Is It Worth a Hoot? Qualms about OWL for Uncertainty Reasoning

Mike Pool[1], Francis Fung[2], Stephen Cannon[1], Jeffrey Aikin[1]

[1] Information Extraction and Transport, Inc.
1911 N. Fort Myer Dr, Suite 600, Arlington, VA 22209
{mpool, scannon, jaikin}@iet.com
[2] Information Extraction and Transport, Inc.
1600 SW Western Blvd, Suite 300, Corvallis, OR 97333
{fung@iet.com}

Abstract. Information Extraction and Transport, Inc. (IET) is developing the Knowledge Elicitation Environment for Probabilistic Event and Entity Relations (KEEPER) system, a tool for eliciting, storing, updating and implementing probabilistic relational models (PRMs)[1,8,16]. The KEEPER elicitation component implements a single ontology for the purposes of constraining and guiding elicitation and providing the semantic bedrock for the reintegration of diverse knowledge sources for reasoning and learning. We have used an extension of the Web Ontology Language (OWL) to implement the ontology and the tools for PRM representation, and we describe the main features of that extension in this paper. This paper offers an informal characterization of OWL_QM, an extension of OWL that supports the representation of PRMs. It is intended to motivate discussion as to whether OWL is an appropriate foundation for addressing the challenge of handling uncertainty on the Semantic Web.

1 Introduction

IET's Knowledge Elicitation Environment for Probabilistic Event and Entity Relations (KEEPER) system is designed to facilitate probabilistic knowledge elicitation from subject matter experts (SMEs) in an environment that maximizes the integration and updating of that knowledge. This process has involved implementing an ontology based elicitation environment that also facilitates learning over disparate data.

An established technology for performing reasoning under uncertainty is the formalism of Bayesian networks (BNs) [14]. Standards for representing BNs have been created and work has been done to extend semantic web languages for purpose of representing BNs, see [2] and [15]. However, BNs are not, in themselves, completely adequate to enable reasoning systems because BNs embody a "flat" representation language in which all domain variables must be represented as nodes in a graph without allowing any abstraction. Within a BN, the fact that one node represents a relation between entities is lost, *i.e.*, it cannot be directly recovered from the information contained only in the BN model. Since BN variables do not fully capture semantics, it is difficult to maintain generality and enforce shared semantics across SMEs.

A more expressive alternative to BNs is the implementation of type-level probabilistic relational models (PRMs) that we discuss below. Upon instantiation, PRMs encode a Bayesian network and probabilistic reasoning tools can be used to reason about properties of the objects instantiated. Thus a PRM can be viewed as an augmentation of an ontological description of a set of entities that not only describes the taxonomic hierarchies and the relationships between entities, but also the probabilistic relationships among the values of various attributes of entities, see [3] and [16]. We discuss the nature of IET's implementation of PRMs in OWL_QM and the motivation for implementing them in OWL. We note that the OWL extensions required to achieve this relatively minimal extension are quite extensive and briefly discuss whether this argues against the use of OWL as an appropriate foundation for uncertainty reasoning.

2 Probabilistic Relational Models

IET's modeling language, Quiddity*Modeler (QM), is a representation language for creating a version of Probabilistic Relational Models (PRMs) that can be implemented with IET's reasoning tools. A PRM is

based on a relational schema consisting of a set *X* of n classes $\{X_1, X_2, \ldots X_n\}$. For each $X_i \in X$, there is a set of attributes, denoted by $A(X_i)$, and for each $A \in A(X_i)$, there is a set of possible values of A, denoted by $V(X_i.A)$. For example, in PRMs for reasoning about vehicles, there may be a Vehicle class and an attribute associated with it may be color. That attribute may be able to take on some set of values each denoting different colors. Also associated with each $X_i \in X$ is a set of reference slots $R(X_i)$ that relates X_i to another class X_j (where j may equal i) indicating how instances of different classes may be related to each other. A range class is associated with each element $R \in R(X_i)$. For example, the Vehicle class may have a reference slot owner with a range Person, thereby linking instances of Vehicle to instances of Person when the PRM is instantiated. A reference slot chain can be created by composing a set of reference slots into a list, e.g., $r_1, r_2, \ldots, r_n$. Attributes of objects defined in terms of their relation to another object are referenced with a slot chain, SC, as follows: $X_i.SC.A$ where X_i is a class, SC is a slot chain in which the first element has the domain X_i and A is an attribute of the class that is in the range of the final element of SC. So, for example, suppose that in a model about car sales it is useful to reference the sales tax rate of the state in which car owners reside. This would be done with the slot chain 'Car.owner.location.taxRate', where 'owner.location' is a slot chain linking the Vehicle class to the Person class (owner) and the Person class to the GeographicLocation class (location).

