual PR-OWL syntax, that link is expressed via the object property `hasMFrag` (which is the inverse of object property `isMFragIn`). Individuals of class `MFrag` are comprised of nodes. Each individual of class `Node` is a random variable (RV) and thus has a mutually exclusive, collectively exhaustive set of possible states. In PR-OWL, the object property `hasPossibleValues` links each node with its possible states, which are individuals of class `Entity`. Finally, random variables (represented by the class `Node` in PR-OWL) have unconditional or conditional probability distributions, which are represented by class `ProbabilityDistribution` and linked to their respective nodes via the object property `hasProbDist`.

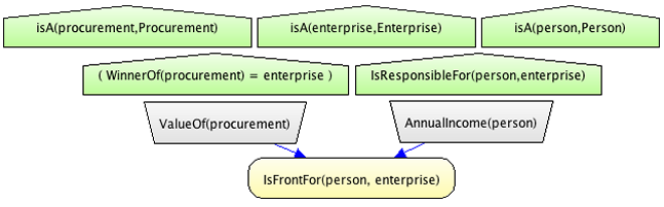

Fig. 1. Front of an Enterprise MFrag.

As a running example, we consider an OWL ontology for the public procurement domain. The ontology defines concepts such as procurement, winner of a procurement, members of a committee responsible for a procurement, etc.

Now, imagine we want to define some uncertain relations about this domain. For example, if an enterprise wins a procurement for millions of dollars, but the responsible person for this enterprise makes less than 10 thousand dollars a year, the responsible person may be a front. That is, we can identify potential fronts by examining the value of the procurement and the income of the responsible person. Figure 1 shows this probabilistic relation defined using PR-OWL in an open-source tool for probabilistic reasoning, UnBBayes [1]. In the figure, we see that a person's income and the value of a procurement influence whether the person is front for the procurement. The green pentagons at the top of the figure show conditions that must be met for the probabilistic relationship to apply; e.g., that the person we are considering as a possible front must be responsible for the enterprise we are examining.

Listing 1.1. Definition of `WinnerOf` RV in PR-OWL 1

```
<owl:Thing rdf:about="#WinnerOf_RV">
  <rdf:type rdf:resource="#Domain_Res"/>
  <hasPossibleValues rdf:resource="#Enterprise"/>
  <isResidentNodeIn rdf:resource="#ProcurementInfo_MFrag"/>
  <hasArgument rdf:resource="#WinnerOf_1"/>
</owl:Thing>

<owl:Thing rdf:about="#WinnerOf_1">
  <rdf:type rdf:resource="#SimpleArgRelationship"/>
  <hasArgNumber rdf:datatype="&xsd;int">1</hasArgNumber>
  <hasArgTerm rdf:resource=
    "#ProcurementInfo_MFrag.procurement"/>
  <isArgumentOf rdf:resource="#WinnerOf_RV"/>
</owl:Thing>

<owl:Thing rdf:about="#ProcurementInfo_MFrag.procurement">
  <rdf:type rdf:resource="#OVariable"/>
  <isOVariableIn rdf:resource="#ProcurementInfo_MFrag"/>
  <isSubsBy rdf:resource="#Procurement"/>
  <isArgTermIn rdf:resource="#WinnerOf_1"/>
</owl:Thing>
```

We would like to be able to tie this fragment of probabilistic knowledge with domain knowledge already represented in an OWL ontology. That is, we might have a database containing instances of persons and enterprises, linked to an OWL ontology defining their semantics (e.g., that persons can be responsible for enterprises). Accessing this information should be trivial once the definitions in the ontology were made available and permission was granted to retrieve data from the database. However, for PR-OWL to make use of this knowledge, there must be a way to link PR-OWL random variables (RVs) with concepts defined

in OWL. The current version of PR-OWL has no standard way to establish such links.

