MODELING UNCERTAINTIES AND BELIEFS USING ONTOLOGIES

Melvin S. Metzger
B.Sc. Thesis, Computer Science

SEPTEMBER 8, 2014

GOETHE-UNIEVRSITY FRANKFURT AM MAIN
DEPARTMENT OF COMPUTER SCIENCE

Table of Contents

Abstract ..... **Error! Bookmark not defined.**
1. Introduction 2
   1.1. The Semantic Web 2
   1.2. RDF and RDFS 3
   1.3. OWL 4
   1.4. Uncertainty 4
   1.5. Illustrations 5
   1.6. Examples 6
2. ontology modeling approach 7
   2.1. W3C's Uncertainty ontology 8
   2.2. International Council of Museums Conceptual Reference Model 14
   2.3. Uncertainty Framework 21
   2.4. Comparing modeling approaches 25
3. Mathematical approaches 28
   3.1. BayesOWL 28
   3.2. Fuzzy OWL 2 32
   3.3. URDF 36
   3.4. Comparing mathematical approaches 38
4. Conclusion 39
Appendix A 40
Appendix B 40
Appendix C 41
References 43

Abstract

Dealing with uncertainties is an important issue in modelling ontologies for the Semantic Web. This thesis presents and compares different representations and mathematical approaches such as fuzzy logic, Bayesian networks and probabilistic logic to model uncertainties using ontologies.

1. Introduction

This section gives an introduction to the Semantic Web, its basis languages, uncertainty and its usefulness in the Semantic Web as well as the illustrations and examples used in this thesis. Section 2 presents and compares three approaches to the problem, using ontology modeling techniques. Section 3 presents and compares three approaches to the problem, based on different mathematical theories. Finally section 4 gives a conclusion of the findings of sections 2 and 3.

1.1. The Semantic Web

The idea of the Semantic Web is to add machine understandable semantics to the World Wide Web (WWW). The leading supporter of the Semantic Web is the World Wide Web Consortium[1] (W3C). The WWW formerly consisted only of HTML pages that on the one hand provide great freedom of publishing to the WWW but on the other hand lack the quality of structuring information so that the semantic meaning can be interpreted by machines.
The intention of the Semantic Web is to add meta-data, i.e. data about data, processible by machines and to make machines "understand" the semantics of web documents and the web of their subsumption. [1]
The Semantic Web is about giving meaning to structured data resulting in a "Web of data" [2] [3].

An example stated in [1] is that a common keyword search over documents may have a load of results suggesting documents that carry the same keyword somewhere in the document but in a completely different context than what the searcher was looking for. If the search engine knows the context searched for and meta-data about the searched documents provides it with the information on the context of each document, the search could give more precise results by matching the context as well as the keyword. Please note that this example was stated in 2003 and today's search engines use far more powerful techniques than simple keyword search. Nevertheless it is a good example to understand the advantages of the Semantic Web. For instance if the searcher is currently visiting an American football page and from there starts a search with the keyword "dolphins" the search engine could get the context "American football" from the current pages meta-data and take it into consideration. The search then could suggest pages with information about the American football team "Miami Dolphins" before suggesting information about the animal. Due to the context the first seems to be a more appropriate result.

The W3C has established languages to provide such *"machine-processible semantics of data"* [1] in order to extend the *"Web of documents"* [2] by a *""Web of data", the sort of data you find in databases"* [2].

[1] http://www.w3.org/

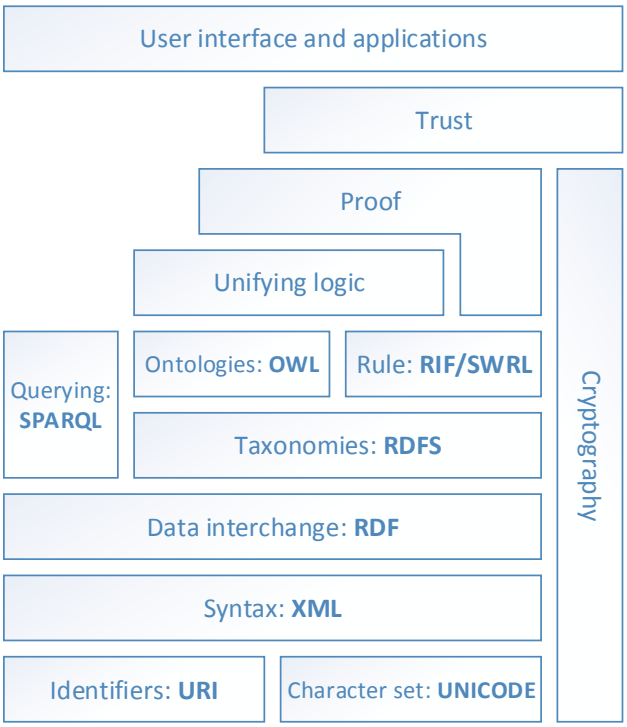

Figure 1: Semantic Web Stack

Figure 1 shows the so called Semantic Web Stack consisting of the different languages and concepts that support the Semantic Web. The figure was adjusted to fit the design of this thesis. The original figure can be found in [4]. In the following, a short introduction into the languages that appear in this thesis will be provided. The other parts of the Semantic Web Stack will not be explained in this thesis. More information on these parts can be found in [1] and [3].

1.2. RDF and RDFS

RDF stands for Resource Description Framework. *"RDF is a standard model for data interchange on the Web."* [5] The Syntax is based on XML and the core elements of RDF are the so called RDF triples that can combine two individuals identified in a relationship. An RDF triple has an intuitive resemblance to natural language. Thus the three elements are called subject, predicate and object, where the predicate denotes a property, the relationship between subject and object. Each of these components are identified with uniform resource identifiers (URI). With RDF there also comes a number of predefined properties as for example *rdf:type* that is *"used to state that a resource is an instance of a class"* [6]. Classes are used to group and properties are used to describe resources. RDFS (RDF Schema) extends RDF by class hierarchies and other useful features. For more information on RDF and RDFS please review [5], [6], [7] and [8].

RDF triples are commonly displayed as graphs as in Figure 2: RDF tripleGraph illustrations are described in greater detail in section 1.5.

Figure 2: RDF triple

1.3. OWL

"The W3C Web ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things." [9]

OWL is syntactically based on RDF and generally based on description logics. It is used to define ontologies that consist of classes, properties and individuals and represent machine-understandable knowledge. Thus OWL is a declarative language and not a programming language. Another name for the current version of OWL is OWL 2. When OWL is mentioned in this document the current version of OWL (OWL 2) is meant. For more information on the difference between OWL 2 and OWL 1 please refer to [10]. Some upcoming sections use OWL to model uncertainties in ontologies and for that, familiarity with OWL is presupposed. For more information on OWL please see to [9] and [10].

1.4. Uncertainty

"In this Report, the term "uncertainty" is intended to encompass a variety of aspects of imperfect knowledge, including incompleteness, inconclusiveness, vagueness, ambiguity, and others" [11].