Given a PRM, we can specify a set of instances of the classes and the relations between them in terms of the relational schema. Some attributes for the instances can be assigned values (for instance, vehicle1.location = Virginia, where Virginia would be an instance of GeographicLocation). Other attributes are uncertain, and part of the PRM definition is a specification of the parents of a given attribute, and a specification for how to construct a conditional probability distribution for an attribute of an instance, which depends on the values for its parent attributes. In this way, a set of instances of a PRM give rise to a Bayesian network that encodes the probabilistic dependencies among the attributes of the instances.

QM implements PRMs and is based loosely on frames [10], a popular knowledge representation approach, and is augmented with various methods to construct structural hypotheses. The fundamental modeling unit is the frame. A frame defines general properties that hold for a class of objects, called *frame instances;* frames are comparable to OWL classes and are used to represent the classes in a relational schema. Frames contain (or, one could also say, are associated with) *slots* that are used to specify attributes of an instance of the frame; they are the descriptive attributes of the relational schema. Each *slot* can have a number of *facets* defined on it. Some of these facet names are reserved words, and their values define the probability model over instances of frame definitions. QM supports frame inheritance, where subframes inherit all slots (and facets defined on them) defined in parent frames. It supports a version of multiple inheritance, where a frame can inherit from multiple parents, but each such parent must inherit from a different direct subframe of the top-level frame, Frame (see [5]).

In addition to frame (class) abstractions organized by an "is-a" hierarchy inherited from the frame system, QM supports mechanisms to express uncertainties about the value of a variable, the reference to an instance, the existence of an instance, and the type of an instance. QM allows for expressing domain knowledge as fragments of Bayesian networks in a modular and compact way, facilitating reuse.

This represents a significant advance over traditional approaches to BN representation. There has been a great deal of interest in extending the Bayesian network formalism to provide greater expressive power (see [8,9,12,17]). IET's frame-language representation overcomes some of the challenges mentioned above, i.e., its semantics allow users to create type level probabilistic models that impose richer semantics and distinguish different objects that hold different properties in the context of a single BN.

3 Why OWL?

While IET has tools in place that allow for the creation of PRMs, the KEEPER tool required an implementation to elicit PRMs, expressed within some uniform language, from SMEs. Our central assumption is that an ontology-based approach is extremely useful for addressing some of the elicitation challenges. Using a fixed ontology allows us to guide and constrain the elicitation; its implementation allows SMEs to use variables that are clearly defined in a language sufficiently expressive to capture intended meaning and facilitate interoperability between the knowledge representations of different users. For more discussion of the importance of an ontology in a knowledge representation environment requiring

elicitation, learning from diverse sources and integration of heterogeneous sources, see [6]. Given the central importance of an ontology in our reasoning system, it appeared to make sense to utilize OWL for the following reasons:

a) On the face of it there appeared to be a relatively simple mapping from QM's frame language to OWL's class-slot description logic.
b) Many tools exist for editing and reasoning with knowledge developed in OWL. The Protégé ontology editor has been a key resource in developing the KEEPER tool. We have also used the Jena reasoning and parsing tools quite extensively.
c) The semantic web is a key potential source of information for the purposes of reasoning and learning. If our models are to be able to use that information they must be developed in an amenable knowledge representation framework. It gives us ready access to the many ontologies that have already been developed in OWL.
d) The semantic web users are potential consumers of tools and knowledge that allows them to deal with the uncertainty inherent in the web.

4 Related Work

Several efforts have been made to extend OWL and/or general description logics for the purposes of representing probabilistic information. We mention some of the more closely related efforts here. First, Ding and Peng [15] have proposed an extension to OWL for representing particular Bayesian networks. This effort provides a means of translating an ontology implementing the set constructors of OWL into a Bayesian network and concerns itself explicitly with set or class memberships rather than relationships between attributes. In this sense, it is a more natural extension to OWL, *i.e.*, insofar as the main point of implementing a description logic is to reason simply about class membership. The work of Koller *et al* [7] in developing P-Classic is similar, i.e., it provides a way to encode the classifiers in terms of a probabilistic extension.