Listing 1.1 presents how the RV `WinnerOf_RV` from Figure 1 is defined in PR-OWL today. This RV is defined as follows:

- It is a domain resident node (line 2)
- Its possible values (range) are instances of `Enterprise` (line 3)
- Its home MFrag is `ProcurementInfo_MFrag` (line 4)
- It has one argument (domain) `WinnerOf_1` (line 5)
- `WinnerOf_1` is the first argument (line 10)
- `WinnerOf_1` is related to the variable `ProcurementInfo_MFrag.procurement` (lines 11-12)
- `ProcurementInfo_MFrag.procurement` is an ordinary variable (line 17)
- `ProcurementInfo_MFrag.procurement` is defined in the `ProcurementInfo_MFrag` (line 18)
- `ProcurementInfo_MFrag.procurement` can only be replaced by instances of `Procurement` (line 19)

Listing 1.2 is a suggested definition of the object property winnerOf in OWL. This property is defined as follows:

- It is an object property (line 1)
- It is a functional property (line 2)
- Its domain is the instances of `Procurement` (line 3)
- Its range is the instances of `Enterprise` (line 4)

Listing 1.2. Definition of `winnerOf` object property in OWL

```
1 <owl:ObjectProperty rdf:about="#winnerOf">
2     <rdf:type rdf:resource="&owl;FunctionalProperty"/>
3     <rdfs:domain rdf:resource="#Procurement"/>
4     <rdfs:range rdf:resource="#Enterprise"/>
5 </owl:ObjectProperty>
```

Comparing the two definitions `winnerOf` and `WinnerOf_RV`, we can see that they are consistent, since their domain/arguments and range/possible values are the same, `Procurement` and `Enterprise`, respectively. However, there is no property that explicitly relates these two concepts, and there is no implicit way of figuring out that they should be related besides the fact that their names are similar (`winnerOf` and `WinnerOf_RV`). Therefore, we would not have access to the semantics of the term `winnerOf` defined in our ontology when defining its probabilistic relations using the new and unrelated term `WinnerOf_RV` defined in our probabilistic ontology.

This simple example demonstrates the need to define a reference from every probabilistic definition involving a concept to its OWL definition. In other words, full compatibility with OWL requires modifications to PR-OWL that guarantee the preservation of OWL's semantics.

A simple solution to this mapping problem is presented in Listing 1.3. By adding the property `defineUncertaintyOf` which states that a random variable defines the uncertainty relations of a specific property, we could state that `WinnerOf_RV` defines the uncertainty of the object property `winnerOf` (line 3). In order to make this definition consistent we would need to add some axioms to our language stating that the possible values of the RV must be the same as the range defined in the property for which this RV defines the uncertainty. Similar axioms would be needed for its domain.

Listing 1.3. Definition of `WinnerOf` RV with mapping information to its OWL concept

```
1  <owl:Thing rdf:about="#WinnerOf_RV">
2    <rdf:type rdf:resource="#Domain_Res"/>
3    <defineUncertaintyOf rdf:resource="#winnerOf"/>
4    <hasPossibleValues rdf:resource="#Enterprise"/>
5    <isResidentNodeIn rdf:resource="#ProcurementInfo_MFrag"/>
6    <hasArgument rdf:resource="#WinnerOf_1"/>
7  </owl:Thing>
8
9  <owl:Thing rdf:about="#WinnerOf_1">
10   <rdf:type rdf:resource="#SimpleArgRelationship"/>
11   <hasArgNumber rdf:datatype="&xsd;int">1</hasArgNumber>
12   <hasArgTerm rdf:resource=
13     "#ProcurementInfo_MFrag.procurement"/>
14   <isArgumentOf rdf:resource="#WinnerOf_RV"/>
15 </owl:Thing>
16
17 <owl:Thing rdf:about="#ProcurementInfo_MFrag.procurement">
18   <rdf:type rdf:resource="#OVariable"/>
19   <isOVariableIn rdf:resource="#ProcurementInfo_MFrag"/>
20   <isSubsBy rdf:resource="#Procurement"/>
21   <isArgTermIn rdf:resource="#WinnerOf_1"/>
22 </owl:Thing>
```

3 Mapping n-ary relations

In Section 2 we presented a simple solution to map OWL concepts to random variables defined in PR-OWL. In this section we will show that the presented solution is not enough to cover the full expressiveness of PR-OWL. In particular, this solution cannot represent uncertainty for n-ary functions and relations.

Imagine extending our example to a situation in which a group of enterprises can win a procurement. Moreover, there will be a price associated with each enterprise on the contract. Therefore, instead of comparing the value of the procurement as a whole to try to identify the owner of the enterprise as a front (as shown on Figure 1), we need to consider only the part of that total associated to that specific enterprise, as shown in Figure 2.

Note that we now have a ternary relation which associates an enterprise, a contract, and the amount awarded by the contract to the enterprise. As a

functional relation, this is represented by the two-argument function `priceOf(-contract,enterprise)`.

