There's No More Need to be a Night OWL: on the PR-OWL for a MEBN Tool Before Nightfall

Shou Matsumoto[1], Rommel N. Carvalho[2], Paulo C.G. Costa[2],
Kathryn B. Laskey[2], Laécio L. dos Santos[1], and Marcelo Ladeira[1]

[1]*Departamento de Ciência da Computação,*
Universidade de Brasília, Brazil

[2]*Center of Excellence in C4I,*
George Mason University, USA

1 Introduction

The area of semantics and services has been extremely active in recent years. Advances in making semantic information explicit and machine-interpretable were key to effectively exploit data from disparate sources. In such context, Web Semantic technologies, such as *ontologies* - explicit formal definition of common set of terms describing and representing domains and their relations - have emerged as an integration framework to record, distinguish, operate, and integrate web resources.

The ability to support inferential reasoning makes ontologies a promising tool for knowledge management. Traditional ontology languages, such as the *Web Ontology Language* - OWL (Patel-Schneider et al., 2004; Heflin, 2004), allow computers to effectively infer multiple meanings for a word, such as "night,"[1] but it lacks the ability to finely grade their respective plausibility within contexts. This limitation comes directly from the fact that inferential reasoning in traditional ontologies relies on different flavors of classical logic to support automated reasoning.

Although acceptable in a relatively small and well-defined domains with complete and consistent information sources, this is a serious limitation when dealing with open world environments such as the Semantic Web, in which information comes from diverse sources and is often incomplete, inaccurate, or unreliable. Such circumstances could, notably, lead to poor knowledge reuse and integration, reinforcing the need for means to represent and to reason with uncertain information in a principled fashion. For instance, representations based on natural language are intrinsically uncertain, and with little effort anyone can find domains that are strictly bounded to dealing with uncertainty.[2] *How can we share, integrate or reuse information when there are so many sources of uncertainty? PR-OWL*, pronounced as "prowl", is an approach to face that shortcoming, by extending OWL to enable probabilistic ontology representation. It is a language based on *MEBN - Multi-Entity Bayesian Network*, a formalism that brings together the expressiveness of first-order logic and the inferential power of Bayesian Networks.

[1] E.g. a time from dusk to dawn, or periods of absence of moral values.
[2] Quantum physics may be a good example of an uncertainty-bounded domain.

This Chapqter invites you to understand the problems beneath the classical models (see Section 2), and to appreciate some fundamental theoretical principles and methods for modeling probabilistic ontologies (see Section 3), presenting you with a step-by-step construction of a simple probabilistic ontology for image-based vehicle identification using an open-source, freely available, Java-based GUI tool (see Section 4). It is intended to help you learn probabilistic ontologies before nightfall, so that you don't fall asleep calculating your probabilities.

2 Problem Description

Traditional ontologies are indeed powerful, but their classical logic basis provides no built-in support for uncertainty. This is a major shortcoming for Semantic Web applications, which are often required to perform automated reasoning in complex, uncertain open-world environments.

Historically, several approaches have been adopted by researchers and engineers to express uncertainty, such as *Dempster-Shafer's belief functions* (Dempster, 1967; Shafer, 1976) and rule-based systems with *Certainty Factors* (Buchanan & Shortliffe, 1984). *Fuzzy logic* models also have increasingly gained popularity for representing and processing degrees of truth about imprecise or vague pieces of informations, becoming a strong candidate for incorporating impreciseness to ontologies (Stoilos et al., 2005; Stoilos et al., 2007). Belief functions' ability to explicitly model degrees of ignorance still makes them very appealing in inconsistency handling (Nikolov et al., 2007) or ontology mapping (Yaghlane & Laamari, 2007).

As a recognition of this issue, the *Uncertainty Reasoning for the World Wide Web Incubator Group* (URW3-XG) (Laskey et al., 2007b) was created as an incubator activity of World Wide Web Consortium (W3C) to better define the challenge of reasoning with and representing uncertain information through the World Wide Web. Its final report (Laskey et al., 2008) provided an important milestone for characterizing the range of uncertainty that affects reasoning on the scale of the World Wide Web, and the issues to be considered in designing a standard representation.

The use of graphical probabilistic models is an appealing way of enabling information systems to coherently exploit uncertain, incomplete information[3]. *Bayesian Networks* (BN) (Pearl, 1988), a graphical, flexible means of expressing joint probability distributions over interrelated hypotheses, is one of the most promising probabilistic approaches to representing uncertainty.

For Semantic Web applications, BNs have the potential to provide well defined semantics, a powerful inference mechanism, and a compact structure. However, BNs have some key limitations hampering their use in the Semantic Web. These limitations are summarized to, but not limited to, the following aspects: (a) the number of variables must be known in advance (*i.e.*, the number of nodes in a BN is fixed); (b) The BN language is not expressive enough to represent problems with repeated structure; (c) since a BN is a directed acyclic graph, no native support for recursive (cyclic) definition is provided.

A number of formalisms have appeared in order to extend the expressiveness of standard probabilistic graphical models. These include *Dynamic Bayesian Networks* (Murphy, 2002), *Object-Oriented Bayesian Networks* (Koller & Pfeffer, 1997), *Probabilistic Relational Models* (Getoor et al., 2001), and *Markov Logic Networks* (Richardson & Domingos, 2004). Choosing a feasible probabilistic extension of an ontology model is a huge step towards the representation and reasoning under uncertainty. Not surprisingly, many of the probabilistic extensions in the literature are based on description logics (DL), a decidable subset of first-order logic (FOL) and a basis for most of OWL profiles. Examples of such DL-based extensions are

[3] A key advantage of graphical models is the ability to express knowledge as interrelated modular components. This ability improves tractability, making the knowledge engineering task feasible.

P-$\mathcal{SHOQ}$(D) (Giurno & Lukasiewicz, 2002), and P-$\mathcal{SHIQ}$(D) (Klinov & Parsia, 2008). A clever usage of markups, allowing OWL ontologies to be translated to probabilistic network models is also a feasible option (Ding, 2005). However, the ability of DL-based models to represent and reason about other commonly occurring kinds of knowledge is limited.[4] Ontologies need a highly expressive language to capture all the relevant informations of a given domain. OWL, so as other ontology languages, rely in variations of FOL, so a probabilistic FOL would be a more natural approach for incorporating probabilistic reasoning to a more general use-case.

Probabilistic Web Ontology Language (*PR-OWL*) was proposed by (Costa, 2005) as a means to provide a basis for representation and reasoning under uncertainty within the Semantic Web context and applications. This probabilistic ontology framework uses *Multi-Entity Bayesian Network (MEBN)* (Costa & Laskey, 2005; Costa & Laskey, 2006) as its underlying logic. MEBN extends BN to the expressiveness of a first-order Bayesian logic. Sections 3.1 and 3.2 provides an overview of the main concepts underlying BN and MEBN respectively. Also, a brief description of PR-OWL is provided in Section 3.3.

As it was once wisely stated, with (representational) power comes responsibility. Sure enough, modeling probabilistic ontologies (POs) is not an easy task. PR-OWL ontologies can become huge, complex, and even intractable as more domain information is included. Also, since MEBN allows for an unlimited number of variables (*i.e.* nodes), a *Local Probability Distribution (LPD)*[5] cannot be declared statically. In other words, since a node can have different conditioners (*i.e.* parents) in a particular situation[6], probabilities must be declared in a dynamic fashion. This hints at the need for a computer-aided visual tool for editing in PR-OWL, with a built-in reasoner and support for dynamic LPD declaration. Fortunately, you will find all this in Section 4 , which uses a sample probabilistic ontology as means to provide a smooth introduction to this fairly complex logic.

3 Solving Uncertainty Using Probabilities

Definition 1 *A probabilistic ontology is an explicit, formal knowledge representation that expresses knowledge about a domain of application. This includes: (a) types of entities existing in the domain; (b) properties*[7] *of those entities; (c) relationships among entities; (d) processes and events that happen with those entities; (e) statistical regularities that characterize the domain; (f) inconclusive, ambiguous, incomplete, unreliable, and dissonant knowledge; (g) uncertainty about all the above forms of knowledge. Where the term entity refers to any concept (real or fictitious, concrete or abstract) that can be described and reasoned about within the domain of application (Costa, 2005).*

The purpose of a probabilistic ontology is to describe knowledge about a domain and its associated uncertainty in a principled, structured, and sharable way, so that it can be applied to support semantic applications working in complex open-world environments. As mentioned previously in Section 2, traditional ontologies have no built-in mechanism for representing or inferring with uncertainty. PR-OWL, which addresses this shortcoming, consists of a set of classes, subclasses and properties that collectively form a framework for building and reasoning with probabilistic ontologies. With MEBN as its underlying logic, PR-OWL ontologies avoid the limitations of Bayesian Networks.

[4] One restrictive aspect of DL languages is their limited ability to represent constraints on the instances that can participate in a relationship.

[5] A LPD is a function to calculate the probability distribution for a random variable, given the values of its local conditioners.

[6] Here, a single "situation" can be thought as a particular configuration of a knowledge base (*i.e.* a set of individuals and evidence items).

[7] In this chapter, *property* is defined as an attribute or abstract quality associated with an individual or common to a concept.