This quote from the W3C's Uncertainty Reasoning for the World Wide Web [11] is best to describe how the term "uncertainty" is used in this thesis. Uncertainty is interesting to the Semantic Web for a number of reasons. Ontologies in the Semantic Web commonly work with the assumption that knowledge is certain. Nevertheless uncertainty is encountered and needs to be handled under a number of circumstances. One of these is Knowledge Integration. When two ontologies are integrated, it is most unlikely that an exact fit is achieved. Concepts of different ontologies could match only partially, individuals from one ontology could match multiple concepts of the other one and so on. For example the concept "hot" in an ontology about the weather is likely to differ partially from the concept "hot" (as the term for temperature) in an ontology about cooking. To represent such incomplete or uncertain matches it is useful to apply some degree of match or membership. Even when two ontologies can be matched exactly, the mass of information accessible in the Semantic Web comes from different sources that could (or rather should) be observed with different degrees of belief or trust. These degrees of trust and belief should also be considered when integrating knowledge. Ontology learning as an automated procedure of generating ontologies from natural language is likely to result in probabilistic or uncertain ontologies as well. In short uncertainty is a side effect of common features of the Semantic Web. [11]

Uncertainties can also be within ontologies. A common example is a weather report that contains information about how the weather is assumed to be and is usually weighted with some degree of certainty. [12] There can also be uncertain knowledge due to incompleteness of the information the knowledge is based on. Such uncertainties need to be modelled within ontologies. It is not satisfactory for every developer who encounters uncertain information to build an own model of uncertainty, because uncertain information should also be exchangeable in a manner that keeps the interpretation of the uncertainty alive. When a package, containing uncertain information that is modelled in some way, is received and the receiver has his own way of modelling and thus interpreting the uncertainty, the uncertain information could falsely be assumed certain and the uncertain information could be lost. This means that it is necessary for sender and receiver to have the same interpretation and model of uncertainty so that the receiver can interpret it in the way the sender intended. This calls for a standardized representation of uncertainty for the Semantic Web. [11]

Benefit can be received from handling uncertainty within the Semantic Web by exploiting partial information as it is typical for a large source as the WWW. "*For example, that an online service deals with greeting cards may be evidence that it also sells stationery. It is clear that search effectiveness could be improved by appropriate use of technologies for handling uncertainty."* [11]

1.5. Illustrations

In the following sections ontology models are illustrated as directional graphs.

A class is displayed with a white rectangle, an individual with a grey ellipse and a data value with a white ellipse with a dashed outline. Different namespaces and individuals are organized in “Swimlanes” which originally are used in cross-functional flowcharts. Most namespaces are denoted using short-cuts. A complete table of used namespaces is given in Appendix A. The namespace of a property or class can also be denoted as prefix as in *rdf:type* where the property type has the namespace *rdf*. For classes such prefixes are disregarded because classes are always located in the “Swimlane” of their namespace. To promote visibility the reoccurring attributes *rdf:type*, *rdfs:subClassOf, rdfs:subPropertyOf* have their own figure. *rdf:type* is illustrated with a dashed, grey arrow, *rdfs:subClassOf* with a solid, thick, grey arrow and *rdfs:subPropertyOf* with a blue dash-dotted arrow.

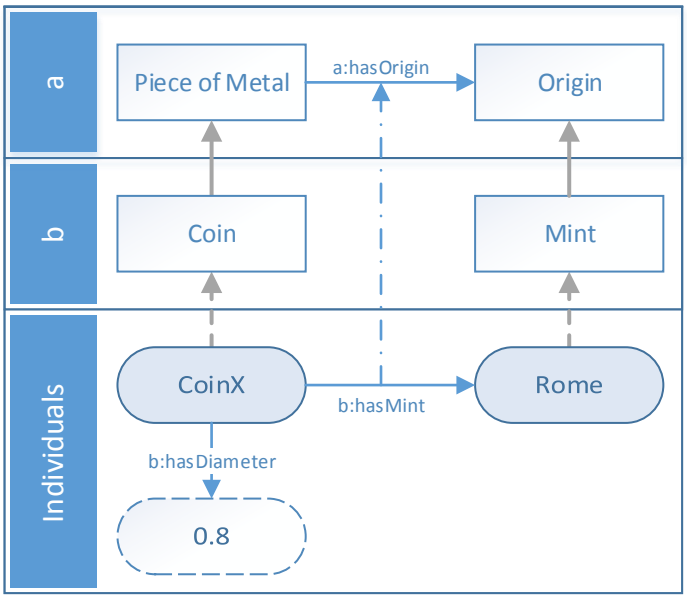

Figure 3: Graph design

Figure 3 is an example of how upcoming graphs are designed. The displayed namespaces do not actually exist and this is no suggestion to how any of the content should be modeled in an ontology.
The individual *CoinX* is of type *Coin* which is a subclass of *Piece of Metal*. The individual *Rome* is of type *Mint* which is a subclass of *Origin*. *CoinX* is linked to *Rome* via the property *hasMint* which is a subproperty of *hasOrigin* linking a *Piece of Metal* to an *Origin*. Also *CoinX* is linked to the data value *0.8* with the property *hasDiameter*. The classes *Coin* and *Mint* come from the namespace *b*. *Piece of Metal*, *Origin* and the property *hasOrigin* come from namespace *a*. Note that the property *hasOrigin* is shown to connect the corresponding classes in its namespace contrary to *hasMint* (*hasDiameter*) which is only shown to connect the individuals. This is to display that *hasOrigin* has domain and range restrictions defined and *hasMint* (*hasDiameter*) does not. For *hasOrigin* the domain is *Piece of Metal* and the range is *Origin*. For *hasMint* (*hasDiameter*) there is no restriction of domain and range defined. Nonetheless *hasMint* being a subproperty of *hasOrigin* derives the domain and range restrictions and they are fulfilled due to the given subclass relations.

1.6. Examples

To show how the different approaches model uncertainties a simple example is used throughout this document. Some coin was minted in Rome, where Rome refers to the mint not the region (as it is defined by nomisma.org[2]), is represented as: *CoinX* has *Rome* as a Mint, where *CoinX* is an instance of the class *Coin* and *Rome* is an instance of the class *Mint*. The property connecting these individuals is *hasMint*.

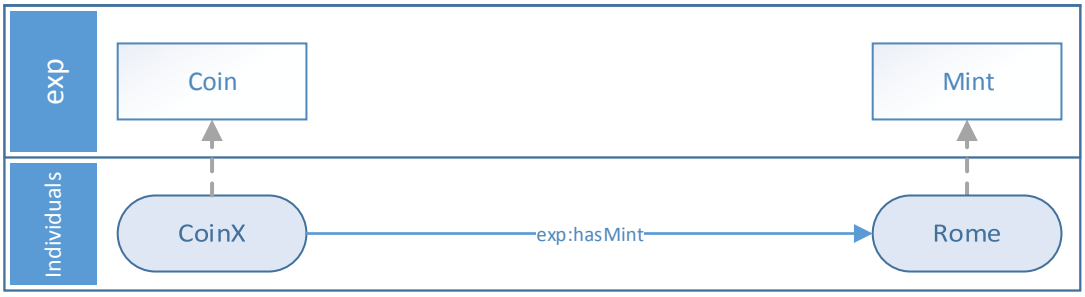

Figure 4: Example used throughout the document

Figure 4 illustrates the simple example used throughout the document.
The classes and property used in this example are leaned on, but not exactly the same as in the Nomisma ontology (NMO). NMO is an ontology developed to represent numismatic concepts for nomisma.org[3]. Especially the property *hasMint* in NMO is intended to be used in a slightly different way than in this example as explained in section 2.1. To differentiate the classes and properties made up for an example from real elements of NMO or others, they are denoted with the namespace "*exp*" standing for example.

[2] http://nomisma.org/id/rome
[3] http://nomisma.org/

2. Ontology modeling approach

The method to integrate uncertainties in ontologies, described in this section is to model uncertainties using existing principals and tools especially OWL and RDF develop ontologies. The main intention is to add a measure of uncertainty to a relation between two individuals. If for example some coin *CoinX* has *Rome* as a Mint, that fact can be described by linking both the coin and the mint via a property (*CoinX hasMint Rome*) as illustrated above.

The problem encountered when adding uncertainty to this relation as in "*CoinX* has *R*ome as a Mint but I am uncertain", is that properties in OWL (and RDF) are binary relations which means that they cannot couple more than two individuals. It is impossible to attach any degree of uncertainty to the property connecting *CoinX* and *Rome*. A solution to this problem as presented by the W3C in [13], is to add an individual representing the property and the appropriate number of properties (in this case three) to link all the relations participants as well as the additional information, namely the degree of certainty to this individual.

