ISWC2005

International Semantic Web Conference
November 6-10 2005
Galway, Ireland

http://iswc2005.semanticweb.org

W3: Uncertainty Reasoning for the Semantic Web

Organisers:

Paulo C. G. Costa,

Kathryn B. Laskey,

Kenneth J. Laskey,

Michael Pool

Monday, 7 November 2005

ISWC 2005 could not take place without the generous support of the following sponsors

DERI

sfi science foundation ireland
fondúireacht eolaíochta éireann

Super Emerald Sponsors

ENTERPRISE IRELAND

IDA IRELAND

Gold Sponsors

dip

Ontotext
Knowledge and Language Engineering Lab of Sirma

knowledgeweb

sekt

Siderean™ SOFTWARE
navigation for the digital universe

storm technology

Silver Sponsors

AGENTLINK

ADVANCED KNOWLEDGE AKT TECHNOLOGIES

france telecom

iSOCO
intelligent software for the networked economy

niwa
WEB Solutions

NOKIA
CONNECTING PEOPLE

ontoprise
know how to use Know-How

JOURNAL OF Web Semantics
SCIENCE, SERVICES & AGENTS ON THE WORLD WIDE WEB

TopQuadrant

ISWC 2005 Organising Committee

General Chair	Mark Musen, Stanford University
Research Track Co-Chair	Yolanda Gil, Information Sciences Institute
Research Track Co-Chair	Enrico Motta, The Open University
Industrial Track Chair	V Richard Benjamins, iSOCO, S.A.
Workshop Chair	Natasha F Noy, Stanford University
Tutorial Chair	R.V. Guha, Google
Poster & Demo Chair	Riichiro Mizoguchi, Osaka University
Semantic Web Challenge	Michel Klein, Vrjie Universiteit Amerdam
Semantic Web Challenge	Ubbo Visser, Universitat Bremen
Doctoral Symposium Co-Chair	Edward Curry, National University of Ireland, Galway
Doctoral Symposium Co-Chair	Enda Ridge, University of York
Meta-Data Chair	Eric Miller, W3C
Sponsorship Chair	Liam O'Móráin, DERI Galway
Local Organising Co-Chair	Christoph Bussler, DERI Galway
Local Organising Co-Chair	Stefan Decker, DERI Galway
Local Organiser	Brian Cummins, DERI Galway
Webmaster	Seaghan Moriarty, DERI Galway
Web Design	Johannes Breitfuss, DERI Innsbruck

URSW 2005 Presentations

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 Relation (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 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 finding 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[14](BNs). There are existing standards for representing BNs 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, adequate to enable reasoning systems because BNs embody a "flat" representation language where 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 others 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 Relation Models (PRMs) that can be implemented with IET's reasoning tools. A PRM is based on a relational schema which consists 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 vehicle, there 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 $R \in R(Xi)$. 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 schema and PRM are 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 can be referenced with a slot chain, SC, as follows: X_i.SC.A where Xi 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 it's useful to reference the sales tax rate of state in which car owners reside in a model about car sales. This would be done as follows: '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 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 representation 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] 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 reading) 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 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 their domains and could be used to relate probability distributions and the likes 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 and 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 not only must we represent the class and the slot but we also represent, explicitly, the relationship between them, i.e., 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 for the bins, the lower and upper bounds, and whether each endpoint is open or closed with other metaproperties have also been created. 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 need 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 implementations 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 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.

A Fuzzy Semantics for Semantic Web Languages

Mauro Mazzieri[1] and Aldo Franco Dragoni[2]

[1] mauro.mazzieri@gmail.com
[2] Università Politecnica delle Marche
Dipartimento di Elettronica, Intelligenza Artificiale e Telecomunicazioni (DEIT)
a.f.dragoni@univpm.it

Abstract. Although the model-theoretic semantics of the languages used in the Semantic Web are crisps, the need arise to extend them to represent fuzzy data, in the same way fuzzy logic extend first-order-logic. We will define a fuzzy counterpart of the RDF Model Theory for RDF (section 2) and RDF Schema (section 3). Last, we show how to implement the extended semantics in inference rules (section 4).

Keywords: Fuzzy Logic, Knowledge Representation, Semantic Web, RDF, RDF Schema.

1 Knowledge representation on the web

The Semantic Web is an extension of the current web in which information is given well-defined meaning[1] by the use of knowledge representation (KR) languages.

The KR languages used (RDF, RDF Schema and OWL) have the characteristics that make them useful on the web[2]:

- the elements of the domain are represented by URI;
- there is no global coherence requirements, as local sources can make assertions independently without affecting each other's expressiveness.

The languages have the ability to describe, albeit not formally, much more than their semantics can express. Their model theory captures only a formal notion of meaning, captured by inference rules; the exact 'meaning' of a statement can depend on many factors, not all accessible to machine processing[3]. This feature can be useful to represent information from fields that require knowledge representation paradigms other than the FOL-like RDF Model Theory or the expressive Description Logic used by OWL. Amongst those paradigms there is fuzzy logic, to represent vague or ambiguous knowledge.

2 Fuzzy RDF

RDF has its own model-theoretic semantics, similar to that of first-order logic. To represent fuzzy data, we will define a syntactic and semantic extension of RDF, similar to the extension from first-order logic to fuzzy logic.

Even if fuzzy data can be simply seen as a juxtaposition of a triple and a number, the model-theoretic approach has well-known theoretical advantages.

We will try to be as plain as possible. Starting from RDF Syntax and RDF Model Theory, we will make as few changes as possible. In the rest of the paper, for the sake of brevity only the changes from RDF Semantics[3] are shown.

2.1 Syntax

The RDF syntax must be extended to add to the triple ⟨subject, predicate, object⟩ a value. Such a value can be taken as a real number in the interval [0, 1], but every bounded real interval will do.

This is not an extension from a 3-elements tuple to a 4-elements tuple as it may seem at a first glance. The added element has a syntactic nature different from the others: it is not an element of the domain of the discourse, but a property related to the formalism used by the language to represent uncertainty and vagueness.

The simple concrete syntax we define is as an extension of the EBNF of N-Triples as given in [4]. Our extension is given in table 1.

N-Triples is a line-based, plain text format for encoding an RDF Graph, used for expressing RDF test cases. A statement has the form `s p o.`, where `s`, `p` and `o` are respectively the subject, the predicate and the object of the statement. Our extended syntax add an optional prefix `n:` to a statement in N-triple notation, where n is a decimal number representing the fuzzy truth-value of the triple. The use of decimal numbers instead of real numbers is only a limitation of the syntax and does not undermine the discussion.

The term *triple*, used in the EBNF for N-Triple, is replaced with the more generic term *statement*. Triple and statement are often used in semantic web literature as a synonym, but we prefer to use the latter to avoid confusion between a plain RDF statement (made actually of three parts) and a fuzzy RDF statement (that, although is still a triple semantically, is made up of four elements).

The fuzzy *value* is defined as optional. This way, the syntax is backward-compatible; the intended semantics is that a statement with the form `s p o.` is equivalent to the statement `1: s p o.`. With such a (syntactic only) default, we could take an inference engine implementing fuzzy RDF, let it parse plain RDF statements, and get the same results of a conventional RDF inference engine. Furthermore, as it would be clear in the description of fuzzy RDF inference rules (section 4), even the complexity of the computation would be of the same order.

We will not give an abstract syntax, nor a RDF/XML based syntax, as they would not be useful. It can be shown that all “physical“ data (i.e., data transmitted between host or processes) can be encoded using plain RDF reified statements. The extended syntax will be used only in the paper to write down the examples.

```
fuzzyNtripleDoc ::= line*
line            ::= ws* ( comment | statement )? eoln
comment         ::= ‘#’ ( character − ( cr | lf ) )*
statement       ::= (value ws+)? subject ws+ predicate ws+ object ws* ‘.’ ws*
value           ::= 1 | 0.[0–9]+
subject         ::= uriref | nodeID
predicate       ::= uriref
object          ::= uriref | nodeID | literal
uriref          ::= ‘<’ absoluteURI ‘>’
nodeID          ::= ‘_:’ name
literal         ::= langString | datatypeString
langString      ::= ‘"’ string ‘"’ ( ‘@’ language )?
datatypeString  ::= ‘"’ string ‘"’ ‘^^’ uriref
language        ::= [a–z]+ (‘-’ [a–z0–9]+ )*
                    encoding a language tag.
ws              ::= space | tab
eoln            ::= cr | lf | cr lf
space           ::= #x20 /* US-ASCII space - decimal 32 */
cr              ::= #xD /* US-ASCII carriage return - decimal 13 */
lf              ::= #xA /* US-ASCII line feed - decimal 10 */
tab             ::= #x9 /* US-ASCII horizontal tab - decimal 9 */
string          ::= character* (with escapes as defined in section Strings of [4])
name            ::= [A-Za-z][A–Za–z0–9]*
absoluteURI     ::= character+ (with escapes as defined in section URI References
                    of [4])
character       ::= [#x20–#x7E] /* US-ASCII space to decimal 126 */
```

Table 1. EBNF for Fuzzy N-Triples

2.2 Simple interpretation

The RDF Model Theory[3] is based on the concept of *extension.* An interpretation satisfies a triple `s p o.` if the couple formed by the interpretation of the subject and the interpretation of the object belongs to the extension of the interpretation of the property.

In this fuzzy counterpart, a couple ⟨subject, object⟩ has a membership degree to the extension of the predicate, given by the number prepended to the statement. The extension is not an ordinary set of couples anymore, but a fuzzy set of couples. In other words, a fuzzy RDF interpretation satisfies a statement `n: s p o.` if the membership degree of the couple, formed by the interpretation of the subject and the interpretation of the object, to the extension of the interpretation of the predicate, is greater or equal than n.

We have chosen not to make the mapping between vocabulary items and domain fuzzy. Instead, the membership of a resource to the domain is fuzzy. This is a step which poses some theoretical problems, in particular when we have to deal with properties in simple interpretations. In RDF interpretation, the property domain IP is a subset of the resource domain IR, so in fuzzy RDF interpretations would be enough to make IP a fuzzy subset of IR; in simple interpretations, instead, there is no formal relation between IP and IR, so when the mapping IS from URI references to $(IR \cup IP)$ becomes fuzzy we need a further device. The chosen solution is to define a domain IDP for properties, so that IP is a fuzzy subset of IDP, and to modify the definition of IS to a mapping URI references $\in V \rightarrow (IR \cup IDP)$. RDF interpretations does not need IDP, as IP can be shown to be a fuzzy subset of IR.

Definition of a simple interpretation A *simple fuzzy interpretation* I of a vocabulary V is defined by:

1. A non empty set IR of resources, called the domain or universe of I
2. A non empty set IDP, called the property domain of I
3. A fuzzy subset IP of IDP, called the set of properties of I
4. A fuzzy mapping $IEXT : IP \rightarrow 2^{IR \times IR}$, i.e. the fuzzy set of pairs $\langle x, y \rangle$ with $x, y \in IR$.
5. A mapping IS from URI references $\in V \rightarrow (IR \cup IDP)$
6. A mapping IL from typed literals $\in V \rightarrow IR$
7. A distinguished subset $LV \subseteq IR$, called the set of literal values, which contains all the plain literals of V

The belonging of an element to the properties domain is strictly related to the use of such element as a property in a statement. Therefore, we have defined a membership degree to the property domain, intuitively related to the truth value of the statements in which the resource is used as a property.

2.3 Denotations for ground graphs

The next step is to define the semantic conditions an interpretation must satisfy in order to be a model for a graph. We state the semantic conditions that relate the membership degree of a couple ⟨subject, object⟩ to an extension and the truth of a fuzzy statement.

We will use the abbreviated Zadeh's notation $A(x) = n$, instead of $\mu_A(x) = n$, to state that the membership degree of the element x to the set A is equal to n [5].

Semantic conditions for ground graphs

– if E is a plain literal $\texttt{aaa} \in V$, then $I(E) = \texttt{aaa}$
– if E is a plain literal $\texttt{aaa@ttt} \in V$, then $I(E) = \langle \texttt{aaa}, \texttt{ttt} \rangle$
– if E is a typed literal $\in V$, then $I(E) = IL(E)$
– if E is a URI reference $\in V$, then $I(E) = IS(E)$
– if E is a ground triple `n: s p o.`, then $I(E) = \texttt{true}$ if $\texttt{s}$, $\texttt{p}$ and $\texttt{o} \in V$, $IP(I(\texttt{p})) \geq n$ and $IEXT(I(\texttt{p}))(\langle I(\texttt{s}), I(\texttt{o}) \rangle) \geq n$, otherwise $I(E) = \mathsf{false}$.
– if E is a ground RDF graph, than $I(E) = \texttt{false}$ if $I(E') = \texttt{false}$ for some triple $E' \in E$, otherwise $I(E) = \texttt{true}$

Only the condition of truth and falsity of a ground statement in the interpretation is affected. The given formulation of the condition has as a consequence that a graph where the same statement appears more than once, with different membership degrees, is equivalent to a graph where the statement appears only once, with a membership degree equal to the maximum of the membership degrees.

Note that whether a statement is a model for a graph or not is not a fuzzy concept; it is either true or false. However, it could be interesting to compute the minimum and maximum membership degree to an extensions a couple must have in an interpretation to be a model of a given graph. This minimum degree has a role similar to the degree of truth of a statement in a knowledge base.

2.4 Simple entailment

The definition of simple interpretation is not affected. A set S of RDF graphs *(simply) entails* a graph E if every interpretation which satisfies every member of S also satisfies E.

Given a graph G and a triple ⟨s, p, o⟩, it could be interesting to compute the minimum and maximum value of n such that G entails `n: s p o.`. Those bounds must be taken in account when we compute the deductive closure of the graph, as it is not unique.

Section 2 of RDF Semantics [3] shows many lemmas that apply to simple interpretations. All of them retain their validity within fuzzy RDF Model Theory, making some adjustments in the proof of some of them. We will show these.

The *Empty Graph Lemma* can be shown using the same proof. The definition of an empty graph is the same as in plain RDF: an *empty graph* is a graph with no statements at all. It is important to note that an empty graph can not be defined as a graph with no not-zero-valued statements. Statements such as `0: s p o.`, although pretty useless, cannot be ignored, as the semantic requirement that `s`, `p` and `o` must belong to the graph's vocabulary still applies.

Subgraph Lemma, *Instance Lemma* and *Merging Lemma* retain both their validity and their proofs with the new semantics.

Interpolation Lemma, *Anonymity Lemma*, *Monotonicity Lemma* and *Compactness Lemma* make use in their proof of a way of constructing an interpretation of a graph using lexical items in the graph itself, the so called *Herbrand interpretation* [6]. To prove the lemmas, we need to construct a similar interpretation for a fuzzy graph.

The *(simple) Herbrand fuzzy interpretation* of G, written $\mathrm{Herb}(G)$, can be defined as follows.

- $LV_{\mathrm{Herb}}(G)$ is the set of all plain literals in G;
- $IR_{\mathrm{Herb}}(G)$ is the set of all names and blank nodes which occur in subject or object position of statements in G;
- $IDP_{\mathrm{Herb}}(G)$ is the set of URI references which occur in the property position of statements in G;

- $IP_{\mathrm{Herb}(G)}(\mathtt{p})$ is the maximum of n for all statements in which `p` occur in property position;
- $IEXT_{\mathtt{Herb}(G)}(\langle \mathtt{s}, \mathtt{o} \rangle)$ is the maximum n for all the statements `n: s p o.` in G
- $IS_{\mathtt{Herb}}(G)$ and $IL_{\mathtt{Herb}}(G)$ are both identity mappings on the appropriate parts of the vocabulary of G.

Using this definition of Herbrand interpretation instead of that in Appendix A of [3], the proofs for cited lemmas still apply.

2.5 RDF Interpretation

RDF Semantic Conditions

- $IP(x) = IEXT(I(\mathtt{rdf:type}))(\langle x, I(\mathtt{rdf:Property}) \rangle)$
- If $"\mathtt{xxx}"\wedge\wedge\mathtt{rdf:XMLLiteral} \in V$ and `xxx` is a well-typed XML literal string, then
  - $IL("\mathtt{xxx}" \wedge \wedge\mathtt{rdf:XMLLiteral})$ is the XML value of `xxx`;
  - $IL("\mathtt{xxx}" \wedge \wedge\mathtt{rdf:XMLLiteral}) \in LV$;
  - $IEXT(I(\mathtt{rdf:type}))$
    $(\langle IL("\mathtt{xxx}" \wedge \wedge\mathtt{rdf:XMLLiteral}),$
    $I(\mathtt{rdf:XMLLiteral})\rangle) = 1$
- If $"\mathtt{xxx}"\wedge\wedge\mathtt{rdf:XMLLiteral} \in V$ and `xxx` is an ill-typed XML literal string, then
  - $IL("\mathtt{xxx}" \wedge \wedge\mathtt{rdf:XMLLiteral}) \notin LV$;
  - $IEXT(I(\mathtt{rdf:type}))$
    $(\langle IL("\mathtt{xxx}" \wedge \wedge\mathtt{rdf:XMLLiteral}),$
    $I(\mathtt{rdf:XMLLiteral})\rangle) = 0$

The first RDF semantic condition has the consequence that IP must be a subset of IR. Given such a fact, there is no more need of IDP, as it was for simple interpretation. IP can be directly defined as a fuzzy subset of IR.

The second and third conditions equal to see the well-formedness of an XML Literal as crisp truth-valued. We could conceive an external machinery that can be considered completely trustworthy as it classify an XML literal as well-formed or not.

RDF axiomatic triples By definition, we give axiomatic triples a unit truth value. Given the (syntactic) convention that a triple `s p o.` is equivalent to the fuzzy statement `1: s p o.`, we can take the table of axiomatic triples of RDF in section 3.1 of [3] and copy it as-is as the table of axiomatic statements of fuzzy RDF.

3 Fuzzy RDF Schema

The path from RDF Schema to Fuzzy RDF Schema follows the same guidelines of the previous section.

The RDFS semantics is conveniently stated in terms of a new semantic construct: the *class* [3]. A class is a resource with a *class extension*, $ICEXT$, which represents a set of things in the universe which all have that class as the object of their `rdf:type` property. Thus, the definition of a *class* roots in the definition of *extension.*

In fuzzy RDF, extensions are fuzzy set of couples; in fuzzy RDFS, *class extensions* are fuzzy sets of domain's elements.

3.1 RDFS Interpretation

A RDFS interpretation define the domains for resources (IR), literals (IL) and literal values (LV) in terms of classes. In fuzzy RDFS they are fuzzy subdomains of IR.

We will give RDFS semantic conditions and axiomatic triples, then we will try to explain the more problematic definitions: domains/ranges (section 3.2) and subproperties/subclasses (section 3.3).

RDFS semantic conditions

- $ICEXT(y)(x) = IEXT(I(\texttt{rdf}:\texttt{type}))(\langle x, y\rangle)$
  - $IC = ICEXT(I(\texttt{rdfs}:\texttt{Class}))$
  - $IR = ICEXT(I(\texttt{rdfs}:\texttt{Resource}))$
  - $IL = ICEXT(I(\texttt{rdfs}:\texttt{Literal}))$
- $ICEXT(y)(u) \geq \min(IEXT(I(\texttt{rdfs}:\texttt{domain}))(\langle x, y\rangle), IEXT(x)(\langle u, v\rangle))$
- $ICEXT(y)(u) \geq \min(IEXT(I(\text{rdfs}:\text{range}))(\langle x, y\rangle), IEXT(x)(\langle u, v\rangle))$
- $IEXT(I(\texttt{rdfs}:\texttt{subPropertyOf}))$ is transitive and reflexive on IP
- If $IEXT(\texttt{rdfs}:\texttt{subPropertyOf})(\langle x, y\rangle) = n$, then $IP(x) \geq n$, $IP(y) \geq n$, $\min_{\langle a,b\rangle}\{1 - IEXT(x)(\langle a, b\rangle) + IEXT(y)(\langle a, b\rangle)\} \geq n$
- $IEXT(I(\texttt{rdfs}:\texttt{subClassOf}))(\langle \text{x}, I(\texttt{rdfs}:\texttt{Resource})\rangle) = IC(x)$
- If $IEXT(\texttt{rdfs}:\texttt{subClassOf})(\langle x, y\rangle) = n$, then $IC(x) \geq n$, $IC(y) \geq n$, $\min_a\{1 - IC(x)(a) + IC(y)(a)\} \geq n$.
- $IEXT(I(\texttt{rdfs:subClassOf}))$ is transitive and reflexive on IC
- $IEXT(I(\texttt{rdfs}:\texttt{subPropertyOf}))(\langle \text{x}, I(\texttt{rdfs}:\texttt{member})\rangle) =$ $ICEXT(I(\texttt{rdfs}:\texttt{ContainerMembershipProperty}))(x)$
- $ICEXT(I(\texttt{rdfs}:\texttt{Datatype}))(x) =$ $IEXT(I(\texttt{rdfs}:\texttt{subClassOf}))(\langle x, I(\texttt{rdfs}:\texttt{Literal})\rangle)$

RDFS axiomatic triples As for RDF axiomatic triples, fuzzy RDFS axioms are the same of plain RDFS, from section 4.2 of RDF Semantics [3].

3.2 Domains and ranges

The semantic condition on domains looks quite complicated. To explain it, we will proceed by grades.

In plain RDF Schema, if $\langle x,y\rangle \in IEXT(I(\texttt{rdfs:domain}))$ and $\langle u,v\rangle \in IEXT(x)$ then $u \in ICEXT(y)$.

In fuzzy set theory, let R be a fuzzy relation on $X \times Y$. Then the domain is defined as $\mathrm{dom}(R)(x) = \sup_y R(x,y)$ [7], i.e. the least upper bound of $R(x,y)$ for all y.

In fuzzy RDFS, we have to deal both with a fuzzy notion of domain, and with a fuzzy assignment of a domain to a property.

Let consider a resource `u` and a class `y`. For each property `x`, we take the minimum between $IEXT(I(\texttt{rdfs:domain}))(\langle x,y\rangle)$ and $IEXT(x)(\langle u,v\rangle)$. Then, following the original RDFS condition, $ICEXT(y)(u)$ must be greater or equal than this value.

The previous condition must hold for every property `x`, so it's equivalent to state that must be taken the maximum value.

The conditions for ranges are analogous.

3.3 Subproperties and subclasses

Subproperties and subclasses are fully analogous concepts. The set inclusion is between extensions for the former, between class extensions of the latter.

To define the semantics of `subClassOf` and `subPropertyOf`, we need a relation of set inclusion between fuzzy sets that takes into account also the degree of the relation of inclusion itself. This relation must be transitive and reflexive.

Zadeh's definition of fuzzy subset [8][3] ($A \subseteq B \iff \forall x \in X \quad A(x) \leq B(x)$) is transitive and reflexive, but is not a fuzzy relation: either the set A is a subset of B, or not. What we need is instead a weaker fuzzy subset relation; a relation that reduces to the Zadeh's one when the subclass/subproperty relation has a unit truth value. It must also maintain the reflexivity and transitivity properties.

Dubois and Prade [7] define *weak inclusion* $\prec_\alpha$ as

$$A \prec_\alpha B \iff x \in (\overline{A} \cup B)_\alpha \; \forall x \in X \ ,$$

where α is a parameter and $(\cdot)_\alpha$ is the α-cut[4]. This relation is transitive only for $\alpha > \frac{1}{2}$.

Other definitions of weak inclusion make use of *inclusion grades*. An inclusion grade $I(A,B)$ is a scalar measure of the inclusion of the set A in the set B. In general, $A \subseteq_\alpha B$ iff $I(A,B) \geq \alpha$, where $\subseteq_\alpha$ denote a weak inclusion with inclusion grade α.

We have chosen to use the inclusion grade defined as [7]:

[3] Again, we use the abbreviation $A(x)$ for the membership function $\mu_A(x)$.

[4] The *α-cut* A_α of A is the set of all elements with a membership value to A greater than α, with $\alpha \in (0,1] \quad A_\alpha = \{x | A(x) \geq \alpha\}$

$$I(A,B) = \inf_{x \in X} \overline{(A \mid - \mid B)}(x)$$

where inf is the infimum and $| - |$ is the *bounded difference*[5].

When $A \subseteq B$, $I(A,B) = 1$ [7].

This inclusion grade could also be written as $I(A,B) = \inf_{x \in X}(1 - \max(0,\ A(x) - B(x))) = \inf_{x \in X} \min(1,\ (1 - A + B))$.

Furthermore, let's suppose that there is at least an x such that $A(x) > B(x)$. Then $I(A,B)$ could be written as $\inf_{x \in X}(1 - A + B)$. The semantic condition requires such measure to be greater than or equal to n, where n is the truth value of the statement. In this case the semantic condition reduces to

$$\inf_{x \in X}(1 - A + B) \geq n\ .$$

It could be interesting to ask how much this definition differs from the condition for classical fuzzy subsets, $A(x) \leq B(x)$.

If $A \subseteq B$, then $I(A,B) = 1$, so the semantic condition holds for any $n \in [0,1]$.

Let's call $d(x)$ the difference $d(x) = A(x) - B(x)$, so that $1 - A + B = 1 - d$. We suppose that there is at least an x such that $A(x) > B(x)$, so $d(x)$ has at least a positive value. The semantic condition could then be written $\inf_{x \in X}(1 - d(x)) \geq n$. The maximum positive value of the difference d equal to $1 - n$.

As n is the truth value of the statement that asserts the relation of subproperty or subclass, and $1 - n$ represent the lack of truth of the same statement, we can conclude that the maximum allowable positive difference between $A(x)$ and $B(x)$ is equal to the lack of truth on the subproperty or subclass relation.

4 Fuzzy RDF entailment rules

RDF Model Theory's entailment rules [3] are all of the same form: add a statement to a graph when it contains triples conforming to a pattern. Each rule has only one or two antecedent statements and derive only one new inferred statement; either $P \vdash R$ or $P, Q \vdash R$.

Given the way fuzzy RDF semantics is defined, the corresponding inference rules for fuzzy RDF are analogous; only the fuzzy truth values of inferred statements must be computed. The simplest possible choice that respect the semantics is:

- With rules as $P \vdash Q$, having only one antecedent, the truth value of the consequent Q is taken to be the same of the antecedent P.
- With rules as $P, Q \vdash R$, the truth value of R is the minimum between the truth values of P and Q.

The inference rules for RDF/RDFS are shown in table 2. They were derived from the rules used by the Sesame[10] forward-chaining inferencer.

[5] $\forall x \in X, \quad (A \mid - \mid B)(x) = \max(0,\ A(x) - B(x))$ [9]

Sesame is a generic architecture for storing and querying RDF and RDF Schema. It makes use of a forward-chaining inferencer to compute and store the closure of its knowledge base whenever a transaction adds data to the repository[11]. Sesame applies RDF-MT inference rules in a optimized way, making use of the dependencies between them to eliminate most redundant inferencing steps.

To obtain a fuzzy RDF storage and inference tool it is only a matter of modify Sesame RDF-MT inferencer, making it compute the correct truth values for inferred statements, and to extend the underlying storage to make room for a truth value (i.e., a number) for each statement.

This shows how a simple proof-of-concept fuzzy RDF inferencer is easy to implement. The starting point is the code base of an inference engine that implements the RDF model theory.

It can be shown that an inference engine implementing such rules is correct: all its rules are *valid*, in the sense that a graph entails any larger graph that is obtained by applying the rules to the original graph. There is no formal proof that it is also complete, but there is not such a proof for plain RDF Model Theory inference rules either.

References

1. Hendler, J., Lassila, O., Berners-Lee, T.: The semantic web. Scientific American (2001) 28–37
2. Berners-Lee, T.: What the semantic web can represent. W3C design issues, World Wide Web Consortium (September 1998)
3. Hayes, P.: RDF Semantics. W3C recommendation, World Wide Web Consortium (10 February 2004)
4. Grant, J., Beckett, D.: RDF test cases. W3C recommendation, World Wide Web Consortium (2004)
   http://www.w3.org/TR/rdf-testcases/.
5. Zadeh, L.A.: A fuzzy set theoretic interpretation of linguistic hedges. Journal of Cybernetics **2** (1972) 4–34
6. Goldfarb, W.D., ed.: Logical Writings of Jacques Herbrand. Harvard University Press, Cambridge (1971)
7. Dubuois, D., Prade, H.: Fuzzy sets and Systems. Academic Press, New York, NJ (1980)
8. Zadeh, L.A.: Fuzzy sets. Information and Control (1965) 338–353
9. Zadeh, L.A.: Calculus of Fuzzy Restrictions. In: Fuzzy Sets and Their Applications to Cognitive and Decision Processes. Academic Press, New York (1975) 1–39
10. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A generic architecture for storing and quering RDF and RDF Schema. In Horrocks, I., Handler, J., eds.: Proceedings of the first International Semantic Web Conference (ISWC 2002), Sardinia, Italy (2002) 54–68
11. Broekstra, J., Kampman, A.: Inferencing and truth maintenance in RDF Schema: exploring a naive practical approach. In: Workshop on Practical and Scalable Semantic Systems (PSSS) 2003. Second International Semantic Web Conference (ISWC), Sanibel Island, Florida, USA (2003)

#	antecedents	consequent
1	iii: xxx aaa yyy	iii: aaa rdf:type rdf:Property
2.1	iii: xxx aaa yyy jjj: aaa rdfs:domain zzz	kkk: xxx rdf:type zzz where kkk = min(iii, jjj)
2.2	iii: aaa rdfs:domain zzz jjj: xxx aaa yyy	kkk: xxx rdf:type zzz where kkk = min(iii, jjj)
3.1	iii: xxx aaa uuu jjj: aaa rdfs:range zzz	kkk: uuu rdf:type zzz where kkk = min(iii, jjj)
3.2	iii: aaa rdfs:range zzz jjj: xxx aaa uuu	kkk: uuu rdf:type zzz where kkk=min(iii, jjj)
4a	iii: xxx aaa yyy	jjj: xxx rdf:type rdfs:Resource
4b	iii: xxx aaa uuu	iii: uuu rdf:type rdfs:Resource
5a.1	iii: aaa rdfs:subPropertyOf bbb jjj: bbb rdfs:subPropertyOf ccc	kkk: aaa rdfs:subPropertyOf ccc where kkk=min(iii, jjj)
5a.2	iii: bbb rdfs:subPropertyOf ccc jjj: aaa rdfs:subPropertyOf bbb	kkk: aaa rdfs:subPropertyOf ccc where kkk=min(iii, jjj)
5b	iii: xxx rdf:type rdf:Property	iii: xxx rdfs:subPropertyOf xxx *reflexivity of rdfs:subPropertyOf*
6.1	iii: xxx aaa yyy jjj: aaa rdfs:subPropertyOf bbb	kkk: xxx bbb yyy where kkk=min(iii, jjj)
6.2	iii: aaa rdfs:subPropertyOf bbb jjj: xxx aaa yyy	kkk: xxx bbb yyy where kkk=min(iii, jjj)
7a	iii: xxx rdf:type rdfs:Class	iii: xxx rdfs:subClassOf rdfs:Resource
7b	iii: xxx rdf:type rdfs:Class	iii: xxx rdfs:subClassOf xxx *reflexivity of rdfs:subClassOf*
8.1	iii: xxx rdfs:subClassOf yyy jjj: yyy rdfs:subClassOf zzz	kkk: xxx rdfs:subClassOf zzz where kkk=min(iii, jjj)
8.2	iii: yyy rdfs:subClassOf zzz jjj: xxx rdfs:subClassOf yyy	kkk: xxx rdfs:subClassOf zzz where kkk=min(iii, jjj)
9.1	iii: xxx rdfs:subClassOf yyy jjj: aaa rdf:type xxx	kkk: aaa rdf:type yyy where kkk=min(iii, jjj)
9.2	iii: aaa rdf:type xxx jjj: xxx rdfs:subClassOf yyy	kkk: aaa rdf:type yyy where kkk=min(iii, jjj)
10	iii: xxx rdf:type rdfs:ContainerMembershipProperty	iii: xxx rdfs:subPropertyOf rdfs:member
11	iii: xxx rdf:type rdfs:Datatype	jjj: xxx rdfs:subClassOf rdfs:Literal
X1	iii: xxx rdf:_* yyy	jjj: rdf:_* rdf:type rdfs:ContainerMembershipProperty *This is an extra rule for list membership properties (_1, _2, _3, ...). The RDF MT does not specify a production for this.*

Table 2. Fuzzy RDF inference rules

PR-OWL: A Bayesian Ontology Language for the Semantic Web

Paulo Cesar G. da Costa, Kathryn B. Laskey
School of Information Technology and Engineering,
George Mason University
4400 University Drive
Fairfax, VA 22030-4444 USA
{pcosta, klaskey}@gmu.edu

Kenneth J. Laskey[#]
MITRE Corporation, M/S H305
7515 Colshire Drive
McLean, VA 22102-7508 USA
klaskey@mitre.org

Abstract. This paper addresses a major weakness of current technologies for the Semantic Web, namely the lack of a principled means to represent and reason about uncertainty. This not only hinders the realization of the original vision for the Semantic Web, but also creates a barrier to the development of new, powerful features for general knowledge applications that require proper treatment of uncertain phenomena. We propose to extend OWL, the ontology language recommended by the World Wide Web Consortium (W3C), to provide the ability to express probabilistic knowledge. The new language, PR-OWL, will allow legacy ontologies to interoperate with newly developed probabilistic ontologies. PR-OWL will move beyond the current limitations of deterministic classical logic to a full first-order probabilistic logic. By providing a principled means of modeling uncertainty in ontologies, PR-OWL will serve as a supporting tool for many applications that can benefit from probabilistic inference within an ontology language, thus representing an important step toward the W3C's vision for the Semantic Web.

1 A Deterministic View of a Probabilistic World

Uncertainty is ubiquitous. If the Semantic Web vision [1] is to be realized, a sound and principled means of representing and reasoning with uncertainty will be required. Existing Semantic Web technologies lack this capability. Our broad objective is to address this shortcoming by developing a Bayesian framework for probabilistic ontologies and plausible reasoning services. As an initial step toward our objective, we introduce PR-OWL, a probabilistic extension to the Web ontology language OWL.

Although our research is focused in the Semantic Web, we are tackling a problem that long predates the WWW: the quest for more efficient data exchange. Clearly, solving that problem requires precise semantics and flexible ways to convey information. While the WWW provides a new presentation medium, and technologies such as XML present new data exchange formats, neither addresses the semantics of data

[#] The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the author.

being exchanged. The SW is meant to fill this gap, and the realization of its goals will require major improvements in technologies for data exchange.

One of the main technical differences between the current World Wide Web and the Semantic Web is that while the first relies on syntactic-only protocols such as HTTP and HTML, the latter adds meta-data annotations as a means to convey shared, precisely defined terms. That is, semantic awareness is exploited to improve interoperability among Web resources. Semantic interoperability requires shared repositories of precisely defined concepts. Such repositories are called ontologies.

One can find many different definitions for the concept of ontology applied to information systems, each emphasizing a specific aspect its author judged most important. Our focus is on ontology's role as a structured form of knowledge representation. Thus, we define an ontology as an explicit, formal representation of knowledge about a domain of application. This includes: types of entities that exist in the domain, properties of those entities, relationships among entities, and processes and events that happen with those entities. In this definition, 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. Ontologies are used for the purpose of comprehensively describing knowledge about a domain in a structured and sharable way, ideally in a format that can be read and processed by a computer.

Semantically aware schemes must be able to represent and appropriately process semantic differences between syntactically identical terms (e.g., "Grape" as a fruit versus John Grape the person). This is not a trivial task. Semantic interoperability requires shared sources of precisely defined concepts, which is exactly where ontologies play a key role. Yet, a traditional ontology can at best list multiple possible senses for a word such as "Grape," with no ability to grade their relative plausibility in a given context. This is inadequate for an open world environment where incomplete information is the rule and plausible reasoning is required.

Current generation Semantic Web technology is based on classical logic, and is lacks adequate support for plausible reasoning. For example, OWL, a W3C Recommendation [2], has no built-in support for probabilistic information and reasoning. This is understandable, given that OWL is rooted in web language predecessors (i.e. XML, RDF) and traditional knowledge representation formalisms (e.g.. Description Logics [3]). This historical background somewhat explains the lack of support for uncertainty in OWL. Nevertheless, it is a serious limitation for a language intended for environments where one cannot simply ignore incomplete information.

A similar historical progression occurred in Artificial Intelligence (AI). From its inception, AI has struggled with how to cope with incomplete information. Although probability theory was initially neglected due to tractability concerns, graphical probability languages changed things dramatically [4]. Probabilistic languages have evolved from propositional to full first-order expressivity (e.g., [5]), and have become the technology of choice for reasoning under uncertainty in an open world [6]. Clearly, the Semantic Web will pose similar uncertainty-related issues as those faced by AI. Thus, just as AI has moved from a deterministic paradigm to embrace probability, a similar path appears promising for ontology engineering.

This path is not yet being followed. The lack of support for representing and reasoning with uncertain, incomplete information seriously limits the ability of current Semantic Web technologies to meet the requirements of the Semantic Web. Our work is an initial step toward changing this situation. We aim to establish a framework that enables full support for uncertainty in the field of ontology engineering and, as a consequence, for the Semantic Web. In order to achieve this goal, we choose to focus on extending OWL so it can represent uncertainty in a principled way.

2 Related Research

One of the main reasons why Semantic Web research is still focused on deterministic approaches has been the limited expressivity of traditional probabilistic languages. There is a current line of research focused on extending OWL so it can represent probabilistic information contained in a Bayesian Network (e.g. [7], [8]). The approach involves augmenting OWL semantics to allow probabilistic information to be represented via additional markups. The result would be a probabilistic annotated ontology that could then be translated to a Bayesian network (BN). Such a translation would be based on a set of translation rules that would rely on the probabilistic information attached to individual concepts and properties within the annotated ontology. BNs provide an elegant mathematical structure for modeling complex relationships among hypotheses while keeping a relatively simple visualization of these relationships. Yet, the limited attribute-value representation of BNs makes them unsuitable for problems requiring greater expressive power.

Another popular option for representing uncertainty in OWL has been to focus on OWL-DL, a decidable subset of OWL that is based on Description Logics [3]. Description Logics are a family of knowledge representation formalisms that represent the knowledge of an application domain (the "world") by first defining the relevant concepts of the domain (its terminology), and then using these concepts to specify properties of objects and individuals occurring in the domain (the world description).

Description logics are highly effective and efficient for the classification and subsumption problems they were designed to address. However, their ability to represent and reason about other commonly occurring kinds of knowledge is limited. One restrictive aspect of DL languages is their limited ability to represent constraints on the instances that can participate in a relationship. As an example, suppose we want to express that for carnivore to be a threat to another carnivore in a specific type of situation it is mandatory that the two individuals of class Carnivore involved in the situation are not the same. Making sure the two carnivores are different in a specific situation is only possible in DL if we actually create/specify the tangible individuals involved in that situation. Indeed, stating that two "fillers" (i.e. the actual individuals of class Carnivore that will "fill the spaces" of concept carnivore in our statement) are not equal without specifying their respective values would require constructs such as *negation* and *equality role-value-maps*, which cannot be expressed in description logic. While equality role-value-maps provide useful means to specify structural properties of concepts, their inclusion makes the logic undecidable [9].

Although the above approaches are promising where applicable, a definitive solution for the Semantic Web requires a general-purpose formalism that gives ontology designers a range of options to balance tractability against expressiveness.