Since PR-OWL is written in OWL, it is possible to import PR-OWL ontologies into regular OWL editors (e.g. OntoStudio[8], Knoodl[9], Protégé[10]) and edit them, although logical reasoning using standard Description Logics (DL) reasoners may not be supported. The latter issue is related to the fact that mixing logical and probabilistic reasoning is still a work in progress by many researchers, including the team behind PR-OWL/MEBN. In spite of the reasoning aspects, building PR-OWL ontologies within a pure OWL editor is a manual, tedious, error prone task that requires deep understanding of the PR-OWL foundations. The development of UnBBayes-MEBN[11] (Carvalho et al., 2008; Carvalho et al., 2010a) began with the main goal of addressing this issue. UnBBayes-MEBN is an open source, plug-in-based Java application that provides an easy way of building probabilistic ontologies and perform plausible reasoning based on the PR-OWL/MEBN framework. It features a graphical user interface (GUI), an application programming interface (API) for saving and loading PR-OWL ontologies, as well as plug-in support for unforeseen extensions. Section 4.5 describes the grammar used in UnBBayes-MEBN that addresses the dynamic LPD construction. We hope this tool will save many "PR-OWLers" from arduous "night-OWLing" work.[12]

3.1 BN Overview

A *Bayesian Network* (BN) (Pearl, 1988) is a graphical model based on Bayesian probability. It consists of a directed acyclic graph and a set of local probability distributions. Each node in the graph represents a random variable[13], and the edges represent direct qualitative dependencies of the random variables. The local probability distribution is a function (usually represented as a *Conditional Probability Table* - CPT) specifying the quantitative information about the "strength" of the dependencies. Each random variable has a set of possible values, and exactly one of the possible values will be the actual value in a given moment (but we are uncertain about which one). The graph and the local distributions together represent a joint probability distribution over the random variables.

Bayesian probability models provide a sound representation and formal calculus for rational degrees of belief or likelihood of a proposition. Reasoning systems based on Bayesian models relies on *Bayes rule*, a theorem which provides a method of updating the probability of a proposition when information is acquired about a related proposition. In the Bayesian view, inference is a problem of belief change as new information becomes available. The standard format of Bayes rule is:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)},$$

$P(X|Y)$, which is the posterior probability of event X, represents our belief in event X after applying Bayes rule with the information collected from event Y. $P(Y)$ is the prior probability of Y, and represents our belief in event Y before obtaining evidence. Similarly, $P(X)$ is the prior probability of X. $P(Y|X)$ is the likelihood of event Y given that event X has occurred.

The left side of Figure 1 illustrates an example for identifying a vehicle's type based on a single report from image sensors. Basically, it states that the place where the vehicle is (*TerrainType*) may indicate the type of vehicle (*ObjectType*). Additionally, it also states that the weather conditions (*Weather*) can cause

[8] OntoStudio's project page: <http://www.ontoprise.de/en/home/products/ontostudio/>

[9] Knoodl's project page: <http://www.knoodl.com/ui/home.html>

[10] Section 4 presents some Protégé usage for PR-OWL ontology.

[11] Interested readers may want to visit UnBBayes project's home page (<http://unbbayes.sourceforge.net/>) to check the development stage of the tool.

[12] In fact, it already spared us from several nights of insomnia, a natural consequence of tedious "manual" work of encoding PR-OWL ontologies.

[13] A random variable denotes a set of attributes or hypotheses which we may be uncertain.

imaging (*ImageTypeReport*) to fail. Interested readers may want to check this example again after reading Section 4.7, because this BN is also a very specific situation in the Vehicle Identification domain.

One of the key strengths of reasoning in BN is its ability to update the beliefs of each random variable via bi-directional propagation of new information through the whole structure. The right side of Figure 1 illustrates the same BN after discovering that the object was on road, the sensor was reporting a wheeled vehicle, and the weather was clear. It is stating that the probability of the object to be a wheeled vehicle is 55.81% now (posterior probability), which is slightly higher than before, which was 45% (prior probability).

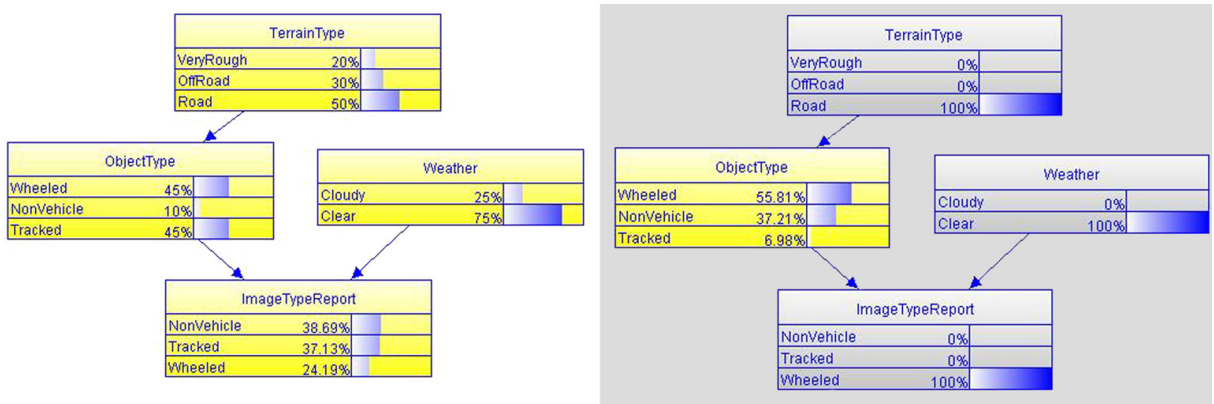

Figure 1: A Bayesian network for vehicle identification based on image sensors (on the left). The same BN after entering evidences/findings is presented on the right side.

Bayesian networks have been successfully applied to create consistent probabilistic representations of uncertain knowledge in various fields, such as language understanding (Charniak & Goldman, 1989), image recognition (Booker & Hota, 1986), and medical diagnosis (Spiegelhalter et al., 1989). However, as mentioned in Section 2, their shortcomings in expressiveness for many real-world applications have become increasingly apparent as they have grown in popularity.

3.2 MEBN Overview

Multi-Entity Bayesian Networks (MEBN) extends Bayesian Networks (BN) to achieve first-order expressiveness. MEBN represents the world as a collection of inter-related entities, their respective attributes, and relations among them. Knowledge about attributes of entities and their relationships is represented as a collection of repeatable patterns, known as *MEBN Fragments* (MFrags). A set of well-defined MFrags that collectively satisfies first-order logical constraints ensuring a unique joint probability distribution is a *MEBN Theory* (MTheory) (Costa & Laskey, 2006). To be considered a complete PR-OWL ontology, a group of MFrags should form a MTheory.

An MFrag represents uncertain knowledge about a collection of related *random variables* (RVs). RVs, also known as "nodes" of an MFrag, represent the attributes and properties of a set of entities. A directed graph represents dependencies among the RVs. Since an MFrag is in fact a template that can be repeatedly instantiated to form *Situation Specific Bayesian Networks* (SSBNs), their RVs usually contain as arguments one or more *ordinary variables*, which are variables that are substituted by instances of entities during the instantiation process. SSBNs are regular BNs that are formed, usually in response to a query, to address a particular situation of a domain knowledge. Since a SSBN is just a regular BN, traditional BN algorithms can be applied to it with no special adaptations. Usually, a SSBN would look like a collection of "similar"

nodes, differing only by their arguments' values.

MEBN categorizes random variables into three different types (see figure 2 for a graphical representation):

- **Resident nodes** these are the actual random variables that form the core subject of an MFrag. MEBN logic requires that the local probabilistic distribution of each resident node should be uniquely and explicitly[14] defined in its home MFrag. The possible values of a resident node can be either an existing entity or an ad-hoc list of mutually exclusive and collectively exhaustive values.

- **Input nodes** these nodes are basically "pointers" referencing to another MFrag's resident node. Input nodes also provide a mechanism to allow resident nodes' re-usage between MFrags.[15] Input nodes influence the probability distribution of the resident nodes that are their children in a given MFrag, but their own distributions are defined someplace else (*i.e.* in their own home MFrags). In a complete MTheory, every input node must point to a resident node in an MFrag.

- **Context nodes** these are boolean random variables representing conditions that must be satisfied to make a distribution in an MFrag valid. Context nodes can represent several types of sophisticated uncertainty patterns, such as the uncertainty about relationships among entities.[16] If we can infer from the knowledge base (i.e. ontology) that the value of a context node is true, the probability distribution of the MFrag will be applied in the inference model. If this value is false, a default distribution will be used instead. If the value is unknown, the context node becomes virtually a parent of all the resident nodes in the same MFrag.

When designating an entity as a possible value for a resident node, we must take extra care, because all individuals in the entity become a possible value of the node. In this case, it is difficult to state the probability distributions explicitly, because the number of possible values is indefinite. On these occasions, it is more common to assign an implicit distribution (usually uniform), and use them mostly as components for context nodes. The set of constraints on reasoning can be adjusted through the entrance of evidences in these "special" context nodes.

MEBN provides a compact way to represent repeated structures in a Bayesian Network. An important advantage of MEBN is that there is no fixed limit on the number of random variable instances, which can be dynamically instantiated as needed. Some may see MFrags as tiny "chunks of knowledge" of a given domain. Since a MTheory is a consistent composition of such "chunks", MEBN (as a formalism) is suitable for use cases addressing knowledge fusion (Laskey et al., 2007a).

3.3 PR-OWL Overview

An immediate workaround for representing uncertainty in languages lacking inbuilt support for it is to use custom annotations with numerical degrees (e.g. custom-designed XML tags to add probabilistic values). However, this approach cannot represent important structural features of the domain, which can result in a substantial loss of information. For instance, (Shafer, 1986) stated that probability is more about structure than it is about numbers. It is important for a language for probabilistic ontologies to be able to represent this structure.