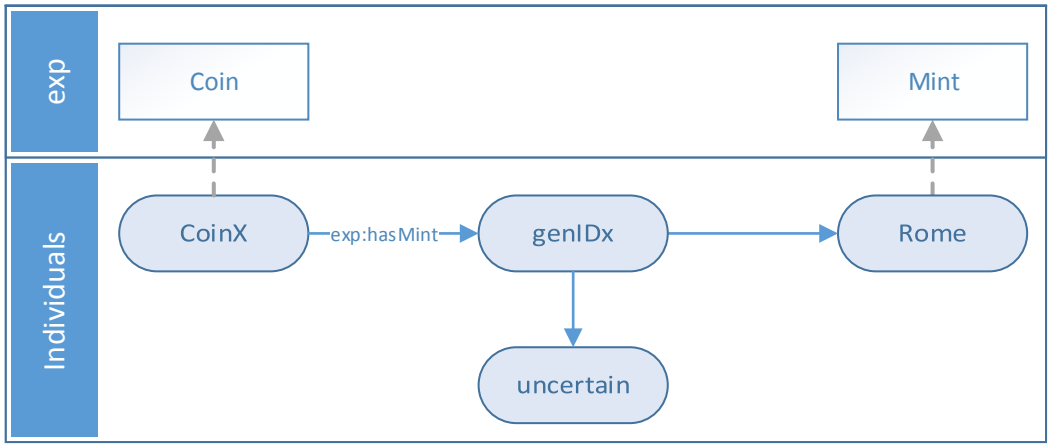

Figure 5: Information addition pattern

This illustration (Figure 5) shows how a generic ID can be used to represent the uncertain relation between *CoinX* and *Rome*. Such a generic ID can be implemented with OWL as an anonymous individual. Note that two of the properties that need to be added are unnamed. Propositions on what properties could be used instead are made in the following sections.

When there is more than one piece of information to add to the relation, this can be done by simply linking another individual representing the additional information to the individual objectifying the relation.

In the following this pattern is called the W3C's information addition pattern. Using this pattern the issue of properties being binary relations can be evaded and n-ary relations can be implemented [13].

The ontologies and modelling approaches described in sections 2.1., 2.2, and 2.3 all follow this pattern in some way.

2.1. W3C's Uncertainty ontology

This section describes the W3C's Uncertainty ontology [11] and how it is partially used to model uncertainties in the Nomisma ontology (NMO).

In the W3C's Uncertainty ontology (UN) the main idea is that an agent says some sentence about some world and this sentence has some kind of uncertainty.

A world in UN *"[..] represents the world about which the Sentence is said."* [11] In the use case *"5.13 Buying Speakers"* [4] provided by the W3C, an assertion about the availability of certain speakers in a certain store is made. In this case the world is the store in which the speakers are available (or not).

A sentence in UN is *"An expression in some logical language that evaluates to a truth-value (formula, axiom, assertion)[..]"* [11]. In the following the sentence is interpreted as a statement annotated by a property or relation between multiple entities and thus is not reduced to a single node but to the context that the node represents.

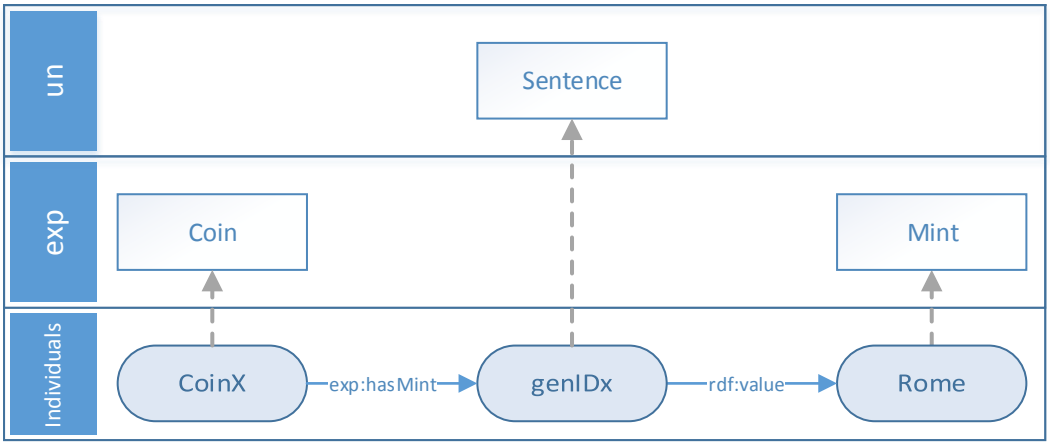

Figure 6: Use of un:Sentence

Figure 6 shows how the statement "*CoinX* has *Rome* as a Mint" can be modelled with a generic ID as intersection. In this example the node *genIDx* for itself is obviously not a sentence since "genIDx" carries no expression that evaluates to a truth-value. However when the context that the node is put in, the properties that point to or from the node, is taken into consideration the node does indeed represent a sentence.

The generic ID is a link between *CoinX* and *Rome* via the properties *hasMint* and *rdf:value* and thus represents the assertion "*CoinX* has *Rome* as a Mint".

It is necessary to point out that this interpretation of *un:Sentence* is imprecise and comes with the unsolved problem of where it ends. For the above example it is relatively simple to say that the sentence represented by *genIDx* is not only this individual but all individuals that are directly connected to it with properties. Thus a boundary could be set so that the context of the assertion represented by a *un:Sentence* consists of every individual that is connected to it with distance 1 (directly with a property) in terms of graph theory. But there can be cases where the semantic context exceeds such a boundary as illustrated in Figure 7, where two alternative mints, that can be considered part of the assertions context, are connected indirectly (distance 2) to the *un:Sentence* node.

[4] http://www.w3.org/2005/Incubator/urw3/XGR-urw3/#speakers

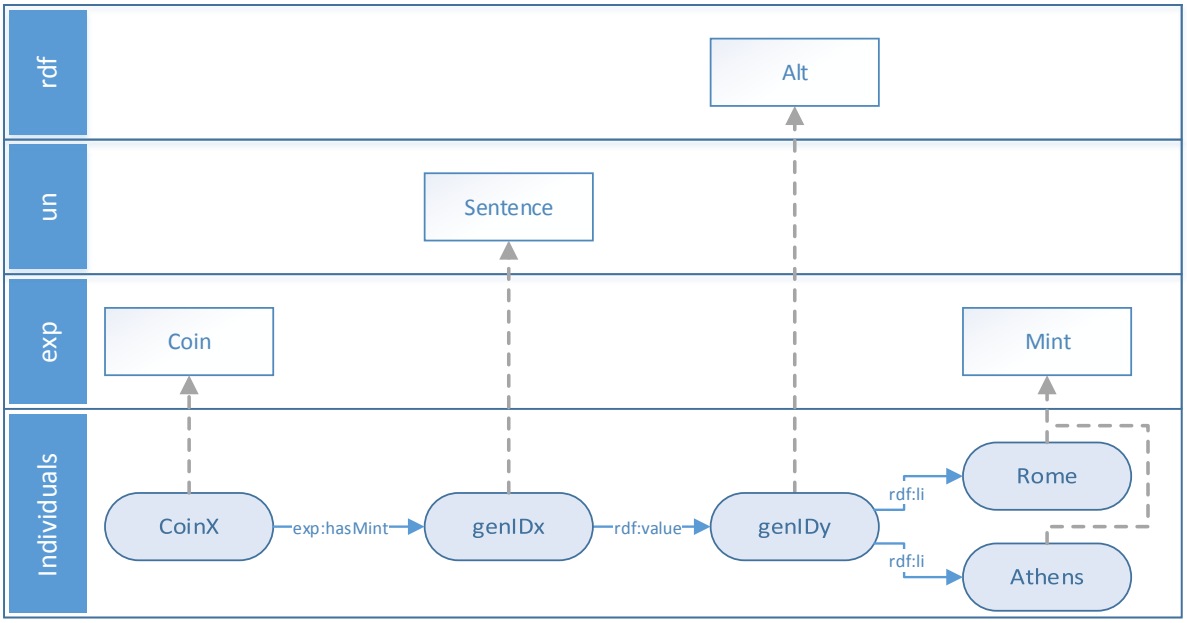

Figure 7: Problem with un:Sentence

For such cases the boundary does not hold and is not adequate. The problem of exceeding the boundary is thinkable with any boundary, so that the problem of grasping the context of an assertion represented by a *un:Sentence* is rather complicated and yet unsolved.