Paulo Costa has developed a very impressive extension to OWL to represent the full MEBN-logic [8] in the OWL framework. The extensions that Costa's work provides will, presumably, subsume the extension provided here but we have continued to maintain our approach as it requires a smaller extension, less parsing and reasoning support, and is more directly translatable into QM.

Our effort is analogous to the Semantic Web Rule Language (SWRL), [4] a proposed extension to OWL, i.e., we are attempting to go beyond a mere description logic; we are not merely looking for classifier tools that will handle uncertainty. In that sense we add our voices to those who have not found the DL focus in semantic web reasoning to be appropriate or adequate. However, we are also willing to settle for less than full blown first-order expressiveness in our language. So, in some sense our examination is meant to determine whether or not OWL is useful for even a modest probabilistic extension as the ability to represent PRMs seems to be a fairly reasonable requirement for any language that is to be implemented for handling uncertainty on the web.

5 OWL Implementation of PRM Constructs

In this section we discuss our extension to OWL, OWL_QM, for eliciting and representing PRMs. While QM is the target language in our example, the approach and concomitant challenges are applicable to representing PRMs in general.

5.1 Representing Quiddity Facets

PRMs implemented in QM focus on the representation of causal links between properties of objects. Suppose that one wants to model the relation between a car's monetary value and the mileage (odometer) and show the probability distributions for these values as well as the causal links. Assume that one of the classes in our relational schema is Car and that monetaryValue and mileage are elements of A(Car). To indicate that there is a causal relation between a car's value and its gas mileage it is necessary to specify the

causal links between monetaryValue and mileage, probability distributions over the values of each attribute, and other metaproperties in terms of these properties. These properties of properties, or more correctly, properties of associations between properties and classes, are called 'facets' in QM.

Our focus in extending OWL to include QM PRMs was the creation of a means to introduce these facets. Since these facets are associated with properties, the natural inclination is to attempt to define these simply as properties of properties. One might assume that the creation of facets would simply involve defining "metaproperties" that had rdf:Property as the value for each of their respective domains and could be used to relate probability distributions and the like to these properties. However, this is not feasible in OWL. A central difference between QM (and most frame languages [10]), and OWL is the fact that rdfs:domain, the property linking a slot to a class, is a global restriction, i.e., it tightly binds the property to a particular class (see [11] and [13]). This means that any property specified on a property P is, in effect, necessarily tied to the class, C, such that (P domain C). So, any metaproperty defined on P will be linked to C as well.

This impacts efforts to translate from OWL to QM as well as efforts to embed notions central to QM in an OWL ontology. Such a restriction cannot be overridden by associating a property with different classes. In QM, and other frame-slot languages, a slot is effectively defined relative to a particular class. So, when one declares facets, like 'distribution', on a particular slot, those facets are interpreted to be associated relative to the frame at which the slot is defined. Consider the following frame and slot definition in QM:

```
frame Car isa Frame
  slot mileage
       facet domain = [good, poor]
       facet distribution = [.5, .5]
```

It is interpreted to mean that the slot mileage, when attached to an instance of Car, can take on the value of either 'good' or 'poor' and the distribution over those two values is [.5,.5]. But note that the Frame to which it is related is central to the definition. As a convenience, QM allows for inheritance so that if a child (subclass) of a frame does not reintroduce the same slot name, then the system infers that the definition of the slot as stated for the parent frame is reapplicable to the child frame. However, it is also possible to redefine the distribution declaration in a subclass, as illustrated in the frame definition below.

```
frame SportUtilityVehicle isa Car
     slot mileage
          facet domain = [good, poor]
          facet distribution = [.3, .7]
```

This facet redefinition is less straightforward in OWL. If we define a distribution as a property of a property, we cannot specify that the distribution applies for instances of some classes on which the base property is defined but not others, i.e., it is less straightforward to override the defaults on a property if the are defined as properties of a property rigidly tied to a class.