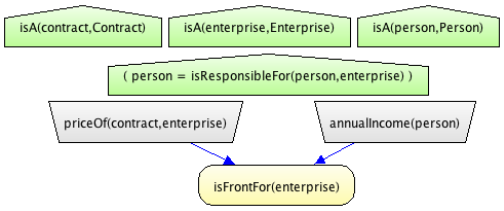

Fig. 2. Front of an Enterprise MFrag using `priceOf(contract,enterprise)`.

Listing 1.4. Problem when trying to define n-ary relations as simple binary relations

```
<owl:ObjectProperty rdf:about="#hasPrice">
    <rdfs:domain rdf:resource="#Contract"/>
    <rdfs:range rdf:resource="#Money"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#hasEnterprise">
    <rdfs:domain rdf:resource="#Contract"/>
    <rdfs:range rdf:resource="#Enterprise"/>
</owl:ObjectProperty>

<Contract rdf:about="#contract1">
    <rdf:type rdf:resource="&owl;Thing"/>
    <hasEnterprise rdf:resource="#enterprise1"/>
    <hasEnterprise rdf:resource="#enterprise2"/>
    <hasPrice rdf:resource="#price1"/>
    <hasPrice rdf:resource="#price2"/>
</Contract>

<Money rdf:about="#price1">
    <rdf:type rdf:resource="&owl;Thing"/>
    <valueOf rdf:datatype="&xsd;float">10000</valueOf>
    <currencyOf rdf:resource="#Dollar"/>
</Money>

<owl:Thing rdf:about="#price2">
    <rdf:type rdf:resource="#Money"/>
    <valueOf rdf:datatype="&xsd;float">500000</valueOf>
    <currencyOf rdf:resource="#Dollar"/>
</owl:Thing>
```

Suppose that we want to represent that `enterprise1` was hired for $10,000.00 and `enterprise2` for $500,000.00 both in `contract1`. The problem is that OWL

supports only binary relations. As shown in Listing 1.4, if we tried to represent this situation using binary relations with the class `Contract`, we would be unable to distinguish whether `enterprise1` has price of $10,000.00, `price1`, or $500,000.00, `price2`.

Listing 1.5. Defining n-ary relations in OWL

```
<owl:ObjectProperty rdf:about="#contractOf">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="_:id1"/>
    <rdfs:range rdf:resource="#Contract"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#enterpriseOf">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="_:id1"/>
    <rdfs:range rdf:resource="#Enterprise"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#priceOf">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="_:id1"/>
    <rdfs:range rdf:resource="#Money"/>
</owl:ObjectProperty>

<owl:Thing rdf:about="#3aryInstance1">
    <rdf:type rdf:resource="_:id1"/>
    <contractOf rdf:resource="#contract1"/>
    <enterpriseOf rdf:resource="#enterprise1"/>
    <priceOf rdf:resource="#price1"/>
</owl:Thing>

<owl:Thing rdf:about="#3aryInstance2">
    <rdf:type rdf:resource="_:id1"/>
    <contractOf rdf:resource="#contract1"/>
    <enterpriseOf rdf:resource="#enterprise2"/>
    <priceOf rdf:resource="#price2"/>
</owl:Thing>
```

As shown in Figure 3, one way to overcome this problem is to create a blank node which has three functions mapping to each of the 3 arguments of our ternary relation. Notice that these 3 binary relations (`contractOf`, `enterpriseOf`, and `priceOf`) have to be functions, otherwise we would have the same problem we had in Listing 1.4. Listing 1.5 presents this representation in OWL (for more details on how to define n-ary relations in OWL, see [5]).

When we try to apply the simple solution given in Section 2, we realize that it is not suitable for RVs with more than one argument. This is due to the fact that we assume the following:

1. The range from the property associated to the `defineUncertaintyOf` has to be the same type as the value of the RV's `hasPossibleValues` property; and
2. The domain from the property associated to the `defineUncertaintyOf` has to be the same type as the only RV's argument (`hasArgument` → `hasArgTerm` → `isSubsBy`).

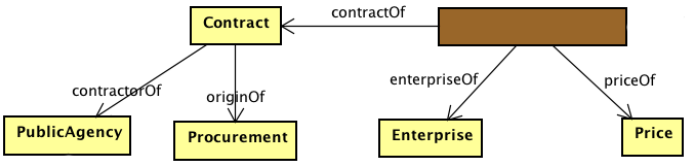

Fig. 3. An initial ontology with an n-ary relation between `Price`, `Enterprise`, and `Contract` using a blank node.