The above interpretation of *un:Sentence* is one of many. For example in use case *"5.13 Buying Speakers"* [5] provided by the W3C a sentence can have sub-sentences and each sentence is represented in a single node where the caption holds the information. While it works for this use case, modelling the above example in a single node would obviously lack a great measure of expressiveness.

[5] http://www.w3.org/2005/Incubator/urw3/XGR-urw3/#speakers

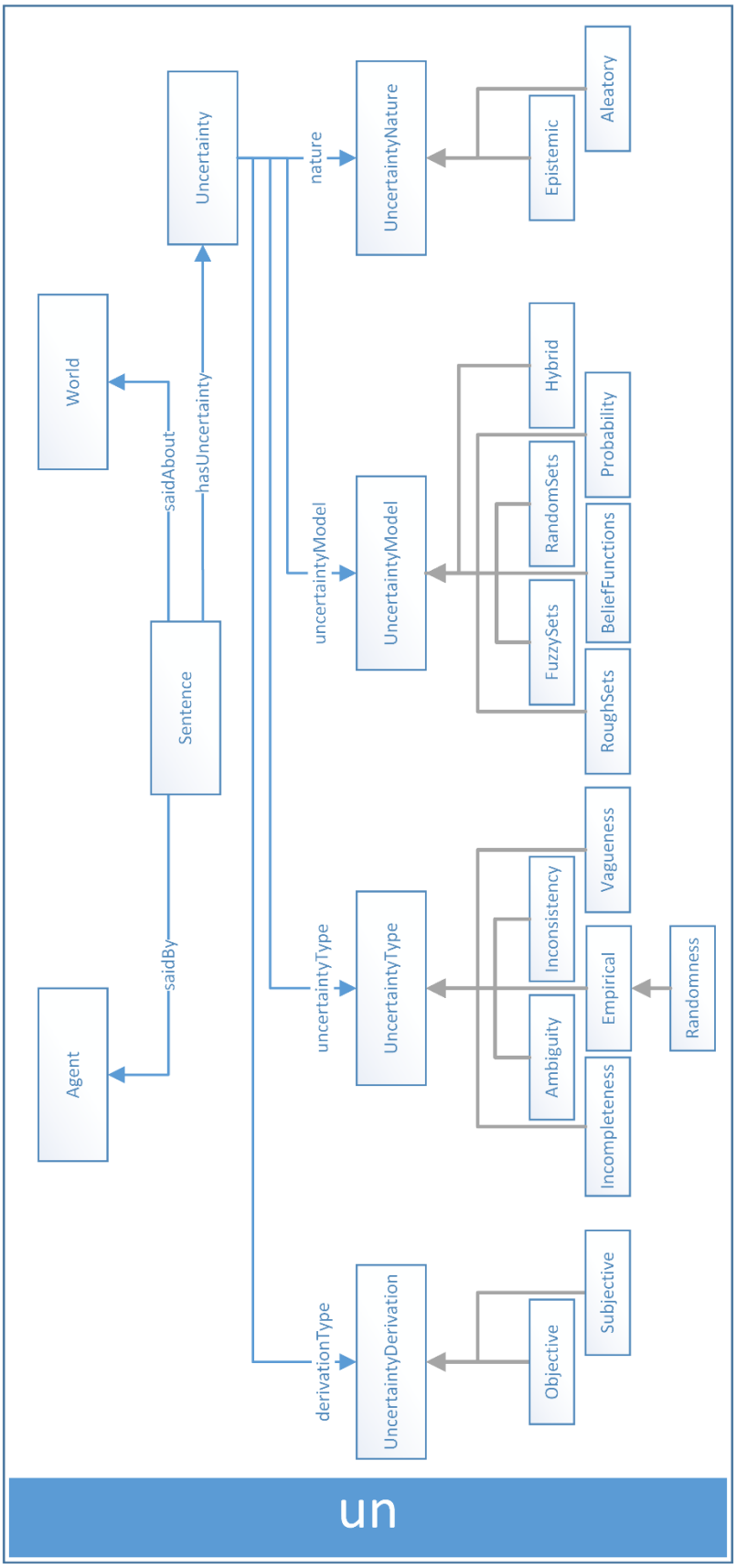

Figure 8: W3C's Uncertainty ontology

Figure 8 displays the W3C's Uncertainty ontology. It is a synopsis of the figures provided by the W3C, adjusted to fit the design of this thesis. The original figures can be found in [11]. Note that no short-cuts for namespaces are used in this illustration because everything is from the same namespace, the UN.

The concept of a *Sentence* said by someone, said about something and having some *Uncertainty* is the core of UN. Furthermore an *Uncertainty* has *Uncertainty-Derivation*, *-Type, -Model and -Nature.*

The *UncertaintyDerivation* specifies weather the *Uncertainty* is derived *Objective* or *Subjective* that is in a formal way or in some kind of subjective judgement or guess.

The *UncertaintyType* gives information whether the uncertainty is due to an *Ambiguity*, *Randomness* (which is a subclass of Empirical), *Vagueness*, *Inconsistency* or *Incompleteness* of the information or the terms in the sentence and the world it is said about.

The *UncertaintyNature* is either *Aleatory* meaning that it is a property of the world or *Epistemic* meaning that the uncertainty is a consequence of the agent's lack of knowledge.

The *UncertaintyModel* is the underlying mathematical theory as for example: *Probability*, *Fuzzy Sets*, etc. (a couple of theories are introduced in section 3).

These are all the segments of the UN.

Since this model is of a very generic nature, it can easily be adapted for more specific use as it is done with NMO.

NMO has a slightly different use of the class *Coin* and the property *hasMint* than previous examples. In NMO there is another class *type_series_item*, used to describe a coin type. The class *Coin* (in NMO) consists of individual *Coins* assigned to a *type_series_item*. This *type_series_item* describes all the features *that Coins assigned* to it have in common. Thus the property *hasMint* in NMO is not intended to connect a *Coin* with a *Mint* but a *type_series_item* with a *Mint*. For example when *type_series_item TypeX* has *Rome* as a Mint, all *Coins* assigned to *TypeX* have the *Mint Rome* in common. Note that this is only the intentional use of the property *hasMint*. NMO does not apply any domain or range constraints on this property so the property could be used in other contexts as well. The *Mint nm:rome*[6] and the uncertainty annotation *nm:unknown_value*[7] refer to URIs defined in the *nm* namespace and are used in coherence with NMO.

The following use case shows how NMO and UN should be combined (UN + NMO) to model the statement: "*TypeX* has *Rome* as a Mint but I am uncertain."

[6] http://nomisma.org/id/rome
[7] http://nomisma.org/id/unknown_value

Use case: UN + NMO hasMint uncertain

Description: *type_series_item TypeX* has *Rome* as a Mint but I am uncertain.

Note: The sentence represented by *genIDx* results from all the properties to and from it.