Pool and Aiken [10] developed an OWL-based interface for the relational probabilistic toolset Quiddity*Suite, developed by IET, Inc. Their constructs provide a very expressive method for representing uncertainty in OWL ontologies. Their work is similar in spirit to ours, but is specialized to the Quiddity*Suite toolset. We focus on the more general problem of enabling probabilistic ontologies for the SW. We employ Multi-Entity Bayesian Networks (MEBN) as our underlying logical basis, thus providing full first-order expressiveness.

3 Multi-Entity Bayesian Networks

The acknowledged standard for logically coherent reasoning under uncertainty is Bayesian probability theory. Bayesian theory provides a principled representation of uncertainty, a logic for combining prior knowledge with observations, and a learning theory for refining the ontology as evidence accrues. The logical basis for PR-OWL is MEBN logic [5], which combines Bayesian probability theory with classical First Order Logic. Probabilistic knowledge is expressed as a set of MEBN fragments (MFrags) organized into MEBN Theories. An MFrag is a knowledge structure that represents probabilistic knowledge about a collection of related hypotheses. Hypotheses in an MFrag may be *context* (must be satisfied for the probability definitions to apply), *input* (probabilities are defined in other MFrags), or *resident* (probabilities defined in the MFrag itself). An MFrag can be instantiated to create as many instances of the hypotheses as needed (e.g., an instance of the "Disease" hypothesis for each patient at a clinic). Instances of different MFrags may be combined to form complex probability models for specific situations. A MEBN theory is a collection of MFrags that satisfies consistency constraints ensuring the existence of a unique joint probability distribution over instances of the hypotheses in its MFrags.

MEBN inference begins when a query is posed to assess the degree of belief in a target random variable given a set of evidence random variables. We start with a generative MTheory, add a set of finding MFrags representing problem-specific information, and specify the target nodes for our query. The first step in MEBN inference is to construct a situation-specific Bayesian network (SSBN), which is a Bayesian network constructed by creating and combining instances of the MFrags in the generative MTheory. When each MFrag is instantiated, instances of its random variables are created to represent known background information, observed evidence, and queries of interest to the decision maker. If there are any random variables with undefined distributions, then the algorithm proceeds by instantiating their respective home MFrags. The process of retrieving and instantiating MFrags continues until there are no remaining random variables having either undefined distributions or unknown values. A SSBN may contain any number of instances of each MFrag, depending on the number of entities and their interrelationships. Next, a standard Bayesian network inference algorithm is applied. Finally, the answer to the query is obtained by inspecting the posterior probabilities of the target nodes.

MEBN logic overcomes the limitations of the attribute-value representation of standard BNs. To understand this limitation, consider a relational database in which some entries are uncertain. A BN can represent only probabilities for a single table, and treats the rows of the table independently of each other. For example, in a medical system, the "Patient" table might include information such as age, smoking history, family history, and whether the patient has emphysema. A BN might represent the probability of emphysema as a function of smoking history, age, and family history. If a patient's family history were unknown, the BN could estimate the probability of emphysema using probabilities for the family history. However, a BN cannot represent relational information such as the increase in the probability of emphysema for all siblings upon learning that one of their parents had emphysema. To incorporate this kind of knowledge in a coherent manner, we need to combine *relational* knowledge (e.g., siblings have the same family history) with attribute-value knowledge (e.g., family history of emphysema increases the likelihood of emphysema).

To draw generalizations about individuals related in various ways, we need first-order expressive power. Description logics are attractive because they provide limited first-order expressivity, yet certain reasoning problems such as classification and subsumption are decidable. Many researchers have worked to identify decidable classes of problems for which efficient probabilistic algorithms exist (e.g., Naïve Bayes classification, in which features are modeled as conditionally independent given an object's class). The ontology language P-SHOQ(D) [11], based on description logics, falls into this class.

We have chosen to base PR-OWL on MEBN logic because of its expressiveness: MEBN can express a probability distribution over models of any finitely axiomatizable first-order theory. As a consequence, there are no guarantees that exact reasoning with a PR-OWL ontology will be efficient or even decidable. On the other hand, a future objective is to identify restricted sub-languages of PR-OWL specialized to classes of problems for which efficient exact or approximate reasoning algorithms exist. It is our view that a general-purpose language for the Semantic Web should be as expressive as possible, while providing a means for ontology engineers to stay within a tractable subset of the language when warranted by the application.

4 Probabilistic Ontologies

Before presenting our probabilistic ontology language, we begin by defining a probabilistic ontology. Intuitively, an ontology that has probabilities attached to some of its elements would qualify for this label, but such a limited definition is inadequate for our purposes. Merely adding probabilities to concepts does not guarantee interoperability with other ontologies that also carry probabilities. More is needed than syntax for including probabilities if we are to justify a new category of ontologies.

A *probabilistic ontology* is an explicit, formal knowledge representation that expresses knowledge about a domain of application. This includes: (*i*) Types of entities that exist in the domain; (*ii*) Properties of those entities; (*iii*) Relationships among entities; (*iv*) Processes and events that happen with those entities; (*v*) Statistical regularities that characterize the domain; (*vi*) Inconclusive, ambiguous, incomplete, unreliable, and dissonant knowledge related to entities of the domain; and (*vii*) Uncertainty about all the above forms of knowledge. In this definition, the term entity refers to any concept (real or fictitious, concrete or abstract) that can be described and reasoned about within the domain.

Probabilistic Ontologies are used for the purpose of comprehensively describing knowledge about a domain and the uncertainty regarding that knowledge in a principled, structured and sharable way, ideally in a format that can be read and processed by a computer. They also expand the possibilities of standard ontologies by introducing the requirement of a proper representation of the statistical regularities and the uncertain evidence about entities in a domain of application.

5 PR-OWL

PR-OWL is an extension that enables OWL ontologies to represent complex Bayesian probabilistic models in a way that is flexible enough to be used by diverse Bayesian probabilistic tools based on different probabilistic technologies. That level of flexibility can only be achieved using the underlying semantics of first-order Bayesian logic, which is not a part of the standard OWL semantics and abstract syntax. Therefore, it seems clear that PR-OWL can only be realized via extending the semantics and abstract syntax of OWL. However, in order to make use of those extensions, it is necessary to develop new tools supporting the extended syntax and implied semantics of each extension. Such an effort would require commitment from diverse developers and workgroups, which falls outside our present scope.

Therefore, in this initial work our intention is to create an upper ontology to guide the development of probabilistic ontologies. Daconta *et al.* define an upper ontology as a set of integrated ontologies that characterizes a set of basic commonsense knowledge notions [12]. In this preliminary work on PR-OWL as an upper ontology, these basic commonsense notions are related to representing uncertainty in a principled way using OWL syntax. If PR-OWL were to become a W3C Recommendation, this collection of notions would be formally incorporated into the OWL language as a set of constructs that can be employed to build probabilistic ontologies.

The PR-OWL upper ontology for probabilistic systems consists of a set of classes, subclasses and properties that collectively form a framework for building probabilistic ontologies. The first step toward building a probabilistic ontology in compliance with our definition is to import into any OWL editor an OWL file containing the PR-OWL classes, subclasses, and properties.

From our definition, it is clear that nothing prevents a probabilistic ontology from being "partially probabilistic". That is, a knowledge engineer can choose the concepts he/she wants to include in the "probabilistic part" of the ontology, while writing the other concepts in standard OWL. In this case, the "probabilistic part" refers to the concepts written using PR-OWL definitions and that collectively form a MEBN Theory. There is no need for all the concepts in a probabilistic ontology to be probabilistic, but at least some have to form a valid MEBN Theory. Of course, only the concepts that are part of the MEBN Theory will be subject to the advantages of the probabilistic ontology over a deterministic one.

The subtlety here is that legacy OWL ontologies can be upgraded to probabilistic ontologies only with respect to concepts for which the modeler wants to have uncertainty represented in a principled manner, make plausible inferences from that uncertain evidence, or to learn its parameters from incoming data via Bayesian learning. While the first two are direct consequences of using a probabilistic knowledge representation, the latter is a specific advantage of the Bayesian paradigm, where learning falls into the same conceptual framework as knowledge representation.

The ability to perform probabilistic reasoning with incomplete or uncertain information conveyed through an ontology is a major advantage of PR-OWL. However, it should be noted that in some cases solving a probabilistic query might be intractable or even undecidable. In fact, providing the means to ensure decidability was the reason why the W3C defined three different version of the OWL language. While OWL Full is more expressive, it enables an ontology to represent knowledge that can lead to undecidable queries. OWL-DL imposes some restrictions to OWL in order to eliminate these cases. Similarly, restrictions of PR-OWL could be developed that limit expressivity to avoid undecidable queries or guarantee tractability. Possible restrictions to be considered for an eventual PR-OWL Lite include (*i*) constraining the language to classes of problems for which tractable exact or approximate algorithms exist; (*ii*) restrict the representation of the conditional probability tables (CPT) to express a tractable and expressive subset of first-order logic; and/or (*iii*) to employ a standard semantic web language syntax to represent the CPTs (e.g. RDF). As an initial step, we chose to focus on the most expressive version of PR-OWL, which does not have expressivity restrictions and provides the ability to represent CPTs in multiple formats.

An overview of the general concepts involved in the definition of a MEBN Theory in PR-OWL is depicted in Figure 1. In this diagram, the ovals represent general classes; and arrows represent major relationships between classes. A probabilistic ontology must have at least one individual of class MTheory, which is a label linking a group of MFrags that collectively form a valid MEBN Theory. In actual PR-OWL syntax, that link is expressed via the object property hasMFrag (which is the inverse of object property isMFragIn).

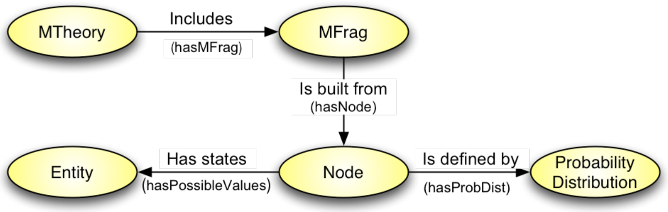


Fig. 1. Overview of a PR-OWL MEBN Theory Concepts

Individuals of class MFrag are comprised of nodes, which can be resident, input, or context nodes (not shown in the picture). Each individual of class Node is a random variable and thus has a mutually exclusive and collectively exhaustive set of possible states. In PR-OWL, the object property hasPossibleValues links each node with its possible states, which are individuals of class Entity. Finally, random variables (represented by the class Nodes in PR-OWL) have unconditional or conditional probability distributions, which are represented by class Probability Distribution and linked to its respective nodes via the object property hasProbDist.

The scheme in Figure 1 is intended to present just a general view and thus fails to show many of the intricacies of an actual PR-OWL representation of a MEBN Theory. Figure 2 shows an expanded version conveying the main elements in Figure 1, their subclasses, the secondary elements that are needed for representing a MEBN Theory and the reified relationships that were necessary for expressing the complex structure of a Bayesian probabilistic model using OWL syntax.

Reification of relationships in PR-OWL is necessary because of the fact that properties in OWL are binary relations (i.e. link two individuals or an individual and a value), while many of the relations in a probabilistic model include more than one individual (i.e. N-ary relations). The use of reification for representing N-ary relations on the Semantic Web is covered by a working draft from the W3C's Semantic Web Best Practices Working Group [13].

Although the scheme in Figure 2 shows all the elements needed to represent a complete MEBN Theory, it is clear that any attempt at a complete description would render the diagram cluttered and incomprehensible. A complete account of the classes, properties and the code of PR-OWL that define an upper ontology for probabilistic systems is given in [14]. These definitions can be used to represent any MEBN Theory.

In its current stage, PR-OWL contains only the basic elements needed to represent any MEBN theory. Such a representation could be used by a Bayesian tool (acting as a probabilistic ontology reasoner) to perform inferences to answer queries and/or to learn from newly incoming evidence via Bayesian learning.

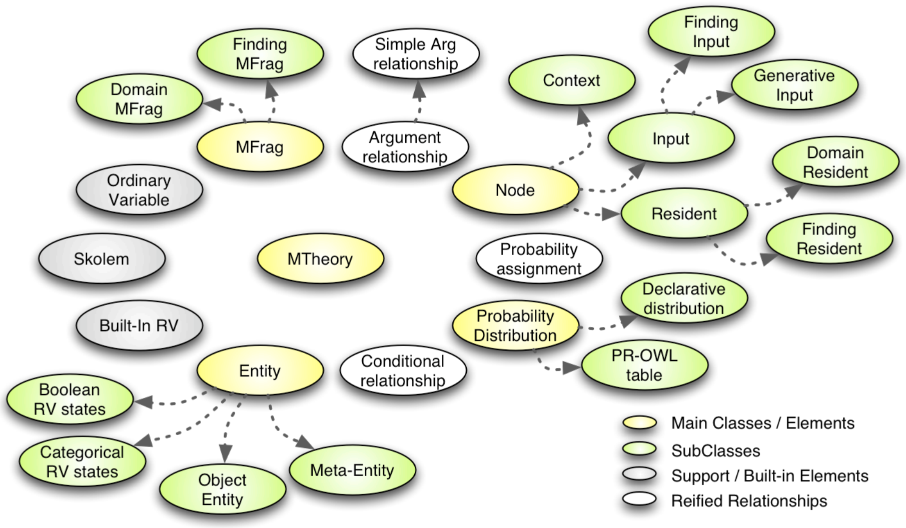
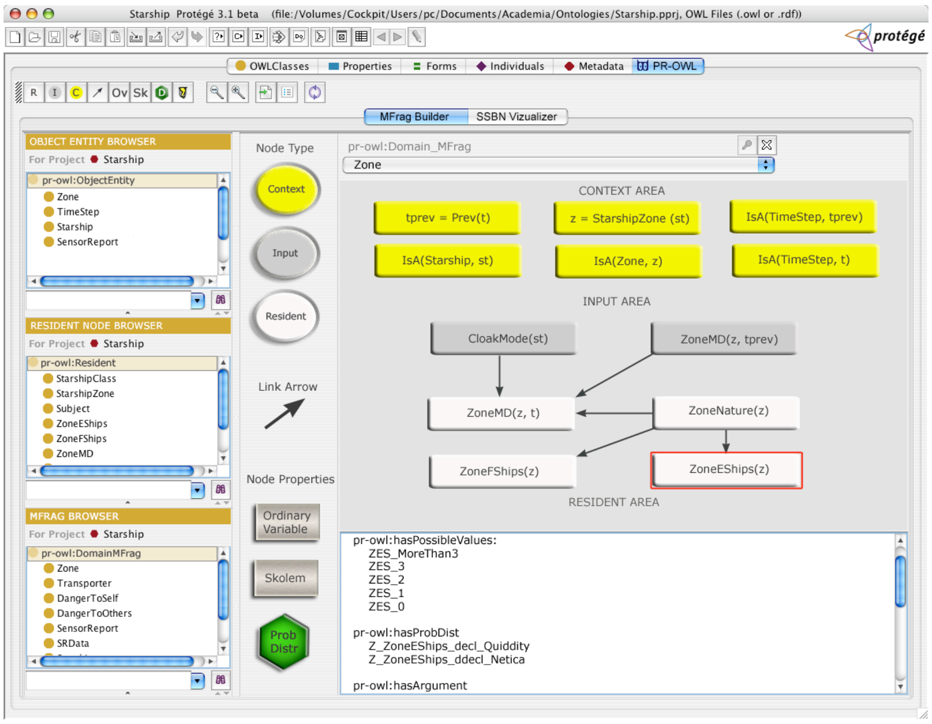


Fig. 2. Elements of a PR-OWL Probabilistic Ontology

However, building MFrags and all their elements in a probabilistic ontology is a manual, error prone, and tedious process. Avoiding errors or inconsistencies requires very deep knowledge of the logic and of the data structure of PR-OWL. Without considering the future paths to be followed by research on PR-OWL (i.e. whether it will be kept as an upper ontology or transformed into an actual extension to the OWL language), the framework discussed here and in greater detail in [14] makes it already possible to facilitate probabilistic ontology usage and editing by developing plugins to current OWL editors. Figure 3 illustrates a plugin concept for the OWL Protégé editor (which is itself a Protégé plugin). The figure illustrates how graphical construction of an MFrag can be performed in a similar fashion to how a BN is constructed in one of the many graphical editors for BNs. In this proposed scheme, in order to build an MFrag a user would select the icon for the type of node he/she wants to create (e.g. resident, input, context, etc.), connect that node with its parents and children, and enter its basic characteristics (i.e. name, probability distribution, etc.) either by double-clicking on it or via another GUI-related facility. Such a plugin would hide from users the complex constructs required to convey the many details of a probabilistic ontology, providing a more intuitive and less error-prone means of constructing and maintaining probabilistic ontologies.



```
pr-owl:hasPossibleValues:
   ZES_MoreThan3
   ZES_3
   ZES_2
   ZES_1
   ZES_0

pr-owl:hasProbDist
   Z_ZoneEShips_decl_Quiddity
   Z_ZoneEShips_ddecl_Netica

pr-owl:hasArgument
```

Fig. 3. Elements of a PR-OWL Probabilistic Ontology

This brief idea of an operational concept barely scratches the surface of the many possibilities for the technology presented here. Implementing a plugin such as the one envisioned here is a development task that is a topic for future research. Nonetheless, the PR-OWL upper ontology definitions take an important first step toward making probabilistic ontologies a reality. By opening the door to wide use of PR-OWL probabilistic ontologies, the present research makes a significant contribution to realizing the Semantic Web vision.

6 Conclusion

This paper describes a coherent, comprehensive probabilistic framework for the Semantic Web, that provides a means of representing probabilistic knowledge and providing web services such as plausible inference and Bayesian learning. The proposed framework is an initial step towards a more comprehensive effort focused on representing uncertainty in the Semantic Web.

A PR-OWL plugin for current OWL ontology editors is a priority for future efforts. The process of writing probabilistic ontologies can be greatly improved via automation of most of the steps in the ontology building, not only for defining MFrags to represent sets of related hypotheses, but also for consistency checking, reified relations and other tasks that demand unnecessary awareness of the inner workings of the present solution. Once implemented, such a plugin has the potential to make probabilistic ontologies a natural, powerful tool for the Semantic Web.

Finally, the most important requirement for adoption of a language is the standardization process. This process goes significantly beyond academic research and thus falls outside the scope of the present work. Nonetheless, we are confident of its feasibility, which we believe we have demonstrated in this effort, and of its desirability, given its potential to help solve many of the obstacles that stand in the way of realizing the W3C's vision for the Semantic Web.

References

1. Berners-Lee, T. and M. Fischetti, *Weaving the Web: the original design and ultimate destiny of the World Wide Web by its inventor*. 1st pbk. ed. 2000, New York: HarperCollins Publishers. ix, 246 p.
2. Patel-Schneider, P.F., P. Hayes, and I. Horrocks, *OWL Web ontology language - Semantics and abstract syntax*, in *W3C Recommendation*. 2004, World Wide Web Consortium: Boston, MA. p. W3C Recommendation.
3. Baader, F., et al., eds. *The Description Logic Handbook: Theory, Implementation and Applications*. First edition ed. 2003, Cambridge University Press: Cambridge, UK. 574.
4. Korb, K.B. and A.E. Nicholson, *Bayesian Artificial Intelligence*. Series in Computer Science and Data. 2003: Chapman & Hall/CRC. 392.
5. Laskey, K.B. and P.C.G. Costa, *Of Klingons and Starships: Bayesian Logic for the 23rd Century*, in *Uncertainty in Artificial Intelligence: Proceedings of the Twenty-first Conference*. 2005, AUAI Press: Edinburgh, Scotland.
6. Heckerman, D., A. Mamdani, and M.P. Wellman, *Real-world applications of Bayesian networks*. Communications of the ACM, 1995. **38**(3): p. 24-68.

7. Ding, Z. and Y. Peng. *A probabilistic extension to ontology language OWL*. in *37th Annual Hawaii International Conference on System Sciences (HICSS'04)*. 2004. Big Island, Hawaii.
8. Gu, T., P.H. Keng, and Z.D. Qing. *A Bayesian approach for dealing with uncertainty contexts*. in *Second International Conference on Pervasive Computing*. 2004. Vienna, Austria: Austrian Computer Society.
9. Calvanese, D. and G. De Giacomo, *Expressive Description Logics*, in *The Description Logic Handbook: Theory, Implementations and Applications*, F. Baader, et al., Editors. 2003, Cambridge University Press: Cambridge, UK. p. 184-225.
10. Pool, M. and J. Aikin, *KEEPER: and Protégé: An elicitation environment for Bayesian inference tools*, in *Workshop on Protégé and Reasoning held at the Seventh International Protégé Conference*. 2004: Bethesda, MD, USA.
11. Giugno, R. and T. Lukasiewicz. *P-SHOQ(D): A probabilistic extension of SHOQ(D) for probabilistic ontologies in the Semantic Web*. in *European Conference on Logics in Artificial Intelligence (JELIA 2002)*. 2002. Cosenza, Italy: Springer.
12. Daconta, M.C., L.J. Obrst, and K.T. Smith, *The Sematic Web: A guide to the future of XML, Web services, and knowledge management*. 2003, Indianapolis, IN: Wiley Publishing, Inc. 312.
13. Noy, N.F. and A. Rector, *Defining N-ary relations on the Semantic Web: Use with individuals*, in *W3C Working Draft*. 2004, World Wide Web Consortium: Boston, MA. p. W3C Working Draft.
14. Costa, P.C.G., *Bayesian Semantics for the Semantic Web*, in *Department of Systems Engineering and Operations Research*. 2005, George Mason University: Fairfax, VA, USA. p. 312. Available at www.pr-owl.org. Available at http://www.pr-owl.org.

Discovery and Uncertainty in Semantic Web Services

Francisco Martín-Recuerda[1] and Dave Robertson[2]

[1] Digital Enterprise Research Institute (DERI), Leopold-Franzens Universität Innsbruck, Technikerstraße 21a, 6020 Innsbruck, Austria
francisco.martin-recuerda@deri.org
http://www.deri.at/

[2] University of Edinburgh, School of Informatics, Centre for Intelligence Systems and their applications, Appleton Tower, Crichton Street, Edinburgh, EH89LE, Scotland
dr@inf.ed.ac.uk
http://www.cisa.informatics.ed.ac.uk/

Abstract. Although Semantic Web service discovery has been extensively studied in the literature ([7], [12], [15] and [10]), we are far from achieving an effective, complete and automated discovery process. Using the incidence calculus [4], a truth-functional probabilistic calculus, and a lightweight brokering mechanism [17], the article explores the suitability of integrating probabilistic reasoning in Semantic Web services environments. We show how the combination of relaxation of the matching process and evaluation of web service capabilities based on a previous historical record of successful executions enables new possibilities in service discovery.

1 Introduction

Discovery composition, invocation and interoperation are the core pillars of the deployment of Semantic Web services [9]. Discovery has been extensively studied in the literature ([7], [21], [12] and [15]). In a recent effort, the authors of [10] have focused on providing a coherent and formal model for Semantic Web services discovery.

Roughly speaking, the relaxation of the matching process between a goal (a functional description of objectives that clients want to achieve using web services) and web services capabilities (functional descriptions of a service) has been based on the following set of matching notions [10]: (i) exact-match, a goal and matched web service capabilities are the same; (ii) plug-in-match, a goal is subsumed by matched web service capabilities; (iii) subsume-match, matched web service capabilities are subsumed by a goal; (iv) intersection-match, a goal and matched web service capabilities have some elements in common; and (v) disjoint-match, a goal and matched web service capabilities does not follow any of the previous definitions. Although matching notions relax the identification of target web services, in a future scenario in which thousands of services can potentially fulfill (or partially fulfill) the objectives described in a goal, a fine-grained classification of matching notions may be necessary for improving the degree of automation of the discovery process. One possible approach is to identify a degree of matching inside of each matching notion. Thus, if we found one thousand web services that follow an intersection-match pattern, we need to distinguish which are the web services that are closer to the goal requested capability.

Brokers [5] bring another interesting approach to the problem of filtering the most promising web services. Brokers are intermediate systems between clients and service providers. They store web service capabilities and interfaces, execute matching processes for each goal that they have received, and manage the interaction between clients and selected web services. Thus after several interactions, brokers can gain valuable knowledge about which web services are providing a good service and which are not. A quality of service historical record can help in the identification of promising web services during the matching process executed in a broker.

Current Semantic Web services frameworks (e.g. OWL-S[1], WSMO[2] and Meteor-S[3]) use first order logic, description logics and logic programs to represent web service and goal capabilities and execute matching processes mostly based on subsumption checking or query-answering. In this article, we address the two problem areas raised above as part of a novel architecture for service matching, based on the incidence calculus. The incidence calculus [4] is a truth-functional probabilistic calculus in which the probabilities of composite formulae are computed from intersections and unions of the sets of worlds for which the atomic formulae hold true. Incidence Calculus can be easily integrated with other logic formalisms like propositional logic and logic programs and facilitate the implementation of a fine-grained matching mechanism based on probabilities and quality of service records.

The experiments were executed on a platform called F-X [17], a modular formal knowledge management system developed at University of Edinburgh. F-X has common roots with WSMO (both follows the main principles of UPML [3]), and can deal with WSMO/OWL-S ontologies and web services that fall into DLP fragment [8]. We will show how to specify service capabilities in F-Broker, and how incidence calculus can be nicely integrated

The paper is structured as follows: section 2 introduces semantic web services, F-X and Incidence Calculus. In section 3, the key implementation efforts are described, and testing results are discussed. Section 4 provides a short review of related work on probabilistic logic in the Semantic Web. Finally, conclusions and future work are included in section 5.

2 Preliminaries

Commonly in a Virtual Travel Agency scenario, customers require services in terms of goals (for instance, "*I want to book the cheapest flight and hotel available. The destination is Galway and I want to go on the 4th of November and back to San Francisco on the 9th of November*"). Airline companies and hotels provide services

[1] http://www.daml.org/services/owl-s/

[2] http://www.wsmo.org/

[3] http://lsdis.cs.uga.edu/projects/meteor-s/

("*to book a flight please provides: origin, destination, departure date, return date, valid passport id and credit-card*"). The broker is the *virtual travel agency* that stores service descriptions related with hotel and flights booking and attend requests from customers. We will show in this section how F-X can become in an efficient virtual travel agency for representing, storing and matching services. First we will introduce F-Broker, the broker component, and then we will describe incidence calculus and how this formalism can be integrated in F-Broker to improve its matching capabilities.

2.1 F-Broker

F-Broker [17] is an automated broker mechanism of F-X with the responsibility to identify the assemblies of knowledge components appropriate to a task we wish to achieve. This information is specified using F-Comp. In a multi-agent environment, agents advertise their *competences* (or capabilities, defined in the knowledge components they contain) simply by sending these to F-Broker, which records the competences and the agents who claim to be able to supply them.

When other agent sends a query, the broker processes it, and constructs an internal description, *brokerable structure*, based in the competences that previously it recorded which describes how the query might be answered. In the final stage the broker translates its brokerable structure into a sequence of *performative statements* describing the messages that will be necessary to establish a communication with the agents that can attend the query. The broker manages the communication between agents (request and providers) sending and receiving messages which the appropriate information to response the query [17].

[17] describes how capabilities and related brokerable structures are represented in previous versions of F-X. Four forms of capability, *C*, each of which is implemented within the expression `cap(K, C)` , denoting that the agent named `K` can deliver capability, `C` in at least one instance or, if not, will signal failure. Valid options for `C` are [17]:

- A **unit goal** of the form $P(A_1,\ldots,A_n)$, where `P` is a predicate name and $A_1,\ldots, A_n$ are its arguments.
- A **conjunctive goal** of the form $(C1\wedge\ldots\wedge Cm)$, where each C_i is a unit goal or a set expression.
- A **set expression** of the form `setof(X,C,S)` , where `C` is either a unit goal or a conjunctive goal; `X` is a tuple of variables appearing in `C`; and `S` is a set of instances of those tuples which satisfy `C`.
- A **conditional goal** of the form $C_c \leftarrow C_p$, where C_c is a unit goal which the agent, *K*, will attempt to satisfy (but will not guarantee to satisfy) if the condition, C_p , is satisfied. C_p is either a unit goal or a conjunctive goal.

Although for simplicity, we will use this version of the capability language, in later versions of FX, capabilities are represented following the next pattern:

```
  service(Agent, Uri, Ontology, [Service1:-Preconditions1, In-
puts1, Outputs1,Externals1], [...],..., [...]).
```

A simple brokerable structure has the form c(K, C), where K is the name of the agent which should be able to deliver the capability and C is a description of the sources of the capability. C can be in any of the following forms [17]:

- A capability available directly from K.
- A term of the form c(K, dq(Q,QC)), where Q is a capability obtainable from K conditional on its other capabilities and QC describes how these capabilities are obtained.
- A term of the form c(K, pdq(Q,QC,QP)), where Q is a capability obtainable from K conditional on its other capabilities and on capabilities external to K, and QC and QP describe how these internal and external capabilities (respectively) are obtained.
- A term of the form c(conj, co(CQ1,CQ2)), where CQ1 and CQ2 are two capability structures which must jointly be satisfied.
- A term of the form c(K, cn(Q, G, c(K1,Q1))), where K1 is the name of an agent different from K which allows capability structure Q to be delivered in combination with capability structure Q1 provided that the correspondence constraints given by G are satisfiable.

Given a query posed by a client, a *broker* tries to find all the possible ways in which agents which have advertised their capabilities might be contacted in order to satisfy that query. It is necessary a formal representation of this sort of combination of capabilities, for which we use what we call a brokerage structure, of the form c(K, C), where K is the name of the agent which should be able to deliver the capability and C is a description of the sources of the capability. C can be in any of the following forms [17][4]:

```
broker(Q,c(K,Q))
  ←cap(K,Q).
broker(Q, c(K, dq(Q,QC)))
  ←cap(K, (Q←C)) ∧
    broker(C,QC).
broker(Q, c(K1,pdq(Q,QC,QP)))
  ←p_cap(K1, (Q←C), P) ∧
    broker(C,QC) ∧
    e_broker(P,K1,QP).
broker((Q1,Q2), c(conj,
co(CQ1,CQ2)))
  ←broker(Q1,CQ1) ∧
    broker(Q2,CQ2).
broker(Q2, c(K2, cn(Q2, G,
c(K1,BQ))))
  ←corr(K1,Q1,K2,Q2,G) ∧
    Broker(Q1, c(K1,BQ)).
```

```
e_broker(Q, Kn, c(K,Q))
  ←cap(K,Q) ∧ not(K=Kn).
e_broker(Q, Kn, c(K, dq(Q,QC)))
  ←cap(K, (Q←C)) ∧ not(K=Kn) ∧
    broker(C,QC).
e_broker(Q, Kn, c(K1,
pdq(Q,QC,QP)))
  ←p_cap(K1, (Q←C), P) ∧
    not(K1=Kn) ∧ broker(C,QC) ∧
    e_broker(P,K1,QP).
e_broker((Q1,Q2), Kn, c(conj,
co(CQ1, CQ2)))
  ←e_broker(Q1,Kn,CQ1) ∧
    e_broker(Q2,Kn,CQ2).
e_broker(Q2, Kn, c(Kn, cn(Q2, G,
c(K1,BQ))))
  ←corr(K1,Q1,Kn,Q2,G) ∧
    broker(Q1, c(K1,BQ)).
```

[4] "corr" represents a correspondence, the equivalent of a bridge in UPML [3].

2.3 Incidence Calculus

Bundy [4] demonstrated that purely numeric probabilistic formalism can derive into contradictory results during the calculation of an uncertainty measure of complex formula. The key result of his analysis is that in general `P(A∧B)•P(A)*P(B)`.

Incidence Calculus [4] reviews the notions of probability theory and introduces an important novelty: "*the probability of a sentence is based on a sample space of elements. Each element defines a situation in a possible world where a sentence can be true or false. The sample space,* $\top$*, contains an exhaustive and disjoint set of elements that for computational reasons should be finite*".

The incidence of a sentence A, i(A), is the subset of W in which sentence A is true. The dependence or independence of two sentences, A and B, is defined by the amount of common points of the result of the intersection between their incidences, $i(A) \cap i(B)$.

The axioms of Incidence Calculus [4] associate a set of theoretic function with each connective, propositional constant and quantifier of Predicate (Propositional) Logic so that the incidence of a complex sentence can be calculated from the incidences of its sub-sentences. The probabilities of composite formulae are computed from intersections and unions of the sets of worlds for which the atomic formulae hold true. Bundy called the resulting system Predicate (Propositional) Incidence Logic [4]:

$i(\top) = \{\}$	$i(\bot) = \{\}$
$i(A) = i(A)$	$i(\neg A) = i(\top) \backslash i(A)$
$i(A \wedge B) = i(A) \cap i(B)$	$i(A \vee B) = i(A) \cup i(B)$
$i(A \rightarrow B) = i(\neg A \vee B) = (i(\top) \backslash i(A)) \cup i(B)$	

Thus, probabilities are calculated in the following way [4]:

$P(\top) = \lvert i(\top) \rvert = 1$	$P(\bot) = \lvert i(\bot) \rvert = 0$
$P(A) = \lvert i(A) \rvert / \lvert i(\top) \rvert$	$P(\neg A) = 1 - \lvert i(A) \rvert / \lvert i(\top) \rvert$
$P(A \wedge B) = \lvert i(A) \cap i(B) \rvert / \lvert i(\top) \rvert$	
$P(A \vee B) = (\lvert i(A) \cup i(B) \rvert - \lvert i(A) \cap i(B) \rvert) / \lvert i(\top) \rvert$	
$P(A \mid B) = \lvert i(A) \cap i(B) \rvert / \lvert i(B) \rvert$	

As an illustration, consider the following set of incidences describing the weather of a given week adopted from [4]:

Suppose there are two propositions, P={rainy, windy} and seven possible worlds, $\top$ ={sunday, monday, tuesday, wednesday, thursday, friday, saturday}. Suppose that each possible world is equally probable (i.e. 1/7), and we learn that rainy is true in four possible worlds (friday, saturday, sunday and monday) and windy is true in three possible worlds (Monday, wednesday and Friday). Therefore, we can derivate the following incidence sets [4]:

i(rainy) = {friday, saturday, sunday, monday}
i(windy)= {monday,wednesday, friday}
i(windy∧rainy)= {monday, friday}

Moreover, we can calculate their probabilities in the following way:

P(rainy) = |i(rainy)| / |i(T)|=4/7
P(windy) = |i(windy)| / |i(T)|=3/7
P(windy∧rainy)= | i(windy)∩i(rainy)| / |i(T)|=2/7

2.3 Travel Agency example, writing capabilities in F-Broker

For simplicity we will use the capability language of an earlier version of F-Broker presented in [17]. We extend the capability language to store in a list the number of incidences in which each atomic capability was execute successfully (a client used this service for a given goal). Initially the set of incidences is empty and after several computations the broker is populating the sets of incidences according with the results in the requests attended. For our traveling scenario capabilities, we can model the services related with an airline company in the following way:

```
n_requests = [1,2,3,4,5, … , 320].

p_capability(airline_aa, ((book_flight(Person, Flight, Ori-
gin, Destination, DepartureDate, ArrivalDate, PurchaseOrder,
Price, Currency, PaymentMethod) :- flight(Flight, Origin,
Destination, DepartureDate, ArrivalDate, Price, Currency)),
pay_order(Person, Nationality, PurchaseOrder, Price, Cur-
rency, PaymentMethod))).

capability(airline_aa, (flight(Flight, Origin, Destination,
DepartureDate, ArrivalDate, Price, Currency), [3,4,5, … ,
301]).

capability(airline_ib, (flight(Flight, Origin, Destination,
DepartureDate, ArrivalDate, Price, Currency), [1,2, … ,
319]).

capability(airline_ba, (flight(Flight, Origin, Destination,
DepartureDate, ArrivalDate, Price, Currency) [6,7, … , 318).

p_capability(financial_vs, pay_order(Person, PurchaseOrder,
Price, Currency, PaymentMethod):-
has_money(Person,Price,Currency, PaymentMethod),
has_passport(Person, Nationality))).

capability(financial_vs,, has_money(Person,Price,Currency,
PaymentMethod), [2,3,4, … , 315]).

capability(financial_ms,, has_money(Person,Price,Currency,
PaymentMethod), [5,6 … , 320]).

capability(financial_amex,, has_money(Person,Price,Currency,
PaymentMethod)[100,105, …, 255]).

capability(police, has_passport(Person, Nationality), [3,4,5,
… , 301]).
```

...

3 Implementation and Results

We present a set of extensions in F-X to allow the system to deal with many OWL-S service profiles, take advantage of a probabilistic mechanism based on Incidence Calculus and relax the matching process.

3.1 From Description Logics to Description Logic Programs.

One of the objectives of the implementation was to test F-Broker with real examples of Semantic Web Services descriptions and also to integrate it in an industrial standard in order to find possible business applications. Many web services are annotated using DAML-S Service Profile descriptions. So we thought that it could be a good idea to provide a translator that semi-automatically converts services descriptions from DAML-S into F-Broker Service Description Language (SDL). One of the difficulties is how to translate DL logical statements into Prolog statements.

Description Logic Programs (DLP)[8], is an expressive fragment of the intersection of Description Logics (DL) [2] and Logic Programs (LP) [13]. An important result of the development of this formalism is DLP-fusion, a bidirectional translation of premises and inferences from DLP fragment of DL to LP, and vice versa from DLP fragment of LP to DL that allows Prolog to describe on expressive subset of DL. The implementation of DLP-Fusion in Prolog is straightforward [14] and with this translator F-Broker is able to import and export knowledge represented using Description Logics.

3.2 Extending matching algorithm

This section describes the necessary extensions to the matching algorithm of F-Broker in order to incorporate subsumption reasoning, matching notions (exact, plug-in, subsume, intersection and disjoint), a fine-grained degree of matching for some of these matching notions, and finally a evaluation algorithm based on historical records. We follow a bottom-up approach in which any new functionality is tested before we continue with the implementations of new refinements.

Subsumption reasoning. A Meta-interpreter for a language is an interpreter for the language written in the language itself [20]. Meta-interpreters are powerful tools that were widely used for implementing the inference engines of many expert systems. Using these features the programmers can modify the behaviour of the interpreter of the language. Goal reduction is the best known and most widely used meta–interpreter that in Prolog is called *Vanilla* [20]. *Vanilla* does not support *subsumption*. So, the first step during the implementation process was the integration of substitution of vanilla meta-interpreter by the simple subsumption meta-interpreter. The integration of the subsumption mechanism with the brokering algorithm is very simple. It is only to add a clause subs in any of the brokerable predicates that compound the brokering algorithm for subsumption checking of terms:

```
brokerable(Q, c(S,Q)) :-
capability(S, Q1),
subs(Q1,Q).
```

Matching notions. The algorithm that evaluates the degree of matching basically compares two lists of terms that belong to a web service capability and a goal, verifies the number of common and no common terms, determines the appropriate notion of matching following the previous classification and returns a value with the notion of matching identified. According to the view described in [10], abstract services and goals are both represented as sets of objects during the service discovery step. Thus, the calculation of the notion of match can be naturally calculated using incidence calculus. The implementation is also simple. We substitute the subsumption clause in the brokerable predicates implemented before for a new clause that call a new algorithm that evaluates and return the notion of match between a capability and goal:

```
brokerable(Q, c(S,Q,Nmatch)) :-
capability(S, Q1),
matchingnotion(Q1,Q,Nmatch),
Nmatch<>"disjoint".
```

Instead of carrying out strings like “disjoint” or “exact”, it should be interesting to carry numeric values that can be reused for the calculation of a joint probability of several composed services.