[14] If local probabilistic distribution is not explicit, UnBBayes-MEBN assumes uniform distribution as default.

[15] It is possible to model "immediate" recursion by pointing input nodes as parents of a resident node within the same MFrag. Special care should be taken to avoid cycles in the resulting SSBN.

[16] The dependencies (relationships) between variables in an MFrag become conditioned by context nodes (which can also be random variables). In this way, the "existence" and "strength" of relationships become uncertain.

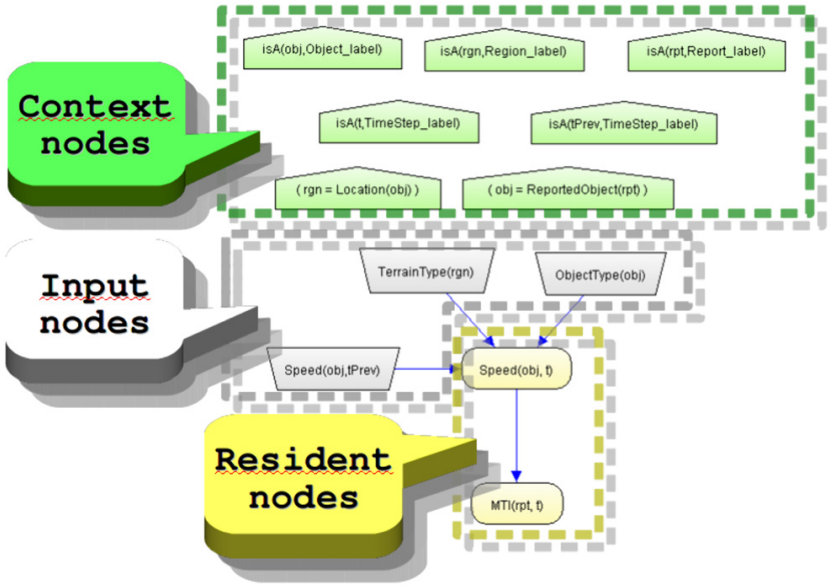

Figure 2: Structure of an MFrag. Directed arrows going from parent to child variables represent dependencies between them. Nodes can contain a list of arguments in parenthesis, which are replaced by unique identifiers when the SSBN instantiation process is triggered.

The *Probabilistic Web Ontology Language* (PR-OWL, pronounced as "prowl") is an OWL extension for representing probabilistic ontologies, thus providing a means of representing subtle features required to express a first-order Bayesian theory. As a consequence of its expressiveness, *there are no guaranties that reasoning with PR-OWL ontology will be efficient or even decidable* (Costa, 2005). PR-OWL structures are in fact MEBN structures, so they can be thought of as a scheme to represent a valid complete MTheory using OWL definitions.[17] Figure 3, extracted from (Costa, 2005), shows the main concepts involved in defining a MTheory in PR-OWL.

In Figure 3, ellipses represent general OWL classes, while arrows represent the main relationships between them. A valid PR-OWL ontology must contain at least an instance of class MTheory, which is basically an individual linking a group of individuals of MFrags that collectively form a MEBN Theory. In PR-OWL syntax, these links are expressed via the *hasMFrag* OWL object property. Its inverse OWL property is *isMFragIn*.

Looking more deeply inside the PR-OWL definitions, individuals of *MFrag* class are comprised of individuals of *Node*. Each individual of *Node* is a random variable, and thus has a mutually exclusive set of possible states. In PR-OWL ontologies, the OWL object property *hasPossibleValues* links each node to its possible states, which are individuals of *Entity*. Finally, RVs have unconditional or local probability distributions represented as individuals of the *ProbabilityDistribution* class and linked to their nodes via the OWL object property *hasProbDist*.

Some may have felt that PR-OWL is merely a simple direct representation of the structures of MEBN in OWL. However, structures of OWL are well mappable to the structure of PR-OWL (see Table 1), which makes PR-OWL a compatible probabilistic extension of OWL. Unfortunately, some standard still needs to be established in order to represent this mapping as a meta-information, so that the inference engine can catch

[17] The current PR-OWL (version 1.05) is an OWL ontology representing MEBN concepts.

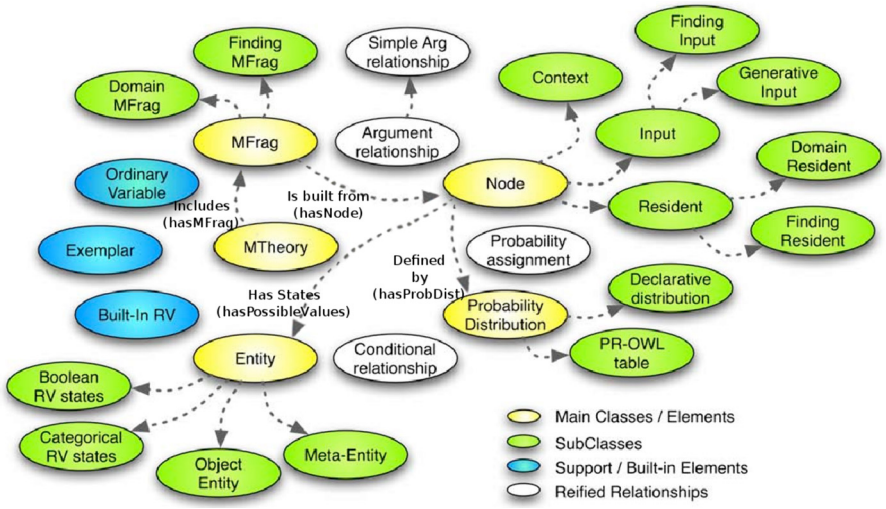

Figure 3: Basic classes composing a PR-OWL ontology. The instantiation of individuals from these classes results in a MTheory, the probabilistic slice of a PR-OWL ontology.

the knowledge derived from both parties. A more elaborate mixture between logical and probabilistic representation/reasoning is still a work in progress. For more information on such standard see the announcement and a glimpse of PR-OWL 2.0 in (Carvalho et al., 2010b; Carvalho et al., 2010c).

OWL	PR-OWL
Classes under *Thing*	Classes under *Entity* (which is also a class under *Thing*)
Object and data properties	*Resident* class (resident nodes)
Assertion components (ABox)	Individuals (instances of *Entity*) and findings
Restrictions on classes and properties	*Context* class (context nodes)

Table 1: A mapping between elements of OWL to PR-OWL.

Major advantages of using PR-OWL include its flexibility and expressiveness, both inherited from first-order expressiveness and inferential power of MEBN. The prospective reader will find useful links on PR-OWL's home-page (<http://www.pr-owl.org>). UnBBayes-MEBN, our PR-OWL visual modeling tool, leverages PR-OWL's usability with a built-in MEBN reasoner. The next section goes through the fundamental steps for building a PR-OWL model using UnBBayes-MEBN, while also providing an overview of the tool's development stage at the time of this writing (Summer 2010).

4 Modeling Your First PR-OWL Ontology

This section provides an approach for modeling an example ontology for the domain of Vehicle Identification[18]. This ontology represents knowledge related to the use of sensor reports and other available informa-

[18]The last release of UnBBayes-MEBN (version 1.4.0) comes with some sample ontologies, including the Vehicle Identification. You may want to use it to follow this section.

tion to classify an object as a vehicle or non-vehicle. A step-by-step modeling process will be presented, in order to smoothly introduce the flow of probabilistic ontology development. However, covering all the aspects that an ontology developer may need to grapple with is a much larger task that is outside the scope of this Chapter. Instead, our objective here is to provide you with a starting point: an initial guide for helping new probabilistic ontology designers to develop POs.

This ontology was built based on our experience using UnBBayes-MEBN (see Section 4.1), as well as Protégé[19] 4.0.2, which we use as a supporting ontology-editing environment. Although probabilistic ontology development is substantially different from object-oriented program design, some ideas used in the former were borrowed from the latter (Costa, 2005; Rumbaugh et al., 1991). While object-oriented programming focuses in the operational properties of classes, an ontology designer should make design decisions based on structural properties. As a result, our structures and relationships will not be strictly "object-oriented".

We start with a fresh MEBN model, and then proceed with an iterative MFrag creation process until achieving a complete MTheory. Finally, we will run some queries against the model and analyze its results.

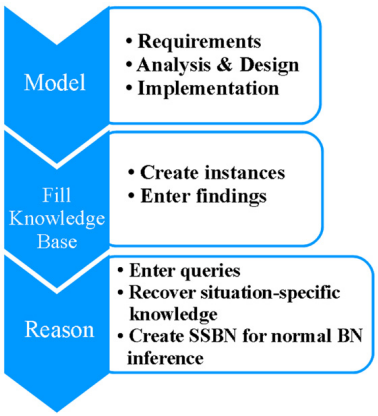

Figure 4: The UMP-SW modeling process. It can be separated into 3 inter-dependent phases: modeling, knowledge base population, and reasoning. Usually, the modeling phase is a iterative process.

Figure 4 presents the Uncertainty Modeling Process for the Semantic Web (UMP-SW) described in detail in (Carvalho et al., 2010a). The three most basic phases of the process of developing a probabilistic ontology are:

1. **Model**: list the requirements, identify the goals, define the domain scope, and implement the ontology.
2. **Fill**: populate the knowledge base, creating individuals (instances) and evidence (findings).
3. **Reason**: enter queries, recover hidden informations, checking if the results are acceptable (usually, with a help from stakeholders), and debug eventual errors.