Used entities: *nmo:type_series_item, nmo:mint, un:Sentence, un:Uncertainty*

Used properties: *nmo:hasMint, un:hasUncertainty, rdf:value, rdf:type*

Used generic IDs:

Name	Description
genIDx	The assertion: *TypeX hasMint Rome* with the uncertainty *nmo:uncertain_value.*

Depiction:

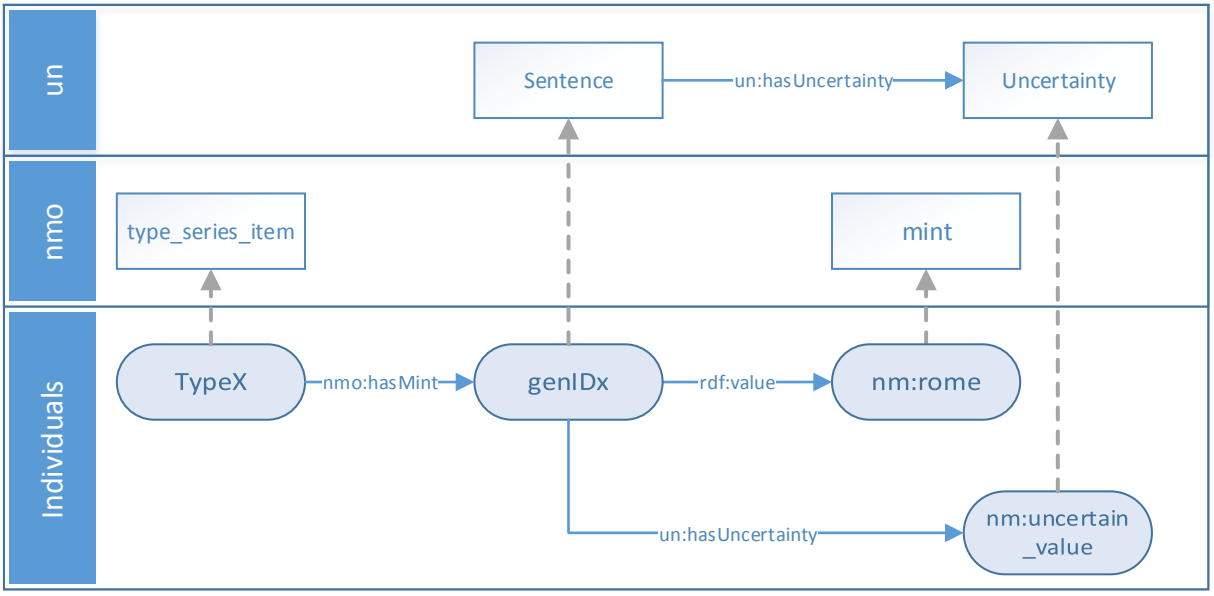

Figure 9: Use case UN + NMO

The *type_series_item* has *Mint genIDx* which has the value *Rome* and the *Uncertainty uncertain_value*. As explained above the generic ID represents a sentence when the context is taken into consideration. Therefore *genIDx* represents the statement that *TypeX* has *Rome* as a Mint and this statement has the *Uncertainty uncertain_value*.

Please be aware that the designation "*uncertain_value*" is rather badly chosen for this context because it refers to a value being uncertain while the Uncertainty ontology refers to a sentence having an uncertainty. It was chosen according to an existing naming convention. It would be more intuitive to simply call the uncertainty "*nmo:uncertain*". The reason why it is still called "*uncertain_value*" is that this designation is already in use in a number of systems and it would take a lot of work changing it. Pretty much everything that can be seen as an uncertainty can be chosen instead of the *nmo:uncertain_value* used here. Gradations of different uncertainties such as "unlikely", "possibly true", "likely", etc. and even simple percentages or decimals representing a probability are thinkable.

This model can be seen as following the information addition pattern introduced before, because the generic ID can be taken as a representation of the property connecting *TypeX* to the *Mint Rome* to facilitate that some information, to be precise the uncertainty, can be added to the relation.

On first sight it is clear that only the essence of the W3C's ontology is used to model the uncertainty with NMO. This use case is based on numismatic terms and especially on the fashion of how and what information was stored by nomisma.org. Up to now regarding uncertainties nomisma.org only describes the fact that some information is uncertain but does not go into greater detail. This is the reason why this use case does not go into greater detail either. Of course the rest of the Uncertainty Ontology could be used to express such greater detail as for example the uncertainty type or derivation.

Since the use case shows how the two ontologies can be connected by representing a *Sentence* with a generic ID, all the features of Uncertainty Ontology can easily be made use of.

2.2. International Council of Museums Conceptual Reference Model

In this section the CIDOC (French: Comité international pour la documentation) Conceptual Reference Model (CRM) [14] is introduced and explained with a couple of use cases. Furthermore problems and fixes stated by the Ontotext Research Space[8] are addressed.

CRM uses a naming convention that attaches the prefix EX to entities and PX to properties, where X is a number.

CIDOC CRM follows an event driven modelling approach.
In addition to some attribute connecting two entities (*E1 CRM Entities*), the event of the attribute assignment is captured in an own entity. This so called *"E13 Attribute Assignment"* has the properties *"P141 assigned attribute to"*, pointing to the entity to which an attribute is assigned and "*P140 assigned*" pointing to the entity which represents the value assigned by the attribute.

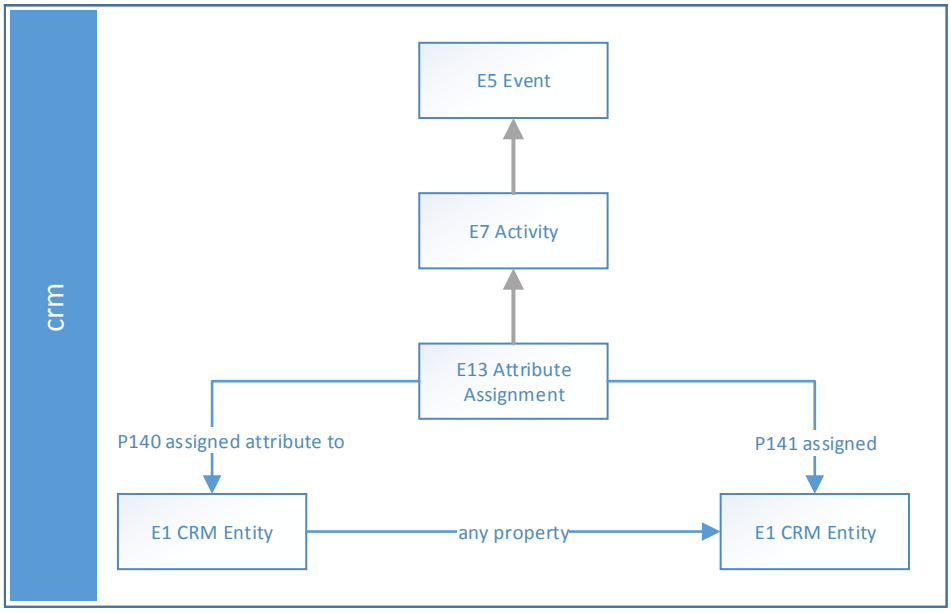

Figure 10: CRM Attribute Assignment

Figure 10 shows the CRMs attribute assignment. Note that an *Attribute Assignment* is an *Activity* which again is an *Event*.

Capturing the assignment of the attribute in an entity rather than only using a property can be mapped to the Information addition pattern. Because the formerly binary relation of the property connecting two entities is embodied by an entity and two properties, it can be extended by any number of participating entities (or individuals), i.e. Information.