Degree of matching notion. The previous algorithm can be improved by using a degree of matching that qualified the goodness of the matching notion identified. To do this, we include a new return variable in the matchingnotion predicate with the value that the incidence calculus algorithm calculates during the evaluation of common terms between capability and goal.

```
brokerable(Q, c(S,Q,Nmatch, Dmatch)) :-
capability(S, Q1),
matchingnotion(Q1,Q,Nmatch, Dmatch),
Nmatch<>"disjoint".
```

Evaluation of historical records. The proposal described in the current section focus the evaluation of the brokerable structures according to an historical record of previous goals. Associated with any atomic service capability there is a list of successful previous goals. This notion of a set of points (previous goals) fits perfectly with the probabilistic mechanism Incidence Calculus introduced in the previous section. In this case, the implementation requires the modification the atomic capabilities that have to maintain a list of values:

```
brokerable(Q, c(S, Q, L)) :-
capability(S, Q1, L),
```

A predicate called *evaluate* finds all the possible broker structures that can satisfied a request and evaluate the different structures according with the information of the history record. During the interaction with the client, the broker should modify the set of previous request of the service that successfully attend the demand of the client:

```
|?- evaluate(time(T), L).
L = [c(sd,time( A),[1,2]),2/4] ?
yes
```

3.3 Discussion

The extended version of F-Broker was tested with a modified version of the ecologic knowledge base [19] and slightly adapted versions of several web services examples from DAML[5], Mindswap[6] and Carnegie-Mellon[7]. [14] shows that the use of incidence calculus does not make significantly worse the performances of the broker with respect to the original version of F-Broker, and the relaxation of the matching process and the filtering of services based on a list of previous experiences of goals improve the matching abilities of the matching algorithm.

In [11] the use of incidence calculus was tested with a more advanced version of F-Broker that includes a lightweight coordination calculus (LCC) [16], a method for specifying agent interaction protocols. Lambert and Robertson use incidence calculus for the evaluation of services based on an historical record. The use of incidence calculus clearly helps to identify most promising services and thus satisfied client goals more efficiently.

[14] identified an important limitation of the use of incidence calculus to evaluate web services based on an historical record of previous goals. This is the incapacity of the system to handle the changes that the environment undergoes in a specific periods of time. For instance, the provider of a service with a large and excellent history record can fall. Any request of the clients that asks for this service will be processed by the broker and the answer will include the service that the provider cannot supply. After many requests another service could overcome the re-cord of the unavailable service, but before this moment the broker will try to execute the wrong service.

4 Related Work

The use of probabilistic logic in the context of the Semantic Web has not been explored in detail. Even the inventor of the Semantic Web, Sir Tim Berners-Lee, mentioned during the dev day lunchtime session at WWW2004 conference[8], that the Se-

[5] http:// www .daml.org/services/examples.html
[6] http://www.mindswap.org/2002/services/
[7] http://www. daml.ri.cmu.edu/ont/TaskModeler/TMont-index.html# Request Realtor1
[8] http://esw.w3.org/mt/esw/archives/000055.html

mantic Web stack does not need a representation of uncertainty. The first serious attempt to incorporate probabilistic reasoning in the Semantic Web was done with P-SHOQ[18]. Unfortunately, this work was not taken into consideration by the Semantic Web Community. A detailed description of an early version of this work can be found in my master thesis, "*Dealing with uncertainty in semantic web services*" [14]. This work was the first attempt to incorporate incidence calculus in a broker for semantic web services. [11] based on this previous experience incorporates the use of incidence calculus in an advance version of F-Broker that includes a lightweight coordination calculus (LCC) [16], a method for specifying agent interaction protocols.

5 Conclusions and Future Work

The relaxation of the matching process and the evaluation web service capabilities based on a previous historical record of successful executions show the feasibility of the use of probabilistic logic in Semantic Web services. Uncertainty is present in functional aspects of Web Services like discovery, composition, interoperation, mediation, monitoring and compensation [1]. In this paper, we focused only in discovery, and in [14], composition is also studied.

Incidence calculus was an excellent choice because its simplicity, rigor and compatibility with other classical logic formalisms. F-Broker provides an excellent test platform for the evaluation of incidence calculus in semantic web services. Although simple, F-Broker provides all basic functionality of a broker and allows the composition of web services capabilities and the execution of services based on an elementary vocabulary inspired in KQML. The code is very compact and clean, and new extensions are easily to include.

Future work will concentrate in the migration of the test platform to more realistic scenarios and the evaluation of other probabilistic logic formalism that combines logic programming with description logics.

Acknowledgements

This work has been partially supported by the SFI (Science Funds Ireland) under the DERI-Lion project, and the European Commission under the project Knowledge Web.

References

1. S. Arroyo and D. Fensel (2004). The Semantic Web Service Usage Process. No published.
2. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider. The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2003.

3. V. R. Benjamins, D. Fensel, E. Motta, S. Decker, M. Gaspari, R. Groenboom, W. Grosso, M. Musen, E. Plaza, G. Schreiber, R. Studer, and B. Wielinga. The Unified Problem-solving Method Development Language UPML, February 1999. Esprit Project 27169 IBROW 3 (An Intelligent Brokering Service for Knowledge-Component Reuse on the World-Wide Web.
4. A. Bundy. Incidence Calculus. In Encyclopedia of Artificial Intelligence, pages 663–668. 1992.
5. K. Decker and K. Sycara. Middle-Agents for the Internet. In Proceedings of ICJCAI-97, January 1997.
6. T. Finin, Y. Labrou, and J. Mayfield. KQML as a Agent Communication Language. Sofware Agents, 1997. J.M. Bredshaw, AAAI Press/MIT Press.
7. J. Gonzalez-Castillo, D. Trastour, and C. Bartolini. Description logics for matchmaking of services. In KI-2001 Workshop on Applications of Description Logics, September 2001.
8. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description Logic Programs: Combining Logic Programs with Description Logics. In Proc. of the Twelfth International World Wide Web Conference (WWW 2003), pages 48–57, 2003.
9. H. Haas and A. Brown (2004). Web Services Glossary. 2004. http://www.w3.org/TR/ws-gloss/
10. U. Keller, R. Lara, and A. Polleres (eds.). WSMO Web Service Discovery. Technical report, DERI, November 2004.
11. D. Lambert and D. Robertson. Matchmaking and Brokering Multi-Party Interactions Using Historical Performance Data. To appear in the Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems, Utrecht 2005.
12. L. Li and I. Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. In WWW'03, Budapest, Hungary, May 2003.
13. J.W. Lloyd. Foundations of logic programming (second extended edition). Springer series in symbolic computation. Springer-Verlag, New York, 1987.
14. F. Martin-Recuerda. Dealing with uncertainty in Semantic Web services. MSc Thesis. University of Edinburgh. 2003.
15. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic Matching of Web Service Capabilities. In ISWC, pages 333–347. Springer Verlag, 2002.
16. Robertson, D.: A lightweight method for coordination of agent oriented web services. In: Proceedings of the 2004 AAAI Spring Symposium on Semantic Web Services, California, USA (2004)
17. D. Robertson. F-X: A Formal Knowledge Management System. (unpublished), August 2001.
18. Thomas Lukasiewicz and Rosalba Giugno. P-SHOQ (Dn) : A Probabilistic Extension of SHOQ(Dn) for Probabilistic Ontologies in the Semantic Web. Technical report. Institut f¨ur Informations systeme, Technische Universität Wien,
April 2002. Technical Report Nr. 1843-02-06.
19. D. Robertson, A. Bundy, R. Muetzelfeldt, M. Haggith, and M Uschold. Eco-Logic: Logic-Based Approaches to Ecological Modeling. MIT Press (Logic Programming Series), 1991.
20. L. Sterling and E. Shapiro. The Art of Prolog: Advanced Programming Techniques, 2nd Edition. MIT Press, 1994.
21. K. Sycara, S. Widoff, M. Klusch, and J. Lu. LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. Autonomous Agents and Multi-Agent Systems, pages 173–203, 2002.

Ontology Learning and Reasoning — Dealing with Uncertainty and Inconsistency

Peter Haase, Johanna Völker

Institute AIFB, University of Karlsruhe, Germany
{pha,jvo}@aifb.uni-karlsruhe.de

Abstract. Ontology Learning from text aims at generating domain ontologies from textual resources by applying natural language processing and machine learning techniques. It is inherent in the ontology learning process that the acquired ontologies represent uncertain and possibly contradicting knowledge. From a logical perspective, the learned ontologies are potentially inconsistent knowledge bases that thus do not allow meaningful reasoning directly. In this paper we present an approach to generate consistent OWL ontologies from learned ontology models by taking the uncertainty of the knowledge into account. We further present evaluation results from experiments with ontologies learned from a Digital Library.

1 Introduction

Ontology Learning from text aims at generating domain ontologies from a given collection of textual resources by applying natural language processing and machine learning techniques. Due to an increasing demand for efficient support in knowledge acquisition, a number of tools for automatic or semi-automatic ontology learning have been developed during the last years. Common to all of them is the need for handling the uncertainty which is inherent in any kind of knowledge acquisition process. Moreover, ontology-based applications which rely on learned ontologies have to face the challenge of reasoning with large amounts of imperfect information resulting from automatic ontology generation systems.

Causes for the imperfection of information can be found thrice. According to [1] imperfection can be due to *imprecision*, *inconsistency* or *uncertainty*. Imprecision and inconsistency are properties of the information itself - either more than one world (in the case of ambiguous, vague or approximate information) or no world (if contradictory conclusions can be derived from the information) is compatible with the given information. Uncertainty means that an agent, i.e. a computer or a human, has only partial knowledge about the truth value of a given piece of information. One can distinguish between objective and subjective uncertainty. Whereas objective uncertainty relates to randomness referring to the propensity or disposition of something to be true, subjective uncertainty depends on an agent's opinion about the truth value of information. In particular, the agent can consider information as unreliable or irrelevant.

In ontology learning, (subjective) uncertainty is the most prominent form of imperfection. This is due to the fact that the results of the different algorithms have to be considered as unreliable or irrelevant due to imprecision and errors introduced during the ontology generation process. There exist different approaches for the representation of uncertainty: Uncertainty can for example be represented as part of the learned ontologies, e.g. using probabilistic extensions to the target knowledge representation formalism, or at a meta-level as application-specific information associated with the learned structures.

In Text2Onto [7], a framework for ontology learning and data-driven ontology evolution, we follow a slightly different approach: In a first step, we apply ontology learning algorithms to generate ontologies based on a *Learned Ontology Model* (LOM), which is independent of a concrete ontology representation language. In the LOM, we represent uncertainty as annotations capturing the confidence about the correctness of the ontology elements. Most importantly, since the LOM does not have any logical semantics, in this step we do not have to consider logical inconsistencies which are often introduced during the ontology learning process. In a second step, we transform the LOM model to a standard logic-based ontology language, in order to be able to apply standard reasoning over the learned ontologies (e.g. for query answering). In our work we build on the OWL ontology language, as it is now the standard for representing ontologies on the web, and – with its grounding in Description Logics – reasoning with OWL ontologies is very well understood and tractable. Because of the uncertain and thus potentially contradicting information in the LOM models, a naive translation of the LOM model to OWL would result in highly inconsistent ontologies, which do not allow meaningful reasoning. We therefore make use of the confidence annotations of the LOM to guide the transformation process.

An obvious alternative approach to dealing with potential inconsistencies is to prohibit primitives that introduce inconsistencies in the first place (e.g. negation, disjointness). However, as shown in [21], semantically rich primitives such as disjointness of concepts can be used for effective semantic clarification in ontologies and thus enables to draw more meaningful conclusions.

As a main contribution of this work we present a transformation that results in an ontology that is (1) consistent and (2) "most likely correct", relying on the certainty information of the LOM model. The transformation is based on the notion of an evaluation function that measures the quality of ontologies with respect to given criteria, i.e. in our case consistency and certainty.

Application Scenario Intelligent search over document corpora in Digital Libraries is one application scenario that shows the immediate benefit of the ability to reason over ontologies automatically learned from text. While search in Digital Libraries nowadays is restricted to structured queries against the bibliographic metadata (author, title, etc.) and to unstructured keyword-based queries over the full text documents, complex queries that involve reasoning over the knowledge present in the documents are not possible. Ontology learning enables obtaining the required formal representations of the knowledge available in the corpus to be able to support such advanced types of search. This application scenario is the subject of a case study within the Digital Library of BT (British Telecom) as part of the SEKT[1] project. One of the key elements

[1] http://www.sekt-project.com/

of the case study is to automatically learn ontologies to enhance search and finally be able support queries of the kind “Find knowledge management applications that support Peer-to-Peer knowledge sharing.” To validate the work the presented in this paper, we performed experiments with data from the BT Digital Library.

Overview of the paper The rest of the paper is organized as follows. In Section 2 we recapitulate the foundations of the OWL ontology language, query answering with OWL ontologies and the role of logical inconsistencies. In Section 3 we introduce the Learned Ontology Model (LOM). In Section 4 we discuss the transformation of LOM models to OWL ontologies. We discuss experimental results in Section 5 and present related work in Section 6 before we conclude in Section 7.

2 Reasoning with OWL

In this section we provide on overview of the OWL ontology language (specifically OWL-DL), typical reasoning tasks and show why standard reasoning with inconsistent ontologies does not yield meaningful results.

OWL-DL is a syntactic variant of the $\mathcal{SHOIN}(\mathbf{D})$ description logic [15]. Hence, although several syntaxes for OWL-DL exist, in this paper we use the traditional description logic notation since it is more compact.

Definition 1 (Ontology). *We use a datatype theory* $\mathbf{D}$*, a set of concept names* N_C*, sets of abstract and concrete individuals* N_{I_a} *and* N_{I_c}*, respectively, and sets of abstract and concrete role names* N_{R_a} *and* N_{R_c}*, respectively.*

The set of $\mathcal{SHOIN}(\mathbf{D})$ concepts *is defined by the following syntactic rules, where* A *is an atomic concept,* R *is an abstract role,* S *is an abstract simple role,* $T_{(i)}$ *are concrete roles,* d *is a concrete domain predicate,* a_i *and* c_i *are abstract and concrete individuals, respectively, and* n *is a non-negative integer:*

$$\begin{aligned} C \rightarrow\ & A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C \mid \geq n\, S \mid \leq n\, S \mid \{a_1, \ldots, a_n\} \mid \\ & \mid \geq n T \mid \leq n T \mid \exists T_1, \ldots, T_n.D \mid \forall T_1, \ldots, T_n.D \\ D \rightarrow\ & d \mid \{c_1, \ldots, c_n\} \end{aligned}$$

A $\mathcal{SHOIN}(\mathbf{D})$ *ontology* O *is a finite set of axioms of the form concept inclusion axioms* $C \sqsubseteq D$*, for* C *and* D *concepts, transitivity axioms* $\mathsf{Trans}(R)$*, role inclusion axioms* $R \sqsubseteq S$ *and* $T \sqsubseteq U$*, concept assertions* $C(a)$*, role assertions* $R(a, b)$*, individual (in)equalities* $a \approx b$*, and* $a \not\approx b$*, respectively.*

The semantics of the $\mathcal{SHOIN}(\mathbf{D})$ description logic is defined via a model-theoretic semantics, which explicates the relationship between the language syntax and the model of a domain: An interpretation $I = (\triangle^I, \cdot^I)$ consists of a domain set $\triangle^I$, disjoint from the datatype domain $\triangle^I_{\mathbf{D}}$, and an interpretation function $\cdot^I$, which maps from individuals, concepts and roles to elements of the domain, subsets of the domain and binary relations on the domain, respectively[2]. An interpretation $\mathcal{I}$ satisfies an ontology O, if it

[2] For a complete definition of the interpretation, we refer the reader to [15].

satisfies each axiom in O. Axioms thus result in semantic conditions on the interpretations. Consequently, contradicting axioms will allow no possible interpretations. This leads us to the definition of a consistent ontology:

Definition 2 (Consistent Ontology). *An ontology O is consistent iff O is satisfiable, i.e. if O has a model.*

To be able to define queries against ontologies, we rely on the notion of entailment: We use $O \models \alpha$ to denote that the ontology O entails the axiom α (alternatively, we say that α is a consequence of the ontology O), iff α holds in any model in which O holds.

Definition 3 (Query and Query Answer). *A query with respect to an entailment relation $\models$ is a pair of an ontology O and an axiom α, written '$O \models \alpha$?'. An answer to a query '$O \models \alpha$?' is a value in the set $\{true, false\}$ as $O \models \alpha$ and $O \not\models \alpha$ respectively.*

Standard entailment as defined above is explosive, i.e. any axiom is a consequence of an inconsistent ontology. Namely, if an ontology O is not consistent, then for any axiom α, $O \models \alpha$. In other words, query answers for inconsistent ontologies are completely meaniningless, as for any query the query answer will be *true*. For a detailed discussion on inconsistencies in OWL ontologies, we refer the reader to [13].

3 LOM - A Learned Ontology Model

We believe, that linguistic evidence with respect to an ontology can be appropriately measured by ontology learning techniques which try to capture the ontological commitment in human language. Since ontology learning algorithms such as implemented in TextToOnto [7] consider the relation of individual ontology elements with the data the ontology has been engineered from, they allow to assess how well the ontology reflects the underlying corpus of data. This is especially relevant for an application scenario as introduced in Section 1, which involves question answering in the context of a Digital Library. In the following we describe the ontology model of Text2Onto and the ontology learning algorithms used in our approach.

A *Learned Ontology Model* (LOM) as used by Text2Onto is a collection of instantiated modeling primitives which are independent of a concrete ontology representation language. These primitives are defined in a declarative fashion which allows for translating the LOM into any knowledge representation language as long as the expressivity of the primitives does not exceed the expressivity of the target language. In Text2Onto we follow a translation-based approach to knowledge engineering. So called *ontology writers* are then responsible for translating instantiated modeling primitives into a specific target knowledge representation language. While a translation to various ontology languages is possible, in the scope of this paper, we focus on the translation to OWL ontologies. The modeling primitives we use in Text2Onto and their correspondences in the OWL ontology model are described by Table 1.

To capture contextual information about ontology elements, such as provenance and certainty in the learning process, we introduce the notion of *rating annotations*.

Modeling Primitive	Explanation	OWL
`concept`	A concept C. Example: *man*, *person*	C
`instance`	An instance a. Example: *John*, *Mary*	a
`subconcept-of`	Concept inheritance. Example: `subconcept-of`(*man*,*person*)	$C_1 \sqsubseteq C_2$
`instance-of`	Concept instantiation. Example: `instance-of`(*John*,*person*).	$C(a)$
`relation`	A relation R between C_1 and C_2. Example: `love`(*person*,*person*)	$C_1 \sqsubseteq \forall R.C_2$
`part-of`	Mereological part-whole relation between C_1 and C_2. Example: `part-of`(*wheel*,*car*)	*part-of*(C_1, C_2)
`equivalence`	Equivalence of concepts C_1 and C_2. Example: `equivalence`(*town*,*city*)	$C_1 \equiv C_2$
`equality`	Equality of instances a_1 and a_2. Example: `equality`(*UN*,*United Nations*)	$a_1 \approx a_2$
`disjointness`	Disjointness of concepts C_1 and C_2. Example: `disjointness`(*man*,*woman*)	$C_1 \sqsubseteq \neg C_2$

Table 1. LOM Modeling Primitives

Definition 4. *Let N denote the set of all possible ontology elements and $\mathcal{X}$ be a suitable representation of a context space, then an* ontology rating annotation *is a partial function $r : N \to \mathcal{X}$.*

In Text2Onto we use these rating annotations to model the certainty of the system about the correctness of a particular ontology element. In particular, we define a special ontology rating annotation

$$r_{conf} : N \to [0, 1]$$

to indicate how confident the system is about the correctness of an ontology element. The confidences are calculated based on different kinds of evidences provided by the ontology learning algorithms that indicate the correctness and the relevance of ontology elements for the domain in question. They can be considered as a corpus-based support for ontology elements.

Algorithms We now describe for each modeling primitive the algorithms used to learn corresponding instances thereof. In particular, we explain the way the confidence and relevance ratings for an instantiated modeling primitive are calculated.

Concepts and Instances Different term weighting measures are used to compute the relevance of a certain concept or instance with respect to the corpus: Relative Term Frequency (RTF), TFIDF, Entropy and the C-value/NC-value method in [17].

Subconcept-of Relations In order to learn subconcept-of relations, we have implemented a variety of different algorithms exploiting the hypernym structure of WordNet [11], matching Hearst patterns [14] in the corpus as well as in the WWW and applying linguistic heuristics mentioned in [24]. The resulting confidence values of these algorithms are then combined through combination strategies as described in [6].

Instance-of Relations In order to assign instances or named entities appearing in the corpus to a concept in the ontology Text2Onto relies on a similarity-based approach extracting context vectors for instances and concepts from the text collection and assigning instances to the concept corresponding to the vector with the highest similarity with respect to their own vector [8]. Alternatively, we also implemented a pattern-matching algorithm similar to the one used for discovering part-of relations.

General Relations To learn general relations, Text2Onto employs a shallow parsing strategy to extract subcategorization frames (e.g. `hit(subj,obj,pp(with))`, transitive + PP-complement) enriched with information about the frequency of the terms appearing as arguments [19]. These subcategorization frames are mapped to relations such as `hit`(*person*,*thing*) and `hit_with`(*person*,*object*). The confidence is estimated on the basis of the frequency of the subcategorization frame as well as of the frequency with which a certain term appears at the argument position. For the purpose of discovering **part-of relations** in the corpus, we developed regular expressions matching lexico-syntactic patterns as described in [5] and implemented an algorithm counting the occurrences of patterns indicating a part-of relation between two terms t_1 and t_2, i.e. part-of(t_1,t_2). The confidence is then calculated by dividing by the sum of occurrences of patterns in which t_1 appears as a part. The results are combined with confidences which can be acquired by consulting WordNet for mereological relations.

Equivalence and Equality Following the assumption that terms are similar to the extent to which they share similar syntactic contexts, we implemented algorithms calculating the similarity between terms on the basis of contextual features extracted from the corpus, whereby the context of a terms varies from simple word windows to linguistic features extracted with a shallow parser. This corpus-based similarity is then taken as the confidence for the equivalence of the corresponding concepts or instances.

Disjointness For the extraction of disjointness axioms we implemented a simple heuristic based on lexico-syntactic patterns. In particular, given an enumeration of noun phrases $NP_1, NP_2, ...(and|or)NP_n$ we conclude that the concepts $C_1, C_2, ...C_k$ denoted by these noun phrases are pairwise disjoint, where the confidence for the disjointness of two concepts is obtained from the number of evidences found for their disjointness in relation to the total number of evidences for the disjointness of these concepts with other concepts.

4 Transforming Learned Ontologies to OWL

In this section we discuss the transformation of learned ontologies as described in the previous section to OWL ontologies (c.f. Section 2). As mentioned before, a naive translation that simply disregards the certainty information (rating annotations) would result in a potentially highly inconsistent knowledge base that would not allow meaningful reasoning. The goal of the transformation therefore is to obtain an ontology that is (1) consistent (to allow meaningful reasoning), and (2) captures the most certain information while disregarding the potentially erronous information. In general, there may be many different consistent ontologies obtained from a LOM. The difficulty is to select the "best" ontology, i.e. the one that will result in most meaningful reasoning.

Evaluation Function In order to able to define what a "good" ontology for a particular context is, we need to be able to measure the quality of the ontology with respect to given set of criteria. We therefore define the notion of an *ontology evaluation function*.

Definition 5. *Let $\mathcal{O}$ be the set of possible ontologies, then an ontology evaluation function e is a function $e : \mathcal{O} \rightarrow [0, 1]$.*

Effectively, the evaluation function provides a total order over the space of possible ontologies and thus allows to compare given ontologies. Here it is important to note that the evaluation function can take the rating annotations into account and thus provides an evaluation measure for a given context. Using the evaluation function, we can define the problem of translating a given learned ontology LOM to a "discrete" and consistent OWL ontology as: $max_{O \subseteq LOM}\ e(O)$.
In other words, we try to find the best ontology O based on the knowledge in LOM that maximizes the evaluation function.

For our particular goal to obtain a consistent ontology capturing the most certain information, we can define an evaluation function as follows:

$$e_{certainty}(O) = \begin{cases} \max\left(\frac{\sum_{\alpha \in O} r_{conf}(\alpha) - t}{\|O\|}, 0\right) & \text{if } O \text{ is consistent} \\ 0 & \text{if } O \text{ is inconsistent} \end{cases} \tag{1}$$

Let us discuss the intuition behind this function. The basic idea is to maximize the certainty of the ontology based on the confidence of its individual axioms, as given by $r_{conf}(\alpha)$. The threshold t is introduced to "filter out" axioms with a confidence below a minimal value: Adding an axiom with a confidence below t will thus decrease the value of ontology. An inconsistent ontology is defined to have "no value".

In general, it will be hard to determine the optimal ontology that maximizes the evaluation function, as one theoretically would need to search entire space of possible consistent ontologies. However, in most cases it is not necessary to prove the optimality of an obtained solution, especially when considering that the rating annotations themselves are already somewhat imprecise. Instead it is possible to exploit heuristics to obtain a "fairly" optimal ontology.

We now outline an algorithm that exploits the behavior of the evaluation function and local characteristics of inconsistencies to maximize the value. It is based on the ideas of consistent ontology evolution as presented in [12]. Consistent ontology evolution ensures the consistency of ontologies when the ontology is changed by mapping consistency conditions that need to be satisfied to resolution functions that resolve introduced inconsistencies. The task of the resolution function consists of two main steps: (1) localizing the inconsistency and (2) generating additional changes that lead to another consistent state.

We treat the transformation of a LOM ontology to a consistent OWL ontology in a similar way as shown in Algorithm 1: Starting with an empty ontology O, we incrementally add all axioms from the learned ontology LOM whose confidence is equal to or greater than the threshold t. If adding the axioms leads to an inconsistent ontology, we localize the inconsistency by identifying a minimal inconsistent subontology. (For the details of this procedure, we refer the reader to [12]). An ontology O' is a minimal inconsistent subontology of O, if O' and every subontology of O' is consistent. Within this minimal inconsistent subontology we then identify the axiom that is most uncertain, i.e. has the lowest confidence value. This axiom will be removed from the ontology, thus resolving the inconsistency.

Algorithm 1 Algorithm for Transforming a LOM into a consistent OWL ontology

```
Require: A learned Ontology $LOM$
 1: $O := \emptyset$
 2: for all $\alpha \in LOM, r_{conf}(\alpha) \geq t$ do
 3:     $O := O \cup \{\alpha\}$
 4:     while $O$ is inconsistent do
 5:         $O' :=$ minimal_inconsistent_subontology$(O, \alpha)$
 6:         $\alpha^- := \alpha$
 7:         for all $\alpha' \in O'$ do
 8:             if $r_{conf}(\alpha') \leq r_{conf}(\alpha)$ then
 9:                 $\alpha^- := \alpha'$
10:             end if
11:         end for
12:         $O := O \setminus \{\alpha^-\}$
13:     end while
14: end for
```

5 Evaluation and Experimental Results

We have applied the approach presented in the previous chapter to ontologies learned from a corpus of 1700 abstracts (from documents about knowledge management) of the BT Digital Library. The learned ontology (LOM) consisted of 938 concepts and 125 instances. For the concepts, 406 subconcept-of relations and 2322 disjoint-concepts relations were identified. For the instances, 143 instance-of relations were obtained (as multiple instantiations is allowed).

For the transformation of the LOM ontology to a discrete OWL ontology, we applied the evaluation function and algorithms presented in the previous section. Here we performed an analysis of the influence of the threshold of uncertainty on the transformation. The results in Table 2 clearly show the connection between the level of uncertainty and inconsistency introduced:

Threshold t	# of Inconsistencies	# of Axioms in Result
0.1	40	1706
0.2	8	705
0.4	3	389
0.8	0	197

Table 2. Influence of certainty threshold t on transformation process

A low threshold t results in more uncertain information being allowed in the target ontology. As a result, the chances for inconsistencies increase. How to choose the "right" threshold t for the transformation process will very much depend on the application scenario, as it essentially means finding a trade-off between the amount of information learned and the confidence in the correctness of the learned information.

In the following we will discuss typical types of inconsistencies and present examples of such inconsistencies that were detected and resolved. The first type of inconsistency involves unsatisfiable concepts (often called incoherent concepts) in the $\mathcal{T}$-Box of the ontology. This can for example happen if two concepts are identified to be disjoint, but at the same time these concepts are in a subconcept-relation (either explicitly asserted or inferred). Interestingly, this type of inconsistency often occurred for concepts for which even for a domain expert the correct relationship is hard to identify, as the following example shows:

Example 1. The relationship between the concepts *Data*, *Information*, and *Knowledge* is a very subtle (often philosophical) one, for which one will encounter different definitions depending on the context. The (inconsistent) definitions learned from our data set stated that $Data$ is a subconcept of both $Information$ and $Knowledge$, while $Information$ and $Knowledge$ are disjoint concepts:

Axiom t	Confidence
$Data \sqsubseteq Information$	1.0
$Data \sqsubseteq Knowledge$	1.0
$Information \sqsubseteq \neg Knowledge$	0.7

The inconsistency was resolved by removing the disjointness axiom, as its confidence value was lowest.

The second type of inconsistencies involves $\mathcal{A}$-Box assertions. Here, typically instances were asserted to be instances of two concepts that were identified to be disjoint. We again present an example:

Example 2. Here $KaViDo$ was identified to be both an instance of $Application$ and a $Tool$ (based on the abstract of [23]), however, $Application$ and $Tool$ were learned to be disjoint concepts:

Axiom t	Confidence
$Application(kavido)$	0.46
$Tool(kavido)$	0.46
$Tool \sqsubseteq \neg Application$	0.3

This inconsistency was again resolved by removing the disjointness axiom.

Other types of inconsistencies involving, for example, domain and range restrictions were not considered in our current experiments, thus being left for future work. Nevertheless, this evaluation showed that inconsistency is an important issue in ontology learning.

6 Related Work

Since building an ontology for a huge amount of data is a difficult and time consuming task a number of tools such as TextToOnto [20], the ASIUM system [10], the Mo'k Workbench [3], OntoLearn [24] or OntoLT [4] have been developed in order to support the user in constructing ontologies from a given set of (textual) data. So far, none of these tools explicitly addresses the problem of uncertainty. Text2Onto implements the first approach towards integrating uncertainty into ontology learning. Obviously, the LOM of Text2Onto is not probabilistic in a strict mathematical sense. Nevertheless, several researchers have already addressed the issue of integrating and reasoning with probabilities in knowledge representation formalisms. [9] for example present a probabilistic extension of the Ontology Language OWL which relies on Bayesian Networks for reasoning. Other researchers have integrated probabilities into first-order logic [2] or description logics [18]. Fuzzy extensions of OWL have been proposed e.g. in [22].

The approach to dealing with inconsistencies presented in this work is based on the idea of obtaining a consistent ontology from a LOM to be then able to derive consistent query answers. A very related approach is that of reasoning with inconsistent ontologies. A typical technique is the selection of a consistent subontology for a given query, which yields a consistent query answer (c.f. [16]). The important question here is how to select the *right* subontology. While current techniques often rely on syntactic selection functions, it would also be possible to rely on the rating annotations available in the LOM to guide the selection function. Another related approach is that of diagnosis and repair of inconsistencies based on techniques such as pinpointing [21]. The pinpointing technique tries to identify and remove a *minimal* set of axioms (in terms of number of axioms) to obtain a consistent ontology, while we try to identify the *most certain* consistent ontology. As there are typically multiple possible pinpoints, a combination of pinpointing with the notion of certainty of our work is an interesting path to explore.

7 Conclusion and Future Work

Ontology learning is a promising technique for automated knowledge acquisition from text corpora. However, as we have shown, uncertainty and inconsistencies are issues that need to be dealt with in order to allow meaningful reasoning over the learned ontologies. In this paper we have presented how uncertainty can be represented in the Learned Ontology Model (LOM) and how such learned ontologies can be transformed to consistent OWL ontologies using the notion of an ontology evaluation function. Our experiments with ontologies learned from documents of a Digital Library show the feasibility and usefulness of the approach. An extensive evaluation will be performed as part of a case study within the SEKT project.

It is important to mention that confidence as generated by ontology learning algorithms represent a data-driven approach to the evaluation of ontologies. There are many other notions of ontology quality and consistency which could be used for the definition of an ontology evolution function. In particular, we will in the future integrate an automatic approach towards the formal evaluation of ontologies by means of the OntoClean methodology as presented in [25].

Acknowledgements Research reported in this paper has been financed by the EU in the IST project SEKT (IST-2003-506826) (`http://www.sekt-project.com/`).

References

1. P. Smets A. Motro. *Uncertainty Management In Information Systems*. Springer, 1997.
2. F. Bacchus. *Representing and Reasoning with Probabilistic Knowledge*. MIT Press, 1990.

3. G. Bisson, C. Nedellec, and L. Canamero. Designing clustering methods for ontology building - The Mo'K workbench. In *Proc. of the ECAI Ontology Learning WS*, 2000.
4. P. Buitelaar, D. Olejnik, and M. Sintek. OntoLT: A protégé plug-in for ontology extraction from text. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2003.
5. E. Charniak and M. Berland. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 57–64, 1999.
6. P. Cimiano, A. Pivk, L. Schmidt-Thieme, and S. Staab. Learning taxonomic relations from heterogeneous sources of evidence. In *Ontology Learning from Text: Methods, Applications and Evaluation*. IOS Press, 2005.
7. P. Cimiano and J. Völker. A framework for ontology learning and data-driven change discovery. In *Proc. of the NLDB'2005*, 2005.
8. P. Cimiano and J. Völker. Towards large-scale, open-domain and ontology-based named entity classification. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP'05)*, SEP 2005.
9. Z. Ding and Y. Peng. A probabilistic extension to ontology language OWL. In *Proceedings of the 37th Hawaii International Conference on System Sciences*, 2004.
10. D. Faure and C. Nedellec. A corpus-based conceptual clustering method for verb frames and ontology. In *Proceedings of the LREC Workshop on Adapting lexical and corpus resources to sublanguages and applications*, 1998.
11. C. Fellbaum. *WordNet, an electronic lexical database*. MIT Press, 1998.
12. P. Haase and L. Stojanovic. Consistent evolution of OWL ontologies. In *Proceedings of the Second European Semantic Web Conference, Heraklion, Greece, 2005*, MAY 2005.
13. P. Haase, F. van Harmelen, Z. Huang, H. Stuckenschmidt, and Y. Sure. A framework for handling inconsistency in changing ontologies. In *Proc. of the Fourth International Semantic Web Conference (ISWC'05)*, NOV 2005.
14. M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545, 1992.
15. I. Horrocks and P. F. Patel-Schneider. Reducing OWL Entailment to Description Logic Satisfiability. *Journal of Web Semantics*, 1(4), 2004.
16. Z. Huang, F. van Harmelen, and A. ten Teije. Reasoning with inconsistent ontologies. In *Proceedings of IJCAI'05*, August 2005.
17. J. Tsuji K. Frantzi, S. Ananiadou. The c-value/nc-value method of automatic recognition for multi -word terms. In *Proceedings of the ECDL*, pages 585–604, 1998.
18. D. Koller, A. Levy, and A. Pfeffer. P-classic: A tractable probabilistic description logic. In *Proceedings of AAAI-97*, pages 390–397, 1997.
19. A. Maedche and S. Staab. Discovering conceptual relations from text. In W. Horn, editor, *Proceedings of the 14th ECAI'2000*, 2000.
20. A. Maedche and S. Staab. Ontology learning. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, pages 173–189. Springer, 2004.
21. S. Schlobach. Debugging and semantic clarification by pinpointing. In *Proceedings of the Second European Semantic Web Conference, Heraklion, Greece, 2005*, pages 226–240, 2005.
22. U. Straccia. Towards a fuzzy description logic for the semantic web (preliminary report). In *Proceedings of the Second European Semantic Web Conference, 2005*, pages 167–181, 2005.
23. O. Tamine and R. Dillmann. Kavido: a web-based system for collaborative research and development processes. *Computers in Industry*, 52(1):29–45, 2003.
24. P. Velardi, R. Navigli, A. Cuchiarelli, and F. Neri. Evaluation of ontolearn, a methodology for automatic population of domain ontologies. In *Ontology Learning from Text: Methods, Applications and Evaluation*. IOS Press, 2005.
25. J. Völker, D. Vrandecic, and Y. Sure. Automatic evaluation of ontologies (AEON). In *Proc. of the Fourth International Semantic Web Conference (ISWC'05)*, NOV 2005.

Controlling Ontology Extension by Uncertain Concepts Through Cognitive Entropy*

Joaquín Borrego-Díaz and Antonia M. Chávez-González

Departamento de Ciencias de la Computación e Inteligencia Artificial.
E.T.S. Ingeniería Informática-Universidad de Sevilla.
Avda. Reina Mercedes s.n. 41012-Sevilla
{jborrego, tchavez}@us.es

Abstract. A logical formalism to support the insertion of uncertain concepts in formal ontologies is presented. It is based on the search of extensions by means of two automated reasoning systems (ARS), and it is driven by what we call *cognitive entropy.*

1 Introduction

The challenge of data management with logical trust arose from the statement of the Semantic Web (SW). An important problem is the need for extending or revising ontologies. Such task is, from the point of view of companies, dangerous and expensive: since every change in ontology would affect the overall knowledge of the organization. It is also hard to be automated, because some criteria for revision cannot be fully formalized. Despite its importance, the tools designed to facilitate the syntactic extension or ontological mapping do not analyze, in general, their effect on the (automated) reasoning.

Our aim is to design tools for extending ontologies in a semi-automated way, that is one of the problems present in several methods for cleaning data in the SW, when it implies ontological revision (see e.g. [1] [3]). The method is based on the preservation by extensions of the notion of *ontology robustness* [8]. (*Lattice categoricity*, described in sect. 3), is going to applied in a special case: the change is induced by the user, who has detected the (cognitive) necessity of adding a *notion*. That is, a vague concept which comprises a set of elements with features roughly shaped by the existing concepts. In Ontological Engineering, careful consideration should be paid to the accurate classification of objects: the notion becomes a *concept* when its behaviour is constrained by new axioms that relate it to the initial concepts. This scenario emphasises the current need for an explanation of the reasoning behind cleaning programs. That is, a *formalized explanation* of the decisions made by systems. Note that such explanations are necessary for the desirable design of logical algorithms to be used by general-purpose cleaning agents [4]. It is evident that the task will need not only specific ARSs for SW, but also those for general purpose. The reason is that some tasks are not directly related to reasoning services for the SW [2] [16] [8]. Among the

* This work is partially supported by the project TIN2004-03884 of Spanish Ministry of Education and Science, cofinanced by FEDER founds.

challenges the problem raises in a dynamic setting as the SW, there are three of them which are specially interesting from the point of view of automated reasoning. They seem to obstruct the design of a fully formalised methodology [4] from classical database field:

- We can not suppose the database to be stable (because new facts could be added in the future).
- Usually, specification of ontology is syntactically complex, so it is very likely that classical axiomatization of database theory becomes inconsistent, even if ontology itself is consistent.
- It is possible that the database does not contain facts about the whole relations of the language.