In theory, there is no substantial difference between traditional (Bechhofer et al., 2003) and probabilistic ontology development. Table 2 illustrates the similarities between their respective steps.

[19] Protégé's project page: <http://protege.stanford.edu/>

Classical ontology	PR-OWL ontology
Defining classes and taxonomies.	**(Model)** Defining entities (andtaxonomies) and MFrags in the ontology.
Specifying properties of a class, determining restrictions.	**(Model)** Defining random variables, ordinary variables allowed values (e.g. types) and restrictions (context nodes).
Adding individuals to ontology and filling in the values for properties.	**(Fill)** Adding instances to entities and filling in findings (evidences) of random variables.
Reasoning over instances, and discovering hidden facts.	**(Reason)** Triggering MEBN reasoning, and inferring probabilities.

Table 2: Equivalences between classical and PR-OWL ontology modeling processes.

Actually, there is no single correct ontology-design methodology. The ideas we present in this section are the ones we found most useful, based on our own empirical process and observed results. Here we discuss general issues that should be considered and comment on some modeling decisions for probabilistic ontology development. The approach is iterative: we start with a rough first draft, and then revise and refine the evolving ontology, filling in the details. Here are some fundamental rules in ontology design that are helpful when making design decisions:

1. *There is always more than one viable alternative to model a domain*: the best solution almost always depends on the application you have in mind, and the extensions that you anticipate. That is, the solution depends on what you are going to use the ontology for, how detailed the ontology is going to be, and how frequently it is going to be modified.

2. *Ontology development is an iterative process*: after we define an initial version of the ontology, we can evaluate it by using applications and/or reasoners, by discussing it with domain experts, or by combining multiple criteria (Gomez-Perez, 1994; Brank et al., 2005). As a result, it is almost certain that you are going to revise the initial ontology. This process of iterative design will likely continue through the entire life-cycle.

3. *Concepts in the ontology should be close to logical/physical objects and relationships of your domain*: an ontology is a model of reality of the world, and the concepts in the ontology should reflect this reality in a intuitive manner.

4.1 On the Prowl for a Golden Hammer - UnBBayes-MEBN

Tools for probabilistic ontology creation are still rare, making the usage of the PR-OWL language very difficult. When building PR-OWL ontologies via a graphical ontology editor such as Protégé, the PR-OWL definitions can be imported. This makes the task a bit easier since it is not necessary to remember all the details and OWL tags that should be filled. However, the input of information is not intuitive, and the user is forced to know all the PR-OWL-specific terms, such as *hasPossibleValues*, *isNodeFrom*, *hasParents*, etc...

Many of these terms could be ignored by the end user and filled in automatically by a software application such as UnBBayes-MEBN, which was designed in a way that enforces the consistency of a MEBN model. UnBBayes-MEBN is a plug-in for the UnBBayes project[20]. It allows the user to build a probabilistic ontology in an intuitive way, eliminating the need to rely on deep knowledge about the PR-OWL

[20]UnBBayes-MEBN plug-in (currently at version 1.7.0) and UnBBayes (currently at a stable version - 4.1.1) belongs to the same open-source project; thus, both can be downloaded at <http://unbbayes.sourceforge.net/>.

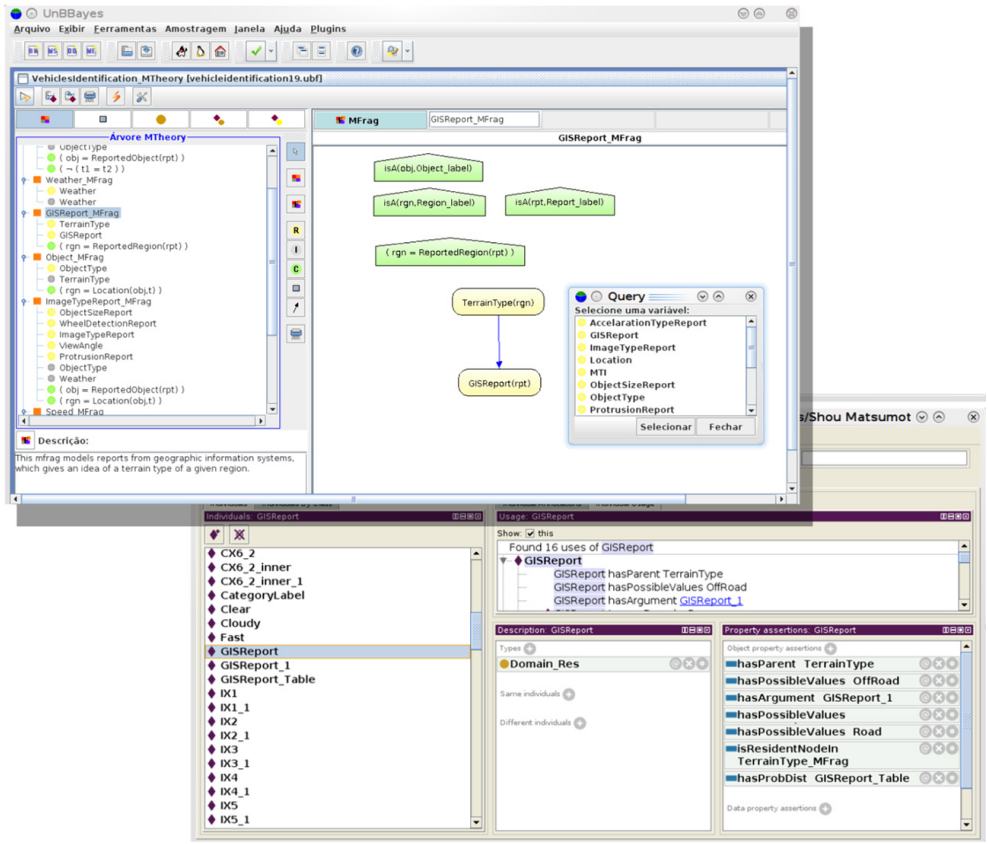

Figure 5: UnBBayes-MEBN (top) has a graph-like representation, abstracting PR-OWL ontologies as MEBN Fragments. Protégé (bottom) exposes taxonomic details about PR-OWL-specific classes and properties.

specification. To follow the PO creation process depicted in this Chapter, we recommend downloading UnBBayes core and UnBBayes-MEBN plug-in[21]. Figure 5 illustrates the difference between Protégé and UnBBayes-MEBN user interfaces when editing PR-OWL ontologies.

Since it was specifically designed for this purpose, UnBBayes-MEBN has several advantages over Protégé when building PR-OWL ontologies. However, the implementation of a complex logic such as MEBN, while still focusing on usability requirements, demands some trade-offs on performance, decidability, expressiveness, and ease of use. As a result, the UnBBayes-MEBN design includes some simplifications intended to provide a working, feasible tool for PO modeling. Fortunately, since UnBBayes provides a plug-in infrastructure, additional features can be added on demand, addressing limitations when needed. Some of the default simplifications include:

- The *IsA* property, a binding between ordinary variables and its type, is supported as a built-in feature[22]. However, because it is not a random variable, no built-in type uncertainty is supported.

[21] The project identifier code for UnBBayes-MEBN plug-in is "unbbayes.prs.mebn". Thus, the download page may refer to this name instead of UnBBayes-MEBN.

[22] In UnBBayes-MEBN, a context node containing "IsA" property is automatically created when we create an ordinary variable in an MFrag.

- Entities can be declared as "ordered" for supporting recursion in an easy way. Thus, the recursive steps and the stop condition in a direct dependency recursion[23] are implicit if a random variable contains an "ordered" entity as its arguments.
- No taxonomy (superclass-subclass relationship) is currently supported for MEBN's entities.
- If a random variable has an indefinite quantity of possible values[24], it can be used as constraints in context nodes, but it must follow the format below:

$$< ordinary\_variable >=< random\_variable > (< argument\_list >).$$

As mentioned in Section 3.3, PR-OWL can be thought as a set of classes and properties representing MEBN elements as OWL format. Thus, PR-OWL files are also OWL files with ".owl" extension. UnBBayes-MEBN currently uses the Java open source Protégé-OWL application programming interface (API) for dwelling with input and output over OWL files. Additionally, UnBBayes and its plug-ins use Java Plug-in Framework (JPF)[25] as the plug-in infrastructure, allowing several extension points, plug-in dependency, version control, and hot-plug.

4.2 Determine the Domain and the Scope

(i) What is the domain (e.g. important terms) covered by the ontology?

(ii) For what purposes are we going to use the ontology?

(iii) For what types of questions should the ontology provide answers?

(iv) Who will use and maintain the ontology?

This is the requirements step in our model phase (see Figure 4 to refresh your knowledge on the phases and the steps). In order to narrow down the scope of your ontology, some basic questions should be answered. The answers to the above questions may change during the design process, but at any given moment they should be regarded as design guidelines limiting the scope of the model. Table 3 summarizes our scope. We also assume, from expert's knowledge, that the Vehicle Identification ontology considers some of the following conditions:

a. Tracked vehicles can travel at max speed 6 kph on very rough terrain where wheeled vehicles cannot travel.

b. Tracked vehicles are more likely to be off-road than wheeled vehicles. On rough off-road terrain, tracked vehicles usually travel at 8-12 kph, and wheeled vehicles usually travel at 5-9 kph. On smooth off-road terrain, wheeled vehicles usually travel at 10-20 kph, and tracked vehicles usually travel at 5-15 kph. On roads, wheeled vehicles usually travel at 35-115 kph, and tracked vehicles usually travel at 15-60 kph.

c. A Moving Target Indicator (MTI) sensor provides approximate velocity for vehicles that are moving, but cannot see stationary objects.