[8] https://confluence.ontotext.com/display/ResearchSpace/Home

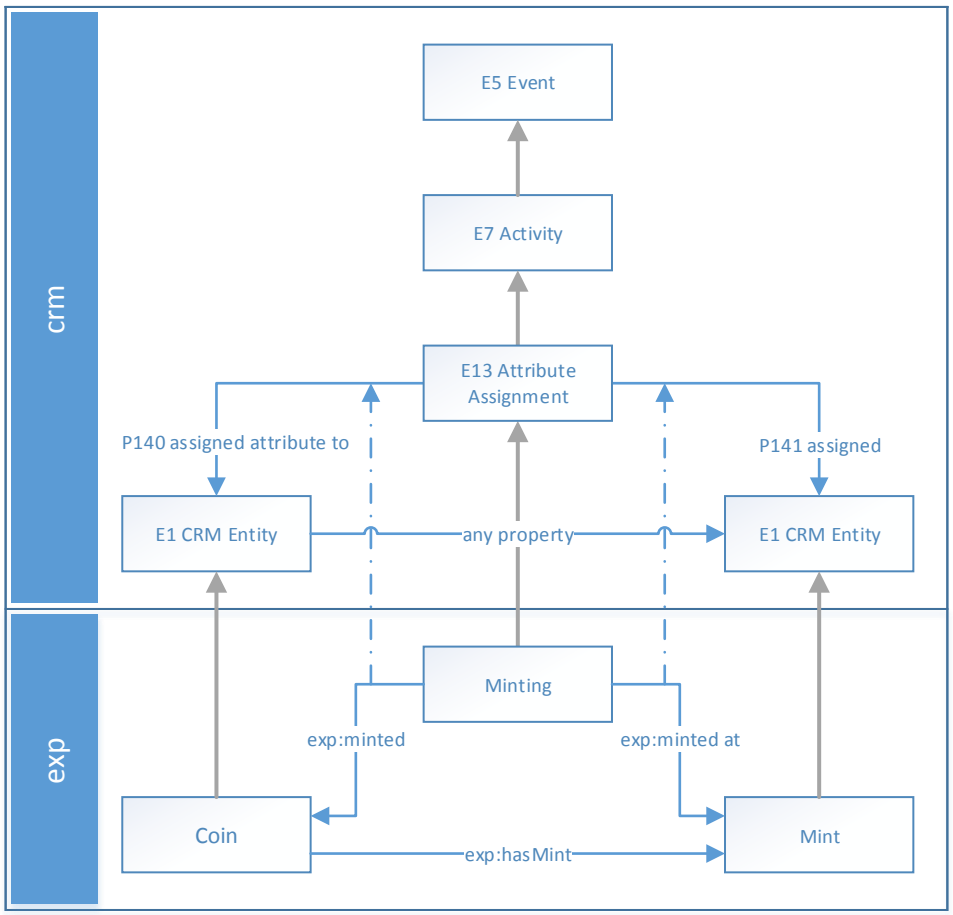

Figure 11: Attribute Assignment extended

Figure 11 shows how CRM and the example used up to now can be combined to model the fact that some *Coin* was minted at some *Mint*. Both *Mint* and *Coin* are subclasses of *E1 CRM Entity* and *Minting* is a subclass of *E13 Attribute Assignment*, while *minted* and *minted at* are sub-properties of *P140 assigned attribute to* respectively *P141 assigned*, following the CRMs recommendation of all extensions being sub-properties or sub-classes of CRM properties or classes.

The link between *Coin* and *Mint* is established in two ways. The *Attribute Assignment Minting* describes the *Activity* or *Event* of a *Coin* being minted and links it to the Mint where it was minted at. The property *hasMint* links the *Coin* directly to the *Mint*.

Here is a full use case on how "*CoinX* has *Rome* as a Mint" can be modelled using CRM.

Use Case: CRM hasMint

Description: CoinX has Rome as a Mint.

Used Entities: *exp:Coin, exp:Mint, exp:Minting, crm:E1 CRM Entity, crm:E13 Attribute Assignment, crm:E7 Activity, crm:E5 Event*

Used Properties: *exp:minted, exp:minted at, exp:hasMint, crm:P140 assigned attribute to, crm:P141 assigned, rdf:subProperyOf, rdf:subClassOf, rdf:type*

Used generic IDs:

Name	Description
genIDx	The *Minting* of *CoinX* at *Rome*

Depiction:

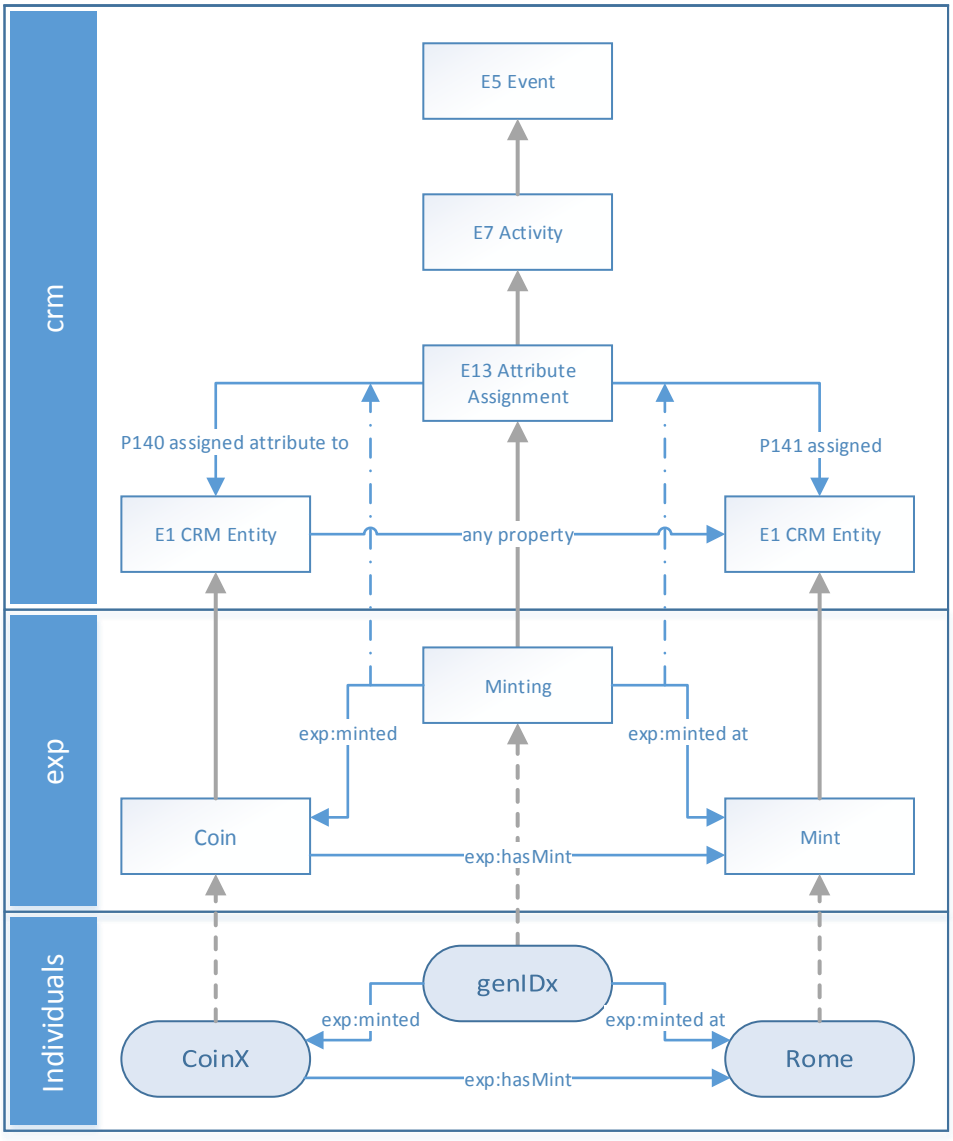

Figure 12: Use case CRM hasMint

As stated in [15] the attribute assignment is a longer path to the actual property, which in turn can be seen as a short-cut to the longer path (long-cut). The long-cut's benefit is a broader capability of expression because it can easily be extended by additional information. Information addition is done by appending properties to the attribute assignment. This is compliant with the W3C's information addition pattern, with a slight but not unimportant difference. That is, that the binary property is kept as short-cut, thus leaving a path to bypass the extended information. As said before, the information addition can be used to add any number of information to the relation in question.

In the following Use Case the CRM Property *P33 used specific technique* is used to model the fact that "*CoinX* has *Rome* as *Mint* and *struck* as *Manufacture*"(the use of *Manufacture* and *struck*[9] is also leaned on to the uses in NMO).