However, the ability to associate a distribution with a property (or slot) and class (frame) is essential to an implementation of a probabilistic ontology extension. The distributions on value ranges for properties of objects will typically change for classes at different levels of the class hierarchy. Since QM treats each facet, (e.g., distribution), as a property of a slot defined relative to some frame, our approach to capturing this information in OWL is to reify the relationship between an OWL class and an OWL property and define our distributions (and other QM facets) as properties of this reified relationship rather than as a property of the OWL property itself. (See [18] for more discussion of reified relationships.) In other words, unlike the situation in QM, in the OWL implementation of QM we must represent the class and the slot and the relationship between them, i.e., the relationship between the slot and the property is reified as an instance of another class we have called 'FrameSlotPairing'. It is this relationship between the slot and the frame, the FrameSlotPairing, rather than the slot on its own, that becomes the "property holder" for our QM facets in OWL_QM.

```
<owl:Class rdf:ID="FrameSlotPairing">
   <rdfs:subClassOf rdf:resource="#DIRECTED-BINARY-RELATION"/>
</owl:Class>
```

Thus, FrameSlotPairing is defined as a subclass of :DIRECTED-BINARY-RELATION, a concept defined in the Protégé metalanguage for OWL. Each FrameSlotPairing instance represents the relationship between one OWL class and one OWL property. The :FROM slot takes as its value the OWL

class involved in the relationship and the :TO slot takes as its value the property involved in the relationship.

The framework for reasoning about the mileage for a car is implemented by creating FrameSlotPairings as follows, where Car and SportUtilityVehicle are classes in the ontology and mileage is a property:

```
<FrameSlotPairing rdf:ID="Car_mileage">
        <protege:FROM rdf:resource="#Car"/>
        <protege:TO rdf:resource="#mileage"/>
</FrameSlotPairing>
<FrameSlotPairing rdf:ID="SUV_mileage">
        <protege:from rdf:resource="#SportUtilityVehicle"/>
        <protege:to rdf:resource="#mileage"/>
</FrameSlotPairing>
```

Facets are then defined as properties having the domain 'FrameSlotPairing' and it becomes possible to define distinct value ranges and distributions as we descend the hierarchy. The distinct distributions, and even ranges, for an SUV's mileage versus a generic car's mileage would be defined as attributes of the two distinct FrameSlotPairings.

5.2 Representing Slot-Chains

A key advantage of PRMs is their ability to represent the way in which causal factors are related to the entity being influenced by using slot chains as defined above, i.e., lists of reference slots that specify the relations between instances of some class and the relationships by which they are related to the attributes of another. Consider a model representing hereditary factors in baldness. A PRM with a Person class that has, as properties, mother, father, and bald can be constructed. The mother and father properties both have Person as their domain, and bald has a boolean range. The link between a person's baldness state and his/her maternal grandfather's baldness is specified by utilizing the slot chain "mother.father.bald", i.e., the bald property of the Person instance in the father slot of the Person instance in the mother slot of the Person instance in question. In QM this relationship is defined as follows:

```
frame Person
        slot mother
                facet domain = Person
        slot father
                facet domain = Person
        slot baldness
                facet domain  = [false, true]
                facet parents=[mother.father.baldness]
```

To implement this in OWL_QM, more representational tools than those used in QM are required. Just as the pairings of slots and frames were reified to represent the association of a slot with a frame, a class called 'Probabilistic Relationship' is reified to represent causal relations. These links are used to link the relevant attributes, as well as to specify how the attributes themselves are linked.

```
<owl:Class rdf:ID="ProbabilisticRelationship">
        <rdfs:subClassOf rdf:resource="#DIRECTED-BINARY-RELATION"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="parent_PR">
        <rdfs:range rdf:resource="#FrameSlotPairing"/>
        <rdfs:domain rdf:resource="#ProbabilisticRelationship"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="child_PR">
        <rdfs:domain rdf:resource="#ProbabilisticRelationship"/>
        <rdfs:range rdf:resource="#FrameSlotPairing"/>
</owl:ObjectProperty>
```

In the above case, an instance of FrameSlotPairing is created, i.e., 'Person_baldness', in which the TO value is 'Person' and the FROM value is 'baldness'. However, the person whose baldness influences the baldness of the person in the child_PR slot must also be specified. Accomplishing this requires a list of

slots showing the chain of relations linking the entity about which reasoning is being performed to the entity or property having a causal influence on it.

```
<FrameSlotPairing rdf:ID="Person_baldness">
  <protege:TO rdf:resource="#baldness"/>
  <protege:FROM rdf:resource="#Person"/>
</FrameSlotPairing>
<owl:Class rdf:ID="SlotList">
        <rdfs:subClassOf rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#List"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="slotList_PR">
        <rdfs:range rdf:resource="SlotList"/>
        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
         >slotList_PR(PR, SL) means that SL is a list of slots (or properties or predicates) by which the
        child  FrameSlotPairing (FSP)  in PR is related to the parent FSP in PR.</rdfs:comment>
        <rdfs:domain rdf:resource="#ProbabilisticRelationship"/>
</owl:ObjectProperty>
```