However, some limitations can be solved weakening the requirements imposed in both database and ontological reasoning [8] [2].

The method proposed is based on the assistance of two ARS, McCune's OTTER and MACE4 (`http://www-unix-mcs.anl.gov`). The first one, OTTER, is an automated theorem prover (ATP) based on resolution and *support set* strategy. The program allows great autonomy: its `auto2` mode suffices to find almost every automated proof that have been required. The second one, MACE4, is an automatic model finder sharing formula syntax with OTTER. It is based on Davis-Putnam-Loveland-Longemann's procedure to decide satisfiability. It has been useful for analyzing the models of the involved theories.

Finally, it would be good to add some information about MACE4. Despite it has not been formally verified to work correctly, once the result by MACE4 is determined, it is not difficult to certify that the models it gives are correct. It is necessary to use OTTER to prove that the list of models is exhaustive. Thus, MACE4 has been used as an automatic assistant to induce new results and investigate the effect of diverse axiomatizations, which must be certified later.

2 Logic-based ontological extensions

Once the need for revision is accepted, the task can be seen, up to some extent -and specially when one designs her/his own logical theory-, from two points of view. The first one considers it like a task similar to belief revision, analyzing it by classic methods of AI. Nevertheless, the effort can be expensive, because it must study once again the impact of revision on the foundational features of the source ontology. The second one has a foundational character. The evolution of ontology should obey ground principles which are accepted on this matter. For example, preserving some sort of backward compatibility, if it is possible is possible (extracted from [14]):

- *The ontology should be able to extend other ontologies with new terms and definitions.*
- *The revision of an ontology should not change the well-formedness of resources that commit themselves to an earlier version of the ontology.*

However, such principles are more adequate if the source ontology is *robust*, in the following sense [4]: *An ontology is robust if its core is clear, stable (except for extensions); if every model of its core exhibits similar properties w.r.t. the core language, and if it is capable of admitting (minor) changes made out of the core without commiting core consistency.* By *core* we understand a portion of ontology that we consider as a sound theory with well known properties, and which is accepted as the best for the concepts involved. We can consider two kinds of extensions:

– *Extension by definition.* Is produces conservative extensions. If definitions are not provided for the new elements, conservation can fail.
– *Ontological insertion*: Essentially new (nondefinable) concepts/relations are inserted. The task is to design good axioms to specify the new ones from core theory.

An interesting case occurs in the task of ATP-aided cleaning of logic databases. The *bottom-up change generation* in ontologies -due to the analysis of track interaction among the Knowledge Base, the ATP and the user- induces ontological revision. It can simulate new elements in ontology to be inserted (such as Skolem noise [2]). We analyze here a slightly different problem, which appears when the user is the person who decides to insert a new concept by collecting a set of data.

The *extension by definition* is the basis of *definitional methodologies* for building formal ontologies. It is based on the following principles [7]:

1. *Ontologies should be based upon a small number of primitive concepts.*
2. *These primitives should be given definite model theoretic semantics.*
3. *Axioms should only be given for the primitive concepts.*
4. *Categorical axiom sets should be sought.*
5. *The remaining vocabulary of the ontology (which may be very large), should be introduced purely by means of definitions.*

In this paper, the first three principles are assumed. The fourth one will be replaced by *lattice categoricity.* Categoricity is a strong requirement that can be hard to achieve and to preserve. Even when it is achieved, the resultant theory may be unmanageable (even undecidable) or unintuitive. This phenomenon might suggest that we restrict the analysis of completeness to coherent parts of the theory. However, it is not a *local* notion: since minor changes commit the categoricity and it is expensive to repeat the logical analysis.

With respect to the last principle, starting with a basic theory, it seems hard to define a new concept/relationship. It is better to consider it only as the starting point to build an ontology, thinking thus that we are in early steps of the process, where ontological insertions are necessary.

Finally (although it is not the topic of this paper), we would like to add that an ontological insertion should be supported by a good theory about its relationship with the original ontology. It should as well be supported by a nice way of expanding a representative class of models of the source theory to the new one. This class of models must contain the *intended* models (those that the ontology designer wants to represent). It can be required an interpretation of the new elements which should be formalised, and an re-interpretation of the older ones, which must be compatible with basic original principles.

3 Lattice categorical theories

In order to solve in practice the several logical problems ontological insertion raises we will analyze the categoricity of the structure of the concepts of the ontology. We are going to take into account compatibility which has been previously mentioned, and we are going to try to obtain definitions of the concepts inserted in the new ontology. We will analyze categoricity of structure of the concepts of ontology. For the sake of clarity, we suppose that the set of concepts has a lattice structure. Actually, this is not a constraint: there are methods to extract ontologies from data which produce such structure (such as the Formal Concepts Analysis [13]) and, in general, the ontology is easy to be extended by definition, verifying lattice structure. Although we think about Description Logics [5] as ontological language (the logical basis for ontology languages as OWL, see `http://www.w3.org/TR/owl-features/`), the definitions are useful for full first order logic (FOL), so we give the definitions in FOL language.

On the one hand, a *lattice categorical* theory is the one that proves the lattice structure of its basic relationships. This notion is weaker than categoricity or completeness. On the other hand, lattice categoricity is a reasonable requirement: the theory must certify the basic relationships among the primitive concepts. In [8] we argued that completeness can be replaced by *lattice categoricity* to facilitate the design of feasible methods for extending ontologies. Let us summarize these ideas.

Given a fixed FOL language, let $\mathcal{C} = \{C_1, \dots, C_n\}$ be a (finite) set of concept symbols, let T be a theory (in the general case, definable concepts in T can be considered). Given $M \models T$, we consider the structure $L(M, \mathcal{C})$, in the language $L_\mathcal{C} = \{\top, \bot, \leq\} + \{c_1, \dots, c_n\}$, whose universe are the interpretations in M of the concepts (interpreting c_i as C_i^M), $\top$ is M, $\bot$ is $\emptyset$ and $\leq$ is the subset relation. Recall that $L(M, \mathcal{C})$ is requested to have a lattice structure is a basic desiderata that we assume from now on for every theory we consider. This requirement simplifies the examples.

The relationship between $L(M, \mathcal{C})$ and the model M itself is based in two facts. The first one, the lattice L can be characterized by a finite set of equations E_L, plus a set of formulas $\Theta_\mathcal{C}$ categorizing the lattice under completion. The second one, there exists a natural translation Π of these $L_\mathcal{C}$-equations into formulas in the FOL language so that if E is a set of equations characterizing $L(M, \mathcal{C})$ (so $L(M, \mathcal{C}) \models E$), then $M \models \Pi(E)$.

Definition 1. *Let E be a $L_\mathcal{C}$-theory. We say that E is a* **lattice skeleton** *(l.s.) for a theory T if E verifies that*

- *There is $M \models T$ such that $L(M, \mathcal{C}) \models E + \Theta_\mathcal{C}$, and*

– *$E + \Theta_{\mathcal{C}}$ has an unique model (modulo isomorphism).*

Every consistent theory has a lattice skeleton [8]. Roughly speaking, the existence of essentially different lattice skeletons makes difficult to reason with the ontology while the existence of only one would make it easy.

Definition 2. *T is called a* **lattice categorical (l.c.) theory** *if whatever pair of lattice skeletons for T are equivalent modulo $\Theta_{\mathcal{C}}$.*

Note that if T is l.c. and E is a l.s. of T, then $T \vdash \Pi(E)$. Note also that every consistent theory T has an extension T' which is lattice categorical: it suffices to consider a model $M \models T$, and then to find a set E of equations such that $\Theta_{\mathcal{C}} + E$ has $L(M, \mathcal{C})$ as only model. The theory $T + \Pi(E)$ (and any consistent extension of it) is l.c.

Finally, we can give a formalization of *robust ontological extension*, based in the categorical extension of the ontology:

Definition 3. *Given two pairs $(T_1, E_1), (T_2, E_2)$ we will say that (T_2, E_2) is a* **lattice categorical extension** *of (T_1, E_1) with respect to the sets of concepts $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively, if $\mathcal{C}_1 \subseteq \mathcal{C}_2$ and $L(T_2, \mathcal{C}_2)$ is an E_1-conservative extension of $L(T_1, \mathcal{C}_1)$.*

For reasoning with the lattice of concepts it suffices to work with a lattice skeleton, so, to simplify, we suppose throughout that T is the self l.s.

3.1 Cognitive support

Once formalized the notion of *lattice categorical extension*, we need to design several functions to advise how to select the best l.c. extension.

Assume that T is a theory, and L is the lattice defined by $\mathcal{C}$ in some $M \models T$. From the point of view of ontology designer, such a model M is the *intended* model that the ontology attempts to represent. Suppose that $\Delta = \{h_1, \ldots h_n\}$ is the set of facts on $\mathcal{C}$, and the user wants to classify some elements that occur in Δ by means of a new concept. We can suppose, to simplify the notation, that every fact explicit in T belongs to Δ. Let $U(\Delta)$ be the universe determined by Δ; that is, $\{\boldsymbol{a} \ : \ \text{exists } C \in \mathcal{C}\ [C(\boldsymbol{a}) \in \Delta]\}$.

Given $C \in \mathcal{C}$ in Δ, we consider

$$|C|^{\Delta} := |\{\boldsymbol{a} \ : \ C(\boldsymbol{a}) \in \Delta\}| \text{ and } |C|^{\Delta}_{T} := |\{\boldsymbol{a} \in U(\Delta) \ : \ T \cup \Delta \models C(\boldsymbol{a})\}|.$$

Definition 4. *The* **cognitive support** *of C with respect to Δ, T and L, is*

$$sup^{L}_{T,\Delta}(C) := \frac{|\{\boldsymbol{a} \in U(\Delta) \ : \ \exists i [C_i \leq^{L} C \wedge T \cup \Delta \models C_i(\boldsymbol{a})]\}|}{|U(\Delta)|}$$

This support estimates the number of facts on the concept C entailed by T, normalized by the size of the universe $U(\Delta)$. Because of the computational complexity of logical reasoning, it can be hard in general to compute it: we need to seek, by logical entailment, the cone of concepts defined by C. However, this computation is trivial for lattice categorical theories:

Proposition 1. *If T is lattice categorical, then* $sup^L_{T,\Delta}(C) = \dfrac{|C|^{\Delta}_{T}}{|U(\Delta)|}$

The proposition holds because if $C_i \leq^L C$, then $T \models C_i \sqsubseteq C$. Thus, if $T \cup \Delta \models C_i(\boldsymbol{a})$, then $T \cup \Delta \models C(\boldsymbol{a})$.

From now on, we suppose that Δ is compounded by facts on atoms of the lattice of concepts (that is, about the most specific concepts). Note, also, that if T is l.c., then L is unique, and we will thus omit the superscript L in that case.

Corollary 1. *If $\mathcal{J} = \{C_1, \dots, C_n\}$ is a Jointly Exhaustive and Pairwise Disjoint (JEPD) set of concepts in L, then $sup_{T,\Delta}(.)$ is a probability measure.*

Proof. It is easily seen that $\sum_{C \in \mathcal{J}} sup^L_{T,\Delta}(C) = 1$.

The **cognitive entropy** of $\mathcal{J}$ is $CH(\mathcal{J}) = -\sum\limits_{C \in \mathcal{J}} sup_{T,\Delta}(C) \log sup_{T,\Delta}(C)$.

3.2 Entropy of ontological extensions

Suppose that the user decides that a set $\{a_1, \dots, a_k\} \subseteq U(\Delta)$ induces a new concept D (provisionally, a notion). Such a notion might not be fully represented by those elements. Also, it is possible that some of them do not belong to the new concept, because of noise in the data. It might also be the case that the concept is constrained by a set Σ of axioms introduced by the user. Furthermore it is also possible that $T + \Sigma$ is not l. c., that is, this theory does not prove the intended lattice induced by $\mathcal{C} \cup \{D\}$. MACE4 provides the collection $\{L_1, \dots, L_m\}$ of the lattices induced by the models of $T + \Sigma$. Let T_i be a lattice skeleton for L_i $(i = 1, \dots, m)$.

Now, we focus our attention on a concrete *level* of the Ontology, where we intend to insert the new concept. The level will be a JEPD $\mathcal{J} = \{C_1, \dots, C_k\}$ of the lattice L verifying that if the new concept D contains some of them,

$$\mathcal{J}^{L_i}_{\restriction D} = \{C_i \in \mathcal{J} \ : \ C_i \leq^{L_i} D\} \neq \emptyset$$

then $\mathcal{J}_i = (\mathcal{J} \setminus \mathcal{J}^{L_i}_{\restriction D}) \cup \{D\}$ is a JEPD in L_i. Since T_i is a l.c. extension of T, the support of D is easy to achieve:

Theorem 1. *In above conditions,* $sup_{T_i,\Delta}(D) = \sum\limits_{C \in \mathcal{J}^{L_i}_{\restriction D}} sup^L_{T,\Delta}(C_i)$

To estimate the conditional entropy of the new extension, we consider a natural definition of *conditional support*:

$$sup_{T_i,T,\Delta}(C'|C) := \frac{|\{\boldsymbol{a} \in U(\Delta) \ : \ T \cup \Delta \models C(\boldsymbol{a}) \wedge T_i \cup \Delta \models C'(\boldsymbol{a})\}|}{|C|^{\Delta}_{T}}$$

This support allows to estimate the amount of new information produced by the extension by standard methods; through the *conditional entropy* associated to the two probability measures. The **conditional cognitive entropy** is :

$$CH(\mathcal{J}||\mathcal{J}_i) = -\sum_{\substack{C' \in \mathcal{J} \\ C \in \mathcal{J}_i}} sup_{T_i,\Delta}(C'|C) \log sup_{T_i,\Delta}(C'|C)$$

This sum can be simplified (assuming $0 \log 0 = 0$): if $C = C'$ or $C, C' \in \mathcal{J}$, then

$$sup_{T_i,T,\Delta}(C'|C) \log sup_{T_i,T,\Delta}(C'|C) = 0$$

and the following property holds:

Proposition 2. *In above conditions,* $sup_{T_i,T,\Delta}(C'|C) = \dfrac{|C'|^{\Delta}_{T}}{|C|^{\Delta}_{T_i}}$

This entropy is similar to Kullback-Leibler distance or relative entropy (see [15]), but using the entailment to classify the elements. It is known that it is minor than the initial entropy. In [12] similar entropies are used, but based on probabilistic assignation. Finally, in order to estimate what is the best extension for our purposes, it is necessary to compute the The **Shannon's diversity index** for each L_i. This index normalizes the amount of information produced by the extension, and is defined as

$$IH(\mathcal{J}_i) = \frac{CH(\mathcal{J}||\mathcal{J}_i)}{\log |\mathcal{J}_i|}$$

The interpretation of the index is as follows: if we select L_i with minimum $IH(\mathcal{J}_i)$, the new information produced by the new concept is minor. This option is the *cautious* one: the reparation of the source ontology is *ligth* and we do not expect big changes in the representation of the intended model. If we select L_i with an upper $IH(J)$, the change of the information is more relevant; we select such an extension if we regard as robust the specification of the concept given by Σ together with the facts. In general, we have to chose the l.c. extension with minor index. Intuitively, in this way we do not change too much the information of the initial ontology.

4 An example

We would like to show a short example in the field of Qualitative Spatial Reasoning (QSR). *Region Connection Calculus* (RCC) [11] is a well-known mereotopological approach to QSR, that we can consider to be a robust ontology. For RCC, the *spatial entities* are non-empty regular sets. The primary relation between them is *connection*, $C(x, y)$, with intended meaning: *"the topological closures of* x *and* y *intersect"*. The basic axioms of RCC are $A_1 := \forall x[C(x, x)]$ and $A_2 := \forall x, y[C(x, y) \rightarrow C(y, x)]$ jointly with a set of definitions on the main spatial relations (fig. 1), and other axioms not used here (see [11]).

$DC(x,y) \leftrightarrow \neg C(x,y)$	(x is disconnected from y)
$P(x,y) \leftrightarrow \forall z[C(z,x) \rightarrow C(z,y)]$	(x is part of y)
$PP(x,y) \leftrightarrow P(x,y) \wedge \neg P(y,x)$	(x is proper part of y)
$EQ(x,y) \leftrightarrow P(x,y) \wedge P(y,x)$	(x is identical to y)
$O(x,y) \leftrightarrow \exists z[P(z,x) \wedge P(z,y)]$	(x overlaps y)
$DR(x,y) \leftrightarrow \neg O(x,y)$	(x is discrete from y)
$PO(x,y) \leftrightarrow O(x,y) \wedge \neg P(x,y) \wedge \neg P(y,x)$	(x partially overlaps y)
$EC(x,y) \leftrightarrow C(x,y) \wedge \neg O(x,y)$	(x is externally connected to y)
$TPP(x,y) \leftrightarrow PP(x,y) \wedge \exists z[EC(z,x) \wedge EC(z,y)]$	(x is a tangential prop. part of y)
$NTPP(x,y) \leftrightarrow PP(x,y) \wedge \neg\exists z[EC(z,x) \wedge EC(z,y)]$	(x is a non-tang. prop. part of y)

Fig. 1. Axioms of RCC

We have proved (using MACE4 and OTTER) that the set of formulas E given in the figure 2 categorises under completion the lattice of the RCC-spatial relationships (given in fig. 3). The set of binary relations formed by the eight (JEPD) relations given in figure 3 is denoted by RCC8. If this set is thought to be a calculus, all possible unions of the basic relations are also used. Another interesting calculus is RCC5, based on $\{DR, PO, PP, PPi, EQ\}$.

$\top \equiv C \sqcup D$	$PO \sqsubseteq \neg P \sqcap \neg Pi \sqcap \neg DR$	$DR \equiv EC \sqcup DC$
$NTPP \sqsubseteq \neg TPP \sqcap \neg Pi \sqcap \neg DR$	$C \equiv O \sqcup EC$	$TPP \sqsubseteq \neg Pi \sqcap \neg DR$
$O \equiv PO \sqcup P \sqcup Pi$	$EQ \sqsubseteq \neg PPi \sqcap \neg DR$	$Pi \equiv EQ \sqcup PPi$
$TPPi \sqsubseteq \neg NTPPi \sqcap \neg DR$	$P \equiv EQ \sqcup PP$	$NTPPi \sqsubseteq \neg DR$
$PPi \equiv TPPi \sqcup NTPPi$	$EC \sqsubseteq \neg DC$	$PP \equiv TPP \sqcup NTPP$

Fig. 2. A skeleton for RCC

Suppose that if we insert a new spatial uncertain relation D expressing "x *and* y *have a isometric overlapping relation*"; that is, D covers *partial overlapping PO* and *extentional equality EQ* relationships. That is, proper part is not possible between isometric objects. This is suggested by the study of spatial relationships among identical objects (e.g. the 2-D spatial configuration of a set of coins). Thus, we consider that the new relation D satisfies

$$RCC + \{\forall x \forall y (PO(x,y) \rightarrow D(x,y)), \forall x \forall y (EQ(x,y) \rightarrow D(x,y))\}$$

or, in terms of skeleton, $E + \{PO \sqsubseteq D, EQ \sqsubseteq D\}$. MACE4 produces seven l.c. extensions (classified according to their lattices in fig. 4). All these extensions can be mereotopologically interpreted [10]. Suppose that the set that motivates the extension is:

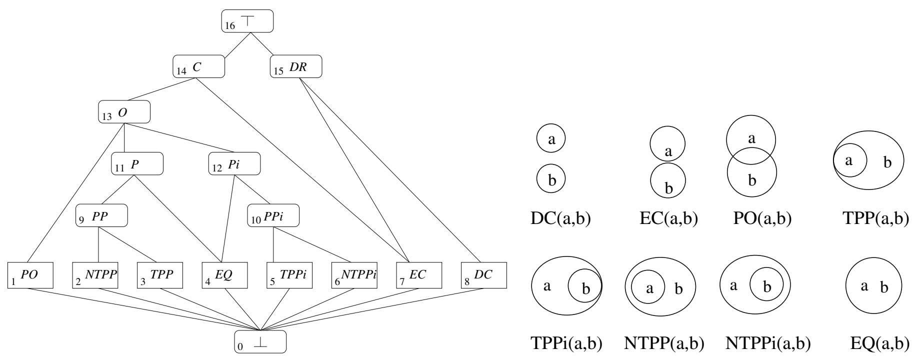


Fig. 3. The lattice of spatial relations of RCC (left) and he relations of RCC8 (right)

$$\Delta := \left\{ \begin{array}{lllll} PO(m_1,m_2) & EQ(m_2,m_3) & EQ(m_3,m_4) & PO(m_1,m_3) & DC(m_4,m_6) \\ DC(m_3,m_5) & PO(m_5,m_1) & NTPP(c_1,m_3) & EC(c_2,m_1) & TPP(c_2,c_5) \\ DC(c_1,c_2) & TPPi(c_5,c_2) & NTPP(m_2,c_4) & DC(m_1,c_3) & TPP(c_1,c_3) \end{array} \right.$$

In this case, $|U(\Delta)| = 15$, and the basic JEPD is the set $\mathcal{J} = \{PO, PP, EQ, PPi, EC, DC\}$. In each L_i, $\mathcal{J}_i$ is a JEPD, so we can assign conditional entropy and Shannon's diversity index to each extension.Thus, $T_2 \equiv E + \{D \equiv PO \sqcup EQ\}$ is the selected l.c. extension because it has the minimum Shannon's index. On the other hand, the user's notion might be inconsistent. For instance, if the user's proposal for Σ' is $\{PO \sqsubseteq D, EQ \sqsubseteq D, P \sqsubseteq D, D \sqsubseteq O\}$, then there is not any l.c. extension, a fact that we have certified using MACE4 and OTTER.

5 Closing Remarks

Although it is usual to study entropy for associating data to concepts in Ontology Learning, it is not usual to consider the *provability from ontology* like a factor, as we do. However, we think, that it will be a key issue in the SW. There are other approaches, but they deal with probabilistic objects. J. Calmet and A. Daemi also use entropy in order to revise or compare ontologies [9] [12]. This is based on the self taxonomy defined by the concepts but provability from specification is not regarded. Conditional entropy has already been considered in the similar task of Abductive Reasoning for learning qualitative relationships/concepts (usually in probabilistic terms, see e.g. [6]). The main difference between this approach and ours is that we work with probability mass distribution of *probable facts* from ontological specifications.

Finally, it should be noted that only some distributions of data will induce the user to decide an ontological insertion. Therefore, although once the distribution of data is determined, the method is fully formalized, the soundness of the extensions still depends on human decisions.

References

1. J. A. Alonso-Jiménez, J. Borrego-Díaz, A. M. Chávez-González and J. D. Navarro-Marín, A Methodology for the Computer–Aided Cleaning of Complex Knowledge

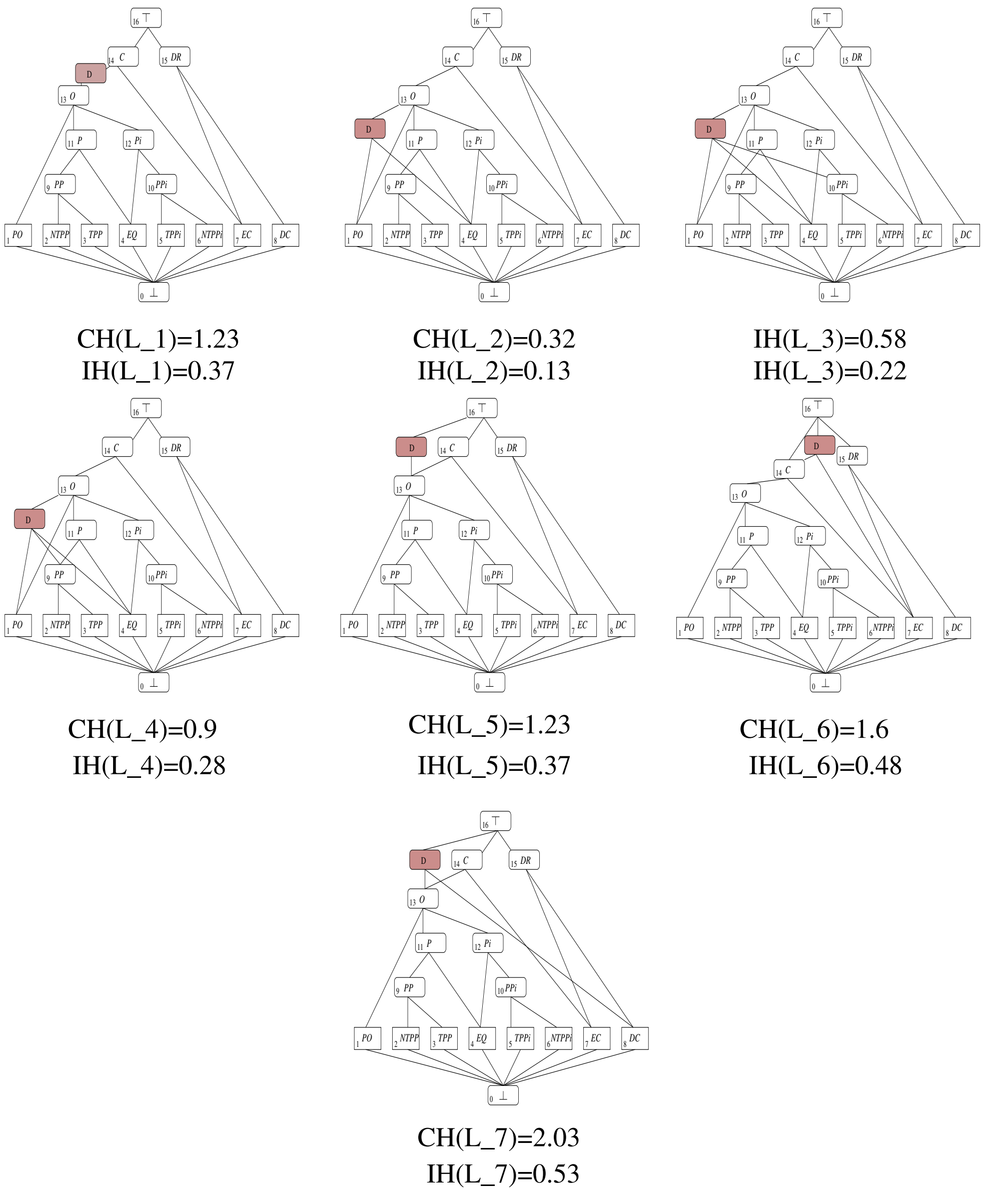


Fig. 4. The seven l.c. extensions by insertion. The grey box denotes the new relation.

Databases. *28th Conf. of IEEE Industrial Electronics Society IECON 2002, pp. 1806-1812, 2003.*

2. J. Alonso-Jiménez, J. Borrego-Díaz, A. M. Chávez González , M. A. Gutiérrez-Naranjo and J. David Navarro-Marín, "Towards a Practical Argumentative Reasoning with Qualitative Spatial Databases". *16th Int. Conf. on Ind. and Eng. App. of AI and Expert Systems IEA/AIE 2003*, LNAI 2718, Springer, 2003, pp. 789-798.
3. J.A. Alonso-Jiménez, Joaquín Borrego-Díaz, Antonia M. Chávez-González, Ontology Cleaning by Mereotopological Reasoning. DEXA Workshop on Web Semantics WEBS-2004, pp. 132-137 (2004).
4. J. A. Alonso-Jiménez, J. Borrego-Díaz, A. M. Chávez-González and F. J. Martín-Mateos, Foundational Challenges in Automated Data and Ontology Cleaning in the Semantic Web, *IEEE Intelligent Systems*, to appear (2005).
5. F. Baader, D. Calvanese, D. McGuinness, D. Nardi and P. Patel-Schneider (eds.), *The Description Logic Handbook. Theory, Implementation and Applications*, Cambridge University Press, 2003.
6. R. Bhatnagar and L. N. Kanal Structural and Probabilistic Knowledge for Abductive Reasoning, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 15(3):233-245 (1993).
7. B. Bennett, The Role of Definitions in Construction and Analysis of Formal Ontologies, in: P. Doherty, J. McCarthy, and M. Williams (eds.) *Logical Formalization of Commonsense Reasoning (2003 AAAI Spring Symp.)*, 27-35, AAAI Press, 2003.
8. J. Borrego-Díaz and A. M. Chávez-González, Extension of Ontologies Assisted by Automated Reasoning Systems, *10th Int. Conf. on Computer Aided Systems Theory, EUROCAST 2005*, to appear in LNCS, Springer, 2005.
9. J. Calmet and A. Daemi, From entropy to ontology, *4th. Int. Symp. From Agent Theory to Agent Implementation AT2AI-4*, 2004.
10. A. M. Chávez-González, Automated Mereotopological Reasoning for Ontology Debugging, Ph.D. Thesis, University of Seville, 2005.
11. A. G. Cohn, B. Bennett, J. M. Gooday and N. M. Gotts. Representing and Reasoning with Qualitative Spatial Relations about Regions, chapter 4, in O. Stock (ed.), *Spatial and Temporal Reasoning*, Kluwer, Dordrecth, 1997.
12. A. Daemi and J. Calmet, From Ontologies to Trust through Entropy, *Proc. of the Int. Conf. on Advances in Intelligent Systems - Theory and Applications*, 2004.
13. B. Ganter and R. Wille, *Formal Concept Analysis, Mathematical Foundations*, Springer, Berlin, 1999.
14. J. Heflin, Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment, Ph.D. Thesis, Univ. of Maryland, College Park, 2001.
15. S. Kotz and N.L. Johnson (eds). *Encyclopedia of Statistical Sciences* vol.4, pp. 421–425. John Wiley and Sons, 1981.
16. D. Tsarkov, A. Riazanov, S. Bechhofer, and I. Horrocks, Using Vampire to Reason with OWL. *2004 Int. Semantic Web Conference (ISWC 2004)*, LNCS 3298 Springer, 2004, pp. 471-485.

The Fuzzy Description Logic f-$\mathcal{SHIN}$

Giorgos Stoilos[1], Giorgos Stamou[1], Vassilis Tzouvaras[1],
Jeff Z. Pan[2] and Ian Horrocks[2]

[1] Department of Electrical and Computer Engineering, National Technical University of Athens, Zographou 15780, Greece

[2] School of Computer Science, The University of Manchester
Manchester, M13 9PL, UK

Abstract. In the Semantic Web information would be retrieved, processed, combined, shared and reused in the maximum automatic way possible. Obviously, such procedures involve a high degree of uncertainty and imprecision. For example ontology alignment or information retrieval are rarely true or false procedures but usually involve confidence degrees or provide rankings. Furthermore, it is often the case that information itself is imprecise and vague like the concept of a "tall" person, a "hot" place and many more. In order to be able to represent and reason with such type of information in the Semantic Web (SW), as well as, enhance SW applications we present an extension of the Description Logic $\mathcal{SHIN}$ with fuzzy set theory. We present the semantics as well as detailed reasoning algorithms for the extended language.

1 Introduction

Uncertainty, like imprecision and vagueness, is a factor that can cause the degradation of the performance of a system. To this end, many applications and domains have incorporated mathematical frameworks that deal with such type of information, resulting in the improvement of their effectiveness. Applications like robotics [1], computer vision [2] and many more have embraced frameworks like fuzzy set theory [3] in order to improve their performance. On the other hand, in the Semantic Web context, little work has been carried out towards this direction. Apart from the fact that uncertainty is many times a feature of information itself, as for example the concepts of a "tall" man, a "fast" car, a "blue" sky and many more, applications like information retrieval, automatic information sharing and reuse are hardly true or false procedures but rather a matter of a degree. The need for covering vagueness in the Semantic Web has been stressed many times the past years [4–6]. It has been pointed out that dealing with such information would improve many Semantic Web applications [7–9].

Knowledge in the SW is usually structured in the form of ontologies [10]. This has led to considerable efforts to develop a suitable ontology language, culminating in the design of the OWL Web Ontology Language [11], which is now a W3C recommendation. The OWL recommendation actually consists of three languages of increasing expressive power, namely OWL Lite, OWL DL and OWL Full. *OWL Lite* and *OWL DL* are, basically very expressive description logics; they are almost[3] equivalent to the $\mathcal{SHIF}(\mathbf{D}^+)$ and $\mathcal{SHOIN}(\mathbf{D}^+)$ DLs. OWL Full is clearly undecidable because it does not impose restrictions on the use of transitive properties. Although the above DL languages are very expressive, they feature expressive limitations regarding their ability to represent vague and imprecise knowledge. As obvious, in order to make applications that use DLs able to cope with vague and uncertain information we have to extend them with a theory capable of representing this kind of information. One such important theory is fuzzy set theory.

In the current paper we extend the results obtained in [9] for fuzzy $\mathcal{SI}$ (f-$\mathcal{SI}$) to the language $\mathcal{SHIN}$, thus creating f-$\mathcal{SHIN}$. $\mathcal{SHIN}$ extends $\mathcal{SI}$ [12] with number restrictions and role hierarchies [13]. Number restrictions give us the ability to restrict the number of objects that a certain object is related to by a specific relation. For example we can state that a car has exactly four wheels, writing $\mathsf{Car} \equiv \mathsf{Vehicle} \sqcap \geq 4\mathsf{hasWheel} \sqcap \leq 4\mathsf{hasWheel}$. But though this definition is correct, it faces many limitations, for example, in the context of image processing where several wheels of a car in an image might be hidden. Hence a detected object can belong to a concept like, $\geq 4\mathsf{hasWheel}$, only to a certain degree. On the other hand role hierarchies allow us to state sub-role/super-role relations, as for example the relation that holds between the hasChild and hasOffspring roles. Regarding expressive power, $\mathcal{SHIN}$ is more expressive than OWL-Lite, ignoring data-types. In the following we will introduce the syntax of f-$\mathcal{SHIN}$ and present a detailed procedure to reason with the extended language.

2 Syntax and Semantics of f-$\mathcal{SHIN}$

In this section we introduce the DL f-$\mathcal{SHIN}$. As pointed out in the fuzzy DL literature [9,14], fuzzy extensions of DLs involve only the *assertion* of individuals to concepts and the semantics of the new language. Hence, as usual we have an alphabet of distinct concept names ($\mathbf{C}$), role names ($\mathbf{R}$) and individual names ($\mathbf{I}$). f-$\mathcal{SHIN}$-roles and f-$\mathcal{SHIN}$-concepts are defined as follows:

Definition 1. *Let $RN \in \mathbf{R}$ be a role name, R an f-$\mathcal{SHIN}$-role, C, D f-$\mathcal{SHIN}$-concepts. Valid f-$\mathcal{SHIN}$-roles are defined by the abstract syntax: $R ::= RN \mid R^-$. The inverse relation of roles is symmetric, and to avoid considering roles such as R^{--}, we define a function* Inv*, which returns the inverse of a role, more precisely $\mathsf{Inv}(R) := RN^-$ if $R = RN$ and $\mathsf{Inv}(R) := RN$ if $R = RN^-$.*

The set of f-$\mathcal{SHIN}$ concepts is the smallest set such that:

1. *every concept name $C \in CN$ is an f-$\mathcal{SHIN}$-concept,*
2. *if C and D are f-$\mathcal{SHIN}$-concepts, R is an f-$\mathcal{SHIN}$-role, S a* simple *f-$\mathcal{SHIN}$-role [15] and $p \in \mathbb{N}$, then $(C \sqcup D)$, $(C \sqcap D)$, $(\neg C)$, $(\forall R.C)$, $(\exists R.C)$, $(\geq pS)$ and $(\leq pS)$ are also f-$\mathcal{SHIN}$ concepts.*

[3] They also provide annotation properties, which Description Logics don't.

Table 1. Semantics of f-$\mathcal{SHIN}$-concepts

$$\begin{array}{rcl}
\top^{\mathcal{I}}(a) & = & 1 \\
\bot^{\mathcal{I}}(a) & = & 0 \\
(\neg C)^{\mathcal{I}}(a) & = & 1 - C^{\mathcal{I}}(a) \\
(C \sqcup D)^{\mathcal{I}}(a) & = & max(C^{\mathcal{I}}(a), D^{\mathcal{I}}(a)) \\
(C \sqcap D)^{\mathcal{I}}(a) & = & min(C^{\mathcal{I}}(a), D^{\mathcal{I}}(a)) \\
(\forall R.C)^{\mathcal{I}}(a) & = & \inf_{b \in \Delta^{\mathcal{I}}}\{max(1 - R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))\} \\
(\exists R.C)^{\mathcal{I}}(a) & = & \sup_{b \in \Delta^{\mathcal{I}}}\{min(R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))\} \\
(\geq pR)^{\mathcal{I}}(a) & = & \sup_{b_1,\ldots,b_p \in \Delta^{\mathcal{I}}} \min_{i=1}^{p} R^{\mathcal{I}}(a,b_i) \\
(\leq pR)^{\mathcal{I}}(a) & = & \inf_{b_1,\ldots,b_{p+1} \in \Delta^{\mathcal{I}}} \max_{i=1}^{p+1}\{1 - R^{\mathcal{I}}(a,b_i)\} \\
(R^{-})^{\mathcal{I}}(b,a) & = & R^{\mathcal{I}}(a,b)
\end{array}$$

A fuzzy $TBox$ is a finite set of fuzzy concept axioms. Let A be a concept name, C a f-$\mathcal{SHIN}$-concept. Fuzzy concept axioms of the form $A \sqsubseteq C$ are called *fuzzy inclusion introductions*; fuzzy concept axioms of the form $A \equiv C$ are called *fuzzy equivalence introductions*. Note that how to deal with *general fuzzy concept inclusions* [12] still remains an open problem in fuzzy concept languages. A fuzzy $RBox$ is a finite set of fuzzy role axioms. Fuzzy role axioms of the form $\mathsf{Trans}(RN)$, where RN is a role name, are called *fuzzy transitive role axioms*; fuzzy role axioms of the form $R \sqsubseteq S$ are called *fuzzy role inclusion* axioms. We use the notation $\overset{*}{\sqsubseteq}$ to denote the transitive-reflexive closure of $\sqsubseteq$. A role R is called sub-role (super-role) of a role S if $R \overset{*}{\sqsubseteq} S$ ($S \overset{*}{\sqsubseteq} R$). A fuzzy $ABox$ is a finite set of fuzzy assertions. A *fuzzy assertion* [14] is of the form $\langle a : C \bowtie n\rangle$, $\langle (a,b) : R \bowtie n\rangle$, where $\bowtie$ stands for $\geq, >, \leq$ or $<$ or $a \not\doteq b$, for $a, b \in \mathbf{I}$. Intuitively, a fuzzy assertion of the form $\langle a : C \geq n\rangle$ means that the membership degree of a to the concept C is at least equal to n. We call assertions defined by $\geq, >$ *positive* assertions, while those defined by $\leq, <$ *negative* assertions [9]. A fuzzy knowledge base Σ is a triple $\langle \mathcal{T}, \mathcal{R}, \mathcal{A}\rangle$, where $\mathcal{T}$ is a fuzzy $TBox$, $\mathcal{R}$ is a fuzzy $RBox$ and $\mathcal{A}$ is a fuzzy $ABox$. A pair of assertions are called *conjugated* if they impose contradicting restrictions. For example, the pair of assertions $\langle \phi \geq n\rangle$ and $\langle \phi < m\rangle$, with $n \geq m$ contradict to each other. In the presence of role hierarchies one should also take into consideration possible sub- or super-roles when checking for such contradictions. For example the assertions $\langle (a,b) : R \geq 0.7\rangle$ and $\langle (a,b) : P \leq 0.4\rangle$, with $P \overset{*}{\sqsubseteq} R$ are conjugated. For a detailed description of the possible conjugated pairs the reader is referred to [14].