So, what happens with the other arguments of the RV? What do they map to? Notice also that there is no argument in `priceOf(contract,enterprise)` that relates to the domain of the OWL property `priceOf`. In other words, there is no argument that "points" to the blank node we defined in Figure 3 and Listing 1.5. Besides, having only the property `defineUncertaintyOf` relating to the OWL property `priceOf` tells us nothing about what `contract` and `enterprise` are and where they come from. As a matter of fact, we need to have a reference to all the binary properties that we use to represent the n-ary relation we want. Therefore, in this case, we also need to have a mapping to both `contractOf` and `enterpriseOf`.

Taking a closer look, we realize that all three properties of interest (`priceOf`, `contractOf`, and `enterpriseOf`) have the same domain (the blank node) and their range, `Money`, `Contract`, and `Enterprise`, map directly to the possible values of our RV of interest, to the argument `contract`, and to the argument `enterprise`, respectively. Listing 1.6 shows a more complex and robust solution that covers this case and any other n-ary relation for which we might want to define uncertainty.

Listing 1.6 states that the RV `priceOf_RV` defines the probabilistic semantics of the property `priceOf`, which already has an OWL semantics (line 3). Lines 4 and 5 ensure that the domain and range from the OWL property match the RV domain (`hasDomain`) and range (`hasPossibleValues`), respectively. Lines 7 and 8 say that this RV has two arguments. Lines 13-15 define the first argument as being the variable `contract`, and lines 30-32 define the second argument as the variable `enterprise`. Lines 22-24 specify that the contract variable is used as the object (`objectIn`) of the OWL property `contracOf`, thus it can only be substituted by (`isSubsBy`) the class that is the range of the `contractOf` property, which is `Contract`. In addition, the domain also has to be the same (`hasDomain`), which, in this case, is the blank node `_:id1`.

Listing 1.6. Robust solution for defining n-ary RVs and mapping them to the OWL concepts that define their semantics

```
1  <owl:Thing rdf:about="#priceOf_RV">
2    <rdf:type rdf:resource="#Domain_Res"/>
3    <defineUncertaintyOf rdf:resource="#priceOf"/>
4    <hasDomain rdf:resource="_:id1"/>
5    <hasPossibleValues rdf:resource="#Money"/>
6    <isResidentNodeIn rdf:resource="#FrontOfEnterprise_MFrag"/>
7    <hasArgument rdf:resource="#priceOf_1"/>
8    <hasArgument rdf:resource="#priceOf_2"/>
9  </owl:Thing>
10
11 <owl:Thing rdf:about="#priceOf_1">
12   <rdf:type rdf:resource="#SimpleArgRelationship"/>
13   <hasArgNumber rdf:datatype="&xsd;int">1</hasArgNumber>
14   <hasArgTerm rdf:resource=
15     "#FrontOfEnterprise_MFrag.contract"/>
16   <isArgumentOf rdf:resource="#priceOf_RV"/>
17 </owl:Thing>
18
19 <owl:Thing rdf:about="#ProcurementInfo_MFrag.contract">
20   <rdf:type rdf:resource="#OVariable"/>
21   <isOVariableIn rdf:resource="#FrontOfEnterprise_MFrag"/>
22   <objectIn rdf:resource="#contractOf"/>
23   <hasDomain rdf:resource="_:id1"/>
24   <isSubsBy rdf:resource="#Contract"/>
25   <isArgTermIn rdf:resource="#priceOf_1"/>
26 </owl:Thing>
27
28 <owl:Thing rdf:about="#priceOf_2">
29   <rdf:type rdf:resource="#SimpleArgRelationship"/>
30   <hasArgNumber rdf:datatype="&xsd;int">2</hasArgNumber>
31   <hasArgTerm rdf:resource=
32     "#FrontOfEnterprise_MFrag.enterprise"/>
33   <isArgumentOf rdf:resource="#priceOf_RV"/>
34 </owl:Thing>
35
36 <owl:Thing rdf:about="#ProcurementInfo_MFrag.enterprise">
37   <rdf:type rdf:resource="#OVariable"/>
38   <isOVariableIn rdf:resource="#FrontOfEnterprise_MFrag"/>
39   <objectIn rdf:resource="#enterpriseOf"/>
40   <hasDomain rdf:resource="_:id1"/>
41   <isSubsBy rdf:resource="#Enterprise"/>
42   <isArgTermIn rdf:resource="#priceOf_2"/>
43 </owl:Thing>
```

The same thing goes for the `enterprise` variable. In lines 39-41 we define that the `enterprise` variable is in fact used as the object (`objectIn`) of the OWL property `enterpriseOf`, thus it can only be substituted by (`isSubsBy`)

the class that is the range of the `enterpriseOf` property, which is `Enterprise`. In addition, the domain also has to be the same (`hasDomain`), which, in this case, is the blank node `_:id1`.