Given these definitions and entity types, the following list is created to link a person's baldness to the baldness attribute of their maternal grandfather.

```
<SlotList rdf:ID="mother_father">
  <rdf:first rdf:resource="#mother"/>
  <rdf:rest>
    <rdf:List>
      <rdf:first rdf:resource="#father"/>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </rdf:List>
  </rdf:rest>
</SlotList>
```

The causal link is then defined as follows:

```
<ProbabilisticRelationship rdf:ID="baldnessLink">
        <parent_PR rdf:resource="Person_baldness"/>
        <child_PR rdf:resource="Person_baldness"/>
        <slotList_PR rdf:resource="mother_father"/>
</ProbabilisticRelationship>
```

In general, consider the following generic instance of ProbabilisticRelationship:

```
<ProbabilisticRelationship rdf:ID="PR1">
        <parent_PR rdf:resource="FrameA_slotA"/>
        <child_PR rdf:resource="FrameB_slotB"/>
        <slotList_PR rdf:resource="Slot_list"/>
</ProbabilisticRelationship>
```

Suppose further that the value of FROM in FrameA_SlotA is FrameA, and the value of TO is slotA, and the value of FROM and TO in FrameB_slotB are FrameB and slotB, respectively and the value of Slot_list = <slot1, slot2, …, slotN>. Assuming that we define a predicate 'causallyInfluences', we can interpret the predicate that ProbabilisticRelationship represents as follows:

```
(implies
 (and
        (instantiates ?FA FrameA)
        (slotA ?FA ?VALA)
        (slot1 ?FA ?S1VAL)
        (slot2 ?S1VAL ?S2VAL)
        …
        (slotN ?Sn-1VAL ?FrameBInstance)
        (slotB ?FrameBInstance ?VALB))
(causallyInfluences ?VALB ?VALA))
```

In this case, the interpretation is that the value of SlotA for an instance FA of FrameA is causally influenced by the value of slotB on the instance of FrameB linked to FA by the given chain of relations.

5.3 Variable Discretizations

When specifying a continuous (i.e., numeric-valued) variable in a PRM, it is often advantageous to specify a discretization of the space of values into bins of ranges. For instance, it may not be tractable to perform probabilistic inference with a continuous expression for the probabilistic relationship between the variable and its parents. Specifying a discretization and constructing a discrete approximation to the conditional probability distribution can allow the inference to be performed at a desired level of accuracy. In addition, different situations may demand discretizations of different granularities for a given variable. To accommodate this, OWL_QM provides a method for specifying discretizations within the Protégé framework. KEEPER allows the user to specify a discretization for a datatype property with a continuous (e.g. double) range associated to a class. For instance, the user may wish to discretize a double-valued OWL property, reportedTemp, when associated to a particular OWL class, into bins, one of which is the bin HighCelsius from [250,500). In order to specify such a discretization, a base class in the base ontology called RangeOfValues has been defined. Properties on this class are used to represent data about bins, such as the lower and upper bounds, and whether each endpoint is open or closed. In order to define a particular partition (e.g., for temperatures), a subclass of RangeOfValues, (e.g., RangeOfTemperatures) is defined. Each instance of RangeOfTemperatures then specifies a particular named bin (e.g., MidCelsius, [70, 250)). Each endpoint should be specified as open or closed. Thus, to specify the discretization described above, we would have four instances of RangeOfTemperatures. The instance that represents the VeryHighCelsius bin is declared in OWL as follows:

```
<RangeOfTemperatures rdf:ID="HighCelsius">
        <discretizationBinName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
          high
        </discretizationBinName>
        <lowerBound rdf:datatype="http://www.w3.org/2001/XMLSchema#float">
          251.0
            </lowerBound>
        <closedLowerBound rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
            true
        </closedLowerBound>
        <upperBound rdf:datatype=http://www.w3.org/2001/XMLSchema#float>
          500.0
        </upperBound>
        <closedUpperBound rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
          false
        </closedUpperBound>
 </RangeOfTemperatures>
```