The semantics of fuzzy DLs are provided by a *fuzzy interpretation* [9, 14]. A fuzzy interpretation is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}\rangle$ where the domain $\Delta^{\mathcal{I}}$ is a non-empty set of objects and $\cdot^{\mathcal{I}}$ is a fuzzy interpretation function, which maps an individual name a to elements of $\mathsf{a}^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ and a concept name A (role name R) to a membership function $\mathsf{A}^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0,1]$ ($R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0,1]$). Moreover, fuzzy interpretations are extended to interpret arbitrary f-$\mathcal{SHIN}$-concepts and roles. The complete set of semantics is depicted in Table 1, where inf stands for the infimum and sup for the supremum of a set. Note that apart from the fuzzy number restrictions, the interpretation of fuzzy concepts and concept constructors is the usual one found in the DL literature [9, 14, 16], where the Gödel conjunction (t(a,b)=min(a,b)), the Gödel disjunction (u(a,b)=max(a,b)) and the Kleen-Dienes fuzzy implication ($\mathcal{J}$ (a,b)=max(1-a,b)) are used for performing the fuzzy set theoretic operations. The semantics of fuzzy number restrictions were first presented in [17]. We chose to follow these semantics because, as pointed out in [17], they are derived by the First-Order formulae of classical number restrictions [17]. In [9] the naming f$_{KD}$-$\mathcal{SI}$ was used due to the usage of the Kleen-Dienes fuzzy implication. Since we also use the same implication here, from now on, we will refer to the extended language as f$_{KD}$-$\mathcal{SHIN}$.

An f$_{KD}$-$\mathcal{SHIN}$-concept C is *satisfiable* iff there exists some fuzzy interpretation $\mathcal{I}$ for which there is some $a \in \Delta^{\mathcal{I}}$ such that $C^{\mathcal{I}}(a) = n$, and $n \in (0, 1]$. A fuzzy interpretation $\mathcal{I}$ satisfies a fuzzy *TBox* $\mathcal{T}$ iff $\forall a \in \Delta^{\mathcal{I}}$, $A^{\mathcal{I}}(a) \leq C^{\mathcal{I}}(a)$ for each $A \sqsubseteq C$ in $\mathcal{T}$ and $A^{\mathcal{I}}(a) = C^{\mathcal{I}}(a)$ for each $A \equiv C$ in $\mathcal{T}$. The semantics of fuzzy inclusion axioms is the usual one found in fuzzy set theory [3]. A fuzzy interpretation $\mathcal{I}$ satisfies a fuzzy *RBox* $\mathcal{R}$ iff $\forall a, b, c \in \Delta^{\mathcal{I}}$, $R^{\mathcal{I}}(a, c) \geq sup_{b \in \Delta^{\mathcal{I}}}\{min(R^{\mathcal{I}}(a, b), R^{\mathcal{I}}(b, c))\}$ for each Trans (R) in $\mathcal{R}$, and $\forall \langle a, b\rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, $R^{\mathcal{I}}(a, b) \leq S^{\mathcal{I}}(a, b)$ for each $R \sqsubseteq S$. Note that the semantics of role inclusion axioms $R \sqsubseteq S$ imply $\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S)$. A fuzzy relation R, defined over the domain $X \times X$, is called *sup-min transitive* iff $R(x, z) \geq \sup_{y \in X} min(R(x, y), R(y, z))$. Given a fuzzy interpretation $\mathcal{I}$, $\mathcal{I}$ satisfies $\langle a : C \geq n\rangle$ if $C^{\mathcal{I}}(a^{\mathcal{I}}) \geq n$, $\mathcal{I}$ satisfies $\langle (a, b) : R \geq n\rangle$ if $R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \geq n$, while $\mathcal{I}$ satisfies $a \not\doteq b$ if $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. The satisfiability of fuzzy assertions with $\leq, >$ and $<$ is defined analogously. A fuzzy interpretation satisfies a fuzzy *ABox* $\mathcal{A}$ if it satisfies all fuzzy assertions in $\mathcal{A}$. In this case, we say $\mathcal{I}$ is a *model* of $\mathcal{A}$. If $\mathcal{A}$ has a model then we say that it is *consistent*. Finally, a fuzzy knowledge base Σ is satisfiable iff there exists a fuzzy interpretation $\mathcal{I}$ which satisfies all axioms in Σ. Moreover, Σ *entails* an assertion $\langle \phi \bowtie n\rangle$ or a fuzzy concept inclusion axiom $C \sqsubseteq D$, written $\Sigma \models \langle \phi \bowtie n\rangle$ or $\Sigma \models C \sqsubseteq D$, iff any model of Σ also satisfies the fuzzy assertion or fuzzy concept inclusion axiom, respectively. The problems of entailment and subsumption can be reduced to fuzzy knowledge base satisfiability as is shown in [14].

Since a fuzzy *ABox* $\mathcal{A}$ might contain many positive assertions for the same individual (pair of individuals), without forming a contradiction, it is in our interest to compute what is the best lower and upper truth-value bounds of a fuzzy assertion. In [14] the concept of *greatest lower bound* of a fuzzy assertion w.r.t. Σ was defined as $glb(\Sigma, \phi) = \sup\{n : \Sigma \models \langle \phi \geq n\rangle\}$, and that of a *least upper bound* as, $lub(\Sigma, \phi) = \inf\{n : \Sigma \models \langle \phi \leq n\rangle\}$, where ϕ represents a crisp assertion of the form $a : C$ or $(a, b) : R$. Observe that $\sup \emptyset = 0$ and $\inf \emptyset = 1$. A procedure to solve the best truth-value bound was provided in [14]. Such a procedure can also be used in our framework.

3 A fuzzy tableau for f$_{KD}$-$\mathcal{SHIN}$ *ABoxes*

Most of the inference services of fuzzy DLs, can be reduced to the problem of consistency checking for *ABoxes* [14]. Consistency is usually checked with tableaux algorithms that try to construct a fuzzy tableau for a fuzzy *ABox* $\mathcal{A}$ [9], which is an abstraction of a model of $\mathcal{A}$ [13]. The tableau has a forest-like structure with nodes representing the individuals that appear in $\mathcal{A}$, and edges between nodes, which represent the relations that hold between two individuals. Each node is labelled with a set of triples of the form $\langle D, \bowtie, n\rangle$, which denote the concept, the type of inequality and the membership degree that the individual of the node has been asserted to belong to D. We call such triples *membership triples.* For triples of a single node, the concepts of conjugated, positive and negative triples can be defined in the obvious way. Since the expansion rules decompose the initial concept, the concepts that appear in triples are sub-concepts of the initial concept. Sub-concepts of a concept D are denoted by $sub(D)$. The set of all sub-concepts that appear within an *ABox* is denoted by $sub(\mathcal{A})$.

Since the De'Morgan laws are satisfied by the operations we use in the current paper [3] all concepts are assumed to be in their *negation normal form* (NNF) [18]. In the following we use the symbols $\rhd$ and $\lhd$ as a placeholder for the inequalities $\geq, >$ and $\leq, <$ and the symbol $\bowtie$ as a placeholder for all types of inequations. Furthermore we use the symbols $\bowtie^-$, $\rhd^-$ and $\lhd^-$ to denote their *reflections.* For example the reflection of $\leq$ is $\geq$ and that of $>$ is $<$.

Definition 2. *Let $\mathcal{A}$ be an f_{KD}-$\mathcal{SHIN}$ ABox, $\boldsymbol{R}_A$ the set of roles occurring in $\mathcal{A}$ together with their inverses, $\mathbf{I}_{\mathcal{A}}$ the set of individuals in $\mathcal{A}$, $\mathcal{X}$ the set $\{\geq, >, \leq, <\}$ and $\mathcal{R}$ a fuzzy RBox. A fuzzy tableau T for $\mathcal{A}$ w.r.t. $\mathcal{R}$ is a quadruple ($\boldsymbol{S}$, $\mathcal{L}$, $\mathcal{E}$, $\mathcal{V}$) such that:*

- *$\boldsymbol{S}$ is a non-empty set of individuals (nodes),*
- *$\mathcal{L} : \boldsymbol{S} \to 2^{sub(A)} \times \mathcal{X} \times [0,1]$ maps each element of $\boldsymbol{S}$ to membership triples,*
- *$\mathcal{E} : \boldsymbol{R}_A \to 2^{\boldsymbol{S}\times\boldsymbol{S}} \times \mathcal{X} \times [0,1]$ maps each role to membership triples,*
- *$\mathcal{V} : \mathbf{I}_{\mathcal{A}} \to \boldsymbol{S}$ maps individuals occurring in $\mathcal{A}$ to elements in $\boldsymbol{S}$.*

For all $s,t \in \boldsymbol{S}$, $C, E \in sub(\mathcal{A})$, and $R \in \boldsymbol{R}_A$, T satisfies:

1. *If $\langle \neg C, \bowtie, n\rangle \in \mathcal{L}(s)$, then $\langle C, \bowtie^-, 1-n\rangle \in \mathcal{L}(s)$,*
2. *If $\langle C \sqcap E, \rhd, n\rangle \in \mathcal{L}(s)$, then $\langle C, \rhd, n\rangle \in \mathcal{L}(s)$ and $\langle E, \rhd, n\rangle \in \mathcal{L}(s)$,*
3. *If $\langle C \sqcup E, \lhd, n\rangle \in \mathcal{L}(s)$, then $\langle C, \lhd, n\rangle \in \mathcal{L}(s)$ and $\langle E, \lhd, n\rangle \in \mathcal{L}(s)$,*
4. *If $\langle C \sqcup E, \rhd, n\rangle \in \mathcal{L}(s)$, then $\langle C, \rhd, n\rangle \in \mathcal{L}(s)$ or $\langle E, \rhd, n\rangle \in \mathcal{L}(s)$,*
5. *If $\langle C \sqcap E, \lhd, n\rangle \in \mathcal{L}(s)$, then $\langle C, \lhd, n\rangle \in \mathcal{L}(s)$ or $\langle E, \lhd, n\rangle \in \mathcal{L}(s)$,*
6. *If $\langle \forall R.C, \rhd, n\rangle \in \mathcal{L}(s)$ and $\langle\langle s,t\rangle, \rhd', n_1\rangle \in \mathcal{E}(R)$ is conjugated with $\langle\langle s,t\rangle, \rhd^-, 1-n\rangle$, then $\langle C, \rhd, n\rangle \in \mathcal{L}(t)$,*
7. *If $\langle \exists R.C, \lhd, n\rangle \in \mathcal{L}(s)$ and $\langle\langle s,t\rangle, \rhd, n_1\rangle \in \mathcal{E}(R)$ is conjugated with $\langle\langle s,t\rangle, \lhd, n\rangle$, then $\langle C, \lhd, n\rangle \in \mathcal{L}(t)$,*
8. *If $\langle \exists R.C, \rhd, n\rangle \in \mathcal{L}(s)$, then there exists $t \in \boldsymbol{S}$ such that $\langle\langle s,t\rangle, \rhd, n\rangle \in \mathcal{E}(R)$ and $\langle C, \rhd, n\rangle \in \mathcal{L}(t)$,*
9. *If $\langle \forall R.C, \lhd, n\rangle \in \mathcal{L}(s)$, then there exists $t \in \boldsymbol{S}$ such that $\langle\langle s,t\rangle, \lhd^-, 1-n\rangle \in \mathcal{E}(R)$ and $\langle C, \lhd, n\rangle \in \mathcal{L}(t)$,*
10. *If $\langle \exists S.C, \lhd, n\rangle \in \mathcal{L}(s)$, and $\langle\langle s,t\rangle, \rhd, n_1\rangle \in \mathcal{E}(R)$ is conjugated with $\langle\langle s,t\rangle, \lhd, n\rangle$, for some $R \sqsubseteq^* S$ with* Trans(R)*, then $\langle \exists R.C, \lhd, n\rangle \in \mathcal{L}(t)$,*

11. If $\langle \forall S.C, \rhd, n\rangle \in \mathcal{L}(s)$ and $\langle\langle s,t\rangle, \rhd', n_1\rangle \in \mathcal{E}(R)$ is conjugated with $\langle\langle s,t\rangle, \rhd^-, 1-n\rangle$, for some $R \sqsubseteq^* S$ with $\mathsf{Trans}(R)$, then $\langle \forall R.C, \rhd, n\rangle \in \mathcal{L}(t)$,

12. $\langle\langle s,t\rangle, \bowtie, n\rangle \in \mathcal{E}(R)$ iff $\langle\langle t,s\rangle, \bowtie, n\rangle \in \mathcal{E}(\mathsf{Inv}(R))$,

13. If $\langle\langle s,t\rangle, \rhd, n\rangle \in \mathcal{E}(R)$ and $R \sqsubseteq^* S$ then, $\langle\langle s,t\rangle, \rhd, n\rangle \in \mathcal{E}(S)$,

14. If $\langle \geq pR, \rhd, n\rangle \in \mathcal{L}(x)$, then $|\{t \in \boldsymbol{S} \mid \langle\langle s,t\rangle, \rhd, n\rangle \in \mathcal{E}(R)\}| \geq p$,

15. If $\langle \leq pR, \lhd, n\rangle \in \mathcal{L}(x)$, then $|\{t \in \boldsymbol{S} \mid \langle\langle s,t\rangle, \lhd^-, 1-n\rangle \in \mathcal{E}(R)\}| \geq p+1$,

16. If $\langle \geq pR, \lhd, n\rangle \in \mathcal{L}(x)$, then $|\{t \in \boldsymbol{S} \mid \langle\langle s,t\rangle, \rhd, n_i\rangle \in \mathcal{E}(R)\}| \leq p-1$, conjugated with $\langle\langle s,t\rangle, \lhd, n\rangle$,

17. If $\langle \leq pR, \rhd, n\rangle \in \mathcal{L}(x)$, then $|\{t \in \boldsymbol{S} \mid \langle\langle s,t\rangle, \rhd', n_i\rangle \in \mathcal{E}(R)\}| \leq p$ conjugated with $\langle\langle s,t\rangle, \rhd^-, 1-n\rangle$,

18. There do not exist two conjugated triples in any label of any individual $x \in \boldsymbol{S}$,

19. If $\langle a : C\bowtie n\rangle \in \mathcal{A}$, then $\langle C, \bowtie, n\rangle \in \mathcal{L}(\mathcal{V}(a))$,

20. If $\langle (a,b) : R\bowtie n\rangle \in \mathcal{A}$, then $\langle\langle \mathcal{V}(a), \mathcal{V}(b)\rangle, \bowtie, n\rangle \in \mathcal{E}(R)$,

21. If $a \not\doteq b \in \mathcal{A}$, then $\mathcal{V}(a) \neq \mathcal{V}(b)$

Properties 10 and 11 are a consequence of the fact that the supremum and infimum restrictions have to be preserved, when relations that have transitive sub-roles participate in negative existential and positive value restrictions. The membership degrees that the concepts are being propagated, in Properties 10 and 11, is the same as in the nodes that cause propagation. The proof of this property is quite technical and omitted here. Properties 14-17 are a direct consequence of the semantics of fuzzy number restrictions and the fact that from the De' Morgan laws we can establish equivalences between negative and positive triples.

Lemma 1. *An f_{KD}-$\mathcal{SHIN}$-ABox $\mathcal{A}$ is consistent w.r.t. $\mathcal{R}$ iff there exists a fuzzy tableau for $\mathcal{A}$ w.r.t. $\mathcal{R}$.*

3.1 The Tableaux Algorithm

In order to decide *ABox* consistency a procedure that constructs a fuzzy tableau for an f$_{KD}$-$\mathcal{SHIN}$ *ABox* has to be determined. In the current section we will provide the technical details for constructing a correct tableaux algorithm. As pointed out in [13] algorithms that decide consistency of an *ABox* work on *completion-forests* rather than on *completion-trees*. This is because an *ABox* might contain several individuals with arbitrary roles connecting them. Such a forest is a collection of trees that correspond to the individuals in the *ABox*.

Nodes in the completion-forest are labelled with a set of triples $\mathcal{L}(x)$ (*node triples*), which contain membership triples. More precisely we define $\mathcal{L}(x)\{\langle C, \bowtie, n\rangle\}$, where $C \in sub(\mathcal{A})$ and $n \in [0,1]$. Furthermore, edges $\langle x,y\rangle$ are labelled with a set $\mathcal{L}(\langle x,y\rangle)$ (*edge triples*) defined as, $\mathcal{L}(\langle x,y\rangle) = \{\langle R, \bowtie, n\rangle\}$, where $R \in \mathbf{R}_A$. The algorithm expands the tree either by expanding the set $\mathcal{L}(x)$, of a node x with new triples, or by adding new leaf nodes.

If nodes x and y are connected by an edge $\langle x,y\rangle$, then y is called a *successor* of x and x is called a *predecessor* of y, *ancestor* is the transitive closure of *predecessor*. A node x is called an $S-neighbour$ of a node x if for some R with

Table 2. Tableaux expansion rules

Rule	Description
$(\neg)$	if 1. $\langle \neg C, \bowtie, n\rangle \in \mathcal{L}(x)$ 2. and $\langle C, \bowtie^{-}, 1-n\rangle \notin \mathcal{L}(x)$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\langle C, \bowtie^{-}, 1-n\rangle\}$
$(\sqcap_{\rhd})$	if 1. $\langle C_1 \sqcap C_2, \rhd, n\rangle \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. $\{\langle C_1, \rhd, n\rangle, \langle C_2, \rhd, n\rangle\} \not\subseteq \mathcal{L}(x)$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\langle C_1, \rhd, n\rangle, \langle C_2, \rhd, n\rangle\}$
$(\sqcup_{\lhd})$	if 1. $\langle C_1 \sqcup C_2, \lhd, n\rangle \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. $\{\langle C_1, \lhd, n\rangle, \langle C_2, \lhd, n\rangle\} \not\subseteq \mathcal{L}(x)$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\langle C_1, \lhd, n\rangle, \langle C_2, \lhd, n\rangle\}$
$(\sqcup_{\rhd})$	if 1. $\langle C_1 \sqcup C_2, \rhd, n\rangle \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. $\{\langle C_1, \rhd, n\rangle, \langle C_2, \rhd, n\rangle\} \cap \mathcal{L}(x) = \emptyset$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C\}$ for some $C \in \{\langle C_1, \rhd, n\rangle, \langle C_2, \rhd, n\rangle\}$
$(\sqcap_{\lhd})$	if 1. $\langle C_1 \sqcap C_2, \lhd, n\rangle \mathcal{L}(x)$, x is not indirectly blocked, and 2. $\{\langle C_1, \lhd, n\rangle, \langle C_2, \lhd, n\rangle\} \cap \mathcal{L}(x) = \emptyset$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C\}$ for some $C \in \{\langle C_1, \lhd, n\rangle, \langle C_2, \lhd, n\rangle\}$
$(\exists_{\rhd})$	if 1. $\langle \exists R.C, \rhd, n\rangle \in \mathcal{L}(x)$, x is not blocked, 2. x has no R-neighbour y connected with a triple $\langle P^{*}, \rhd, n\rangle$, $P \sqsubseteq^{*} R$ and $\langle C, \rhd, n\rangle \in \mathcal{L}(y)$ then create a new node y with $\mathcal{L}(\langle x, y\rangle) = \{\langle R, \rhd, n\rangle\}$, $\mathcal{L}(y) = \{\langle C, \rhd, n\rangle\}$,
$(\forall_{\lhd})$	if 1. $\langle \forall R.C, \lhd, n\rangle \in \mathcal{L}(x)$, x is not blocked, 2. x has no R-neighbour y connected with a triple $\langle P^{*}, \lhd^{-}, 1-n\rangle$, $P \sqsubseteq^{*} R$ and $\langle C, \lhd, n\rangle \in \mathcal{L}(y)$ then create a new node y with $\mathcal{L}(\langle x, y\rangle) = \{\langle R, \lhd^{-}, 1-n\rangle\}$, $\mathcal{L}(y) = \{\langle C, \lhd, n\rangle\}$,
$(\forall_{\rhd})$	if 1. $\langle \forall R.C, \rhd, n\rangle \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. x has an R-neighbour y with $\langle C, \rhd, n\rangle \notin \mathcal{L}(y)$ and 3. $\langle *, \rhd^{-}, 1-n\rangle$ is conjugated with the positive triple that connects x and y then $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\langle C, \rhd, n\rangle\}$,
$(\exists_{\lhd})$	if 1. $\langle \exists R.C, \lhd, n\rangle \in \mathcal{L}(x)$, x is not indirectly blocked and 2. x has an R-neighbour y with $\langle C, \lhd, n\rangle \notin \mathcal{L}(y)$ and 3. $\langle *, \lhd, n\rangle$ is conjugated with the positive triple that connects x and y then $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\langle C, \lhd, n\rangle\}$,
$(\forall_{+})$	if 1. $\langle \forall R.C, \rhd, n\rangle \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. there is some P, with $\mathsf{Trans}(P)$, and $P \sqsubseteq^{*} R$, x has a P-neighbour y with, $\langle \forall P.C, \rhd, n\rangle \notin \mathcal{L}(y)$, and 3. $\langle *, \rhd^{-}, 1-n\rangle$ is conjugated with the positive triple that connects x and y then $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\langle \forall P.C, \rhd, n\rangle\}$,
$(\exists_{+})$	if 1. $\langle \exists R.C, \lhd, n\rangle \in \mathcal{L}(x)$, x is not indirectly blocked and 2. there is some P, with $\mathsf{Trans}(P)$, and $P \sqsubseteq^{*} R$, x has a P-neighbour y with, $\langle \exists P.C, \lhd, n\rangle \notin \mathcal{L}(y)$, and 3. $\langle *, \lhd, n\rangle$ is conjugated with the positive triple that connects x and y then $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\langle \exists P.C, \lhd, n\rangle\}$,
$(\geq_{\rhd})$	if 1. $\langle \geq pR, \rhd, n\rangle \in \mathcal{L}(x)$, x is not blocked, 2. there are no p R-neighbours $y_1, ..., y_p$ connected to x with a triple $\langle P^{*}, \rhd, n\rangle$, $P \sqsubseteq^{*} R$, 3. and $y_i \neq y_j$ for $1 \leq i < j \leq p$ then create p new nodes $y_1, ..., y_p$, with $\mathcal{L}(\langle x, y_i\rangle) = \{\langle R, \rhd, n\rangle\}$ and $y_i \neq y_j$ for $1 \leq i < j \leq p$
$(\leq_{\lhd})$	if 1. $\langle \leq pR, \lhd, n\rangle \in \mathcal{L}(x)$, x is not blocked, then apply $(\geq_{\rhd})$-rule for the triple $\langle \geq (p+1)R, \lhd^{-}, 1-n\rangle$
$(\leq_{\rhd})$	if 1. $\langle \leq pR, \rhd, n\rangle \in \mathcal{L}(x)$, x is not indirectly blocked, 2. there are $p+1$ R-neighbours $y_1, ..., y_{p+1}$ connected to x with a triple $\langle P^{*}, \rhd', n_i\rangle$, $P \sqsubseteq^{*} R$, 3. which is conjugated with $\langle P^{*}, \rhd^{-}, 1-n\rangle$, and there are two of them y, z, with no $y \neq z$ and 4. y is neither a root node nor an ancestor of z then 1. $\mathcal{L}(z) \rightarrow \mathcal{L}(z) \cup \mathcal{L}(y)$ and 2. if z is an ancestor of x then $\mathcal{L}(\langle z, x\rangle) \longrightarrow \mathcal{L}(\langle z, x\rangle) \cup \mathsf{Inv}(\mathcal{L}(\langle x, y\rangle))$ else $\mathcal{L}(\langle x, z\rangle) \longrightarrow \mathcal{L}(\langle x, z\rangle) \cup \mathcal{L}(\langle x, y\rangle)$ 3. $\mathcal{L}(\langle x, y\rangle) \longrightarrow \emptyset$ 4. Set $u \neq z$ for all u with $u \neq y$
$(\geq_{\lhd})$	if 1. $\langle \geq pR, \lhd, n\rangle \in \mathcal{L}(x)$, x is not indirectly blocked, then apply $(\leq_{\rhd})$-rule for the triple $\langle \leq (p-1)R, \lhd^{-}, 1-n\rangle$
$(\leq_{r_{\rhd}})$	if 1. $\langle \leq pR, \rhd, n\rangle \in \mathcal{L}(x)$, 2. there are $p+1$ R-neighbours $y_1, ..., y_{p+1}$ connected to x with a triple $\langle P^{*}, \rhd', n_i\rangle$, $P \sqsubseteq^{*} R$, 3. conjugated with $\langle P^{*}, \rhd^{-}, 1-n\rangle$, and there are two of them y, z, both root nodes, with no $y \neq z$ then 1. $\mathcal{L}(z) \rightarrow \mathcal{L}(z) \cup \mathcal{L}(y)$ and 2. For all edges $\langle y, w\rangle$: i. if the edge $\langle z, w\rangle$ does not exist, create it with $\mathcal{L}(\langle z, w\rangle) = \emptyset$ ii. $\mathcal{L}(\langle z, w\rangle) \longrightarrow \mathcal{L}(\langle z, w\rangle) \cup \mathcal{L}(\langle y, w\rangle)$ 3. For all edges $\langle w, y\rangle$: i. if the edge $\langle w, z\rangle$ does not exist, create it with $\mathcal{L}(\langle w, z\rangle) = \emptyset$ ii. $\mathcal{L}(\langle w, z\rangle) \longrightarrow \mathcal{L}(\langle w, z\rangle) \cup \mathcal{L}(\langle w, y\rangle)$ 4. Set $\mathcal{L}(y) = \emptyset$ and remove all edges to/from y 5. Set $u \neq z$ for all u with $u \neq y$ and set $y \doteq z$
$(\geq_{r_{\lhd}})$	if 1. $\langle \geq pR, \lhd, n\rangle \in \mathcal{L}(x)$, then apply $(\leq_{r_{\rhd}})$-rule for the triple $\langle \leq (p-1)R, \lhd^{-}, 1-n\rangle$

$R \sqsubseteq^* S$ either y is a successor of x and $\mathcal{L}(\langle x, y\rangle) = \langle R, \bowtie, n\rangle$ or y is a predecessor of x and $\mathcal{L}(\langle y, x\rangle) = \langle \mathsf{Inv}(R), \bowtie, n\rangle$. We then say that the edge triple *connects* x and y to a degree of n.

A node x is *blocked* iff it is not a root node and it is either directly or indirectly blocked. A node x is directly blocked iff none of its ancestors is blocked, and it has ancestors x', y and y' such that: (i) y is not a root node, (ii) x is a successor of x' and y a successor of y', (iii) $\mathcal{L}(x) = \mathcal{L}(y)$ and $\mathcal{L}(x') = \mathcal{L}(y')$ and, (iv) $\mathcal{L}(\langle x', x\rangle) = \mathcal{L}(\langle y', y\rangle)$. In this case we say that y blocks x. A node y is indirectly blocked iff none of its ancestors is blocked, or it is a successor of a node x and $\mathcal{L}(\langle x, y\rangle) = \emptyset$.

The algorithm initializes a forest $\mathcal{F}_{\mathcal{A}}$ to contain a root node x_0^i, for each individual $a_i \in \mathbf{I}_{\mathcal{A}}$ occurring in the *ABox* $\mathcal{A}$ and additionally $\{\langle C_i, \bowtie, n\rangle\} \cup \mathcal{L}(x_0^i)$, for each assertion of the form $\langle a_i : C_i \bowtie n\rangle$ in $\mathcal{A}$, and an edge $\langle x_0^i, x_0^j\rangle$ if $\mathcal{A}$ contains an assertion $\langle (a_i, a_j) : R_i \bowtie n\rangle$, with $\{\langle R_i, \bowtie, n\rangle\} \cup \mathcal{L}(\langle x_0^i, x_0^j\rangle)$ for each assertion of the form $\langle (a_i, a_j) : R_i \bowtie n\rangle$ in $\mathcal{A}$. At last we initialize the relation $\not\doteq$ as $x_0^i \not\doteq x_0^j$ if $a_i \not\doteq a_j \in \mathcal{A}$ and the relation $\doteq$ to be empty. $\mathcal{F}_{\mathcal{A}}$ is then expanded by repeatedly applying the rules from Table 2. We use the notation R^* to denote either the role R or the role returned by $\mathsf{Inv}(R)$, and the notation $\langle *, \bowtie, n\rangle$, to denote any role that participates in such a triple.

For a node x, $\mathcal{L}(x)$ is said to contain a clash if it contains one of the following: (a) two conjugated pairs of triples, (b) one of the triples $\langle \bot, \geq, n\rangle$, $\langle \top, \leq, n\rangle$, with $n > 0$, $n < 1$, $\langle \bot, >, n\rangle$, $\langle \top, <, n\rangle$ $\langle C, <, 0\rangle$ or $\langle C, >, 1\rangle$, (c) some triple $\langle \leq pR, \rhd, n\rangle \in \mathcal{L}(x)$ and x has $p+1$ R-neighbours $y_0, ..., y_p$, connected to x with a triple $\langle P^*, \rhd', n_i\rangle$, $P \sqsubseteq^* R$, which is conjugated with $\langle P^*, \rhd^-, 1-n\rangle$, and $y_i \not\doteq y_j$, for all $0 \leq i < j \leq p$, or (d) some triple $\langle \geq pR, \lhd, n\rangle \in \mathcal{L}(x)$ and x has p R-neighbours $y_0, ..., y_{p-1}$, connected to x with a triple $\langle P^*, \rhd, n_i\rangle$, $P \sqsubseteq^* R$, which is conjugated with $\langle P^*, \lhd, n\rangle$, and $y_i \not\doteq y_j$, for all $0 \leq i < j \leq p$. A completion-forest is *clash-free* if none of its nodes contains a clash, and it is *complete* if none of the expansion rules is applicable.

Lemma 2. *Let $\mathcal{A}$ be an f_{KD}-$\mathcal{SHIN}$ ABox and $\mathcal{R}$ a fuzzy RBox. Then*

1. *when started for $\mathcal{A}$ and $\mathcal{R}$ the tableaux algorithm terminates*
2. *$\mathcal{A}$ has a fuzzy tableau w.r.t. $\mathcal{R}$ if and only if the expansion rules can be applied to $\mathcal{A}$ and $\mathcal{R}$ such that they yield a complete and clash-free completion forest.*

4 Related Work

Much work has been carried out towards combining fuzzy logic and description logics during the last decade. The initial idea was presented by Yen in [19], where a *structural subsumption* algorithm was provided in order to perform reasoning. The DL language used was a sub-language of the basic DL $\mathcal{ALC}$. Reasoning in fuzzy $\mathcal{ALC}$ was latter presented in [14], as well as in other approaches [20, 21], where an additional concept constructor, called membership manipulator was included in the extended language. In all these approaches tableaux decision procedures were presented for performing reasoning services. The operations used to interpret the concept constructors in all these approaches were the same ones as in our context. Approaches towards more expressive DLs, are presented in [16], where the DL is $\mathcal{ALCQ}$, and in [17], where the language is $\mathcal{SHOIN}(\mathbf{D}^+)$. The former one also included fuzzy quantifiers, which is a new novel idea for fuzzy DLs. Unfortunately, in both these approaches only the semantics of the extended languages were provided and no reasoning algorithms. As far as we know the most expressive fuzzy DL presented till now, which also covers reasoning, is f_{KD}-$\mathcal{SI}$, appeared in [9]. The present work provides an extension of the latter one to an even more expressive DL, namely $\mathcal{SHIN}$.

5 Conclusions

The importance and role that uncertainty, like vagueness (fuzziness) and imprecision, plays in the Semantic web context, as well as to many applications that use DLs to capture, represent and perform reasoning with domain knowledge has been stressed many times in the literature [4–8]. To this extent we have presented an extension of the very expressive description logic $\mathcal{SHIN}$ with fuzzy set theory. Description logics are very powerful and expressive logical formalisms, which are used by ontology creation languages in the Semantic Web context. Moreover, fuzzy set theory is one very important theory for capturing and dealing with vagueness. Additionally, we have presented a detailed reasoning algorithm for deciding fuzzy ABox consistency. In order to achieve this goal we have provided an investigation of the properties of fuzzy cardinalities, in order to provide sound rules for such types of concept constructors. As far as future directions are concerned, these will include the extension of the $\mathcal{SHOIN}(\mathcal{G})$ description with fuzzy set theory. $\mathcal{SHOIN}(\mathcal{G})$ extends $\mathcal{SHIN}$ with nominals [22] and datatype groups [23].

Acknowledgements.

This work is supported by the FP6 Network of Excellence EU project Knowledge Web (IST-2004-507482).

References

1. Kim, W., Ko, J., Chung, M.: Uncertain robot environment modelling using fuzzy numbers. Fuzzy Sets and Systems **61** (1994) 53–62
2. Krishnapuram, R., Keller, J.: Fuzzy set theoretic approach to computer vision: An overview. In: IEEE International Conference on Fuzzy Systems. (1992) 135–142
3. Klir, G.J., Yuan, B.: Fuzzy Sets and Fuzzy Logic: Theory and Applications. Prentice-Hall (1995)
4. Stamou, G., Tzouvaras, V., Pan, J., Horrocks, I.: A fuzzy extension of swrl, W3C Workshop on Rule Languages for Interoperability (2005)
5. Matheus, C.: Using ontology-based rules for situation awareness and information fusion, W3C Work. on Rule Languages for Interoperability (2005)

6. Kifer, M.: Requirements for an expressive rule language on the semantic web, W3C Workshop on Rule Languages for Interoperability (2005)
7. Zhang, L., Yu, Y., Zhou, J., Lin, C., Yang, Y.: An enhanced model for searching in semantic portals. In: Int. WWW Conference Committee. (2005)
8. Bechhofer, S., Goble, C.: Description Logics and Multimedia - Applying Lessons Learnt from the GALEN Project. In: KRIMS 96 Workshop on Knowledge Representation for Interactive Multimedia Systems, ECAI 96, Budapest (1996)
9. Stoilos, G., Stamou, G., Tzouvaras, V., Pan, J., Horrocks, I.: A fuzzy description logic for multimedia knowledge representation, Proc. of the International Workshop on Multimedia and the Semantic Web (2005)
10. Lassila, O., R.Swick, R.: Resource Description Framework (RDF) Model and Syntax Specification – W3C Recommendation 22 February 1999. Technical report, World Wide Web Consortium (1999)
11. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., eds., L.A.S.: OWL Web Ontology Language Reference (2004)
12. Horrocks, I., Sattler, U.: A description logic with transitive and inverse roles and role hierarchies. Journal of Logic and Computation **9** (1999) 385–410
13. Horrocks, I., Sattler, U., Tobies, S.: Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In MacAllester, D., ed.: CADE-2000. Number 1831 in LNAI, Springer-Verlag (2000) 482–496
14. Straccia, U.: Reasoning within fuzzy description logics. Journal of Artificial Intelligence and Research **14** (2001) 137–166
15. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99). Number 1705 in LNAI, Springer-Verlag (1999) 161–180
16. Sánchez, D., Tettamanzi, G.: Generalizing quantification in fuzzy description logic. In: Proceedings 8th Fuzzy Days in Dortmund. (2004)
17. Straccia, U.: Towards a fuzzy description logic for the semantic web. In: Proceedings of the 2nd European Semantic Web Conference. (2005)
18. Hollunder, B., Nutt, W., Schmidt-Schaus, M.: Subsumption algorithms for concept description languages. In: European Conference on Artificial Intelligence. (1990) 348–353
19. Yen, J.: Generalising term subsumption languages to fuzzy logic. In: In Proc of the 12th Int. Joint Conf on Artificial Intelligence (IJCAI-91). (1991)
20. Tresp, C., Molitor, R.: A description logic for vague knowledge. In: In proc of the 13th European Conf. on Artificial Intelligence (ECAI-98). (1998)
21. Hölldobler, S., Khang, T.D., Störr, H.P.: A fuzzy description logic with hedges as concept modifiers. In: Proceedings InTech/VJFuzzy'2002. (2002) 25–34
22. Horrocks, I., Sattler, U.: Ontology reasoning in the SHOQ(D) description logic. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence. (2001)
23. Pan, J.Z.: Web Ontology Reasoning in the SHOQ(Dn) Description Logic. In: Carlos Areces and Maartin de Rijke, editors,Proceedings of the Methods for Modalities 2 (M4M-2). (2001) ILLC, University of Amsterdam.

A Generic Framework for Description Logics with Uncertainty

Volker Haarslev, Hsueh-Ieng Pai, and Nematollaah Shiri

Concordia University
Dept. of Computer Science & Software Engineering
Montreal, Quebec, Canada
{haarslev, hsueh_pa, shiri}@cse.concordia.ca

Abstract. We propose an extension to Description Logics (DLs) with uncertainty which unifies and/or generalizes a number of existing frameworks for DLs with uncertainty. To this end, we first give a classification of these frameworks and identify the essential features as well as properties of the various combination functions allowed in the underlying uncertainty formalisms they model. This also allows us express the semantics of the DL elements in a flexible manner. We illustrate how various DLs with uncertainty can be expressed in our generic framework.

1 Introduction

Ever since Tim Berners-Lee introduced the vision of the Semantic Web [2], attempts have been made on making Web resources more machine-interpretable by giving Web resources a well-defined meaning through semantic markups. One way to encode such semantic markups is using ontologies. Over the last few years, a number of ontology languages have been developed, most of which have a foundation based on *Description Logics* (DLs) [1]. The family of DLs is a subset of first-order logic (FOL) that is considered to be attractive as it keeps a good compromise between expressive power and computational tractability.

Uncertainty management has been a challenge for over two decades in database (DB) and artificial intelligence (AI) research (see [10, 12]), and has recently attracted the attention of the DL community. *Uncertainty* is a form of deficiency or imperfection commonly found in the real-world information/data. A piece of information is uncertain if its truth is not established definitely.

Despite of the popularity of DLs, it has been realized that the standard DL framework is inadequate to model uncertainty. For example, in the medical domain, one might want to express that: "It is very likely that an obese person would have heart disease", where "obese" is a vague concept that may vary across regions or countries, and "likely" shows the uncertain nature of the knowledge. Such knowledge cannot be expressed within the current scope of DL.

Recently, a number of proposals have been put forward which extend DLs with uncertainty. Some of them deal with the vagueness aspect while others deal with the uncertainty aspect. We do not intend to compare which extension is better. In fact, different applications may require using different aspects. It may even be desired in some cases to model different aspects within the same application.

Following the parametric approach proposed in [11], we propose a generic DL framework with uncertainty in this paper as a unifying umbrella for several existing frameworks for DLs with uncertainty. This not only provides a uniform access over knowledge that has been encoded using DL with various kinds of uncertainty, but also allows one to study various problems, including semantics, query processing and optimization, design, and implementation of such frameworks in a framework-independent manner.

The rest of this paper is organized as follows. Section 2 presents a classification of existing frameworks of uncertainty in DL. In section 3, we present our generic framework for DL with uncertainty in detail, along with examples illustrating how it can represent uncertainty modeled in the frameworks considered. Concluding remarks and some future directions are presented in section 4.

2 Approaches to DL with Uncertainty

On the basis of their mathematical foundation and the type of uncertainty modeled, we can classify existing proposals of DLs with uncertainty into three approaches: fuzzy, probabilistic, and possibilistic approach.