4 The bridge joining OWL and PR-OWL

The key to building the bridge that connects the deterministic ontology defined in OWL and its probabilistic extension defined in PR-OWL is to understand how to translate one to the other. On the one hand, given a concept defined in OWL, how should its uncertainty be defined in PR-OWL in a way that maintains its semantics defined in OWL? On the other hand, given a random variable defined in PR-OWL, how should it be represented in OWL in a way that respects its uncertainty already defined in PR-OWL? Examples of our proposed translation were given above. Here, a schematic is given in Figure 4 for the 2-way mapping between triples and random variables. Functions and predicates are considered as separate cases.

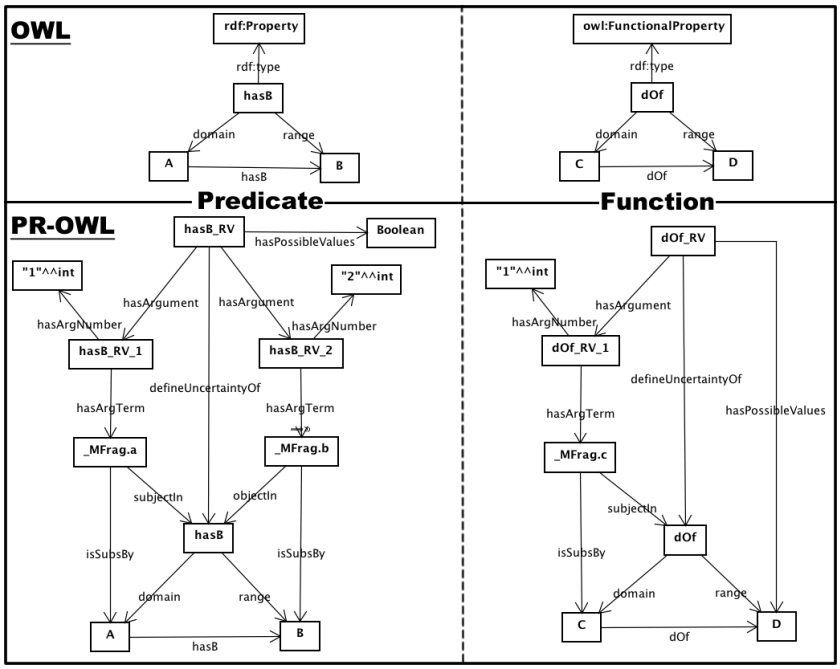

Fig. 4. The bridge joining OWL and PR-OWL.

If a property (`hasB` or `dOf`) is defined in OWL, then its domain and range are already represented (`A` and `B`; `C` and `D`, respectively). The first thing to be done is

to create the corresponding RV in PR-OWL (`hasB_RV` and `dOf_RV`, respectively) and link it to this OWL property through the property `defineUncertaintyOf`.

For binary relations, the domain of the property (`A` and `C`, respectively) will usually be the type (`isSubsBy`) of the variable (`_MFrag.a` and `_MFrag.c`, respectively) used in the first argument (`hasB_RV_1` and `dOf_RV_1`, respectively) of the RV. For n-ary relations see Section 3.

If the property is non-functional (`hasB`), then it represents a predicate that may be true or false. Thus, instead of having the possible values of the RV in PR-OWL (`hasB_RV`) being the range of the OWL property (`B`), it must be `Boolean`. So, its range (`B`) has to be mapped to the second argument (`hasB_RV_2`) of the RV, the same way the domain (`A`) was mapped to the first argument (`hasB_RV_1`) of the RV. On the other hand, if the the property is functional (`dOf`), the possible values of its RV (`dOf_RV`) must be the same as its range (`B`).