This declaration also specifies that, in the QM model (and, thus, the resulting BNs), any slot that is discretized using this bin declaration will have, in its domain of possible values, a value with the name "high". To map a discretization to a FrameSlotPairing, the OWL object property called discretization on the FrameSlotPairing class is used. This is an example of another facet defined on the class of FrameSlotPairings.

```
<owl:ObjectProperty rdf:ID="discretization">
    <rdfs:domain rdf:resource="#FrameSlotPairing"/>
</owl:ObjectProperty>
```

5.4 Representing Probability Distributions and Tables

We have discussed the representational apparatus required to define the probabilistic distributions in a probabilistic relationship. To specify the distributions for a range of values, we must give a probability value for every possible combination of possible states of the parent variables in the relationship and the different values that the attribute or slot can take. So, for example, if an attribute has three possible values and it has two parent attributes, each of which can take on two different values, then it is necessary to state twelve different probability values corresponding to each possible combination of variable-value states. A probability distribution is defined by creating an instance of ConditionalProbabilityTable in which the probability values will be stored. associatedCPT relates a FrameSlotPairing to its associated

ConditionalProbabilityTable. If a FrameSlotPairing has no parents, then the associated table becomes a simple one row table. The values in a ConditionalProbability are contained in instances of CPTCell, each of which is linked to a ConditionalProbabilityTable via the cptCell property.

```
<owl:ObjectProperty rdf:ID="cptCell">
        <rdfs:domain rdf:resource="#ConditionalProbabilityTable"/>
        <rdfs:range rdf:resource="#CPTCell"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="CPTCell">
        <rdfs:subClassOf rdf:resource="#AbstractEntity"/>
</owl:Class>
```

A cell is defined by three attributes. These include an AttributeValuePairList, linked to the cell by the property attributeValueList, which is a list of ordered pairs representing the association of each parent attribute with one of its possible values. relevantValue specifies the value of the attribute in the FrameSlotPairing under consideration. If the slot in the FrameSlotPairing under consideration can take on the values 'true' or 'false', then the value of relevantValue for any cell (cptCell) associated with the FrameSlotPairings's ConditionalProbabilityTable will be either 'true' or 'false'. cellValue gives the probability of that value given the state of the parents as specified in the AttributeValuePairList for that cell.

```
<owl:FunctionalProperty rdf:ID="attributeValueList">
        <rdfs:range rdf:resource="#AttributeValuePairList"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
        <rdfs:domain rdf:resource="#CPTCell"/>
</owl:FunctionalProperty>
<owl:DatatypeProperty rdf:ID="cellValue">
        <rdfs:domain rdf:resource="#CPTCell"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="relevantValue">
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <rdfs:domain rdf:resource="#CPTCell"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
```

The AttributeValuePairList class is the set of lists of AttributeValuePairings. An AttributeValuePairing is an abstract object with two properties. One of the properties, parentSlot, specifies a particular attribute, one that is a parent to the attribute in question; the other one specifies one of the possible values for that slot. A slot is specified with a ProbabilisticRelationship instance. The actual attribute having the causal influence will be the slot in the FrameSlotPairing that is the value of parent_PR in the ProbabilisticRelationship. We refer to the ProbabilisticRelationship to clearly disambiguate, as the same FrameSlotPairing could play causal roles in different ways. If the baldness of both my maternal and paternal grandfather influence the probability of my own baldness, then the operative FrameSlotPairing would be Person_baldness in both instances, but we must disambiguate which person's baldness plays which causal role. Referring to the relevant ProbabilisticRelationship instance does that because the slot list associated with the ProbabilisticRelationship can be used to perform the disambiguation.

```
<owl:Class rdf:ID="AttributeValuePairing">
        <rdfs:subClassOf rdf:resource="#AbstractEntity"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="parentSlot">
        <rdfs:domain rdf:resource="#AttributeValuePairing"/>
        <rdfs:range rdf:resource="#ProbabilisticRelationship"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="value">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:domain rdf:resource="#AttributeValuePairing"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

```
</owl:DatatypeProperty>
```

Consider how this would be implemented for the PRM that we just mentioned, i.e., let us suppose that in our PRM, the user wants to assert that if one's maternal grandfather is bald then the probability that that person will be bald is .7 and if the maternal grandfather isn't bald, then the probability that the person will be bald is only .2. To create a cell for this table, a ConditionalProbabilityTable is associated with the relevant FrameSlotPairing, Person_baldness:

```
<FrameSlotPairing rdf:ID="Person_baldness">
 <associatedCPT>
  <ConditionalProbabilityTable rdf:ID="CPT_Person_baldness"/>
 </associatedCPT>
 <protege:TO rdf:resource="#baldness"/>
 <protege:FROM rdf:resource="#Person"/>
</FrameSlotPairing>
```

The cells in the table are then defined. Consider how to create the cell specifying that when the maternal grandfather is bald, the probability that a person will be bald is .7. First, an AttributeValuePairing is created associating the baldness attribute of the grandfather with the value "true". The value of parentSlot will be the ProbabilisticRelationship created above, i.e., PersonBaldness, and we link that to the value 'true'. The representation is as follows:

```
<AttributeValuePairing rdf:ID="GrandfatherBaldness_true">
 <Value rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >true</Value>
 <parentSlot>
  <ProbabilisticRelationship rdf:ID="baldnessLink"/>
 </parentSlot>
```

A list is then made of all the parent variable values for that cell. Since there is only one parent to consider, the list is made up of only one element, the description of the state in which the grandfather is bald.

```
<AttributeValuePairList rdf:ID="AttributeValuePairList_grandfather_bald">
 <rdf:first>
  <AttributeValuePairing rdf:ID="GrandfatherBaldness_true">
   <Value rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >true</Value>
   <parentSlot>
    <ProbabilisticRelationship rdf:ID="baldnessLink"/>
   </parentSlot>
  </AttributeValuePairing>
 </rdf:first>
 <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
</AttributeValuePairList>
```

This list then becomes one of the values in an instance of CPTCell. It is also necessary to specify that the probability that the person will be bald, i.e., that 'baldness == true' for that person will therefore be .7. This cell is then associated with the original cell and the following representation results:

```
<FrameSlotPairing rdf:ID="Person_baldness">
 <associatedCPT>
  <ConditionalProbabilityTable rdf:ID="CPT_Person_baldness">
   <cptCell>
    <CPTCell rdf:ID="Person_bald_grandfather_bald">
     <cellValue rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal"
     >0.7</cellValue>
     <attributeValueList>
      <AttributeValuePairList rdf:ID="AttributeValuePairList_grandfather_bald">
       <rdf:first>
        <AttributeValuePairing rdf:ID="GrandfatherBaldness_true">
         <Value rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
         >true</Value>
         <parentSlot>
          <ProbabilisticRelationship rdf:ID="baldnessLink"/>
         </parentSlot>
        </AttributeValuePairing>
```