[23] We have a direct dependency recursion when an input node is parent of the resident node to which it refers to.
[24] A random variable with indefinite quantity of possible values can be created by selecting an entity as its possible value.
[25] More information about JPF can be found at <http://jpf.sourceforge.net/>.

d. An imaging sensor usually distinguishes vehicles from other objects, and usually reports correctly whether a vehicle is tracked or wheeled. Cloud cover can interfere with the ability of the imaging sensor to distinguish vehicles from other objects.

e. The Geographic Information System (GIS) usually reports accurately whether a location is on the road, off-road, or off-road on very rough terrain. However, there are occasional errors in the terrain database.

f. The speed of a object is not likely to change suddenly over time.

Domain scope	**Description**
Name	Vehicle Identification.
Goals	Given an object and several sensor reports, classify the object in Tracked Vehicle, Wheeled Vehicle or Non-Vehicle.
Problem type	Classification.
Important terms	Object (e.g. non-vehicle, vehicle - tracked, wheeled), location/region (e.g. road, off-road, rough), weather (e.g. clear, not clear), speed (e.g. stationary, slow, fast), reports (e.g. satellite image, MTI, GIS).
Sample usage	Military strategic decision (e.g. evaluation of enemy's potential, for interception).
Starting point	Empty ontology (from scratch).
Type	Probabilistic ontology (PR-OWL version 1.05).
Process goal	Introduction to PR-OWL modeling.
Target users	Probabilistic ontology learners.
Why is it relevant?	An ontology language with no support for uncertainty is not suitable for this domain, because it integrates sources of incomplete and error-prone informations, requiring a plausible reasoning with uncertainty. Traditional Bayesian models would not be appropriate either, as the speed of objects may change over time (and place), and there are numerous combinations of reports to analyze, this domain needs support for (temporal) recursion and unlimited quantity of random variables (a random variable may have an unspecified number of dependencies - parents).

Table 3: Scope description of Vehicle Identification ontology. Scope identification is usually a result of requirement analysis. Therefore, it is common that such information becomes available only after an initial evaluation.

You may consider reusing existing ontologies as well. It is always worth checking if someone else has done related work, so that we can refine or extend existing sources for a particular need. Reusing existing ontologies may also be a requirement if a system needs to interact with other controlled applications. Finally, since our goal is to model a "first" probabilistic ontology, specifically for this example case study our starting point will be an empty ontology (*i.e.* we will be starting from scratch).

4.3 Identify Entities

This section addresses the analysis and implementation steps of the modeling phase. In MEBN terminology, an *entity* is a concept in a domain; that is, a *Class* in an ontological terminology (thus, instances of an entity are individuals of a class). It is very important to spend some serious thinking to define what are the type of entities your model will have. This is an often overlooked part of the modeling process, but it will spare us a lot of time when performed correctly.

Figure 6 presents a screen capture of entity creation in UnBBayes-MEBN. For Vehicle Identification ontology, we will keep it simple, containing only four types of entities:

- **Object:** represents the objects to be categorized as vehicle (tracked or wheeled) or non-vehicle.
- **Region:** represents location of the object to be categorized.
- **Report:** represents information provided by sensors.
- **TimeStep:** this is a time-frame, representing a single moment of a domain. Modeling time as entities (thus, having a number of individuals of time steps) indicates that the model is discrete[26].

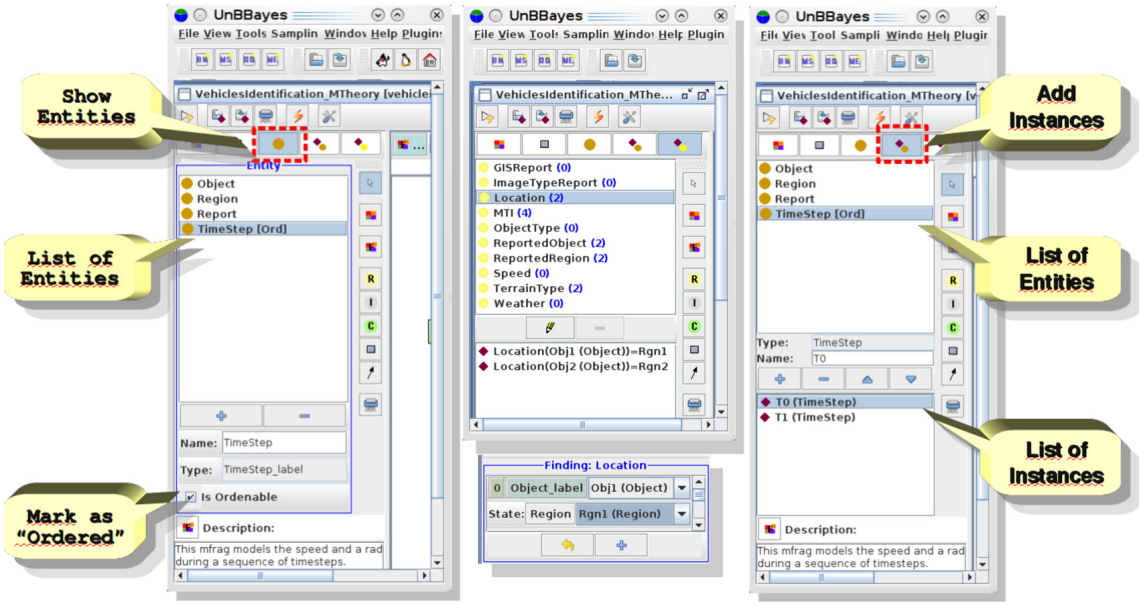

Figure 6: Entity manipulation is shown in the left window, while instances (individuals) manipulation is shown in the right window. The center window shows how to insert evidence (findings).

We choose not to add entities for known categorical values (e.g. weather or region's type, since the former is only "clear" or "cloudy"; and the latter is only "on-road", "off-road", and "rough"), because it adds undesired complexity to our model. Usually, there is no need to create entities for a known fixed set of values, since they can be represented as properties (random variables and their possible values).

4.4 Define Behaviors and Properties - MFrags

Once the entities are defined, we must describe their probabilistic properties and relationships through a collection of inter-dependent random variables (input nodes, resident nodes, and context nodes) and ordinary variables (for which individuals in the given situation will be substituted); all within the MFrag-based structure of a MEBN model. From a general point of view, MFrag's components in a probabilistic ontology can encode several ideas or patterns of a domain, as summarized in Table 4.

[26] A discrete time-step model represents events in subsequent steps, instead of using a continuous scale. This is a common approach for models based on Bayesian Networks, although they are capable of representing continuous values as well. Because time steps are sequential, they must follow a linear order, which is a built-in property in UnBBayes-MEBN (see Figure 6).

Ideas/patterns	Description
Entity's properties	e.g. "type" of *Object* (tracked vehicle, wheeled vehicle, non-vehicle), "weather" of *Region* (clear, cloudy).
Physical/abstract parts	If an entity is thought as "structured", then we can represent its parts (e.g. GIS data, image data, and MTI data for *Report*).
References to entities	This is usually represented as a random variable having an entity *A* as argument and entity *B* as its possible value (*e.g. ReportedRegion*(*rpt*), whose *rpt* is an ordinary variable having *Report* as its type, and *ReportedRegion*'s possible values are individuals of *Region*). N-ary relationships can be represented similarly by using multiple arguments. This type of relationship can also be simulated by a boolean random variable with two or more arguments, so that its value is true when the entities in its arguments are related. Thus, the possible values will be predefined, and the probabilistic distribution will be finely configurable by LPDs.
Dependencies	These are represented as arrows, pointing to resident nodes, and described by local probability distribution, which determines the probability of a random variable given its dependencies (see Section 4.5).
Restrictions	Context nodes can represent restrictions in the domain.

Table 4: These are the main ideas to be represented by MFrags. References to a single individual are represented as *findings*; thus, they are not exactly MFrag's components.

Considering the mappings presented in Table 4, we can identify the following MFrags and variables in the Vehicle Identification ontology.[27] . See Figure 7 to visualize the MFrags, including dependencies and contexts.

- ***Reference_MFrag***: it gathers random variables representing references between different entities. The resident nodes of this MFrag have an unspecified number of possible values, because these values have been specified as an existing entity. Therefore, these resident nodes will be mostly used as components of a context node.
  - Location(Object) = Region : associates Region to Object.
  - ReportedRegion(Report) = Region : associates Region to Report.
  - ReportedObject(Report) = Object : associates Object to Report.
  - isA(obj, Object_label[28]) : this is a context node indicating that the ordinary variable "obj" is of type "Object". Because they represent identical concepts, explanations about the other "isA" nodes will be omitted from now on.
- ***Speed_MFrag***: it represents the MTI and speed of objects.
  - Speed(Object,TimeStep) = {Stationary, Slow, Medium, Fast, VeryFast}[29] : represents speed of an object in a given moment. As you may have noticed, this is more finely-graded than the

[27] Resident node's signatures follow the format: <Name> (<Argument_Types>) = <Possible_ Values>

[28] The system has automatically added the "_label" suffix because of an aspect of implementation: a "type" of an ordinary variable should not share the same internal name with an existing entity.