The fuzzy approach, based on fuzzy set theory [19], deals with the vagueness in the knowledge, where a proposition is true to only some degree. For example, the statement "Jason is obese with degree 0.4" indicates Jason is slightly obese. Here, the value 0.4 is the degree of membership that Jason is in concept obese.

The probabilistic approach, based on the classical probability theory, deals with the uncertainty due to lack of knowledge, where a proposition is either true or false, but one does not know for sure which one is the case. Hence, the certainty value refers to the probability that the proposition is true. For example, one could state that: "The probability that Jason would have heart disease given that he is obese lies in the range $[0.8, 1]$."

Finally, the possibilistic approach, based on possibility theory [20], allows both certainty (necessity measure) and possibility (possibility measure) be handled in the same formalism. For example, by knowing that "Jason's weight is above 80 Kg", the proposition "Jason's weight is 80 Kg" is necessarily true with certainty 1, while "Jason's weight is 90 Kg" is possibly true with certainty 0.5.

3 A Generic Framework for DL with Uncertainty

To incorporate uncertainty into DL, each component of the classical DL framework needs to be extended, as shown in Fig. 1. To be more specific, any framework for DL with uncertainty consists of the following three components.

1. *Description Language with Uncertainty:* The syntax and semantics of the description language are extended to enable expression of uncertainty.

2. *Knowledge Bases with Uncertainty:* A knowledge base is composed of the *intensional* knowledge, i.e., TBox (for terminological axioms) and RBox (for role axioms), both extended with uncertainty, and *extensional* knowledge, i.e., ABox (for assertions), with uncertainty.
3. *Reasoning with Uncertainty:* The DL framework is equipped with reasoning services that take into account the presence of uncertainties in DL theories during the reasoning process.

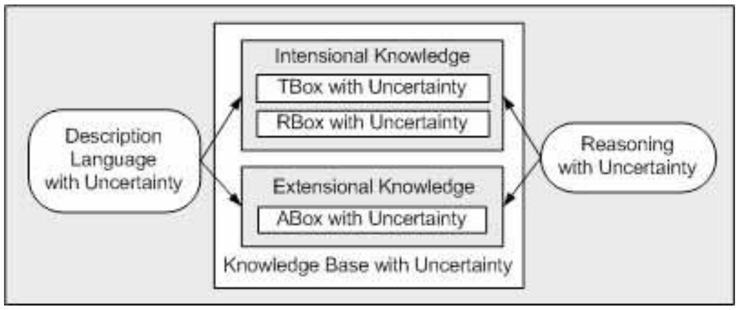


Fig. 1. DL Framework with Uncertainty

In what follows, we focus more on the first two components and discuss them in detail. The third component, the reasoning aspect, is under investigation. This section ends with some examples illustrating how different notions of uncertainty could be represented using our generic framework.

3.1 Description Language with Uncertainty

To provide a generic extension to a description language, one needs to develop a way to represent certainty values, and assign semantics to each element in the description language.

Representation of Certainty Values. To represent the certainty values, we take a *lattice-based approach* followed in the parametric framework [11]. That is, we assume that certainty values form a complete lattice shown as $\mathcal{L} = \langle \mathcal{V}, \preceq \rangle$, where $\mathcal{V}$ is the certainty domain, and $\preceq$ is the partial order defined on $\mathcal{V}$. We also use b to denote the bottom or least element in $\mathcal{V}$, and use t to denote the top or greatest value in $\mathcal{V}$. The least upper bound operator (the join operator) in $\mathcal{L}$ is denoted by $\oplus$, its greatest lower bound (the meet operator) is denoted by $\otimes$, and its negation operator is denoted by $\sim$.

The certainty lattice can be used to model both *qualitative* and *quantitative* certainty values. An example for the former is classical logic which uses the binary values $\{0, 1\}$. For the latter, an example would be a family of multi-valued logics such as fuzzy logic which uses $[0, 1]$ as the certainty domain.

Assignment of Semantics to Description Language. The generic framework treats each type of uncertainty formalism as a special case. Hence, it would be restrictive to consider any specific function to describe the semantics of the description language constructors (e.g., fixing *min* as the function to determine the certainty of concept conjunction). An alternative is proposed in our generic framework to allow a user to specify the functions that are appropriate to define the semantics of the description language element at axiom or assertion level. We elaborate more on this in section 3.2.

To make sure that the combination functions specified by a user make sense, we assume the following properties for various certainty functions to be reasonable. Most of these properties were recalled from [11], and are reasonable and justified when we verify them against existing extensions of DL with uncertainty. To present these properties, we consider the description language constructors in $\mathcal{ALC}$ [1]. We assume that the reader has the basic knowledge about $\mathcal{ALC}$.

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation, where $\Delta^{\mathcal{I}}$ is the domain and $\cdot^{\mathcal{I}}$ is an interpretation function that maps description language elements to some certainty value in $\mathcal{V}$.

Atomic Concept. The interpretation of an atomic concept A is a certainty value in the certainty domain, i.e., $A^{\mathcal{I}}(a) \in \mathcal{V}$, for all individuals $a \in \Delta^{\mathcal{I}}$. For example, in the fuzzy approach, the interpretation of an atomic concept A is defined as $A^{\mathcal{I}}(a) \in [0, 1]$, that is, the interpretation function assigns to every individual a in the domain, a value in the unit interval that indicates its membership to A.

Atomic Role. Similar to atomic concepts, the interpretation of an atomic role R is a certainty value in the certainty domain, i.e., $R^{\mathcal{I}}(a, b) \in \mathcal{V}$, for all individuals $a, b \in \Delta^{\mathcal{I}}$.

Top/Universal Concept. The interpretation of the top or universal concept $\top$ is the greatest value in $\mathcal{V}$, that is, $\top^{\mathcal{I}} = t$. For instance, in fuzzy, probabilistic, and possibilistic approaches, the interpretation of $\top$ is 1, or true in standard logic.

Bottom Concept. The interpretation of the bottom concept $\bot$ is the least value in $\mathcal{V}$, that is, $\bot^{\mathcal{I}} = b$. For example, in fuzzy, probabilistic, and possibilistic approaches, the interpretation of $\bot$ is 0, or false in standard logic.

Concept Negation. Given a concept C, the interpretation of concept negation $\neg C$ is defined by the negation function $\sim$, which is a mapping from $\mathcal{V}$ to $\mathcal{V}$, and should satisfy the following properties:

- Boundary Conditions: $\sim b = t$ and $\sim t = b$.
- Double Negation: For each certainty value $\alpha \in \mathcal{V}$, we have that $\sim(\sim\alpha) = \alpha$.

In our presentation here, we consider the negation operator $\sim$ in the certainty lattice as the default negation function. Other properties of the negation function, such as monotonicity (i.e., $\forall\alpha, \beta \in \mathcal{V}$, if $\alpha \preceq \beta$, then $\sim\alpha \succeq \sim\beta$), may be imposed if necessary. The most common interpretation of $\neg C$ is (1 − certainty of C), where $\mathcal{V} = [0, 1]$.

Before we present the properties of functions that are appropriate to describe the semantics of concept conjunction and disjunction, we first identify a set of desired properties which *combination function* f should satisfy. These functions are used to combine a collection of certainty values into one value. We then identify a subset of these properties suitable for describing the semantics of logical formulas on the basis of concept conjunction and disjunction. Note that, since f is used to combine a collection of certainty values into one, we describe f as a binary function from $\mathcal{V} \times \mathcal{V}$ to $\mathcal{V}$. This view is clearly without loss of generality and, at the same time, useful for implementing functions.

1. Monotonicity: $f(\alpha_1, \alpha_2) \preceq f(\beta_1, \beta_2)$, whenever $\alpha_i \preceq \beta_i$, for $i = 1, 2$.
2. Bounded Value (Above): $f(\alpha_1, \alpha_2) \preceq \alpha_i$, for $i = 1, 2$.
3. Bounded Value (Below): $f(\alpha_1, \alpha_2) \succeq \alpha_i$, for $i = 1, 2$.
4. Boundary Condition (Above): $\forall \alpha \in \mathcal{V}$, $f(\alpha, b) = \alpha$ and $f(\alpha, t) = t$.
5. Boundary Condition (Below): $\forall \alpha \in \mathcal{V}$, $f(\alpha, t) = \alpha$ and $f(\alpha, b) = b$.
6. Commutativity: $\forall \alpha, \beta \in \mathcal{V}$, $f(\alpha, \beta) = f(\beta, \alpha)$.
7. Associativity: $\forall \alpha, \beta, \delta \in \mathcal{V}$, $f(\alpha, f(\beta, \delta)) = f(f(\alpha, \beta), \delta)$.

Concept Conjunction. Given concepts C and D, the interpretation of concept conjunction $C \sqcap D$ is defined by the conjunction function f_c that should satisfy properties 1, 2, 5, 6, and 7. The monotonicity property is required so that the reasoning is monotone, i.e., whatever that has been proven so far will remain true for the rest of the reasoning process. The bounded value property is included so that the interpretation of the certainty values makes sense. Note that this property also implies the boundary condition (property 5). The commutativity property supports reordering of the arguments of the conjunction operator, and associativity ensures that different evaluation order of a conjunction of concepts does not change the result. These properties are useful during the runtime evaluation used by the reasoning procedure. Examples of conjunctions include the usual product $\times$ and min functions, and bounded difference defined as $bDiff(\alpha, \beta) = max(0, \alpha + \beta - 1)$.

Concept Disjunction. Given concepts C and D, the interpretation of concept disjunction $C \sqcup D$ is defined by the disjunction function f_d that should satisfy properties 1, 3, 4, 6, and 7. The monotonicity, bounded value, boundary condition, commutativity, and associativity properties are required for similar reasons as the conjunction case. Some common disjunction functions are: the standard max function, the probability independent function defined as $ind(\alpha, \beta) = \alpha + \beta - \alpha\beta$, and the bounded sum function defined as $bSum(\alpha, \beta) = min(1, \alpha + \beta)$.

Role Value Restriction. Given a role R and a role filler C, the interpretation of role value restriction $\forall R.C$ is defined as follows:

$\forall a \in \Delta^{\mathcal{I}}, (\forall R.C)^{\mathcal{I}}(a) = \otimes_{b \in \Delta^{\mathcal{I}}} \{ f_d({\sim} R^{\mathcal{I}}(a, b), C^{\mathcal{I}}(b)) \}$

The intuition behind this definition is to view $\forall R.C$ as the open first order formula $\forall b.\ R(a, b) \to C(b)$, where $R(a, b) \to C(b)$ is equivalent to $\neg R(a, b) \vee C(b)$, and $\forall$ is viewed as a conjunction over the elements of the domain. To be more specific, the semantics of $\neg R(a,b)$ is captured using the negation function $\sim$ as $\sim R^{\mathcal{I}}(a,b)$, the semantics of $\neg R(a,b) \vee C(b)$ is captured using the disjunction function as $f_d(\sim R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))$, and $\forall b$ is captured using the meet operator in the lattice $\otimes_{b \in \Delta^{\mathcal{I}}}$.

Role Exists Restriction. Given a role R and a role filler C, the interpretation of role exists restriction $\exists R.C$ is defined as follows:

$\forall a \in \Delta^{\mathcal{I}}$, $(\exists R.C)^{\mathcal{I}}(a) = \oplus_{b \in \Delta^{\mathcal{I}}} \{f_c(R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))\}$

The intuition for this property is that we view $\exists R.C$ as the open first order formula $\exists b.\ R(a,b) \wedge C(b)$, where $\exists$ is viewed as a disjunction over the elements of the domain. To be more specific, the semantics of $R(a,b) \wedge C(b)$ is captured using the conjunction function as $f_c(R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))$, and $\exists b$ is captured using the join operator in the lattice $\oplus_{b \in \Delta^{\mathcal{I}}}$.

Additional Inter-Constructor Properties. In addition to the above-mentioned properties, we also assume that the following inter-constructor properties hold:

– De Morgan's Rule: $\neg(C \sqcup D) \equiv \neg C \sqcap \neg D$ and $\neg(C \sqcap D) \equiv \neg C \sqcup \neg D$.
– Negating Quantifiers Rule: $\neg\exists R.C \equiv \forall R.\neg C$ and $\neg\forall R.C \equiv \exists R.\neg C$.

The above two properties are needed to convert a concept description into *negation normal form* (NNF), i.e., the negation operator appears only in front of a concept name. Note that the above properties affect the type of negation, conjunction, and disjunction functions that may be used in the generic framework.

3.2 Knowledge Bases with Uncertainty

As in the classical counterpart, a *knowledge base* Σ in the generic framework is a triple $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a TBox, $\mathcal{R}$ is an RBox, and $\mathcal{A}$ is an ABox.

An interpretation $\mathcal{I}$ *satisfies* a knowledge base Σ (or $\mathcal{I} \models \Sigma$) iff it satisfies each element of Σ (i.e., $\mathcal{T}$, $\mathcal{R}$, and $\mathcal{A}$). We say that Σ is *satisfiable* (or $\Sigma \not\models \bot$) iff there exists an interpretation $\mathcal{I}$ that satisfies Σ. Similarly, Σ is *unsatisfiable* (or $\Sigma \models \bot$) iff there does not exist any interpretation $\mathcal{I}$ that satisfies Σ.

To provide a generic extension to the knowledge base, there is a need to give a syntactical and semantical extension to both the intensional (TBox and RBox) and extensional knowledge (ABox).

A TBox $\mathcal{T}$ consists of a set of terminological axioms expressed in the form $\langle C \sqsubseteq D, \alpha \rangle \langle f_c, f_d, f_p \rangle$ or $\langle C \equiv D, \alpha \rangle \langle f_c, f_d, f_p \rangle$, where C and D are concepts, $\alpha \in \mathcal{V}$ is the certainty that the axiom holds, f_c is the conjunction function used as the semantics of concept conjunction and part of the role exists restriction, f_d is the disjunction function used as the semantics of concept disjunction and part of the role value restriction, and f_p is the propagation function that is used to propagate the truth value from the LHS of the subsumption to the RHS.

Similar to the description language case, we have identified a set of properties that should hold for a propagation function. In general, the propagation function f_p is a combination function, and should satisfy the monotonicity and bounded value (above) properties, as specified in section 3.1. Note that these two properties are a subset of those required by the conjunction functions, and they are needed for similar reasons as the conjunction case. Some commonly used propagation functions are the algebraic product $\times$ and the standard *min* function.

As usual, the concept definition $\langle C \equiv D, \alpha\rangle\langle f_c, f_d, f_p\rangle$ is defined as $\langle C \sqsubseteq D, \alpha\rangle$ $\langle f_c, f_d, f_p\rangle$ and $\langle D \sqsubseteq C, \alpha\rangle\langle f_c, f_d, f_p\rangle$.

The RBox is similar to TBox except that we have role axioms instead of terminological axioms. In addition, only a propagation function is specified, but not a conjunction or disjunction functions. Since existing DL frameworks with uncertainty do not allow role conjunction or role disjunction, we do not consider them in the generic framework either.

An ABox $\mathcal{A}$ consists of a set of assertions of the form $\langle a : C, \alpha\rangle\langle f_c, f_d, -\rangle$ or $\langle (a, b) : R, \alpha\rangle\langle -, -, -\rangle$, where a and b are individuals, C is a concept, R is a role, $\alpha \in \mathcal{V}$, f_c is the conjunction function, and f_d is the disjunction function. Note that, unlike in axioms, the propagation function is not needed in the assertion, hence we use "$-$" as the placeholder to keep the uniformity of the presentation.

3.3 Knowledge Representation

In this section, we first illustrate the capabilities of the generic framework for representing classical DL. We then show how existing DLs with uncertainty can be represented by the generic framework.

*Example 1 (*Classical DL*).* The classical DL knowledge base can be represented in the generic framework as follows. The certainty lattice is defined as $\mathcal{L} = \langle \mathcal{V}, \leq\rangle$, where $\mathcal{V} = \{0, 1\}$, with *max* as the join operator $\oplus$ and *min* as the meet operator $\otimes$. Also, the negation operator is defined as $\sim 0 = 1$ and $\sim 1 = 0$. The certainty value associated with each axiom and assertion is set to 1. Finally, the conjunction function (f_c) is *min*, the disjunction function (f_d) is *max*, and the propagation function (f_p) is *min*. For example, consider the following classical knowledge base:

$$\begin{aligned}\mathcal{T}=\{&\langle Parent \equiv Person \sqcap \exists(hasChildByBirth.Person \\ &\qquad \sqcup\ hasStepChild.Person \sqcup\ hasAdoptedChild.Person)\rangle, \\ &\langle Mother \equiv Parent \sqcap Female\rangle\}\end{aligned}$$

$$\begin{aligned}\mathcal{A}=\{&\langle Mary : Person \sqcap Female\rangle, \\ &\langle (Mary, Joe) : hasStepChild\rangle,\ \langle Joe : Person\rangle\}\end{aligned}$$

This knowledge base can be represented in the generic framework as follows:

$$\begin{aligned}\mathcal{T}=\{&\langle Parent \equiv Person \sqcap \exists(hasChildByBirth.Person \sqcup hasStepChild. \\ &\qquad Person \sqcup\ hasAdoptedChild.Person),\ 1\rangle\ \langle min, max, min\rangle, \\ &\langle Mother \equiv Parent \sqcap Female, 1\rangle\langle min, -, min\rangle\}\end{aligned}$$

$$\begin{aligned}\mathcal{A}=\{&\langle Mary : Person \sqcap Female, 1\rangle\langle min, -, -\rangle, \\ &\langle (Mary, Joe) : hasStepChild, 1\rangle\langle -, -, -\rangle,\ \langle Joe : Person, 1\rangle\langle -, -, -\rangle\}\end{aligned}$$

*Example 2 (*Fuzzy DL*).* Most of the proposed fuzzy DL ('most' because our framework supports only $\mathcal{ALC}$) can be represented in the generic framework by setting the certainty lattice as $\mathcal{L} = \langle \mathcal{V}, \leq \rangle$, where $\mathcal{V} = \mathcal{C}[0,1]$ is the set of closed intervals $[\alpha, \beta]$ in [0, 1] such that $\alpha \preceq \beta$. The negation operator in this case is defined as $\sim([a_1, a_2]) = [1 - a_2, 1 - a_1]$. In [5, 13, 15, 16, 18], the meet operator is inf (infimum) and the join operator is sup (supremum). On the other hand, [17] uses min as the meet operator, and max as the join. The conjunction function used in all these proposals is min, whereas the disjunction function uses max. Note that existing proposals rarely allow certainty values to be associated with both axioms and assertions. Moreover, they do not discuss how to combine the certainty value of an assertion with the certainty value of an axiom, and hence existing frameworks do not specify any propagation function.

As an example, suppose we have the following fuzzy knowledge base:

$\mathcal{T}=\{\langle (\mathit{Old} \sqcup \mathit{WellEducated}) \sqcap \mathit{LikesLearning} \sqsubseteq \mathit{Knowledgeable} \geq 0.8 \rangle\}$

$\mathcal{A}=\{\langle \mathit{John} : \mathit{Old} \geq 0.6 \rangle, \langle \mathit{John} : \mathit{WellEducated} \geq 0.7 \rangle,$
$\langle \mathit{John} : \mathit{LikesLearning} = 0.9 \rangle\}$

Then, we could infer that John is knowledgeable with degree in [0.7, 0.9].
This knowledge base can be represented in the generic framework as follows:

$\mathcal{T}=\{\langle (\mathit{Old} \sqcup \mathit{WellEducated}) \sqcap \mathit{LikesLearning} \sqsubseteq \mathit{Knowledgeable}, [0.8, 1] \rangle$
$\langle \mathit{min, max, min} \rangle\}$

$\mathcal{A}=\{\langle \mathit{John} : \mathit{Old}, [0.6, 1] \rangle \langle -, -, - \rangle, \langle \mathit{John} : \mathit{WellEducated}, [0.7, 1] \rangle \langle -, -, - \rangle,$
$\langle \mathit{John} : \mathit{LikesLearning}, [0.9, 0.9] \rangle \langle -, -, - \rangle\}$

We obtain $\langle \mathit{John} : \mathit{Knowledgeable}, [0.7, 0.9] \rangle \langle -, -, - \rangle$ as the inference result.

*Example 3 (*Probabilistic DL*).* A possible certainty lattice for probabilistic DL is $\mathcal{L} = \langle \mathcal{V}, \leq \rangle$, where $\mathcal{V} = \mathcal{C}[0, 1]$, with negation operator sets to be $\sim([a_1, a_2])$ $= [1 - a_2, 1 - a_1]$. Note that this allows us to express both interval probability (such as [0.4, 0.8]) and exact probability (e.g., [0.8, 0.8]).

Currently existing probabilistic extensions to DLs, such as [3, 4, 7, 8], support mainly conditional constraints. In the generic framework, we view a rule as a conditional statement. As such, let α be some value from the certainty lattice, a conditional constraint of the form $P(A \mid B) : \alpha$ can be expressed as the rule $A \overset{\alpha}{\leftarrow} B$ in IB framework of logic programming, which in turn can be expressed as $\langle B \sqsubseteq A, \alpha \rangle$ in our generic framework.

For illustration purpose, we show how a simple knowledge base with conditional constraints from [4] can be represented in our generic framework. Consider a knowledge base with the following expressions:

$\mathcal{T}=\{\langle \mathit{PacemakerPatient} \sqsubseteq \mathit{HeartPatient} \rangle,$
$\langle (\exists \mathit{hasPrivateInsurance}.\{\mathit{Yes}\} | \mathit{HeartPatient})[0.9, 1] \rangle\}$

$\mathcal{A}=\{\langle (\mathit{PacemakerPatient} | \{\mathit{John}\})[0.8, 1] \rangle\}$

Then, one could infer that $\{\langle (\exists \mathit{hasPrivateInsurance}.\{\mathit{Yes}\} | \{\mathit{John}\})[0.72, 1] \rangle\}$
The above knowledge base can be expressed in the generic framework as follows:

$\mathcal{T}=\{\langle \mathit{PacemakerPatient} \sqsubseteq \mathit{HeartPatient}, [1, 1] \rangle \langle -, -, \times \rangle,$
$\langle \mathit{HeartPatient} \sqsubseteq \exists \mathit{hasPrivateInsurance}.\{\mathit{Yes}\}, [0.9, 1] \rangle \langle -, -, \times \rangle\}$

$\mathcal{A}=\{\langle \mathit{John} : \mathit{PacemakerPatient}, [0.8, 1] \rangle \langle -, -, - \rangle\}$

With the first axiom and the first assertion, we infer that John is HeartPatient with probability $\times([0.8, 1], [1, 1]) = [0.8, 1]$ (since f_p in the first axiom is $\times$). Then, this assertion together with the second axiom allow us to infer that John has private insurance with probability $\times([0.8, 1], [0.9, 1]) = [0.72, 1]$ (since the f_p in the second axiom is $\times$), as inferred in [4].

Unlike in other uncertainty formalisms, reasoning with probability requires extra information/knowledge. Hence, although our framework can easily handle simple probabilities, such as independent events and mutually exclusive events, more complex probability modes such as positive/negative correlation [9], ignorance [9], and conditional independence [14] are still under investigation.

Example 4 (Possibilistic DL). Hollunder [6] is the only proposal that gives a possibilistic extension to DL. Here, the possibility (Π) and necessity (N) degrees can be represented by the certainty lattice $\mathcal{L} = \langle \mathcal{V}, \preceq \rangle$, where $\mathcal{V} = \mathcal{C}[0, 1]$, with negation operator sets to be $\sim([a_1, a_2]) = [1 - a_2, 1 - a_1]$. The conjunction and propagation functions are min, and the disjunction function is max.
As an example, in [6], we have the following knowledge base:
$\mathcal{T}$=$\{\langle \exists owns.Porsche \sqsubseteq RichPerson \sqcup CarFanatic \geq N0.8 \rangle,$
$\langle RichPerson \sqsubseteq Golfer \geq \Pi 0.7 \rangle\}$
$\mathcal{A}$=$\{\langle Tom : \exists owns.Porsche \geq N1 \rangle, \langle Tom : \neg CarFanatic \geq N0.7 \rangle\}$
Then, according to [6], one could infer that $\langle Tom : Golfer \geq \Pi 0.7 \rangle$.
The above knowledge base can be simulated in the generic framework as follows:
$\mathcal{T}$=$\{\langle \exists owns.Porsche \sqsubseteq RichPerson \sqcup CarFanatic, [0.8, 1] \rangle$
$\langle min, max, min \rangle, \langle RichPerson \sqsubseteq Golfer, [0.7, 1] \rangle \langle -, -, min \rangle\}$
$\mathcal{A}$=$\{\langle Tom : \exists owns.Porsche, [1, 1] \rangle \langle min, -, - \rangle,$
$\langle Tom : \neg CarFanatic, [0.7, 1] \rangle \langle -, -, - \rangle\}$
The result of the inference is $\langle Tom : Golfer, [0.7, 1] \rangle \langle -, -, - \rangle$, as obtained in [6].

4 Conclusion and Future Work

We have proposed a generic framework that allows us to unify and/or generalize various extensions of DLs with uncertainty, taking an axiomatic approach. In particular, we abstracted away both the underlying notion of uncertainty (fuzzy logic, probability, possibilistic logic), and the way in which the constructors in the description language are interpreted. This was done by identifying the essential properties that various combination functions should possess in order to realize our unifying framework. We are currently investigating ways to specify generic reasoning services that the proposed framework should support, for which we have some partial result. Other future work includes extending this framework to a more expressive DL, for instance $\mathcal{SHOIN}$. A study of suitable query processing and optimization techniques is an important future work. An implementation of the proposed generic framework is underway.

Acknowledgements: This work is supported in part by Natural Sciences and Engineering Research Council (NSERC) of Canada, and by ENCS, Concordia University.

References

1. Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.
2. Berners-Lee, T., Hendler, J., and Lassila, O. The Semantic Web. *Scientific American*, 284(5), May 2001.
3. Ding, Z., Peng, Y., and Pan, R. A Bayesian approach to uncertainty modeling in OWL ontology. In *Proceedings of AISTA-04*, Luxembourg, November 2004.
4. Giugno, R. and Lukasiewicz, T. P-$\mathcal{SHOQ}$(D): A probabilistic extension of $\mathcal{SHOQ}$(D) for probabilistic ontologies in the Semantic Web. In *Proceedings of the JELIA-02*, pages 86–97, London, UK, 2002. Springer-Verlag. Lecture Notes In Computer Science; Vol. 2424.
5. Hölldobler, S., Khang, T. D., and Störr, H.-P. A fuzzy description logic with hedges as concept modifiers. In *Proceedings of InTech/VJFuzzy-02*, pages 25–34, Hanoi, Vietnam, 2002. Science and Technics Publishing House.
6. Hollunder, B. An alternative proof method for possibilistic logic and its application to terminological logics. In *Proceedings of UAI-94*, pages 327–335, San Francisco, CA, 1994. Morgan Kaufmann.
7. Jaeger, M. Probabilistic reasoning in terminological logics. In *Proceedings of KR-94*, pages 305–316, 1994.
8. Koller, D., Levy, A. Y., and Pfeffer, A. P-CLASSIC: A tractable probablistic description logic. In *Proceedings of AAAI-97*, pages 390–397, Providence, Rhode Island, July 1997. AAAI Press.
9. Lakshmanan, L.V.S. and Sadri, F. Probabilistic deductive databases. In *Proceedings of ILPS-94*, pages 254–268, Ithaca, NY, November 1994. MIT Press.
10. Lakshmanan, L.V.S. and Shiri, N. Logic programming and deductive databases with uncertainty: A survey. In *Enclyclopedia of Computer Science and Technology*, volume 45, pages 153–176. Marcel Dekker, Inc., New York, 2001.
11. Lakshmanan, L.V.S. and Shiri, N. A parametric approach to deductive databases with uncertainty. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):554–570, 2001.
12. Parsons, S. Current approaches to handling imperfect information in data and knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 8(3):353–372, 1996.
13. Sánchez, D. and Tettamanzi, A. G. B. Generalizing quantification in fuzzy description logics. In *Proceedings of Fuzzy Days-04*. Springer-Verlag, 2004.
14. Shenoy P.P. Conditional independence in valuation-based systems. *Journal of Approximate Reasoning*, 10(3):203–234, 1994.
15. Straccia, U. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.
16. Straccia, U. Transforming fuzzy description logics into classical description logics. In *Proceedings of JELIA-04*, pages 385–399. Springer-Verlag, 2004. Lecture Notes In Computer Science; Vol. 3229.
17. Stracia, U. A fuzzy description logic. In *Proceedings of AAAI-98*, pages 594–599, Menlo Park, CA, USA, 1998. AAAI Press.
18. Tresp, C. and Molitor, R. A description logic for vague knowledge. In *Proceedings of ECAI-98*, pages 361–365, Brighton, UK, 1998. John Wiley and Sons.
19. Zadeh, L. A. Fuzzy sets. *Information and Control*, 8:338–353, 1965.
20. Zadeh, L. A. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1(1):3–28, 1978.

Stratified Probabilistic Description Logic Programs

Thomas Lukasiewicz*

Dipartimento di Informatica e Sistemistica, Università di Roma “La Sapienza”
Via Salaria 113, I-00198 Rome, Italy
lukasiewicz@dis.uniroma1.it

Abstract. In previous work, we have introduced probabilistic description logic programs (or pdl-programs), which are a combination of description logic programs (or dl-programs) under the answer set and well-founded semantics with Poole's independent choice logic. Such programs are directed towards sophisticated representation and reasoning techniques that allow for probabilistic uncertainty in the Rules, Logic, and Proof layers of the Semantic Web. In this paper, we continue this line of research. We concentrate on the special case of stratified probabilistic description logic programs (or spdl-programs). In particular, we present an algorithm for query processing in such pdl-programs, which is based on a reduction to computing the canonical model of stratified dl-programs.

1 Introduction

The *Semantic Web* initiative [2,9] aims at an extension of the current World Wide Web by standards and technologies that help machines to understand the information on the Web so that they can support richer discovery, data integration, navigation, and automation of tasks. The main ideas behind it are to add a machine-readable meaning to Web pages, to use ontologies for a precise definition of shared terms in Web resources, to make use of KR technology for automated reasoning from Web resources, and to apply cooperative agent technology for processing the information of the Web.

The Semantic Web consists of several hierarchical layers, where the *Ontology layer*, in form of the *OWL Web Ontology Language* [30,18] (recommended by the W3C), is currently the highest layer of sufficient maturity. OWL consists of three increasingly expressive sublanguages, namely *OWL Lite*, *OWL DL*, and *OWL Full*. OWL Lite and OWL DL are essentially very expressive description logics with an RDF syntax [18]. As shown in [16], ontology entailment in OWL Lite (resp., OWL DL) reduces to knowledge base (un)satisfiability in the description logic $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$). On top of the Ontology layer, the *Rules*, *Logic*, and *Proof layers* of the Semantic Web will be developed next, which should offer sophisticated representation and reasoning capabilities. As a first effort in this direction, *RuleML* (Rule Markup Language) [3] is an XML-based markup language for rules and rule-based systems, whereas the OWL Rules Language [17] is a first proposal for extending OWL by Horn clause rules.

A key requirement of the layered architecture of the Semantic Web is to integrate the Rules and the Ontology layer. In particular, it is crucial to allow for building rules on top

* Alternate address: Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Vienna, Austria; e-mail: lukasiewicz@kr.tuwien.ac.at.

of ontologies, that is, for rule-based systems that use vocabulary from ontology knowledge bases. Another type of combination is to build ontologies on top of rules, which means that ontological definitions are supplemented by rules or imported from rules. Towards this goal, the works [7,8] have proposed *description logic programs* (or *dl-programs*), which are of the form $KB = (L, P)$, where L is a knowledge base in a description logic and P is a finite set of description logic rules (or *dl-rules*). Such dl-rules are similar to usual rules in logic programs with negation as failure, but may also contain *queries to* L in their bodies, which are given by special atoms (on which possibly default negation may apply). Another important feature of dl-rules is that queries to L also allow for specifying an input from P, and thus for a *flow of information from* P *to* L, besides the flow of information from L to P, given by any query to L. Hence, description logic programs allow for building rules on top of ontologies, but also (to some extent) building ontologies on top of rules. In this way, additional knowledge (gained in the program) can be supplied to L before querying. The semantics of dl-programs was defined in [7] and [8] as an extension of the answer set semantics by Gelfond and Lifschitz [12] and the well-founded semantics by Van Gelder, Ross, and Schlipf [29], respectively, which are the two most widely used semantics for nonmonotonic logic programs. The description logic knowledge bases in dl-programs are specified in the well-known description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$.

In [22], towards sophisticated representation and reasoning techniques that also allow for modeling probabilistic uncertainty in the Rules, Logic, and Proof layers of the Semantic Web, we have presented *probabilistic description logic programs* (or *pdl-programs*), which generalize dl-programs under the answer set and well-founded semantics by probabilistic uncertainty. They have been developed as a combination of dl-programs with Poole's independent choice logic (ICL) [25].

It is important to point out that Poole's ICL is a powerful representation and reasoning formalism for single- and also multi-agent systems, which combines logic and probability, and which can represent a number of important uncertainty formalisms, in particular, influence diagrams, Bayesian networks, Markov decision processes, and normal form games [25]. Furthermore, Poole's ICL also allows for natural notions of causes and explanations as in Pearl's structural causal models [10].

In this paper, we continue this line of research. We concentrate on the special case of stratified pdl-programs (or *spdl-programs*). In particular, as a main new contribution, we present an algorithm for query processing in spdl-programs. It is based on a reduction to computing the canonical model of stratified dl-programs, which can be done by a finite sequence of finite fixpoint iterations. This shows especially that query processing in spdl-programs is conceptually easier than query processing in general pdl-programs, which is reducible to computing the set of all answer sets of general dl-programs and solving linear optimization problems. To my knowledge, this paper and [22] are the first works that combine description logic programs with probabilistic uncertainty.

The rest of this paper is organized as follows. In Section 2, we recall the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ as well as stratified description logic programs. Section 3 defines stratified probabilistic description logic programs, and Section 4 deals with query processing in such programs. In Section 5, we discuss related work. Section 6 summarizes the main results and gives an outlook on future research.

2 Preliminaries

In this section, we first recall the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$. We then recall positive and stratified *description logic programs* (or *dl-programs*) under their canonical semantics [7], which combine description logics and normal programs. They consist of a knowledge base L in a description logic and a finite set of description logic rules P. Such rules are similar to usual rules in logic programs with negation as failure, but may also contain *queries to* L, possibly default negated.

2.1 $\mathcal{SHIF}$(D) and $\mathcal{SHOIN}$(D)

We first describe $\mathcal{SHOIN}(\mathbf{D})$. We assume a set $\mathbf{D}$ of *elementary datatypes*. Each $d \in \mathbf{D}$ has a set of *data values*, called the *domain* of d, denoted $\mathrm{dom}(d)$. Let $\mathrm{dom}(\mathbf{D}) = \bigcup_{d \in \mathbf{D}} \mathrm{dom}(d)$. A *datatype* is either an element of $\mathbf{D}$ or a subset of $\mathrm{dom}(\mathbf{D})$ (called *datatype oneOf*). Let $\mathbf{A}$, $\mathbf{R}_A$, $\mathbf{R}_D$, and $\mathbf{I}$ be nonempty finite pairwise disjoint sets of *atomic concepts*, *abstract roles*, *datatype roles*, and *individuals*, respectively. Let $\mathbf{R}_A^-$ denote the set of all inverses R^- of abstract roles $R \in \mathbf{R}_A$.

A *role* is an element of $\mathbf{R}_A \cup \mathbf{R}_A^- \cup \mathbf{R}_D$. *Concepts* are inductively defined as follows. Every $C \in \mathbf{A}$ is a concept, and if $o_1, o_2, \ldots \in \mathbf{I}$, then $\{o_1, o_2, \ldots\}$ is a concept (called *oneOf*). If C and D are concepts and if $R \in \mathbf{R}_A \cup \mathbf{R}_A^-$, then $(C \sqcap D)$, $(C \sqcup D)$, and $\neg C$ are concepts (called *conjunction*, *disjunction*, and *negation*, respectively), as well as $\exists R.C$, $\forall R.C$, $\geq nR$, and $\leq nR$ (called *exists*, *value*, *atleast*, and *atmost restriction*, respectively) for an integer $n \geq 0$. If $d \in \mathbf{D}$ and $U \in \mathbf{R}_D$, then $\exists U.d$, $\forall U.d$, $\geq nU$, and $\leq nU$ are concepts (called *datatype exists*, *value*, *atleast*, and *atmost restriction*, respectively) for an integer $n \geq 0$. We write $\top$ and $\bot$ to abbreviate $C \sqcup \neg C$ and $C \sqcap \neg C$, respectively, and we eliminate parentheses as usual.

An *axiom* is of one of the following forms: (1) $C \sqsubseteq D$, where C and D are concepts (*concept inclusion*); (2) $R \sqsubseteq S$, where either $R, S \in \mathbf{R}_A$ or $R, S \in \mathbf{R}_D$ (*role inclusion*); (3) $\mathrm{Trans}(R)$, where $R \in \mathbf{R}_A$ (*transitivity*); (4) $C(a)$, where C is a concept and $a \in \mathbf{I}$ (*concept membership*); (5) $R(a, b)$ (resp., $U(a, v)$), where $R \in \mathbf{R}_A$ (resp., $U \in \mathbf{R}_D$) and $a, b \in \mathbf{I}$ (resp., $a \in \mathbf{I}$ and $v \in \mathrm{dom}(\mathbf{D})$) (*role membership*); and (6) $a = b$ (resp., $a \neq b$), where $a, b \in \mathbf{I}$ (*equality* (resp., *inequality*)). A *knowledge base* L is a finite set of axioms. For decidability, number restrictions in L are restricted to simple $R \in \mathbf{R}_A$ [19].

The syntax of $\mathcal{SHIF}(\mathbf{D})$ is as the above syntax of $\mathcal{SHOIN}(\mathbf{D})$, but without the oneOf constructor and with the *atleast* and *atmost* constructors limited to 0 and 1.

For the semantics of $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, we refer the reader to [16].