```
        </rdf:first>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
       </AttributeValuePairList>
      </attributeValueList>
      <relevantValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >true</relevantValue>
     </CPTCell>
    </cptCell>
          …
   </ConditionalProbabilityTable>
  </associatedCPT>
```

Note that the above represents but one cell in the CPT for a fairly simple distribution. The markup required for a more elaborate table is much more involved.

6 Conclusion

We have presented the basic components required to extend the OWL language to represent PRMs. This work is of a kind with recent proposals to extend OWL by merging it with RuleML, insofar as we are seeking to find a relatively lightweight extension of OWL that will extend expressiveness without an overwhelming increase in representational complexity. However, the development of this extension presented some surprises in terms of the representational complexity needed to implement OWL for PRM representation. A number of facts seem to argue against OWL as an appropriate foundation for something sufficiently expressive to handle probabilistic models.

a) Despite the *prima facie* analogous structure, the mapping from OWL classes and properties to PRM structures such as QM frames and slots is not meaning preserving, particularly with respect to the semantics of overriding. In particular, since OWL does not directly support attaching additional information at a "slot-at-class" level, additional structure must be constructed in OWL to support the mapping.
b) The implementation of PRMs in OWL requires the construction of numerous higher order entities like FrameSlotPairings, SlotChains, ProbabilisticRelationships, as well as slots that have classes and slots as domains and ranges. The construction and implementation of such higher order entities is not well supported in OWL.
c) Representation of probabilistic structures and distributions requires the extensive use of lists. Like higher order entities, lists in OWL languages are second-class citizens in OWL enjoying minimal support in terms of parsing, reasoning or even editing and construction in the extant OWL tools. They are implemented in highly complex structures that are difficult to manage and use.
d) Related to (a), (b), and (c), representation of PRMs in OWL is extremely complicated as compared to the native syntax of most PRM languages. A PRM that is representable in twenty lines in a well-suited format could quite conceivably require hundreds or thousands of lines in OWL_QM. In this sense, OWL_QM is disanalogous with a SWRL extension. Of course, OWL is designed for machine readability rather than human readability. Nevertheless, parsimony appears to argue against an OWL implementation of uncertainty models.

Much of this complexity of structure and bulkiness of model specifications can be hidden from users and kept in the background. Extant tools for wizard development available in ontology development GUIs allow us to implement these representational tools with reasonable effort. Nevertheless, it is natural to ask whether OWL would be the base for semantic web reasoning had it not been for the fact that it was the language with which development began. An important question to ask here is whether OWL is the correct foundation for probabilistic representation or whether a more reasonable approach would be to dismiss the OWL paradigm and introduce or adapt a competing representational paradigm for uncertainty reasoning in the semantic web context.

If the representational complexity described above is rendered mostly irrelevant because of the ability to implement effective editing tools, the next steps in this process is to develop the abstract syntax and

formal semantics for this extension. For these purposes we would look to [1] as a possibly compatible approach to formally expressing the semantics.

References

1. Costa, Paulo, Bayesian Semantics for the Semantic Web, PhD Dissertation, 2005.
2. Cozman, F.G. The Interchange Format for Bayesian Networks, http://www.cs.cmu.edu/~fgcozman/Research/InterchangeFormat/.
3. Getoor, Lise, Nir Friedman, Daphne Koller, and Avi Pfeffer. Learning Probabilistic Relational Models. In Saso Dzeroski and Nada Lavrac, eds., Relational Data Mining, Springer-Verlag, New York, 2001.
4. Horrocks, Ian, *et al*, SWRL: A Semantic Web Rule Language Combining OWL and RuleML, May, 2004, http://www.w3.org/Submission/2004/SUBM-SWRL-20040521.
5. IET, Inc., Quiddity Technical Guide, 2005, see www.quiddity.com.
6. IET, Inc., The Role Of Ontologies in Probabilistic Knowledge Representation, IET Technical Report, Oct. 2004.
7. Koller, D., A. Levy, A. Pfeffer, P-Classic: A Tractable Probabilistic Description Logic. Proceedings of the AAAI Fourteenth National Conference on Artificial Intelligence, 1997.
8. Laskey, Kathryn. First-Order Bayesian Logic. Technical Report, George Mason University Department of Systems Engineering and Operations Research, April 2005.
9. Laskey Kathryn, Suzanne Mahoney, and Edward Wright. Hypothesis Management in Situation-Specific Network Construction. In Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference, Morgan Kaufmann Publishers, San Mateo, California, 2001.
10. Minsky, M. "A Framework for Representing Knowledge." in P. H. Winston (Ed.) The Psychology of Computer Vision, NY:McGraw-Hill, 1975.
11. Mcguinness, D, Van Harmelen, Frank, OWL Web Ontology Language Overview, W3C Recommendation, February 10, 2004. http://www.w3.org/TR/2004/REC-owl-features-20040210/.
12. Ngo, Liem, and Peter Haddawy. Answering Queries from Context-Sensitive Probabilistic Knowledge Bases. Theoretical Computer Science, 171:147-171, 1996.
13. Patel-Schneider, Peter, Hayes, Patrick, Horrocks, Ian. OWL Web Ontology Language Semantics and Abstract Syntax, February 2004. http://www.w3.org/TR/owl-semantics/.
14. Pearl, Judea. Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann, San Mateo, CA, 1998.
15. Zhongli Ding and Yung Peng, A Probabilistic Extension to Ontology Language OWL, Proceedings of the 37th Hawaii International Conference on System Sciences, 2004.
16. Russell, Stuart, and Peter Norvig, "Artificial Intelligence: A Modern Approach", 2nd edition, Prentice-Hall, Upper Saddle River, NJ. 2003.
17. David J. Spiegelhalter, Andrew Thomas, and Nicky Best. Computation on Graphical Models. Bayesian Statistics, 5: 407-425, 1996.
18. Tudorache, Tania, Representation and Management of Reified Relationships in Protégé, Protégé Conference, Bethesda Maryland, July, 2004.