[29] Let's assume *Slow* is less than 15 km/h, *Medium* is around 16-59 km/h, and *Fast* is above 60 km/h. We may consider "*VeryFast*" as fast as 100 km/h.

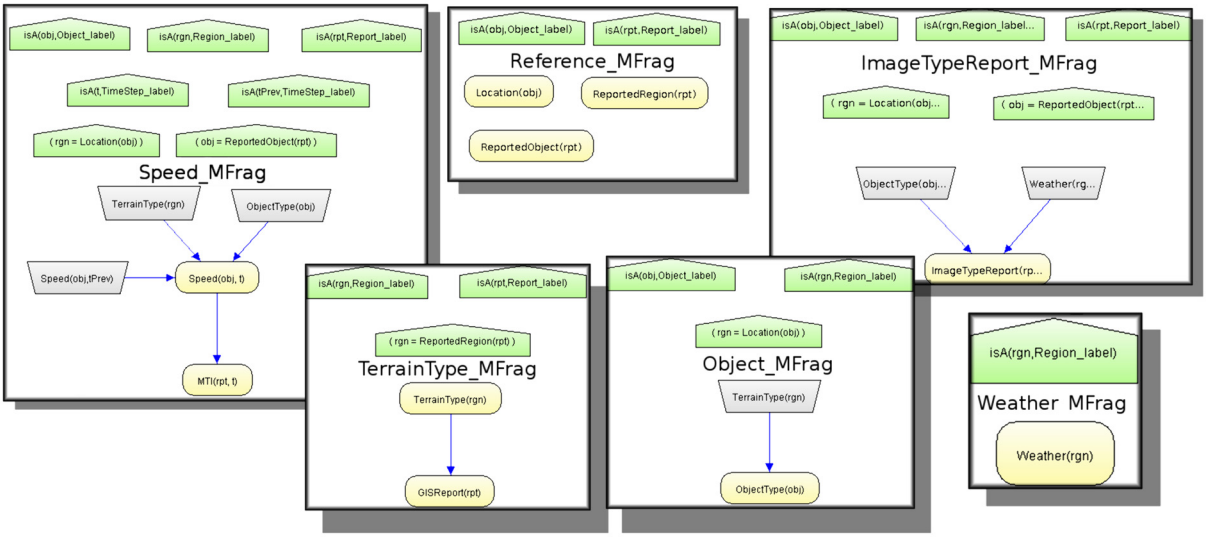

Figure 7: This is the MTheory (consistent set of MFrags) for Vehicle Identification Ontology. Now, we just need to set local probability distribution (see Section 4.5) for each resident nodes in order to complete our modeling phase.

ones presented in Table 3. This is in order to represent eventual oscillations of speed over time, although it is not a strictly vital aspect of this domain.

- MTI(Report,TimeStep) = {Stationary, Slow, Medium, Fast, VeryFast} : represents the speed tracked by MTI.
- rgn = Location(obj) : this is a context node stating that "rgn" is the location where "obj" is. Explanations about identical context nodes will be omitted from now on.
- obj = ReportedObject(rpt) : this is a context node stating that "obj" is the subject of report "rpt". Explanations about identical context nodes will be omitted from now on.

- ***ImageTypeReport_MFrag***: it represents reports from images (e.g. satellite images).
  - ImageTypeReport(Report) = {Tracked, Wheeled, NonVehicle} : represents the type of an object inferred from an image.
- ***Object_MFrag***: it represents object's properties.
  - ObjectType(Object) = {Tracked, Wheeled, NonVehicle} : represents the type of the analyzed object. This is likely our query node (goal).
- ***TerrainType_MFrag***: it represents the terrain property of regions and GIS information.
  - TerrainType(Region) = {Road, OffRoad, VeryRough} : represents the actual terrain type of a region.
  - GISReport(Report) = {Road, OffRoad, VeryRough} : represents the terrain type obtained from a GIS report.

  - rgn = ReportedRegion(rpt) : this is a context node stating that “rgn” is the region reported by the report “rpt”.

- ***Weather_MFrag***: it represents the weather property of a region.

  - Weather(Region) = {Clear, Cloudy} : represents the weather condition (obtained from weather reports).

Note that *Speed_MFrag* contains a temporal recursion[30], as the distribution of *Speed(obj,t)* depends on *Speed(obj,tPrev)*. Thankfully, MEBN/PR-OWL offers enough expressiveness to represent temporal recursion. Inquisitive readers may ask how could this recursion ever stop, since no context node indicating that *tPrev* precedes *t* is provided. Because *t* and *tPrev* are both ordinary variables of type *TimeStep*, which contains a finite number of individuals related by a linear order, recursive steps and stop conditions can be handled implicitly.

4.5 Local Probability Distribution in a Dynamic Fashion

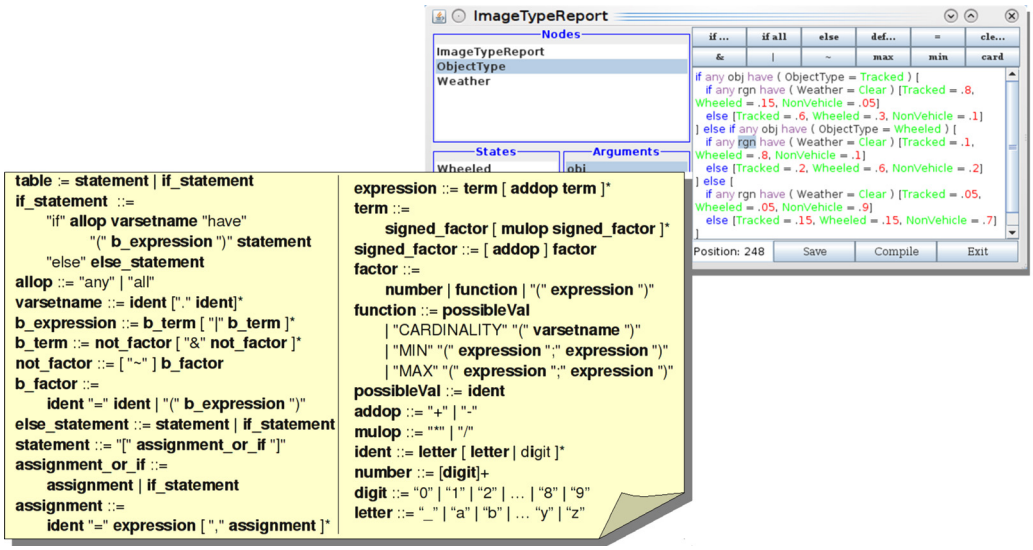

Figure 8: Dynamic LPD follows this grammar. The window at the top right corner shows how LPD script is edited in UnBBayes-MEBN.

UnBBayes-MEBN provides a flexible way for declaring the local probability distribution (LPD) of resident nodes. This is done by using a special-purpose script, whose grammar is described in Figure 8. The *varsetname* is a dot-separated list of ordinary variables, and it refers to parent nodes containing all of those ordinary variables as arguments. *MIN*, *MAX* and *CARDINALITY* are respectively a minimal function, maximum function, and a function to count the number of parent's combinations, having *varsetname* as its arguments and satisfying *b_expression*. If no script is provided, UnBBayes-MEBN assumes uniform distribution (all values are equally probable). Resident nodes in *Reference_MFrag* do not contain LPD script; thus, they use an uniform distribution. See Figure 9 for Vehicle Identification ontology's LPD scripts.

[30] We call it temporal recursion because iteration is proceeded on a time step variable *t*.

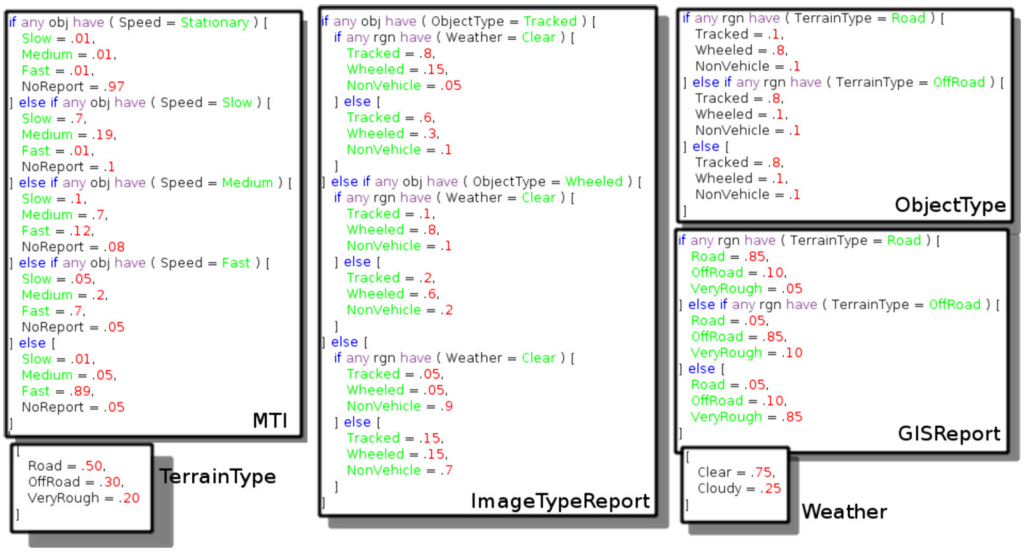

Figure 9: These are the LPDs for the resident nodes in Vehicle Identification Ontology. Because the LPD for *Speed* is too large, it is not addressed here for a better visibility.