Example 2.1. An online store (such as *amazon.com*) may use a description logic knowledge base to classify and characterize its products. For example, suppose that (1) textbooks are books, (2) personal computers and cameras are electronic products, (3) books and electronic products are products, (4) every product has at least one related product, (5) only products are related to each other, (6) tb_ai and tb_lp are textbooks, which are related to each other, (7) pc_ibm and pc_hp are personal computers, which are related to each other, and (8) ibm and hp are providers for pc_ibm and pc_hp, respectively. This knowledge is expressed by the following description logic knowledge base L_1:

$$Textbook \sqsubseteq Book;\ \ PC \sqcup Camera \sqsubseteq Electronics;\ \ Book \sqcup Electronics \sqsubseteq Product;$$

$\mathit{Product} \sqsubseteq\ \geq 1\, \mathit{related};\ \geq 1\, \mathit{related} \sqcup \geq 1\, \mathit{related}^- \sqsubseteq \mathit{Product};$
$\mathit{Textbook}(\mathit{tb\_ai});\ \mathit{Textbook}(\mathit{tb\_lp});\ \mathit{PC}(\mathit{pc\_ibm});\ \mathit{PC}(\mathit{pc\_hp});$
$\mathit{related}(\mathit{tb\_ai}, \mathit{tb\_lp});\ \mathit{related}(\mathit{pc\_ibm}, \mathit{pc\_hp});$
$\mathit{provides}(\mathit{ibm}, \mathit{pc\_ibm});\ \mathit{provides}(\mathit{hp}, \mathit{pc\_hp}).$

2.2 Syntax of Description Logic Programs

We assume a function-free first-order vocabulary Φ with nonempty finite sets of constant and predicate symbols, and a set $\mathcal{X}$ of variables. A *term* is a constant symbol from Φ or a variable from $\mathcal{X}$. If p is a predicate symbol of arity $k \geq 0$ from Φ and $t_1, \ldots, t_k$ are terms, then $p(t_1, \ldots, t_k)$ is an *atom*. A *negation-as-failure* (*NAF*) *literal* is an atom a or a default-negated atom $\mathit{not}\, a$. A *normal rule* r is of the form

$$a \leftarrow b_1, \ldots, b_k, \mathit{not}\, b_{k+1}, \ldots, \mathit{not}\, b_m\ ,\ \ m \geq k \geq 0\,, \qquad (1)$$

where $a, b_1, \ldots, b_m$ are atoms. We refer to a as the *head* of r, denoted $H(r)$, while the conjunction $b_1, \ldots, b_k, \mathit{not}\, b_{k+1}, \ldots, \mathit{not}\, b_m$ is the *body* of r; its *positive* (resp., *negative*) part is $b_1, \ldots, b_k$ (resp., $\mathit{not}\, b_{k+1}, \ldots, \mathit{not}\, b_m$). We define $B(r) = B^+(r) \cup B^-(r)$, where $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$. A *normal program* P is a finite set of normal rules. Informally, a dl-program consists of a description logic knowledge base L and a generalized normal program P, which may contain queries to L. In such a query, it is asked whether a certain description logic axiom or its negation logically follows from L or not. Formally, a *dl-query* $Q(\mathbf{t})$ is either

(a) a concept inclusion axiom F or its negation $\neg F$; or
(b) of the forms $C(t)$ or $\neg C(t)$, where C is a concept and t is a term; or
(c) of the forms $R(t_1, t_2)$ or $\neg R(t_1, t_2)$, where R is a role and t_1, t_2 are terms.

A *dl-atom* has the form $DL[S_1\, op_1 p_1, \ldots, S_m\, op_m\, p_m; Q](\mathbf{t})$, where each S_i is a concept or role, $op_i \in \{\uplus, \cupdot\}$, p_i is a unary resp. binary predicate symbol, $Q(\mathbf{t})$ is a dl-query, and $m \geq 0$. We call $p_1, \ldots, p_m$ its *input predicate symbols*. Intuitively, $op_i = \uplus$ (resp., $op_i = \cupdot$) increases S_i (resp., $\neg S_i$) by the extension of p_i. A *dl-rule* r is of form (1), where any $b \in B(r)$ may be a dl-atom. A *dl-program* $\mathit{KB} = (L, P)$ consists of a description logic knowledge base L and a finite set of dl-rules P. *Ground terms*, *atoms*, *literals*, etc., are defined as usual. The *Herbrand base* of P, denoted HB_P, is the set of all ground atoms with standard predicate symbols in P and constant symbols in Φ. Let $\mathit{ground}(P)$ be the set of all ground instances of dl-rules in P w.r.t. HB_P.

Example 2.2. Consider the dl-program $\mathit{KB}_1 = (L_1, P_1)$, where L_1 is the description logic knowledge base from Example 2.1, and P_1 is the following set of dl-rules:

(1) $\mathit{pc}(\mathit{pc\_1});\ \mathit{pc}(\mathit{pc\_2});\ \mathit{pc}(\mathit{pc\_3});$
(2) $\mathit{brand\_new}(\mathit{pc\_1});\ \mathit{brand\_new}(\mathit{pc\_2});$
(3) $\mathit{vendor}(\mathit{dell}, \mathit{pc\_1});\ \mathit{vendor}(\mathit{dell}, \mathit{pc\_2});\ \mathit{vendor}(\mathit{dell}, \mathit{pc\_3});$
(4) $\mathit{avoid}(X) \leftarrow DL[\mathit{Camera}](X), \mathit{not}\, \mathit{offer}(X);$
(5) $\mathit{offer}(X) \leftarrow DL[\mathit{PC} \uplus \mathit{pc}; \mathit{Electronics}](X), \mathit{not}\, \mathit{brand\_new}(X);$

(6) $provider(P) \leftarrow vendor(P, X), DL[PC \uplus pc; Product](X)$;
(7) $provider(P) \leftarrow DL[provides](P, X), DL[PC \uplus pc; Product](X)$;
(8) $similar(X, Y) \leftarrow DL[related](X, Y)$;
(9) $similar(X, Z) \leftarrow similar(X, Y), similar(Y, Z)$.

The above dl-rules express that (1) pc_1, pc_2, and pc_3 are additional personal computers, (2) pc_1 and pc_2 are brand new, (3) $dell$ is the vendor of pc_1, pc_2, and pc_3, (4) a customer avoids all cameras that are not on offer, (5) all electronic products that are not brand new are on offer, (6) every vendor of a product is a provider, (7) every entity providing a product is a provider, (8) all related products are similar, and (9) the binary similarity relation on products is transitively closed.

2.3 Semantics of Positive Description Logic Programs

In the sequel, let $KB = (L, P)$ be a dl-program. An *interpretation* I relative to P is any $I \subseteq HB_P$. We say that I is a *model* of $a \in HB_P$ under L, denoted $I \models_L a$, iff $a \in I$. We say that I is a *model* of a ground dl-atom $a = DL[S_1\, op_1\, p_1, \ldots, S_m\, op_m p_m; Q](\mathbf{c})$ under L, denoted $I \models_L a$, iff $L \cup \bigcup_{i=1}^{m} A_i(I) \models Q(\mathbf{c})$, where $A_i(I) = \{S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}$, for $op_i = \uplus$; and $A_i(I) = \{\neg S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}$, for $op_i = \cup\!\!\!-$. A ground dl-atom a is *monotonic* relative to $KB = (L, P)$ iff $I \subseteq I' \subseteq HB_P$ implies that if $I \models_L a$ then $I' \models_L a$. In this paper, we consider only monotonic ground dl-atoms, but observe that one can also define dl-atoms that are not monotonic; see [7]. We say that I is a *model* of a ground dl-rule r iff $I \models_L H(r)$ whenever $I \models_L B(r)$, that is, $I \models_L a$ for all $a \in B^+(r)$ and $I \not\models_L a$ for all $a \in B^-(r)$. We say that I is a *model* of a dl-program $KB = (L, P)$, denoted $I \models KB$, iff $I \models_L r$ for every $r \in ground(P)$. We say that KB is *satisfiable* (resp., *unsatisfiable*) iff it has some (resp., no) model.

We say that $KB = (L, P)$ is *positive* iff no dl-rule in P contains default-negated atoms. Like ordinary positive programs, every positive dl-program KB is satisfiable and has a unique least model, denoted M_{KB}, that canonically characterizes its semantics.

2.4 Semantics of Stratified Description Logic Programs

We next define stratified dl-programs and their canonical semantics. They are intuitively composed of hierarchic layers of positive dl-programs linked via default negation. Like ordinary stratified normal programs, they are always satisfiable and can be assigned a canonical minimal model via a number of iterative least models.

For any dl-program $KB = (L, P)$, we denote by DL_P the set of all ground dl-atoms that occur in $ground(P)$. An *input atom* of $a \in DL_P$ is a ground atom with an input predicate of a and constant symbols in Φ. A *(local) stratification of* $KB = (L, P)$ is a mapping $\lambda\colon HB_P \cup DL_P \rightarrow \{0, 1, \ldots, k\}$ such that (i) $\lambda(H(r)) \geq \lambda(b')$ (resp., $\lambda(H(r)) > \lambda(b')$) for each $r \in ground(P)$ and $b' \in B^+(r)$ (resp., $b' \in B^-(r)$), and (ii) $\lambda(a) \geq \lambda(b)$ for each input atom b of each $a \in DL_P$, where $k \geq 0$ is the *length* of λ. For $i \in \{0, \ldots, k\}$, let $KB_i = (L, P_i) = (L, \{r \in ground(P) \mid \lambda(H(r)) = i\})$, and let HB_{P_i} (resp., $HB^{\star}_{P_i}$) be the set of all $b \in HB_P$ such that $\lambda(b) = i$ (resp., $\lambda(b) \leq i$). A dl-program $KB = (L, P)$ is *(locally) stratified* iff it has a stratification λ of some length $k \geq 0$. We define its iterative least models $M_i \subseteq HB_P$ with $i \in \{0, \ldots, k\}$ as follows:

(i) M_0 is the least model of KB_0;
(ii) if $i > 0$, then M_i is the least model of KB_i such that $M_i | HB^{\star}_{P_{i-1}} = M_{i-1} | HB^{\star}_{P_{i-1}}$.

The canonical model of the stratified dl-program KB, denoted M_{KB}, is then defined as M_k. Observe that M_{KB} is well-defined, since it does not depend on a particular λ. Furthermore, M_{KB} is in fact a minimal model of KB.

3 Stratified Probabilistic Description Logic Programs

In this section, we define stratified probabilistic dl-programs as a combination of dl-programs with Poole's independent choice logic (ICL) [25]. Poole's ICL is based on ordinary acyclic logic programs under different "atomic choices", where each atomic choice along with an acyclic logic program produces a first-order model, and one then obtains a probability distribution over the set of first-order models by placing a distribution over the different atomic choices. In stratified probabilistic dl-programs, we here use stratified dl-programs rather than ordinary acyclic logic programs.

3.1 Syntax

We assume a function-free first-order vocabulary Φ with nonempty finite sets of constant and predicate symbols, and a set of variables $\mathcal{X}$, as in Section 2. We use HB_{Φ} (resp., HU_{Φ}) to denote the Herbrand base (resp., universe) over Φ. In the sequel, we assume that HB_{Φ} is nonempty. We define *classical formulas* by induction as follows. The propositional constants *false* and *true*, denoted $\bot$ and $\top$, respectively, and all atoms are classical formulas. If ϕ and ψ are classical formulas, then also $\neg\phi$ and $(\phi \wedge \psi)$. A *conditional constraint* is of the form $(\psi|\phi)[l, u]$ with reals $l, u \in [0, 1]$ and classical formulas ϕ and ψ. We define *probabilistic formulas* inductively as follows. Every conditional constraint is a probabilistic formula. If F and G are probabilistic formulas, then also $\neg F$ and $(F \wedge G)$. We use $(F \vee G)$, $(F \Leftarrow G)$, and $(F \Leftrightarrow G)$ to abbreviate $\neg(\neg F \wedge \neg G)$, $\neg(\neg F \wedge G)$, and $(\neg(\neg F \wedge G) \wedge \neg(F \wedge \neg G))$, respectively, and adopt the usual conventions to eliminate parentheses. *Ground terms*, *ground formulas*, *substitutions*, and *ground instances* of probabilistic formulas are defined as usual.

A *choice space* C is a set of pairwise disjoint and nonempty sets $A \subseteq HB_{\Phi}$. Any member $A \in C$ is called an *alternative* of C and any element $a \in A$ an *atomic choice* of C. A *total choice* of C is a set $B \subseteq HB_{\Phi}$ such that $|B \cap A| = 1$ for all $A \in C$. A *probability* μ on a choice space C is a probability function on the set of all total choices of C. Since C and all its alternatives are finite, μ can be defined by (i) a mapping $\mu \colon \bigcup C \rightarrow [0, 1]$ such that $\sum_{a \in A} \mu(a) = 1$ for all $A \in C$, and (ii) $\mu(B) = \Pi_{b \in B}\mu(b)$ for all total choices B of C. Intuitively, (i) associates a probability with each atomic choice of C, and (ii) assumes independence between the alternatives of C.

A *probabilistic dl-program* (or *pdl-program*) $KB = (L, P, C, \mu)$ consists of a dl-program (L, P), a choice space C such that (i) $\bigcup C \subseteq HB_P$ and (ii) no atomic choice in C coincides with the head of any dl-rule in $ground(P)$, and a probability μ on C. A *stratified probabilistic dl-program* (or *spdl-program*) is a pdl-program $KB{=}(L, P, C, \mu)$ where (L, P) is stratified. A *probabilistic query* to KB has the form $?F$ or the form

$?(\beta|\alpha)[L,U]$, where F is a probabilistic formula, β, α are classical formulas, and L, U are variables. The *correct answer* to $?F$ is the set of all substitutions θ such that $F\theta$ is a consequence of KB. The *tight answer* to $?(\beta|\alpha)[L,U]$ is the set of all substitutions θ such that $?(\beta|\alpha)[L,U]\theta$ is a tight consequence of KB. In the following paragraphs, we define the notions of *consequence* and *tight consequence*.

Example 3.1. Consider the spdl-program $KB_1 = (L_1, P_1, C_1, \mu_1)$, where L_1 is as in Example 2.1, and P_1 is as in Example 2.2 except that the dl-rules (4) and (5) are replaced by the dl-rules (4') and (5'), respectively, and the dl-rules (10) and (11) are added:

(4') $\mathit{avoid}(X) \leftarrow DL[\mathit{Camera}](X), \mathit{not\ offer}(X), \mathit{avoid\_pos}$;
(5') $\mathit{offer}(X) \leftarrow DL[PC \uplus pc; \mathit{Electronics}](X), \mathit{not\ brand\_new}(X), \mathit{offer\_pos}$;
(10) $\mathit{buy}(C,X) \leftarrow \mathit{needs}(C,X), \mathit{view}(X), \mathit{not\,avoid}(X), \mathit{v\_buy\_pos}$;
(11) $\mathit{buy}(C,X) \leftarrow \mathit{needs}(C,X), \mathit{buy}(C,Y), \mathit{also\_buy}(Y,X), \mathit{a\_buy\_pos}$.

Furthermore, let C_1 be given by $\{\{\mathit{avoid\_pos}, \mathit{avoid\_neg}\}, \{\mathit{offer\_pos}, \mathit{offer\_neg}\}, \{\mathit{v\_buy\_pos},\ \mathit{v\_buy\_neg}\}, \{\mathit{a\_buy\_pos},\ \mathit{a\_buy\_neg}\}\}$, and let $\mu_1(\mathit{avoid\_pos}) = 0.9$, $\mu_1(\mathit{avoid\_neg}) = 0.1$, $\mu_1(\mathit{offer\_pos}) = 0.9$, $\mu_1(\mathit{offer\_neg}) = 0.1$, $\mu_1(\mathit{v\_buy\_pos}) = 0.7$, $\mu_1(\mathit{v\_buy\_neg}) = 0.3$, $\mu_1(\mathit{a\_buy\_pos}) = 0.7$, and $\mu_1(\mathit{a\_buy\_neg}) = 0.3$.

Here, the new dl-rules (4') and (5') express that the dl-rules (4) and (5) actually only hold with the probability 0.9. Furthermore, (10) expresses that a customer buys a needed product that is viewed and not avoided with the probability 0.7, while (11) says that a customer buys a needed product x with probability 0.7, if she bought another product y, and every customer that previously had bought y also bought x.

In a probabilistic query, one may ask for the tight probability bounds that a customer c buys a needed product x, if (i) c bought another product y, (ii) every customer that previously had bought y also bought x, (iii) x is not avoided, and (iv) c has been shown product x (the result to this query may, e.g., help to decide whether it is useful to make a customer automatically also view product x when buying y):

$$?(\mathit{buy}(c,x) \mid \mathit{needs}(c,x) \wedge \mathit{buy}(c,y) \wedge \mathit{also\_buy}(y,x) \wedge \mathit{view}(x) \wedge \mathit{not\,avoid}(x))[L,U]\,.$$

3.2 Semantics

A *world* I is a subset of HB_Φ. We use $\mathcal{I}_\Phi$ to denote the set of all worlds over Φ. A *variable assignment* σ maps each variable $X \in \mathcal{X}$ to an element of HU_Φ. It is extended to all terms by $\sigma(c) = c$ for all constant symbols c from Φ. The *truth* of classical formulas ϕ in I under σ, denoted $I \models_\sigma \phi$ (or $I \models \phi$ when ϕ is ground), is inductively defined by:

- $I \models_\sigma p(t_1, \ldots, t_k)$ iff $p(\sigma(t_1), \ldots, \sigma(t_k)) \in I$;
- $I \models_\sigma \neg\phi$ iff not $I \models_\sigma \phi$; and $I \models_\sigma (\phi \wedge \psi)$ iff $I \models_\sigma \phi$ and $I \models_\sigma \psi$.

A *probabilistic interpretation* Pr is a probability function on $\mathcal{I}_\Phi$ (that is, since $\mathcal{I}_\Phi$ is finite, a mapping $Pr\colon \mathcal{I}_\Phi \to [0,1]$ such that all $Pr(I)$ with $I \in \mathcal{I}_\Phi$ sum up to 1). The *probability* of a classical formula ϕ in Pr under a variable assignment σ, denoted $Pr_\sigma(\phi)$ (or $Pr(\phi)$ when ϕ is ground), is defined as the sum of all $Pr(I)$ such that $I \in \mathcal{I}_\Phi$ and $I \models_\sigma \phi$. For classical formulas ϕ and ψ with $Pr_\sigma(\phi) > 0$, we use $Pr_\sigma(\psi|\phi)$ to abbreviate $Pr_\sigma(\psi \wedge \phi) \,/\, Pr_\sigma(\phi)$. The *truth* of probabilistic formulas F in Pr under a variable assignment σ, denoted $Pr \models_\sigma F$, is inductively defined as follows:

- $Pr \models_\sigma (\psi|\phi)[l,u]$ iff $Pr_\sigma(\phi) = 0$ or $Pr_\sigma(\psi|\phi) \in [l,u]$;
- $Pr \models_\sigma \neg F$ iff not $Pr \models_\sigma F$; and $Pr \models_\sigma (F \wedge G)$ iff $Pr \models_\sigma F$ and $Pr \models_\sigma G$.

A probabilistic interpretation Pr is a *model* of a probabilistic formula F iff $Pr \models_\sigma F$ for every variable assignment σ. We say that Pr is the *canonical model* of an spdl-program $KB = (L, P, C, \mu)$ iff every world $I \in \mathcal{I}_\Phi$ with $Pr(I) > 0$ is the canonical model of $(L, P \cup \{p \leftarrow \mid p \in B\})$ for some total choice B of C such that $Pr(I) = \mu(B)$. Notice that every KB has a unique canonical model Pr. A probabilistic formula F is a *consequence* of KB, denoted $KB \,\|\!\sim F$, iff every model of KB is also a model of F. A conditional constraint $(\psi|\phi)[l,u]$ is a *tight consequence* of KB, denoted $KB \,\|\!\sim_{tight} (\psi|\phi)[l,u]$, iff l (resp., u) is the infimum (resp., supremum) of $Pr_\sigma(\psi|\phi)$ subject to all models Pr of KB and all variable assignments σ with $Pr_\sigma(\phi) > 0$. Here, we assume that $l = 1$ and $u = 0$, when $Pr_\sigma(\phi) = 0$ for all models Pr of KB and all σ.

4 Query Processing

The canonical model of an ordinary positive (resp., stratified) normal logic program P has a fixpoint characterization in terms of an immediate consequence operator T_P, which generalizes to dl-programs. This can be used for a bottom-up computation of the canonical model of a positive (resp., stratified) dl-program, and thus also for computing the canonical model of an spdl-program and for query processing in spdl-programs.

4.1 Canonical Models of Positive Description Logic Programs

For a dl-program $KB = (L, P)$, define the operator T_{KB} on the subsets of HB_P as follows. For every $I \subseteq HB_P$, let

$$T_{KB}(I) \;=\; \{H(r) \mid r \in \mathit{ground}(P),\, I \models_L \ell \text{ for all } \ell \in B(r)\}\,.$$

If KB is positive, then T_{KB} is monotonic. Hence, T_{KB} has a least fixpoint, denoted $\mathit{lfp}(T_{KB})$. Furthermore, $\mathit{lfp}(T_{KB})$ can be computed by finite fixpoint iteration (given finiteness of P and the number of constant symbols in Φ). For every $I \subseteq HB_P$, we define $T^i_{KB}(I) = I$, if $i = 0$, and $T^i_{KB}(I) = T_{KB}(T^{i-1}_{KB}(I))$, if $i > 0$.

Theorem 4.1. *For every positive dl-program $KB = (L, P)$, it holds that $\mathit{lfp}(T_{KB}) = M_{KB}$. Furthermore, $\mathit{lfp}(T_{KB}) = \bigcup_{i=0}^{n} T^i_{KB}(\emptyset) = T^n_{KB}(\emptyset)$, for some $n \geq 0$.*

4.2 Canonical Models of Stratified Description Logic Programs

We next describe a fixpoint iteration for stratified dl-programs. Using Theorem 4.1, we can characterize the canonical model M_{KB} of a stratified dl-program $KB = (L, P)$ as follows. Let $\widehat{T}^i_{KB}(I) = T^i_{KB}(I) \cup I$, for all $i \geq 0$.

Theorem 4.2. *Suppose $KB = (L, P)$ has a stratification λ of length $k \geq 0$. Define $M_i \subseteq HB_P$, $i \in \{-1, 0, \ldots, k\}$, as follows: $M_{-1} = \emptyset$, and $M_i = \widehat{T}^{n_i}_{KB_i}(M_{i-1})$ for $i \geq 0$, where $n_i \geq 0$ such that $\widehat{T}^{n_i}_{KB_i}(M_{i-1}) = \widehat{T}^{n_i+1}_{KB_i}(M_{i-1})$. Then, $M_k = M_{KB}$.*

4.3 Query Processing in Stratified Probabilistic Description Logic Programs

Fig. 1 shows Algorithm canonical_model, which computes the canonical model Pr of a given spdl-program $KB = (L, P, C, \mu)$. This algorithm is essentially based on a reduction to computing the canonical model of stratified dl-programs (see step (4)), which can be done using the above finite sequence of finite fixpoint iterations.

```
Algorithm canonical_model

Input: spdl-program $KB = (L, P, C, \mu)$.
Output: canonical model $Pr$ of $KB$.

1. for every interpretation $I \in \mathcal{I}_\Phi$ do
2.   $Pr(I) := 0$;
3. for every total choice $B$ of $C$ do begin
4.   compute the canonical model $I$ of the stratified dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$;
5.   $Pr(I) := \mu(B)$;
6. end;
7. return $Pr$.
```

Fig. 1. Algorithm canonical_model

Fig. 2 shows Algorithm tight_answer, which computes tight answers $\theta = \{L/l, U/u\}$ for a given query $?(\beta|\alpha)[L, U]$ to a given spdl-program KB. The algorithm first computes the canonical model of KB in step (1) and then the tight answer in steps (2)–(8).

```
Algorithm tight_answer

Input: spdl-program $KB = (L, P, C, \mu)$ and probabilistic query $?(\beta|\alpha)[L, U]$.
Output: tight answer $\theta = \{L/l, U/u\}$ for $?(\beta|\alpha)[L, U]$ to $KB$.

1. $Pr :=$ canonical_model$(KB)$;
2. $l := 1$;
3. $u := 0$;
4. for every ground instance $\beta'|\alpha'$ of $\beta|\alpha$ do begin
5.   $l := \min(l, Pr(\beta'|\alpha'))$;
6.   $u := \max(u, Pr(\beta'|\alpha'))$;
7. end;
8. return $\theta = \{L/l, U/u\}$.
```

Fig. 2. Algorithm tight_answer

5 Related Work

Related approaches can be roughly divided into (a) description logic programs with non-probabilistic uncertainty, (b) probabilistic generalizations of description logics, and (c) probabilistic generalizations of web ontology languages. Note that related work on description logic programs without uncertainty is discussed in [7,8,22].

As for (a), Straccia [28] combines description logic programs with *non-probabilistic uncertainty* using interval annotations. To my knowledge, the present paper and [22] are the first ones on description logic programs with *probabilistic uncertainty*.

As for (b), Giugno and Lukasiewicz [13] present a probabilistic generalization of the expressive description logic $\mathcal{SHOQ}(\mathbf{D})$ behind DAML+OIL, which is based on lexicographic probabilistic reasoning. In earlier work, Heinsohn [15] and Jaeger [20] present probabilistic extensions to the description logic $\mathcal{ALC}$, which are essentially based on probabilistic reasoning in probabilistic logics. Koller et al. [21] present a probabilistic generalization of the CLASSIC description logic, which uses Bayesian networks as underlying probabilistic reasoning formalism. Note that fuzzy description logics, such as the ones by Straccia [26,27], are less closely related to probabilistic description logics, since fuzzy uncertainty deals with vagueness, rather than ambiguity and imprecision.

As for (c), especially the works by Costa [4], Pool and Aikin [24], and Ding and Peng [6] present probabilistic extensions to OWL. In particular, Costa's work [4] is semantically based on multi-entity Bayesian networks, while [6] has a semantics in standard Bayesian networks. In closely related work, Fukushige [11] proposes a basic framework for representing probabilistic relationships in RDF. Finally, Nottelmann and Fuhr [23] present pDAML+OIL, which is a probabilistic generalization of the web ontology language DAML+OIL, and a mapping to stratified probabilistic datalog.

6 Summary and Outlook

We have continued the research on probabilistic dl-programs. We have focused on the special case of stratified probabilistic dl-programs. In particular, we have presented an algorithm for query processing in such probabilistic dl-programs, which is based on a reduction to computing the canonical model of stratified dl-programs.

A topic of future research is to further enhance stratified probabilistic dl-programs towards a possible use for Web Services. This may be done by exploiting and generalizing further features of Poole's ICL for dynamic and multi-agent systems [25].

Acknowledgments. This work has been supported by a Heisenberg Professorship of the German Research Foundation. I thank the reviewers of this paper for their constructive comments, which helped to improve this work.

References

1. G. Antoniou. Nonmonotonic rule systems on top of ontology layers. In *Proceedings ISWC-2002*, *LNCS* 2342, pp. 394–398.
2. T. Berners-Lee. *Weaving the Web*. Harper, San Francisco, CA, 1999.
3. H. Boley, S. Tabet, and G. Wagner. Design rationale for RuleML: A markup language for Semantic Web rules. In *Proceedings SWWS-2001*, pp. 381–401.
4. P. C. G. da Costa. Bayesian semantics for the Semantic Web. Doctoral Dissertation, George Mason University, Fairfax, VA, USA, 2005.
5. C. V. Damásio. The W^4 Project, 2002. `http://centria.di.fct.unl.pt/~cd/projectos/w4/index.htm`.

6. Z. Ding and Y. Peng. A Probabilistic extension to ontology language OWL. In *Proceedings HICSS-2004*.
7. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. In *Proceedings KR-2004*, pp. 141–151.
8. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Well-founded semantics for description logic programs in the Semantic Web. In *Proc. RuleML-2004*, *LNCS* 3323, pp. 81–97.
9. D. Fensel, W. Wahlster, H. Lieberman, and J. Hendler, editors. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2002.
10. A. Finzi and T. Lukasiewicz. Structure-based causes and explanations in the independent choice logic. In *Proceedings UAI-2003*, pp. 225–232.
11. Y. Fukushige. Representing probabilistic knowledge in the Semantic Web. In *Proceedings of the W3C Workshop on Semantic Web for Life Sciences*, Cambridge, MA, USA, 2004.
12. M. Gelfond and V. Lifschitz. Classical negation in logic programs and deductive databases. *New Generation Computing*, 17:365–387, 1991.
13. R. Giugno and T. Lukasiewicz. P-$\mathcal{SHOQ}(\mathbf{D})$: A probabilistic extension of $\mathcal{SHOQ}(\mathbf{D})$ for probabilistic ontologies in the Semantic Web. In *Proc. JELIA-2002*, *LNCS* 2424, pp. 86–97.
14. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logics. In *Proceedings WWW-2003*, pp. 48–57.
15. J. Heinsohn. Probabilistic description logics. In *Proceedings UAI-1994*, pp. 311–318.
16. I. Horrocks and P. F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In *Proceedings ISWC-2003*, *LNCS* 2870, pp. 17–29.
17. I. Horrocks and P. F. Patel-Schneider. A proposal for an OWL Rules Language. In *Proceedings WWW-2004*, pp. 723–731.
18. I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
19. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proceedings LPAR-1999*, *LNCS* 1705, pp. 161–180.
20. M. Jaeger. Probabilistic reasoning in terminological logics. In *Proc. KR-1994*, pp. 305–316.
21. D. Koller, A. Levy, and A. Pfeffer. P-CLASSIC: A tractable probabilistic description logic. In *Proceedings AAAI-1997*, pp. 390–397.
22. T. Lukasiewicz. Probabilistic description logic programs. In *Proceedings ECSQARU-2005*, *LNCS* 3571, pp. 737–749.
23. H. Nottelmann and N. Fuhr. pDAML+OIL: A probabilistic extension to DAML+OIL based on probabilistic Datalog. In *Proceedings IPMU-2004*.
24. M. Pool and J. Aikin. KEEPER and Protégé: An elicitation environment for Bayesian inference tools. In *Proceedings of the Workshop on Protégé and Reasoning* held at the *7th International Protégé Conference*, 2004.
25. D. Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94:7–56, 1997.
26. U. Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.
27. U. Straccia. Towards a fuzzy description logic for the Semantic Web (preliminary report). In *Proceedings ESWC-2005*, *LNCS* 3532, pp. 167–181.
28. U. Straccia. Uncertainty and description logic programs: A proposal for expressing rules and uncertainty on top of ontologies. Technical Report ISTI-2004-TR, CNR Pisa, 2004.
29. A. van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.
30. W3C. OWL web ontology language overview, 2004. W3C Recommendation (10 February 2004). Available at `www.w3.org/TR/2004/REC-owl-features-20040210/`.

Modeling Degrees of Conceptual Overlap in Semantic Web Ontologies

Markus Holi and Eero Hyvönen
Helsinki University of Technology, Media Technology,
Helsinki Institute for Information Technology (HIIT), and University of Helsinki
P.O. Box 5500, FI-02015 TKK, FINLAND
http://www.cs.helsinki.fi/group/seco/
email: firstname.lastname@tkk.fi

Abstract

Semantic Web ontologies are based on crisp logic and do not provide well-defined means for expressing uncertainty. We present a new probabilistic method to approach the problem. In our method, degrees of subsumption, i.e., overlap between concepts can be modeled and computed efficiently using Bayesian networks based on RDF(S) ontologies.

1 Introduction

Ontologies are based on crisp logic. In the real world, however, relations between entities often include subtleties that are difficult to express in crisp ontologies. RDFS [rdf, 2004] and OWL [owl, 2003] do not provide standard ways to express partial overlap and degrees of overlap in general.

This paper presents a method for modeling degrees of overlap between concepts. In the following we first introduce the principles of our method. Then a notation that enables the representation of degrees of overlap between concepts in an ontology is presented after which a method for doing inferences based on the notation will be described. For a more detailed presentation of the method see [Holi, 2004].

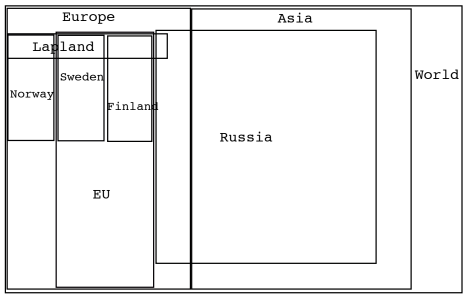
2 Modeling Uncertainty in Ontologies

Figure 1 illustrates various countries and areas in the world. There are important properties in the figure, that are not modeled in a crisp partonomy. For example, EU is a bigger part of Europe than Lapland, and Russia partly overlaps Europe and Asia.

Our method enables the representation of overlap in concept hierarchies, including class hierarchies and partonomies, and the computation of overlap between a *selected* concept and every other, i.e. *referred* concept in the hierarchy. The overlap value is defined as follows:

$Overlap = \frac{|Selected \cap Referred|}{|Referred|} \in [0, 1]$.

Intuitively, the overlap value has the following meaning: The value is 0 for disjoint concepts (e.g., Lapland and Asia) and 1, if the referred concept is subsumed by the selected one. High values lesser than one imply, that the meaning of the selected concept approaches the meaning of the referred one.



```
World 37*23 = 851
Europe 15*23 = 345
Asia 18*23 = 414
EU  8*21 = 168
Sweden 4*9 = 36
Finland 4*9 = 36
Norway 4*9 = 36
Lapland 13*2 = 26  Lapland&(Finland | Sweden | Norway) = 8
Lapland&EU = 16 Lapland&Russia = 2
Russia 18*19 = 342  Russia&Europe = 57  Russia&Asia = 285
```

Figure 1: A Venn diagram illustrating countries, areas, their overlap, and size in the world.

3 Representing Overlap

A concept hierarchy can be viewed as a set of sets and can be represented by a Venn diagram.

If A and B are sets, then A must be in one of the following relationships to B.

1. A is a subset of B, i.e. $A \subseteq B$.
2. A partially overlaps B, i.e. $\exists x, y : (x \in A \wedge x \in B) \wedge (y \in A \wedge y \notin B)$.
3. A is disjoint from B, i.e. $A \cap B = \emptyset$.

Based on these relations, we have developed a simple graph notation for representing overlap in a concept hierarchy as an acyclic *overlap graph*. Here concepts are nodes, and a number called *mass* is attached to each node. The mass of concept A is a measure of the size of the set corresponding to A, i.e. $m(A) = |s(A)|$, where $s(A)$ is the set corresponding to A. A solid directed arc from concept A to B denotes crisp subsumption $s(A) \subseteq s(B)$, a dashed arrow denotes disjointness $s(A) \cap s(B) = \emptyset$, and a dotted arrow represents quantified partial subsumption between concepts, which means that the concepts partially overlap in the Venn diagram. The amount of overlap is represented by the *partial overlap value* $p = \frac{|s(A) \cap s(B)|}{|s(A)|}$.

In addition to the quantities attached to the dotted arrows, also the other arrow types have implicit overlap values. The overlap value of a solid arc is 1 (crisp subsumption) and the value of a dashed arc is 0 (disjointness). The quantities of the arcs emerging from a concept must sum up to 1. This means that either only one solid arc can emerge from a node or several dotted arcs (partial overlap). In both cases, additional dashed arcs can be used (disjointness). Intuitively, the outgoing arcs constitute a quantifi ed partition of the concept. Thus, the dotted arrows emerging from a concept must always point to concepts that are mutually disjoint with each other.

Notice that if two concepts overlap, there must be a directed (solid or dotted) path between them. If the path includes dotted arrows, then (possible) disjointness between the concepts must be expressed explicitly using the disjointness relation. If the directed path is solid, then the concepts necessarily overlap.

4 Computing the Overlaps

Computing the overlap is easiest when there are only solid arcs, i.e. complete subsumption relation between concepts. If there is a directed solid path from A (selected) to B (referred), then overlap $o = \frac{|s(A) \cap s(B)|}{|s(B)|} = \frac{m(A)}{m(B)}$. If there is a mixed path then the computation is not as simple. To exploit the simple case we transform the graph into a solid path structure according to the following principle:

Transformation Principle 1 *Let A be the direct partial subconcept of B with overlap value o. In the solid path structure the partial subsumption is replaced by an additional* middle concept, *that represents $s(A) \cap s(B)$. It is marked to be the complete subconcept of both A and B, and its mass is $o \cdot m(A)$.*

If A is the selected concept and B is the referred one, then the overlap value o can be interpreted as the conditional probability

$$P(B' = true|A' = true) = \frac{|s(A) \cap s(B)|}{|s(B)|} = o, \qquad (1)$$

where $s(A)$ and $s(B)$ are the sets corresponding to the concepts A and B. A' and B' are boolean random variables such that the value $true$ means that the corresponding concept is a match to the query, i.e, the concept in question is of interest to the user.

Based on the above, we chose to use the solid path structure as a Bayesian network topology. In the Bayesian network the boolean random variable X' replaces the concept X of the solid path structure. The effi cient evidence propagation algorithms developed for Bayesian networks [Finin and Finin, 2001] to take care of the overlap computations.

The joint probability distribution of the Bayesian network is defi ned by conditional probability tables (CPT) $P(A'|B'_1, B'_2, \dots B'_n)$ for nodes with parents $B'_i, i = 1 \dots n$, and by prior marginal probabilities set for nodes without parents. The CPT $P(A'|B'_1, B'_2, \dots B'_n)$ for a node A' can be constructed by enumerating the value combinations (true/false) of the parents $B'_i, i = 1 \dots n$, and by assigning:

$$P(A' = true|B'_1 = b_1, \dots B'_n = b_n) = \frac{\sum_{i \in \{i : b_i = true\}} m(B_i)}{m(A)} \qquad (2)$$

The value for the complementary case $P(A' = false|B'_1 = b_1, \dots B'_n = b_n)$ is obtained simply by subtracting from 1.

By instantiating the nodes corresponding to the selected concept and the concepts subsumed by it as evidence (their values are set "true"), the propagation algorithm returns the overlap values as posterior probabilities of nodes. The query results can then be ranked according to these posterior probabilities.

5 Discussion

Overlap graphs are simple and can be represented in RDF(S) easily. Using the notation does not require knowledge of probability theory. The concepts can be quantifi ed automatically, based on data records annotated according to the ontology, for example.

The problem of representing uncertainty in ontologies has been tackled previously by using methods of fuzzy logic, rough sets [Stuckenschmidt and Visser, 2000] and Bayesian networks [Ding and Peng, 2004; Gu and H.K. Pung, 2004].

Acknowledgments

Our research was funded mainly by the National Technology Agency Tekes.

References

[Ding and Peng, 2004] Z. Ding and Y. Peng. A probabilistic extension to ontology language owl. In *Proceedings of the Hawai'i Internationa Conference on System Sciences*, 2004.

[Finin and Finin, 2001] F. V. Finin and F. B. Finin. *Bayesian Networks and Decision Graphs*. Springer-Verlag, 2001.

[Gu and H.K. Pung, 2004] T. Gu and D.Q. Zhang H.K. Pung. A bayesian approach for dealing with uncertain contexts. In *Advances in Pervasive Computing*, 2004.

[Holi, 2004] M. Holi. Modeling uncertainty in semantic web taxonomies, 2004. Master of Science Thesis. Department of Computer Science, University of Helsinki, http://ethesis.helsinki.fi /julkaisut/mat/tieto/pg/holi/.

[owl, 2003] *OWL Web Ontology Language Guide*, 2003. http://www.w3.org/TR/2003/CR-owl-guide-20030818/.

[rdf, 2004] *RDF Vocabulary Description Language 1.0: RDF Schema*, 2004. http://www.w3.org/TR/rdf-schema/.

[Stuckenschmidt and Visser, 2000] H. Stuckenschmidt and U. Visser. Semantic translation based on approximate reclassifi cation. In *Proceedings of the 'Semantic Approximation, Granularity and Vagueness' Workshop*, 2000.

Modeling the Non-Expected Choice: A Weighted Utility Logit[*]

Pia Koskenoja
Tampere University of Technology
Institute of Transportation Engineering
Room FA 208, P O Box 541 (Korkeakoulunkatu 8), FI-33101 Tampere, Finland
pia.koskenoja@tut.fi

Abstract

This work derives and simulates two choice models applying the weighted utility theory, a generalization of the expected utility theory. It shows one set of assumptions, which justify the practice of including the mean and the variance of a risky alternative into a linear utility function of the choice model. A Monte Carlo simulation provides empirical evidence on the robustness of the models.