Use Case:	CRM Manufacture
Description:	*CoinX* has *Rome* as a Mint and *Struck* as *Manufacture*.
Note:	*exp:Minting* now is domain to the property *exp:hasManufacture*, which is used as sub-property of *crm:P33 used specific technique*. This is valid because *exp:Minting* is indirectly (via *crm:E13 Attribute Assignment*) a subclass of *crm:E7 Activity* and *P33* has *E7* as a domain.
Used Entities:	*exp*:*Manufacture*, *crm:E29 Design or Procedure, exp:Coin, exp:Mint, exp:Minting, crm:E1 CRM Entity, crm:E13 Attribute Assignment, crm:E7 Activity, crm:E5 Event*
Used Properties:	*exp:hasManufacture, crm:P33 used specific technique, exp:minted, exp:minted at, exp:hasMint, crm:P140 assigned attribute to, crm:P141 assigned, rdf:subProperyOf, rdf:subClassOf, rdf:type*

Used generic IDs:

Name	Description
genIDx	The *Minting* of *CoinX* being minted at *Rome* with the technique of manufacture *struck*

[9] http://nomisma.org/id/struck

Depiction:

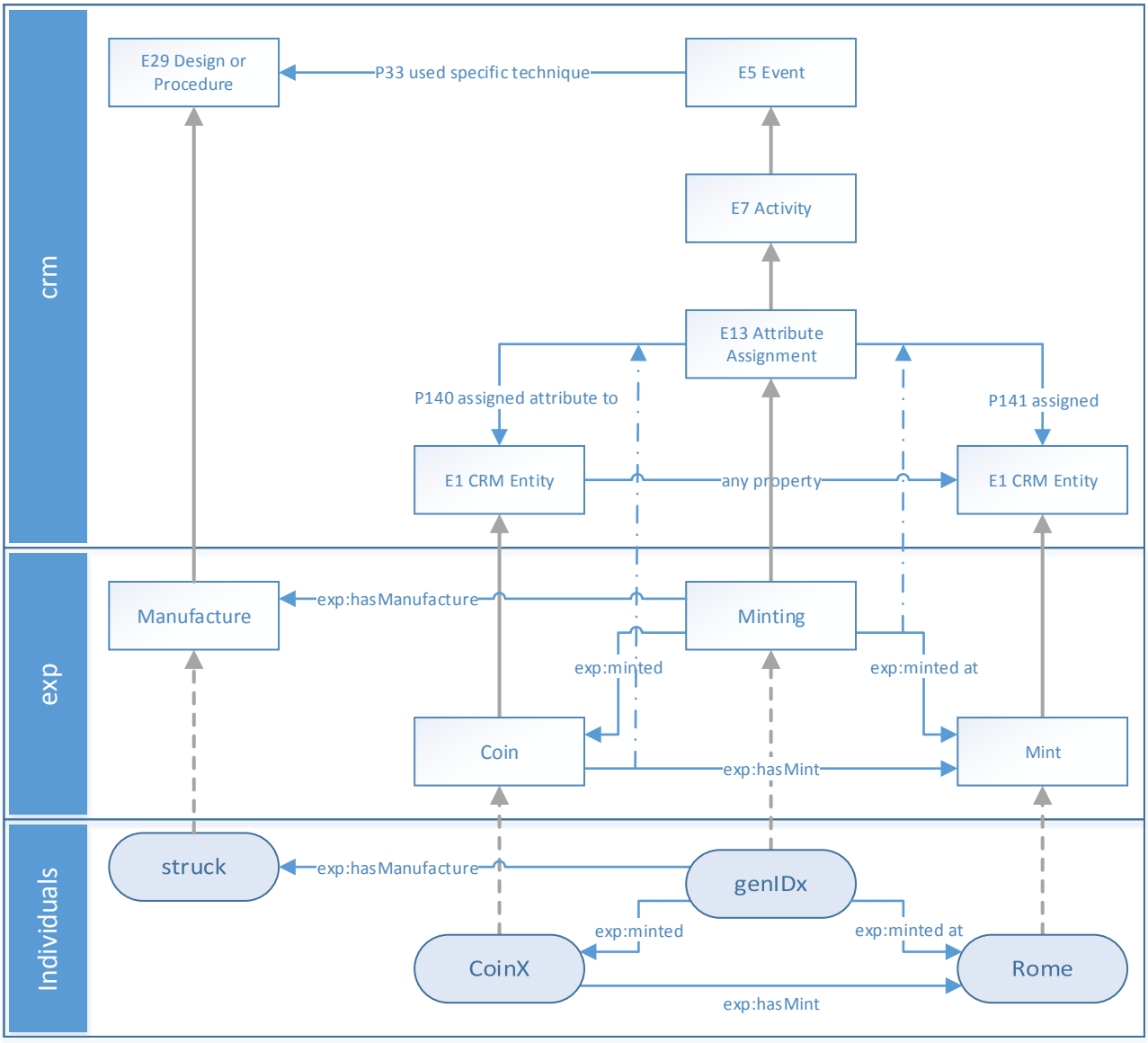

Figure 13: Use case CRM Manufacture

In [15] the authors point out a couple of problems regarding the CRM's attribute assignment. One of those problems is that the "*fully-articulated path*" [15] (called long-cut) does not always imply an instance of the corresponding short-cut property as it is claimed by the CRM. They also bring the very good example, that a long-cut may contain information about the status of the attribute assignment such as *"Formerly Thought To Be (i.e. not currently considered true)"* [15] while the short-cut lacks the capability of expressing the status and should be considered true. In this case the long-cut does not actually imply the short-cut.

Another problem is that the attribute assignment (long-cut) is not directly linked to the corresponding property (short-cut). When the same two individuals are connected with more than one short-cut, and there is a long-cut also linking these individuals, it is unclear which one of the short-cuts corresponds with the long-cut. Now when additional information is added via the long-cut, it is unclear to which one of the short-cuts this information belongs.

One can easily imagine an analogue problematic when adding some degree of uncertainty to the long-cut. This could result in the problem that a statement is falsely considered certain when following the short-cut, although it is meant to be uncertain and vice versa.

The Ontotext Research Space published a model of uncertainty [16] (namespace: bmo) using the CIDOC-CRM. In this publication the property *PX likelihood* links an uncertainty (of type *E55 Type*) to some *EX Association*. *EX Association* being a subclass of *Attribute Assignment* captures the event of associating something with something else in the same manner as the *Attribute Assignment* does. This is similar to the way the *Minting* is implemented in the above example and comes with the same problems. To cope with the latter of the above problems the long-cut is assigned to the short-cut via *PX property* connecting *the EX Association* to the corresponding property. An example of the problem and how *PX property* solves it can be found in Appendix B.

So what happens is nothing else but a long-cut via the *EX Association* is extended by some likelihood. The corresponding short-cut does not hold the information about the likelihood which is solely attached to the long-cut.

The following use case shows how CRM and the Ontotext Research Space's work can be combined to model the fact that "type series item *CoinX* has *Rome* as *Mint* but I am uncertain".