4.6 Creating Individuals - Filling Up Your Knowledge Base

This Section addresses the "fill" phase of our modeling procedure. After defining a probabilistic ontology, we can populate the knowledge base by adding instances (individuals of entities) and findings (evidences, known facts). Let's start adding two individuals[31] for each entity, and some findings to our knowledge base. Figure 6 in Section 4.3 also has shown how the finding's insertion would look like in UnBBayes-MEBN. See below for the inserted individuals and findings. You should be able to understand the meanings of these findings by looking at the variable's definitions at Section 4.4.

```
Object: Obj1, Obj2.
Region: Rgn1, Rgn2.
Report: Rpt1, Rpt2.
TimeStep: T0, T1.
Location(Obj1 (Object))=Rgn1; Location(Obj2 (Object))=Rgn2;
MTI(Rpt2 (Report),T1 (TimeStep))=Fast; MTI(Rpt2 (Report),T0 (TimeStep))=Medium;
MTI(Rpt1 (Report),T0 (TimeStep))=Slow; MTI(Rpt1 (Report),T1 (TimeStep))=Slow;
ReportedObject(Rpt2 (Report))=Obj2; ReportedObject(Rpt1 (Report))=Obj1;
ReportedRegion(Rpt2 (Report))=Rgn2; ReportedRegion(Rpt1 (Report))=Rgn1;
TerrainType(Rgn2 (Region))=Road; TerrainType(Rgn1 (Region))=VeryRough.
```

4.7 Reasoning - SSBN Generation

This section addresses the reasoning phase of our procedure, which helps with evaluating and debugging our probabilistic ontology. The reasoning process in PR-OWL ontology is basically an automatic generation of SSBN, in order to determine the values (probabilities) of a set of query nodes, so that users or semantic web applications can trigger actions based on the observed values. Details on SSBN generation algorithm

[31] Figure 6 in Section 4.3 has shown how to add individuals in UnBBayes-MEBN.

is beyond our scope, and it will not described in this Chapter. Interested readers will find a detailed account in (Laskey, 2007), while would also want to download UnBBayes-MEBN source code to analyze the algorithm's implementation.

As mentioned in Table 3 (Section 4.2), Vehicle Identification ontology aims for classifying an object as *Tracked Vehicle*, *Wheeled Vehicle* or *Non-Vehicle*. This is done by querying the *ObjectType* node. Figure 10 displays the results, indicating that *Obj1* is likely a tracked vehicle, while *Obj2* is likely a wheeled vehicle. SSBN usually does not contain nodes that do not influence a query node. Since no evidences about image sensors or weather reports were found in the knowledge base, they were not influencing the query nodes; thus, they were ignored by the SSBN construction algorithm.

Other types of questions can be answered by querying the corresponding nodes. Inquisitive readers may want to replicate the BN of figure 1. The referred BN can be obtained by removing all the evidences of MTI of *Rpt2* and then querying *ImageTypeReport(Rpt2)*.

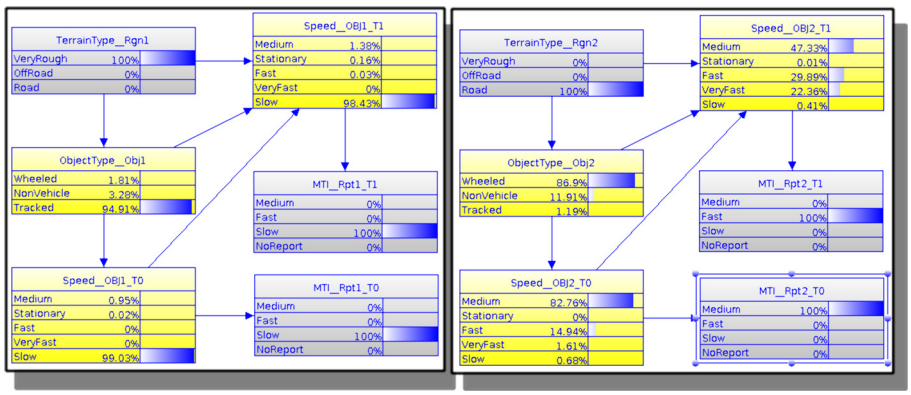

Figure 10: Resulting SSBNs for a query in *ObjectType*, using *Obj1* (left SSBN) and *Obj2* (right SSBN) as arguments. Once generated, the SSBN can be treated as a traditional Bayesian network (including the mechanisms for the propagation of evidences).

4.8 Extending Your Model

As emphasized throughout this chapter, ontology modeling is an iterative process so you may want to extend your model in the future. In order to simulate such situation, we will now add the following observation to our model: *(a) A image-based system usually detects wheels on trucks or other wheeled vehicles, but not on tracked vehicles like tanks. (b) When viewed from the side, tanks usually show a protrusion near the top of the vehicle. When viewed from above, they rarely show a protrusion. (c) The image processing software usually distinguishes correctly between small and large vehicles. (d) Weather can change over time, (e) and so can the location of a vehicle*. Figure 11 shows the major changes in the model, and Figure 12 shows the LPD scripts for the new nodes.

Actually, the only unchanged MFrag in Figure 11 was *TerrainType_MFrag*. The observations (a), (b) and (c) were addressed to *ImageTypeReport_MFrag*. Observation (d) was addressed to *Weather_MFrag*, also affecting *ImageTypeReport_MFrag*. Observation (e) was addressed to *Reference_MFrag*, affecting every single MFrags using *Location* node.

In order to reason with the new model, the findings in its knowledge base must also reflect all the changes. Let's now add the following findings. Note that the individuals remain the same as presented in Section 4.6. Figure 13 shows the resulting SSBN.

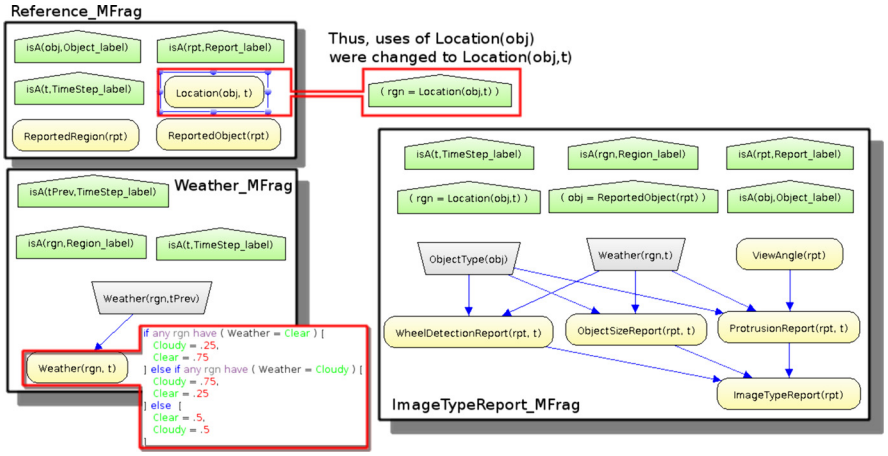

Figure 11: These are the changes on our model.

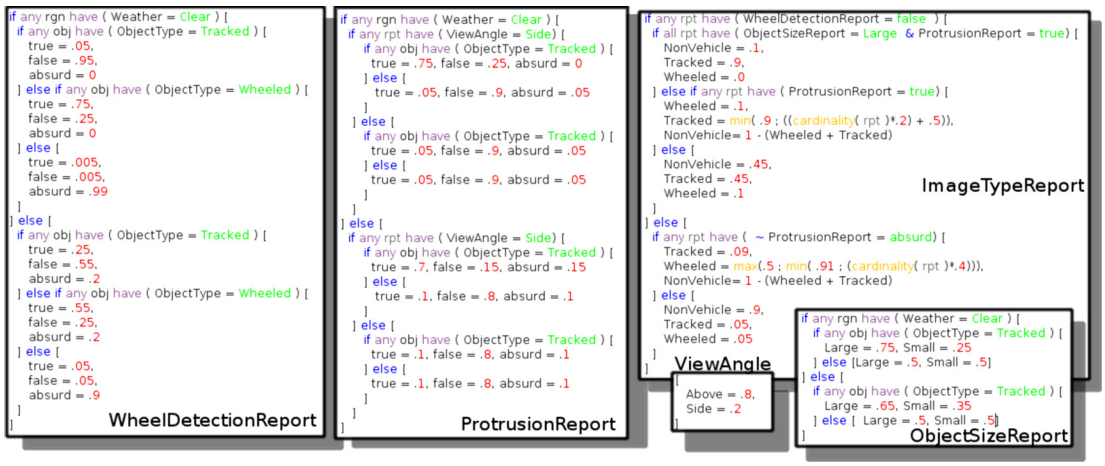

Figure 12: LPD for new nodes. *ImageTypeReport* is now an example of MAX, MIN and CARDINALITY usage.

```
ImageTypeReport(Rpt1 (Report))=Tracked.
Location(Obj1 (Object),T1 (TimeStep))=Rgn1;
Location(Obj1 (Object),T0 (TimeStep))=Rgn2.
ObjectSizeReport(Rpt1 (Report),T1 (TimeStep))=Large.
ProtrusionReport(Rpt1 (Report),T1 (TimeStep))=true.
ReportedObject(Rpt1 (Report))=Obj1.
ReportedRegion(Rpt1 (Report))=Rgn1.
ViewAngle(Rpt1 (Report))=Side.
Weather(Rgn1 (Region),T1 (TimeStep))=Clear.
WheelDetectionReport(Rpt1 (Report),T1 (TimeStep))=false.
```