1 Introduction

Allais paradox shows that our choices commonly violate the axioms of von Neumann-Morgenstein expected utility theory. But we still commonly apply the expected utility theory when we model our choices. One possible remedy to this discrepancy is to build a choice model that uses one generalization of the expected utility theory, the weighted utility theory.

This paper presents two binomial logit models, which assume that the decision maker has weighed utility preferences. The models have been written into a context of a transportation problem, but naturally they can be applied to any choice between two risky alternatives.

Axiomatically weighted utility differs from expected utility by a weaker version of the independence axiom. Weighted utility was first axiomatized by Chew and MacCrimmon, [1979]. Chew [1982] proved that weighted utility behavior cannot be derived from expected utility by transforming the risky variables. Further axiomatic work has been continued by Chew [1983], Fishburn [1981, 1983] and Nakamura [1984, 1985]. Fishburn [1988] contains an informative presentation of the weighted utility theory.

The descriptive strength of weighted utility has been tested in empirical laboratory experiments [Chew and Waller, 1986; Camerer 1989; Conlisk 1989]. I do not know of any choice models where weighted utility is applied.

* The support of Yrjö Jahnson Foundation and NSF grants DMS 9313013 and DMS 9208758 are gratefully acknowledged.

2 Utility Functionals of the Logit Models

Following the tradition of logit models I formulate a utility function that is separable in attributes. The simplest utility function has one sure attribute and one risky attribute. In a transportation context these can be monetary cost of travel and travel time, respectively. In the case of discrete distribution of the risky alternative, the utility functional is:

$$V(\cdot) = b_0 - b_1 \sum_i \frac{p(t_i)\, w(t_i)\, U(t_i)}{\sum_m p(t_m)\, w(t_m)} - b_2 c$$

where $p(t_i)$ denotes the probability of possible travel time outcome t_i, $w(t_i)$ the weight the decision maker places on the outcome t_i, $U(t_i)$ the utility of the outcome t_i, and c the sure monetary cost.

An exponential works well as the weight function.

$$w(t_i) = \exp(\alpha t_i)$$

If $\alpha = 0$, the weight function gets a value one throughout the domain and reduces the weighted utility expression to an expected utility. If $\alpha > 0$, the traveler emphasizes the potential of longer travel times. Correspondingly, if $\alpha < 0$, the traveler behaves as if he would consider the shorter travel times as "more weighty" than what expected utility would warrant.

For the model with continuous distribution of the risky attribute the assumptions are: $t \sim N(\mu,\sigma^2)$, $U = -b_1 t$, and $w(t) = \exp(\alpha t)$. With these assumptions the utility functional is:

$$V(\cdot) = b_0 - b_1 \frac{\frac{1}{\sqrt{2\pi\sigma}} \int t \cdot \exp(\alpha t) \cdot \exp\left(\frac{-(t - \quad {}^2)}{2\sigma^2}\right) dt}{\frac{1}{\sqrt{2\pi\sigma}} \int \exp(\alpha t) \cdot \exp\left(\frac{-(t - \quad {}^2)}{2\sigma^2}\right) dt} - b_2 c.$$

This form has the welcome property that it simplifies to

$$V(\cdot) = b_0 - b_1(\quad + \alpha\sigma^2) - b_2 c\ .$$

This is a welcome find because it justifies the commonly practiced ad hoc inclusion of the risky attribute's variance as a fully separate explanatory variable in addition to the mean in the utility expression of an estimated choice model. On the other hand, it demonstrates that this common practice is not compatible with the expected utility theory. A demonstration of this property in a 3-outcome space is available from the author by request.

2.1 Parameter restrictions

It is customary to require that a utility function exhibits risk aversion and monotonicity.

Risk aversion is defined to mean that the utility of the expected outcome is preferred to the utility of a gamble. Assuming two arbitrary outcomes, the requirement of risk aversion simplifies to a requirement that the ratio of weight functions of the outcomes cannot equal to one, that is, α should not equal zero. This requirement reflects the fact that this particular formulation of weighted utility reduces to expected utility only in the case of risk neutrality.

Monotonicity of utility function in outcomes generalizes into a requirement that the utility functional exhibits first order stochastic dominance (FSD). For the discrete model it is possible to arbitrarily define the range of outcomes as $[L_1, L2]$ and thus the range for $V[\{p(t_i)\}]$ as $[-b_1L_2,-b_1L_1]$. The definitions lead to two conditions for FSD: $\alpha < 1/(L_2-L_1)$ and $\alpha > 0$. If the risky attribute has an infinite range of outcomes, the decision maker violates monotonicity if she is risk averse, that is, if her $\alpha \neq 0$.

3 Monte Carlo Simulations

The Monte Carlo simulations consisted of rounds of first creating the true choices according to three models: a continuous risky attribute, a discrete risky attribute, and a sure attribute, and later taking the created choice data as given and estimating the three models on each data set.

The weighted utility formulations worked well. In all the simulation runs the continuous model specification gave more consistent results than the discrete one, which should be expected due to the simpler functional form. The true b-parameters were more consistently retrieved in both specifications than α. When true value of α was set to strongly violate FSD, only the continuous model specification was able to converge reliably and retrieve the correct values. But when true α was set to 0.15, which still moderately violated FSD, the discrete model formulation converged each time and the mean of the 50 parameter estimates (0.1864) was within two standard deviations of the true value of 0.15. When the true α-value was set to not violate FSD, the weighted utility models retrieved the true parameters very well. The same held when the true behavior was created by mean value utility, that is to say the true α had a value zero.

When the true behavior was generated by weighted preferences, but was estimated by mean value utility model, the estimated parameters were consistently downward biased towards a point where their proportions stay true. This demonstration is something that should be taken into account in the interpretations of models where the ratio of parameters is assumed to not contain a risk premium for the unreliable attribute, like in the value-of-time estimation. If the true preferences driving the choices comply with weighted utility, the parameters estimated from a mean value utility model will produce estimates that include a risk premium.

4 Conclusions

The model simulations demonstrated that the weighted utility logit models give reliable estimates in a wide range of true weighted utility risk preferences. Especially the discrete version of the model poses possibilities for situations where the decision maker tends to succumb to Allais paradox and bases his decisions on a small number of perceived possible realizations of the risky alternative.

References

[Camerer, 1989] C.F. Camerer. An Experimental Test of Several Generalized Utility Theories. *Journal of Risk and Uncertainty*. 2:61-104, 1989.

[Chew, 1982] S.H.Chew. *A Mixture set axiomatization of weighted utility theory*. Discussion Paper 82-4, College of Business and Public Administration, University of Arizona, Tuscon, 1982.

[Chew, 1983] S.H. Chew. A generalization of the quasilinear mean with applications to the measurement of income inequality and decision theory resolving the Allais paradox. *Econometrica,* 51:1065-1092, 1983.

[Chew and MacCrimmon 1979] S.H. Chew and K.R. MacCrimmon. *Alpha-nu choice theory: a generalization of expected choice theory.* Working paper 669, Faculty of Commerce and Business Administration, University of British Columbia, Vancouver, Canada, 1979.

[Chew and Waller 1986] S.H. Chew and W.S. Waller. Empirical Tests of Weighted Utility Theory. *Journal of Mathematical Psychology*,30:55-72, 1986.

[Conlisk 1989] J. Conlisk. Three variants on the Allais Example. *American Economic Review,*79:392-407, 1989.

[Fishburn 1981] Fishburn. An axiomatic characterization of skew-symmetric bilinear functionals, with applications to utility theory. *Economic Letters,*8:311-313, 1981.

[Fishburn 1983] Fishburn. Transitive measurable utility. *Journal of Economic Theory* 31:293-317, 1983.

[Fishburn 1988] Fishburn. *Nonlinear Preference and Utility Theory.* Johns Hopkins University Press, Baltimore, 1988.

[Nakamura 1984] Y. Nakamura. *Nonlinear measurable utility analysis*. Ph.D. dissertation, University of California, Davis, 1984.

[Nakamura 1985] Y. Nakamura. *Weighted linear utility.* Preprint. Department of Precision Engineering. Osaka University, Osaka, Japan, 1985.

Ontology based analysis of experimental data

Andrea Splendiani
Institute Pasteur, Unité de Biologie Systémique, rue du dr. Roux 25-28, 75015 Paris, France
University of Milano-Bicocca, DISCO, via Bicocca degli Arcimboldi 8, 20126 Milano, Italy
andrea.splendiani@unimib.it

Abstract

We address the problem of linking observations from reality to a semantic web based knowledge base. Concepts in the biological domain are increasingly being formalized through ontologies, with an increasing adoption of semantic web standards. At the same time biology is becoming a data-centric science, since the increasing availability of high throughput technologies yields a humanly intractable amount of data describing the behavior of biological systems at the molecular level. This creates the need for automated support to interpret biological data given the pre-existing knowledge about the biological systems under study. While this is currently addressed through the analysis of attributes associated to biological entities, the availability of ontologies that represent biological systems makes it possible to improve the extent to which pre-existing knowledge can be used. The semantic web, in particular, provides a framework to integrate and create a formalized biological knowledge base. Linking ontological knowledge to observed data is inherently approximate, because of the quality of observations, the relation between observed data and entities and the classification of entities. We present an overall framework project and its current development status.

1 Introduction

Scientific exploration constantly involves relating experimental evidence to existing knowledge. In the Life Science fields existing knowledge is commonly encoded in a corpus of scientific publications. This knowledge is used by scientists to design and interpret experiments that in turn lead to new discoveries. Traditionally this meant to relate a limited amount of experimental evidence to pre-defined hypotheses.

Recently, the availability of high-throughput technologies, such as DNA sequencing, mRNA and proteomic profiling is challangingchallenging this existing paradigm. Such technologies allow the observation of the behavior of biological systems at the molecular level on a system-wide basis. This means that a humanly intractable amount of data is available, most of which does not relate to previous hypotheses. Thus relating such a large scale experimental evidence to the existing biological knowledge is essential in order to understand the phenomenon under study.

Given the vast amount of data generated by high throughput technologies, this necessitates automated support.

At the same time there is an increasing availability of structured biological information, in the semantic web framework in particular. The Gene Ontology[Ashburner et al., 2000] initiative provides ontologies describing functions of gene products. It currently encompasses more than 17000 terms linked by relations of inheritance and containment and it is available in RDF. MGED-ontology[1] provides an ontology to describe attributes relevant to mRNA experiments in OWL, and the BioPAX[The BioPAX workgroup] initiative is defining a common standard to represent biological pathways and interaction networks in OWL. This last initiative is of particular interest since it provides a common ontological framework for the unification of different resources. The availability of such resources makes it possible to partially automatize the association between experimental evidence and existing knowledge to effectively lead data analysis.

2 Ontologies and data analysis

Focusing on ontologies that describe the behavior of biological systems at the molecular level, there is a range of ontologies that vary in scope and depth. While available knowledge of some biological systems is enough to build causal models, in general such knowledge is limited and most of ontologies have a low ontological commitment. When dealing with system-wide observations, this second class of ontologies is most relevant.

For instance, in the case of mRNA profiling, the behavior of thousands of genes in a cell in response to some sort of stimuli is observed. For each gene, measures of its activity are provided. These data are usually related to Gene Ontology to derive a functional characterization of the cell response.

Associations of genes to specific classes in Gene Ontology are determined based on available knowledge. By its semantics, association of a gene to a class implies association of a gene to its super classes too. Thus a gene is annotated with a set of classes that act as attributes describing specific biological functions.

[1] www.mged.org

It is common practice to define a subset of relevant genes from experimental data and to study the incidence of these attributes derived from Gene Ontology through statistical tests[Beissbarth et al., 2004; Maere et al., 2005].

Sometimes relations of inheritance and independence are used to measure “conceptual distances” among genes[Joslyin et al., 2004].

3 Uncertainty

Uncertainty plays a key role in the task of associating experimental evidence to ontological knowledge, at several levels.

Uncertainty in the definition and relations between classes plays a limited role. There is not a specific support for uncertainty in OWL, and the definition of ontologies is an ongoing task where crisp definitions are valuable.

Association between instances and classes is one point where uncertainty plays a critical role. Almost every ontology encodes a confidence in the association through “evidence codes” (describing the kind of supporting evidence) and eventually a p-value or citations of relevant scientific literature. See [Karp et al., 2004] for an example of an ontology for experimental evidence.

Association between data and ontologies is then inherently uncertain. Uncertainty may come not only from the experimental setup and measurements, but also from the biological source of variability, and from misconceptions or omissions in the available knowledge.

4 Our project

The way experimental data are associated to existing ontologies now does not take into account all the information encoded in ontologies and does not provide a way to reason over related uncertainty. We plan to overcome these limitations by providing a framework for approximate reasoning based in ontologies.

In particular, we focus on OWL ontologies describing biological pathways and on mRNA data. Given an ontology representing a collection of pathways and related concepts (including evidence support), and a set of experimental data, we define a new ontology as the union of the two, representing observed evidence and the previous knowledge.

Thus we plan to use the structure of previous knowledge to compute plausibility of concepts being pertinent to observed conditions. This can be done through a rule based approach, where inherent structure of pathways ontologies would ensure convergence of plausibility distributions.

5 Current development

We have developed an infrastructure where ontologies can be merged and represented. This is based on the Cytoscape[Shannon et al., 2003] software for molecular interaction analysis which is used as a link to experimental data and an interactive visualizer for RDF ontologies. Rule systems for unifying the ontologies and graph transformations to represent views of ontologies are also provided.

Based on this, we plan to provide a Bayesian network along with the ontology, or possible derivation of that, and to update the plausibility of nodes associated to concepts given evidence encoded in roots nodes. This updates involves considering both the experimental evidence, and uncertainty assessment related to it.

6 Conclusion

The Life Science community is one of the early adopters of semantic web technologies. The need to represent and integrate a vast amount of different information is pushing the development of this technology. The analysis of high-throughput data poses naturally the need of approximate reasoning and uncertainty representation.

References

[Ashburner et al., 2000] Ashburner M, et al. *Gene ontology: tool for the unification of biology.* The Gene Ontology Consortium. Nat Genetics 2000 May;25(1):25-9

[The BioPAX workgroup] The BioPAX workgroup. *BioPAX: Biological Pathways Exchange*. www.biopax.org

[Beissbarth et al., 2004] Beissbarth T, Speed TP. *GOstat: find statistically overrepresented Gene Ontologies within a group of genes*. Bioinformatics 2004 Jun12;20(9):1464-5

[Maere et al., 2005] Maere S, Heymans K, Kuiper M. *BiNGO: a Cytoscape plugin to assess overrepresentation of Gene Ontology categories in biological networks.* Bioinformatics 2005 Aug 21; 3448-3449

[Karp et al., 2004] Karp PD, Paley S, Krieger CJ, Zhang P. *An evidence ontology for use in pathway/genome databases*. Pac Symp Biocomput. 2004; 190-201

[Joslyin et al., 2004] Joslyn CA, Mniszewski SM, Fulmer A, Heaton G. *The gene ontology categorizer.* Bioinformatics 2004 Aug 4;20 Suppl 1:I169-I177

[Shannon et al., 2003] Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B, Ideker Y. Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res. 2003 Nov;13(11):2498-504

Position Paper: Paraconsistent Reasoning for the Semantic Web

S. Schaffert[1], **F. Bry**[2], **P. Besnard**[3], **H. Decker**[4], **S. Decker**[5], **C. Enguix**[5], **A. Herzig**[3]
[1] Salzburg Research Forschungsgesellschaft, Salzburg, Austria
[2] Ludwig-Maximilians-Universität München, Germany
[3] Institut de Recherche en Informatique de Toulouse, France
[4] Instituto Tecnológico de Informática, Valencia, Spain
[5] Digital Enterprise Research Institute, Galway, Ireland

Abstract

Due to the Semantic Web's decentralised and distributed management, contradictory information is and will remain frequent. However, classical reasoning systems fail to work properly in the presence of inconsistencies, because they implicitly or explicitly assume the *ex contradictione quod libet (ECQL)* principle stating that anything follows from contradictory premises. *Paraconsistent reasoning* challenges this ECQL principle.

Stressing practical cases of reasoning on the Web, this position paper first argues that paraconsistent reasoning is likely to become a key issue for successful deployment of the Semantic Web. Then, it briefly introduces the main approaches to date to paraconsistent reasoning.

1 Introduction

Classical and other logic, upon which modern computing is based, requires the complete absence of contradictions. With the classical *ex contradictione quod libet* (ECQL) rule (or *principle of explosion*), everything, and thus nothing useful at all, can be inferred from a contradiction. For instance, from a contradiction in a train information system can be derived that the moon is made of green cheese. Nonetheless, inconsistencies play an important role in practice (Section 2).

Paraconsistent logics are a rather novel direction in mathematical logics that challenge the ECQL principle in order to allow "reasonable" reasoning in the presence of inconsistencies without introducing more problems than are already present in the data. Several different approaches to paraconsistent logics exist and are briefly outlined in Section 3.

We conclude this article with a perspective for paraconsistent reasoning on the Semantic Web (Section 4).

2 Cases for Paraconsistent Reasoning

Distributed Information Systems

In distributed information systems, like the online information systems of European railways companies, contradictory information is frequent. For example, the German railway company might give different arrival times for trains to Paris than the French railway company, because construction works on the track in France have not been entered into the German system. Human beings can easily cope with such inconsistencies in various ways (e.g. identify which information is more likely or "don't care"). Reasoning systems on the (Semantic) Web must equally be able to derive useful conclusions from the "inconsistency-free" premises.

Coping with Change

Belief change is the field of artificial intelligence devoted to the rational change of belief in the light of new evidence. E.g., a train timetable might be updated with new train connections that have to be taken into account in further reasoning. Likewise, train connections might have been removed making previously drawn conclusions invalid.

In practice, changes like updates to an information system may cause inconsistencies that cannot be discarded. Standard methods for belief change are based on classical logic and hence accept the ECQL principle. As a consequence, they cannot be used for deriving useful conclusions in presence of updates causing contradictions.

Inconsistencies Welcome!

In some situations, inconsistencies are even desirable. This is, e.g., the case when contradictory viewpoints are present and need to be reconciled. For instance, two ontologies describing appartment rental offers and appartment sale offers might well inconsistently describe preferences and prices for city areas. This obviously should neither prevent considering both ontologies nor deriving *meaningful* conclusions in the same reasoning context (like helping in taking a decision for buying or renting an appartment). Obviously, human beings are capable of doing so without applying the ECQL principle, and so should automated reasoning systems on the Web.

Another example is policy reasoning. At the beginning of a negotiation towards selling/buying a Web service, the policies of the buyer and seller might be contradictory. Instead of applying the ECQL principle, a reasoning system should strive to overcome the inconsistencies, i.e. find a way to pass a contract acceptable for both the service buyer and seller without requiring them to change their policies.

"Dialetheias"

In practice, there are cases where contradictions are inherent to the problem, so-called "dialetheias". Since such cases arise in knowledge modelling, they will also arise on the Semantic Web. This is in particular the case with the well known

Liar's Paradox where a sentence states its own falsity ("this sentence is not true").

On the Semantic Web, dialetheias might easily arise though reification, especially of RDF statements, and through modalities – such as "A believes B" or "A does not believe what B states" – that are needed e.g. for policy reasoning. Liar sentences can also be indirect consequences of statements that are themselves unproblematic, e.g. when combining knowledge from different Web resources.

3 Approaches to Paraconsistent Reasoning

Most approaches to paraconsistent logic and reasoning allow a formula F and its negation $\neg F$ to hold in an interpreation (or "model"). Major approaches of paraconsistent logics and reasoning are stressed below:

Relevant Logics
Relevant logics have been first proposed by Anderson and Belnap. Semantics for such logics based on "different worlds" have been developed by Routley and Meyer. Conjunction and disjunction behave in the usual way, but each world w has an associated world w^* such that $\neg A$ is true in w iff A is false in w^* (not in w). As a consequence, if A is true in w and false in w^*, then $A \wedge \neg A$ is true in w. Note that requiring $w^* = w$ yields the standard classical logic.

Many-Valued Systems
A multi-valued logic is a logic with more than two truth values. The formulas that hold in a multi-valued interpretations are those which have a specific truth-value, the so-called designated formulas. A multi-valued logic is paraconsistent if it allows both a formula and its negation to be designated.

The simplest approach uses three truth values: true and false, like in classical logic, and a third truth-value denoting "both truth and false" such that if a formual F has this third truth-value in an interpreation, then so does also $\neg F$. Considering the real numbers between 0 and 1 instead of discrete values results in a paraconsistent fuzzy logic.

Non-Adjunctive Systems
A non-adjunctive logic is a logic in which one cannot conclude A from $A \wedge B$. The first non-adjunctive logic, and also the first paraconsistent logic, ever proposed is the discussive (or discursive) logic of Jaskowski. In dicussive logic, several contributors state "opinions". Each opinion is consistent in itself but might be inconsistent with another opinion. A modal logic (S5) is used to define interpretations: a world corresponds to a contributor, and in it, all the contributor's sentences are true. Thus, $A \wedge \neg A$ can hold in an interpretation consisting of several worlds, but not in a single world.

Non-Truth-Functional Logics
Non-truth functional logics have been introduced by da Costa. Their idea is to make negation "non-truth-functional" while keeping the other connectives like in standard, e.g. classical, logics. Seeing an interpretation as a function mapping formulas to 0 (false) or 1 (true), a non-truth funtional logic gives rise to defining the truth-value of $\neg A$ independently of that of A (while keeping the usual functional dependencies of the truth-value of $A \wedge B$, $A \vee B$, $A \Rightarrow B$, etc. to the truth values of A and B).

4 Paraconsistency on the Semantic Web

We believe that dealing with inconsistencies will play a central role in the emergence of the Semantic Web. Paraconsistent reasoning provides foundations and techniques that will allow future applications to function properly in the presence of inconsistencies. In particular, we think that paraconsistent reasoning will influence the following areas:

Paraconsistency in Ontology Reasoning
Ontology reasoning (e.g. instance checking) on the Semantic Web is usually based on reasoning techniques, e.g. the tableaux calculus, developed for *description logics*. Therefore, a first step towards an "inconsistency-aware" Semantic Web will be to adapt existing reasoning algorithms using techniques from paraconsistent reasoning.

Paraconsistency in Query Languages
Querying data plays a very important role on the Semantic Web, as indicated by the multitude of existing Semantic Web query languages. Building upon ontology reasoning, Semantic Web query languages will likely need to be adapted so as to work in the presence of inconsistencies.

Paraconsistency and Trust
In a distributed environment like the Semantic Web, where anyone can author content, trust is a key issue. Conflicts with classical logic are apparent: for example, different sources might make conflicting assertions about the trustworthiness of a site, and users might be interested in more fine-grained levels of trust besides the binary "trusted" or "not trusted".

References

[1] N.D. Belnap. A Useful Four-valued Logic: How a computer should think. In A.R. Anderson, N.D. Belnap, and J.M. Dunn, editors, *Entailment: The Logic of Relevance and Necessity*. Princeton University Press, 1992.

[2] P. Besnard and A. Hunter, editors. *Handbook of Deasible Reasoning and Uncertainty Management Systems*, volume 2. Kluwer Academic Publishers, 1998.

[3] François Bry. An Almost Classical Logic for Logic Programming and Nonmonotonic Reasoning. In *Paraconsistent Computational Logic 2002*, 2002.

[4] N.C.A. da Costa. On the Theory of Inconsistent Formal Systems. *Notre Dame Journal of Formal Logic*, 15(4), 1974.

[5] J.M. Dunn and G. Restall. Relevance Logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 6. Kluwer Academic Publishers, 2nd edition, 2002.

[6] S. Jaskowski. Propositional Calculus for Contradictory Deductive Systems. In *Studia Logica*, volume 24. 2001.

[7] G. Priest. Paraconsistent Belief Revision. In *Theoria*, volume 67. 2001.

[8] G. Priest. Paraconsistent Logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 6. Kluwer Academic Publishers, 2nd edition, 2002.

Representing Probabilistic Relations in RDF

Yoshio Fukushige
Network Development Center
Matsushita Electric Industrial Co., Ltd.
4-5-15 Higashi-shinagawa, Shinagawa-ku, Tokyo 140-8632, Japan
fukushige.yoshio@jp.panasonic.com

Abstract

Probabilistic inference will be of special importance when one needs to know how much we can say with what all we know given new observations. Bayesian Network is a graphical probabilistic model with which one can represent probabilistic relations intuitively and several effective algorithms for inference are developed. This paper reports a now ongoing work in its design stage which provides a vocabulary for representing probabilistic knowledge in a RDF graph which is to be mapped to a Bayesian Network to do inference on it.

1 Introduction

In the real world, especially in the scientific fields like Life Science, or in applications like contents classification and recommendation, it is often the case that relationship between resources holds probabilistically, or we can make statements only with uncertainty. Such relationships can be well described with probabilistic model. And probabilistic inference will be of special importance when one needs to know how much we can say with what we know incompletely.

In this position paper, I report my ongoing work which provides a vocabulary for representing probabilistic knowledge in a RDF graph. I introduce Bayesian Networks and list the requirements for the representing language and candidate vocabulary.

2 Bayesian Network

A Bayesian Network(BN) (Pearl 88) [1] is a graphical model to represent probabilistic relations. It is a directed acyclic graph (DAG), representing probabilistic dependencies among a set of propositions. A node represents a set of exhaustive and exclusive set of propositions (called 'variable' or 'partition'). A link represents a direct dependency between the variables. Each node is accompanied with a conditional probability table (CPT) that represents the probabilistic relationship between the variables. The posterior probability distributions ("beliefs")for each variable could be calculated by propagating beliefs.

Figure 1 shows an example illustration of a BN (CPTs are not shown). It has 5 nodes and 5 links connecting them.

Advantages of employing BN are among others:

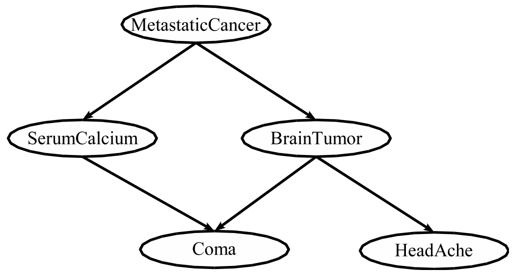


Figure 1: example Bayesian Network

- the relations are expressed by a graph, which is a familiar notion in the Semantic Web, and thus intuitive and easy to understand
- effective calculation algorithms (including simulation methods) have been developed

3 Requirements for the representation language

The aim of this work is not to just represent Bayesian networks in the Semantic Web, but to get a language (or extension vocabulary) which can describe probabilistic relations in a way that is Semantic Web compatible and easy to map to a BN. It is to put together the distributed information in the Semantic Web, and do probabilistic inference in the BN.

The components of a BN are nodes and links and CPT's attached to the nodes. A node represents a set of exhaustive and mutually exclusive propositions (called partition).

The representation language should be able to express:

- a partition, i.e. a set of exhaustive and exclusive propositions
- propositions in such a way that they are distinguished from ground facts
- a probability with which a proposition holds with/without certain conditions

4 Vocabulary for RDF representation

RDF is a W3C standard as one of the fundamental building blocks of the Semantic Web. By representing probabilistic relations in RDF, one gets advantage of reusing existing vocabularies and tools for RDF processing, and one can treat the probabilistic relations themselves as resources and incorporate them into RDF graphs.

To provide a vocabulary that satisfy the requirements above, I introduced the following classes and predicates:

classes `prob:Partition,`
`prob:ProbabilsticStatement,`
`prob:Clause, prob:Probability,`

predicates `prob:predicate,`
`prob:hasProbability, prob:condition,`
`prob:case, prob:about`

Details are omitted because of the limit of the space. Details are to be available at<http://www.w3.org/2005/08/08-prob/>.

Points to note are:

- Conditions are linked with `prob:Partition`'s, not with each cases.
- Introduction of `prob:Clause`'s
  `prob:Clause` is a generalization of the RDF reification. `prob:Clause`represents has one `prob:predicate` and zero or more 'terms.' (cf. the pattern 2 of N-ary relationship representations in [9])

The following is an example graph which represents a probabilistic relation: "if cond0, then case1 has probability prob1 and case2 has probability prob2" (in a Turtle [2] serialization)

```
[a prob:Partition;
 prob:condition :cond0;
 prob:case
  [a prob:ProbabilisticStatement;
   prob:about :case1;
   prob:hasProbability :prob1],
  [a prob:ProbabilisticStatement;
   prob:about :case2;
   prob:hasProability :prob2].
].
```

5 Related works

(Ding & Peng 2004) [4] and (Gu, Pung & Zhang 2004) [5] are close works to this. They proposes to augment OWL to allow probabilistic markups, and provides a set of transformation rules to convert the probabilistically annotated OWL ontology into a BN.

(Holi & Hyvönen 2004) [6] is an attempt to express and calculate overlapping of concepts in RDF(S) and OWL using BN.

Works in combinatorial use of BNs and Description Logics includes, among others, (Koller, Levy & Pfeffer 1997) [7] which presents P-CLASSIC; a probabilistic version of the Description Logic CLASSIC, and (Yelland 2000) [8] which incorporates Description Logics into BNs.

While others address T-Box knowledge, (Fukushige 2004)[3] proposes a method to encode probabilistic relations in A-Box, which is in the same direction with this work.

6 Conclusion and future works

This position paper reported a ongoing work which provides a vocabulary for representing probabilistic knowledge in a RDF graph.

Open issues (other than implementing issues) include:

- Relationship with rule languages
- How to standardize Query Languages against BN store
- How to learn BNs from data or/and partial description in RDF.
- How to deal with / avoid cyclic probabilistic description in RDF
- How to deal with continuous probabilistic distributions
- Examination of computational complexity

References

[1] Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference, Morgan Kaufmann, 1988

[2] Becket, D.: Turtle - Terse RDF Triple Language, http://www.ilrt.bris.ac.uk/discovery/2004/01/turtle/

[3] Fukushige, Y.:Representing Probabilistic Knowledge in the Semantic Web, position paper for the W3C Workshop on Semantic Web for Life Sciences, http://www.w3.org/2004/09/13-Yoshio/PositionPaper.html, 2004

[4] Ding, Zh., Peng, Y.: A Probabilistic Extension to Ontology Language OWL, Proc. of the 37th Hawaii International Conference in System Sciences, 2004

[5] Gu, T., Pung, H.K., Zhang, D.Q.: A Bayesian Approach for Dealing with Uncertain Contexts, Proc. of the Second International Conference on Pervasive Computing (Pervasive 2004) ,"Advances in Pervasive Computing", Austrian Computer Society, vol. 176, 2004

[6] oli, M., Hyvönen, E.: A Method for Modeling Uncertainty in Semantic Web Taxonomies, Proceedings of the 13th international World Wide Web conference, pp.296-297, 2004

[7] Koller, D., Levy, A., Pfeffer, A.: P-CLASSIC: A tractable probabilistic description logic, Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97), pp.390-397, 1997

[8] Yelland, P.M.: An Alternative Combination of Bayesian Networks and Description Logics, Proceedings of the KR2000 Conference, pp.225-234, 2000

[9] Noy, N., Rector, A.: Defining N-ary Relations on the Semantic Web: Use With Individuals, W3C Working Draft, http://www.w3.org/TR/2004/WD-swbp-n-aryRelations-20040721/, 2004

Statistical Reasoning – A Foundation for Semantic Web Reasoning

Shashi Kant • Evangelos Mamas
Massachusetts Institute of Technology
Cambridge, MA 02139
skant@sloan.mit.edu • emamas@sloan.mit.edu

Abstract

There has been considerable debate as to the merits and the applicability of probabilistic or statistical reasoning to Semantic Web. Much of this debate seems to have centered on the applicability of statistical methods in a supposedly deterministic setting. In this paper, we argue that statistical reasoning ("reasoning with uncertainty") need not be a substitute for traditional Description Logic (DL) / First-Order Logic (FOL) reasoning, instead statistical methods can serve as a complement to logic-based reasoning systems in two ways: (i) Offer a meta-reasoning (or audit) mechanism to validate logical reasoning, and (ii) Act as a "filler" where Ontological information either does not exist, or is insufficient to reason conclusively.

1 Introduction

Much of the Semantic Web effort has focused on the design and development of Ontologies and related technologies. This approach presupposes that a critical mass of Ontologies will exist that can sufficiently and accurately respond to reasoning queries. As Sir Tim Berners-Lee puts it [Berners-Lee, 1998]: "*The choice of classical logic for the Semantic web is not an arbitrary choice among equals. Classical logic is the only way that inference can scale across the web*."

However, a pure logic-based approach looks increasingly implausible given the paucity of Ontologies and the difficulty in constructing and maintaining Ontologies. Just like the World Wide Web (WWW) had a ready and mature platform to run on i.e. the Internet - which had been in existence for a long time prior to the emergence of the WWW, we feel that the Semantic Web needs an underlying platform, upon which Ontologies can function and interoperate.

We argue that this platform should be a web of statistical "metadata" – which expresses semantic relations in probabilistic terms. Such systems (e.g. Bayesian Networks, Probabilistic Relational Models) have also been in existence for a while and are used in various Machine Learning and AI applications such as Machine Vision, Speech Recognition, and Robotics etc. The Semantic Web would do well to re-use some of these efforts in building this underlying framework.

2 Ontologies and Probabilistic Models

We introduce the notion that Probabilistic Graph Models (PGM) or Bayesian Networks can be viewed as *fuzzy* Ontologies; conversely an Ontology can be viewed as a *crisper* Bayesian Networks. In our proposed architecture, there may not be a clear dividing line between them. A good way of visualizing this relation would be to view Ontologies and Bayesian Networks as ships floating in a sea of statistical "metadata". We use this metaphor to describe the notion that the sea of statistical metadata fills-in the gaps between the islands of Ontologies. Lately there have been some efforts to develop Probabilistic Ontologies by annotating OWL or RDF Ontologies with probabilistic information e.g. BayesOWL [Ding,Peng 2004]. We argue against this approach, and suggest that probabilistic and logic-based reasoning approaches should be viewed as orthogonal to each other. It makes most sense to keep the Ontological information separate from the statistical data, along the lines of how the WWW operates - in which an HTML page *links* to a "FTP" site or a "mailto" to an email hyperlink and the necessary protocols invoked only when clicked.

Figure 1 illustrates a hierarchical mechanism of aligned Ontologies and Bayesian Networks. At the very top are the top-level Ontologies on which there is general agreement and acceptance, at the bottom are the fuzzier, grayer-scale Bayesian Networks which represent relations between resources using probabilistic mechanisms.

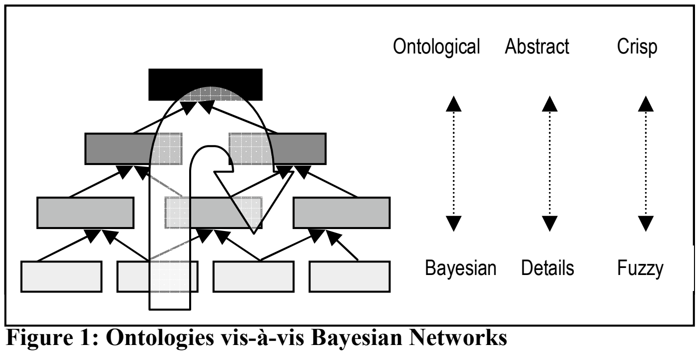


Figure 1: Ontologies vis-à-vis Bayesian Networks

We suggest, that probabilistic (or statistical) information be encoded using any of the widely accepted Bayesian Interchange Formats such as XML-BIF [Cozman, 1998], or Microsoft Research's XBN [Microsoft, 1998] or Hugin.net [Jensen, 2004] format. We propose that the Ontological model encapsulate what it is designed for - expressing logical relations between resources, and the probabilistic model express the statistical relation between them. We do not see a need to mix-and-match as they offer very different views on the same information-set and are perceptually orthogonal.

3 A Hybrid Reasoning Model

Reasoning using Ontologies is based on predicate logic and belongs in the classical tradition of monotonic deductive reasoning i.e. propositions are either true or false. But this proposed framework provides a mechanism for handling fuzzier, incomplete and inaccurate inputs. In this model, reasoning can be performed using a "bottom-up" approach where a query unanswered by a pure Ontological match is extended further up the hierarchy (Fig 1.) until all required information is found. An adjunct application might be to validate traditional reasoning with a mathematical confidence level (meta-reasoning).

Some examples of the reasoning activities possible using this system are:

1. **Deductive Reasoning**: Deductive reasoning allows a system to deduce information given a set of (possibly incomplete and erroneous) information. For example, it can *deduce* that the best course to learn "Machine Vision", "Genomics" and "Political Science" at MIT is *most probably* "6.804J Computational Cognitive Science" even though the course does not directly teach Political Science. It is making a best-guess fit for the requirements [OCW, 2005].
2. **Abductive Reasoning**: Abductive reasoning allows a system to infer the possible causes for a certain effect. For example, the possible courses for learning Artificial Intelligence at MIT are 6.803, 6.825 etc. This is the equivalent of diagnostic reasoning in Bayesian Networks [OCW, 2005].
3. **Monotonic reasoning, non-monotonic reasoning and default values**: Traditional DL-based Ontologies can represent information for monotonic reasoning. For example, one might declare that Universities in the US have a GPA scale of 4.0, but MIT uses a 5.0 GPA scale – so the system *monotonically* cannot reason with that information unless it has been explicitly encoded.

This kind of *non-monotonic* reasoning is possible with the proposed approach.

4 Conclusion

"Reasoning with Uncertainty" is probably a misnomer to describe the efforts required in this area - a more appropriate phraseology would be "reasoning without certainty". While the difference may seem pedantic, the underlying notion is that "uncertainty" is not a state unto itself, but merely the absence of certainty. In a Semantic Web sense, it is a state where Ontological information is non-existent, incomplete or inconclusive. Statistical reasoning could therefore be the bedrock upon which DL/FOL based querying and reasoning can be performed.

This means that the semantic web can operate in areas currently out-of-bounds because of a lack of Ontological information. We therefore hypothesize that statistical "metadata" could be the building-block of the Semantic Web leading to better and more accurate reasoning mechanisms.

5 Acknowledgments

We would like to acknowledge the gracious help and support of the members of the World Wide Web Consortium (W3C) and faculty of MIT-CSAIL for their help and support. We would especially like to thank Ralph Swick and Sir Tim Berners-Lee for their critique and feedback.

6 References

[Berners-Lee, 1998] Tim Berners-Lee. Axioms of Web Architecture: n, 1998, Available at: http://www.w3.org/DesignIssues/Rules.html, Accessed on June 20, 2005.

[Cozman, 1998] Fabio Gagliardi Cozman, The Interchange Format for Bayesian Networks, 1998, Available at: http://www-2.cs.cmu.edu/afs/cs/user/fgcozman/www/Research/InterchangeFormat/, Accessed on June 23, 2005.

[Microsoft, 1998] Microsoft Research, XML Belief Network File Format: Main Page, 1998, Available at: http://research.microsoft.com/dtas/bnformat/ Accessed on June 23, 2005.

[Jensen, 2004] Finn V. Jensen, A Brief Overview of the Three Main Paradigms of Expert Systems, Available at: http://developer.hugin.com/Getting_Started/Paradigms/, Accessed on June 23, 2005.

[OCW, 2005] MIT Open Courseware, Massachusetts Institute of Technology, Available at: http://ocw.mit.edu, Accessed on June 23, 2005.

[Ding, Peng, 2004] Zhongli Ding and Yun Peng. A Probabilistic Extension to Ontology Language OWL, in *Proceedings of the 37th Hawaii International Conference on System Sciences* - 2004