It is important to note that not only is the RV linked to the OWL property by the `defineUncertaintyOf`, but also to the variables by either `subjectIn` or `objectIn`, depending on what they refer to (domain or range of the OWL property, respectively). This feature is especially important when dealing with n-ary relations, where each variable will be associated with a different OWL property (see Section 3) for details).

Finally, if the RV is already defined in PR-OWL with all its arguments and its possible values, the only thing that needs to be done is to create the corresponding OWL property, link the RV to it using the `defineUncertaintyOf` and make sure that the domain and range of the property matches the RV definition, as explained previously.

The mapping described in this Section provides the basis for a formal definition of consistency between a PR-OWL probabilistic ontology and an OWL ontology, in which rules in the OWL ontology correspond to probability one assertions in the PR-OWL ontology. A formal notion of consistency can lead to development of consistency checking algorithms.

5 Conclusion

Although the semantics was not formally defined, this paper provided both the syntax and a more in depth description of one of the major changes in PR-OWL 2: a formal mapping between OWL concepts and PR-OWL random variables. First, the importance of a formal mapping was justified through an example. Second, a simple solution sufficient for 2-way relations was presented. Next, a more complex and robust solution covering n-ary random variables was presented. Finally, a schematic was given for how to do the mapping back and forth between PR-OWL random variables and OWL triples (both predicates and functions).

As future work, this schematic will be formally defined by explicitly defining its semantics. This will be a major contribution of PR-OWL 2. Moreover, a formalization of an algorithm for performing the mapping from OWL concepts to PR-OWL RVs, and vice-versa, will be proposed. In addition, PR-OWL 2 will address other issues described in [2].

Acknowledgments. The authors would like to thank the Brazilian Office of the Comptroller General (CGU) for their active support since 2008 and for providing the human resources necessary to conduct this research.

References

1. UnBBayes - the UnBBayes site. http://unbbayes.sourceforge.net/.
2. Rommel Novaes Carvalho, Kathryn B. Laskey, and Paulo Cesar G. Costa. Compatibility formalization between PR-OWL and OWL. In *Proceedings of the First International Workshop on Uncertainty in Description Logics (UniDL) on Federated Logic Conference (FLoC) 2010*, Edinburgh, UK, July 2010.
3. Paulo C. G Costa. *Bayesian Semantics for the Semantic Web*. PhD, George Mason University, July 2005. Brazilian Air Force.
4. Zhongli Ding, Yun Peng, and Rong Pan. BayesOWL: uncertainty modeling in semantic web ontologies. In *Soft Computing in Ontologies and Semantic Web*, pages 3–29. 2006.
5. Patrick Hayes and Alan Rector. Defining n-ary relations on the semantic web. http://www.w3.org/TR/swbp-n-aryRelations/, 2006.
6. Jochen Heinsohn. Probabilistic description logics. In *Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pages 311–318, Seattle, Washington, USA, 1994. Morgan Kaufmann.
7. Daphne Koller, Alon Levy, and Avi Pfeffer. P-CLASSIC: a tractable probabilistic description logic. *IN PROCEEDINGS OF AAAI-97*, pages 390—397, 1997.
8. Kathryn Blackmond Laskey. MEBN: a language for first-order bayesian knowledge bases. *Artif. Intell.*, 172(2-3):140–178, 2008.
9. Kenneth Laskey and Kathryn Laskey. Uncertainty reasoning for the world wide web: Report on the URW3-XG incubator group. URW3-XG, W3C, 2008.
10. Thomas Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6-7):852–883, April 2008.
11. J Z Pan, G Stoilos, G Stamou, V Tzouvaras, and I Horrocks. f-SWRL: a fuzzy extension of SWRL. *Journal of Data Semantics VI*, 4090/2006:28–46, 2006.
12. Livia Predoiu and Heiner Stuckenschmidt. Probabilistic extensions of semantic web languages - a survey. In *The Semantic Web for Knowledge and Data Management: Technologies and Practices*. Idea Group Inc, 2008.
13. Umberto Straccia. A fuzzy description logic for the semantic web. In *FUZZY LOGIC AND THE SEMANTIC WEB, CAPTURING INTELLIGENCE*, pages 167—181. Elsevier, 2005.
14. Jia Tao, Zhao Wen, Wang Hanpin, and Wang Lifu. PrDLs: a new kind of probabilistic description logics about belief. In *New Trends in Applied Artificial Intelligence*, pages 644–654. 2007.