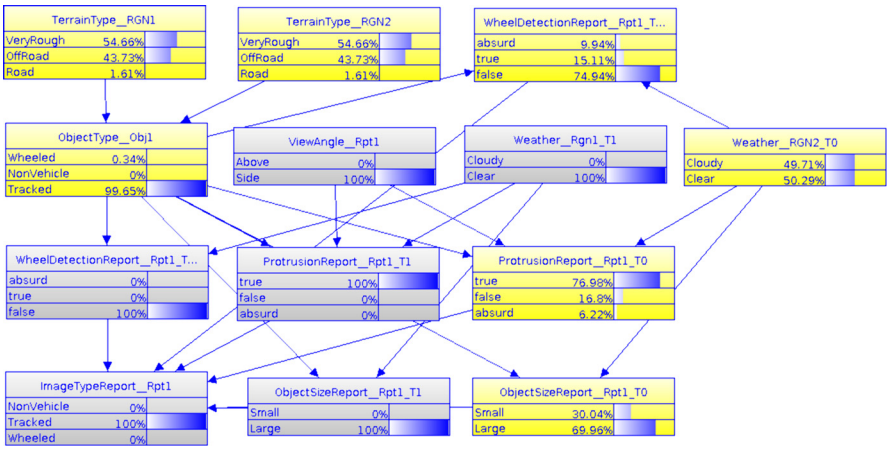

Figure 13: A query on *ObjectType*(*Obj1*) in the new ontology results in this SSBN. This SSBN is larger because we added different findings to the knowledge base, affecting the D-Separation (Geiger et al., 1990) of some nodes. *Obj1* is likely a tracked vehicle.

5 Conclusion

Ontologies provide the "semantic glue" to enable knowledge sharing and to support interoperability among people and computer systems. Traditional ontologies do not provide adequate support for uncertainty, a fundamental characteristic of a complex open-world environment. As part of an overall effort to address such important issue, this chapter presented a framework and some general procedures for modeling uncertain domains using MEBN/PR-OWL. By modeling an introductory ontology for vehicle classification problem using UnBBayes-MEBN, a Java-based probabilistic ontology editor and inference engine, we hope to have helped our readers in forming the initial foundations for a solid basis on probabilistic ontology concepts. This would allow them to produce models that coherently incorporate multiple sources of noisy, incomplete, unreliable information in Semantic Web applications, as well as being capable to perform plausible reasoning.

References

Bechhofer, S., Horrocks, I., & Patel-Schneider, P. F. (2003). Tutorial on OWL. 2nd International Semantic Web Conference (ISWC2003). http://www.cs.man.ac.uk/ horrocks/ISWC2003/Tutorial/.

Booker, L. B. & Hota, N. (1986). Probabilistic reasoning about ship images. In *Second Annual Conference on Uncertainty in Artificial Intelligence* Philadelphia, PA.

Brank, J., Grobelnik, M., & Mladenič, D. (2005). A survey of ontology evaluation techniques. In *In In Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005*.

Buchanan, B. G. & Shortliffe, E. H. (1984). Rule-based expert systems: The MYCIN. In *Experiments of the Stanford Heuristic Programming Project* MA, USA: Addison-Wesley.

Carvalho, R., Santos, L., Matsumoto, S., Ladeira, M., & Costa, P. (2008). *A GUI Tool for Plausible Reasoning in the Semantic Web Using MEBN*, (pp. 17–45). Springer-Verlag Berlin Heidelberg New York, USA.

Carvalho, R., Santos, L., Matsumoto, S., Ladeira, M., Costa, P., & Laskey, K. (2010a). *UnBBayes: Modeling Uncertainty for Plausible Reasoning in the Semantic Web*, (pp. 1–28). IntechWeb, Croatia.

Carvalho, R. N., Laskey, K. B., & Costa, P. C. G. (2010b). 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.

Carvalho, R. N., Laskey, K. B., & Costa, P. C. G. (2010c). PR-OWL 2.0 - Bridging the gap to OWL semantics. In *Proceedings of the 6th Uncertainty Reasoning for the Semantic Web (URSW 2010) on the 9th International Semantic Web Conference (ISWC 2010)* Shanghai, China.

Charniak, E. & Goldman, R. P. (1989). A semantics for probabilistic quantifier-free first-order languages with particular application to story understanding. In *Eleventh International Joint Conference on Artificial Intelligence* Detroit, Michigan, USA.

Costa, P. (2005). *Bayesian Semantics for the Semantic Web*. PhD thesis, Department of Systems Engineering and Operational Research, George Mason University.

Costa, P. & Laskey, K. (2005). Multi-Entity Bayesian Networks without Multi-Tears. http://hdl.handle.net/1920/456. Draft, Department of Systems Engineering and Operations Research, George Mason University, Fairfax, VA, USA.

Costa, P. & Laskey, K. (2006). PR-OWL: A Framework for Probabilistic Ontologies. In *Proceedings of the Fourth International Conference on Formal Ontology in Information Systems* Baltimore, USA.

Dempster, A. P. (1967). Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*, 38, 325–339.

Ding, Z. (2005). *BayesOWL: A Probabilistic Framework for Semantic Web*. PhD thesis, Computer Science and Electrical Engineering, University of Maryland.

Geiger, D., Verma, T., & Pearl, J. (1990). d-separation: From theorems to algorithms. In *UAI '89: Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence* (pp. 139–148). Amsterdam, The Netherlands, The Netherlands: North-Holland Publishing Co.

Getoor, L., Friedman, N., Koller, D., & Pfeffer, A. (2001). Learning probabilistic relational models. In S. Dzeroski & N. Lavrac (Eds.), *Relational Data Mining* New York, USA: Springer-Verlag.

Giurno, R. & Lukasiewicz, T. (2002). P-$\mathcal{SHOQ}$(D): A Probabilistic Extension of $\mathcal{SHOQ}$(D) for Probabilistic Ontologies in the Semantic Web. In *European Conference on Logics in Artificial Intelligence (JELIA 2002)* Cosenza, Italy.

Gomez-Perez, A. (1994). Some ideas and examples to evaluate ontologies.

Heflin, J. (2004). OWL Web Ontology Language - Use Cases and Requirements (W3C Recommendation). www.w3.org/TR/2004/REC-webont-req-20040210.

Klinov, P. & Parsia, B. (2008). Probabilistic modeling and owl: A user oriented introduction to P-$\mathcal{SHIQ}$(D).

Koller, D. & Pfeffer, A. (1997). Object-Oriented Bayesian Networks. In *Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)* San Francisco, CA, USA.

Laskey, J., Laskey, B., Costa, P., Kokar, M. M., Martin, T., & Lukasiewicz, T. (2008). Uncertainty Reasoning for the World Wide Web, W3C Incubator Group Report. http://www.w3.org/2005/Incubator/urw3/XGR-urw3-20080331/.

Laskey, K. (2007). MEBN: A Language for First-Order Bayesian Knowledge Bases. *Artificial Intelligence*.

Laskey, K., Costa, P., Wright, E., & Laskey, K. (2007a). Probabilistic Ontology for Net-Centric Fusion. In *Proceedings of the Tenth International Conference on Information Fusion* Quebec, Canada.

Laskey, K., Laskey, K., & Costa, P. (2007b). Uncertainty Reasoning for the World Wide Web Incubator Group Charter (W3C Incubator Activity). www.w3.org/2005/Incubator/urw3/charter.

Murphy, K. P. (2002). Dynamic bayesian networks: Representation, inference and learning.

Nikolov, A., Uren, V., Motta, E., & de Roeck, A. (2007). Using the dempster-shafer theory of evidence to resolve abox inconsistencies. In *3rd Workshop on Uncertainty Represention for the Semantic Web* Busan, Korea.

Patel-Schneider, P., Hayes, P., & Horrocks, I. (2004). OWL Web Ontology Language - Semantics and Abstract Syntax (W3C Recommendation). http://www.w3.org/TR/owl-semantics/.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA, USA: Morgan Kaufmann Publishers.

Richardson, M. & Domingos, P. (2004). *Markov Logic Networks*. Technical report, University of Washington, Seattle, WA.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. (1991). *Object-oriented modeling and design*. Englewood Cliffs, New Jersey: Prentice Hall.

Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton, NJ, USA: University Press.

Shafer, G. (1986). The construction of probability arguments. *Boston University Law Review 66 799-823*.

Spiegelhalter, D. J., Franklin, R., & Bull, K. (1989). Assessment, criticism, and improvement of imprecise probabilities for a medical expert system. In *Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 257–285). Mountain View, CA.

Stoilos, G., Stamou, G., Z., P. J., Tzouvaras, V., & Horrocks, I. (2007). Reasoning with very expressive fuzzy description logics. *Journal of Artificial Intelligence Research*, 30(8), 273–320.

Stoilos, G., Stamou, G. B., Tzouvaras, V., Pan, J. Z., & Horrocks, I. (2005). The fuzzy description logic f-$\mathcal{SHIN}$. *First Workshop on Uncertainty Reasoning for the Semantic Web (URSW-2005)*, (pp. 67–76).

Yaghlane, B. B. & Laamari, N. (2007). OWL-CM : OWL combining matcher based on belief functions theory. In *2nd International Workshop on Ontology Matching (OM-2007)* Busan, Korea.