Use Case:	CRM & BMO uncertain
Description:	*CoinX* has *Rome* as a Mint but I am uncertain.
Note:	*Minting* is a sub-class of *bmo:EX Association* which is similar to (and sub-class of) *crm:E13 Attribute Assignment*. The property *bmo:PX likelihood* is used similar to *exp*:*hasManufacture* in the use case "CRM Manufacture" and extends the long-cut (*Attribute Association*) by some likelihood as additional information.
Used Entities:	*crm:E55 Type, bmo:EX Association, exp:Coin, exp:Mint, exp:Minting, crm:E1 CRM Entity, crm:E13 Attribute Assignment, crm:E7 Activity, crm:E5 Event*
Used Properties:	*bmo:PX likelihood, bmo: PX property, exp:minted, exp:minted at, exp:hasMint, crm:P140 assigned attribute to, crm:P141 assigned, rdf:subProperyOf, rdf:subClassOf, rdf:type*

Used generic IDs:

Name	Description
genIDx	The *Minting* of *CoinX* at *Rome* with the likelihood *uncertain*

Depiction:

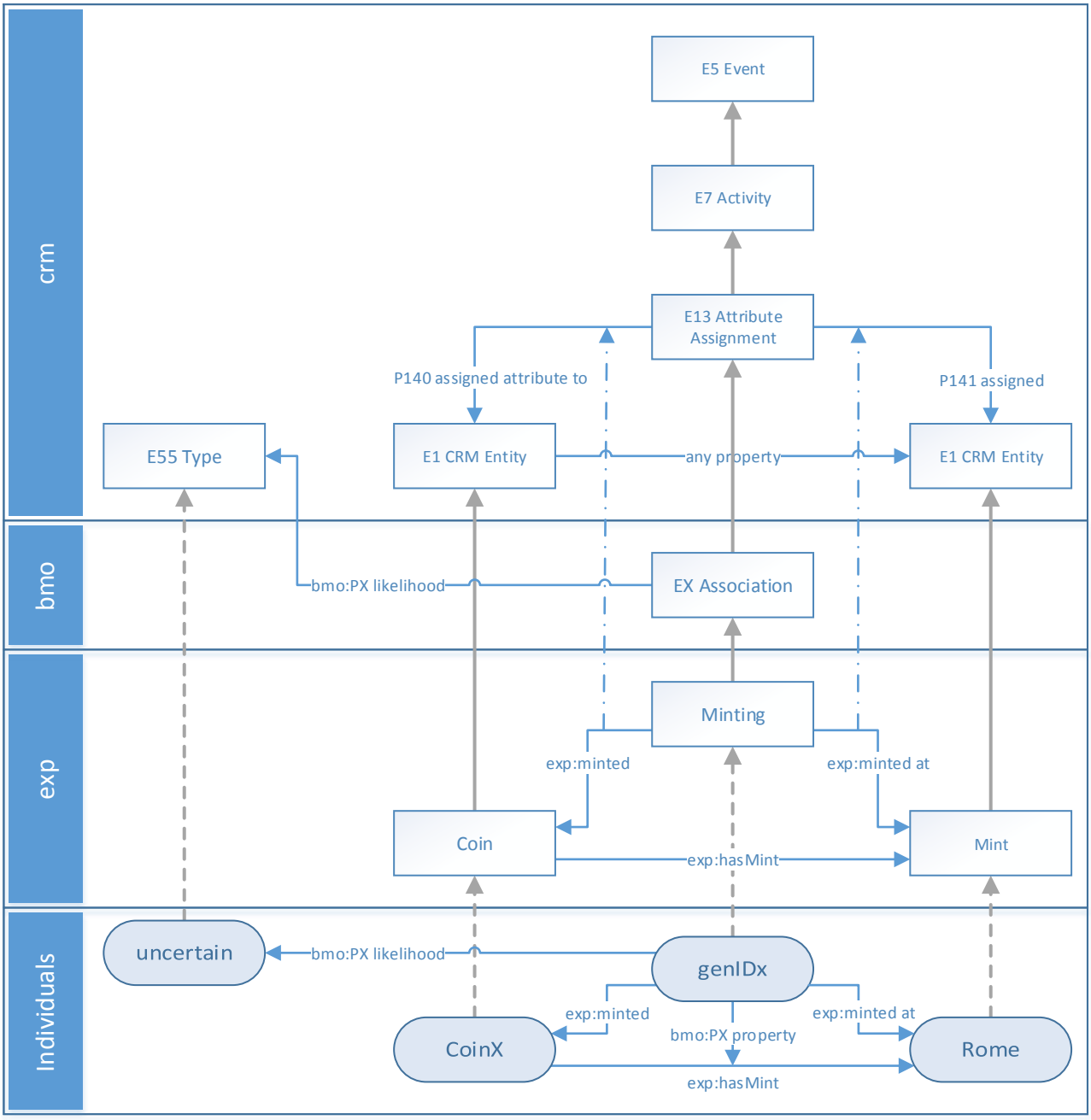

Figure 14: Use case CRM & BM uncertain

Simply put this model uses the information addition pattern (as it was previously shown on the example on the technique of manufacture) to extend the long-cut of a property by some likelihood. The likelihood attribute points out, to what likelihood the statement annotated by the attribute assertion respectively the attribute, can be considered true. For this example the likelihood *"uncertain"* means that the assrtion that *CoinX* was minted at *Rome* is uncertain to turn out true.

2.3. Uncertainty Framework

This section describes how *"a pattern-based Framework for Uncertainty Representation in ontologies"* (called uncertainty framework in this thesis) as it is presented in [17] can be used to model uncertainty. The key principle of this framework is to strictly separate ontologies handling uncertainties from those which do not hold any kind of uncertainty and especially in this point it differs from the previously discussed models. By separating the ontologies and building different layers of base and uncertainty ontologies, this framework works not only with the modeling but also with the architecture of ontologies to reach certain advantages.

The uncertainty part of the framework is founded on fuzzy theory. In fuzzy theory an individual can be in some class to some degree of certainty or in some relation with another Individual to some degree of certainty. These ideas are used to build an ontology containing uncertainties either on top of an already existing ontology or from scratch by following a simple pattern.

Here is how it is done:

$$Let \bowtie \in \{\leq, <, \geq, >\} \; and \; n \in [0,1]$$

2.3.1. Fuzzy instantiation

If the base ontology has information of some individual a being an instance of a class C and some degree of uncertainty shall be attached, in a fuzzy manner this is expressed as follows:

$$\langle a : C \bowtie n \rangle$$

Stating that individual a is in class C with certainty $\bowtie n$.

To hold the degree of uncertainty the framework adds a fuzzy-instantiation node and links it to the class C, the instance a and the corresponding type $\bowtie$ and value n of the uncertainty degree.

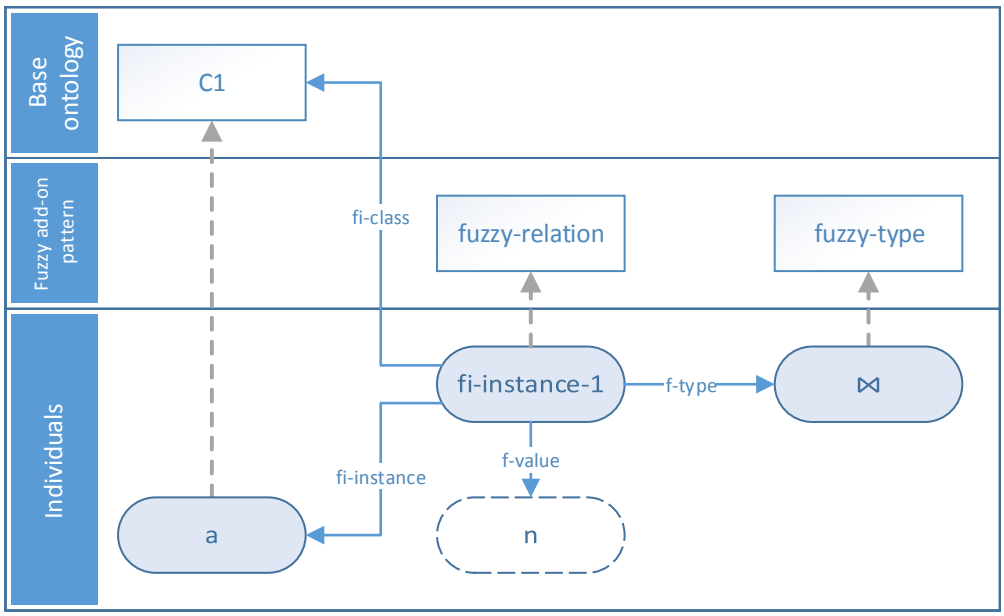

Figure 15: Fuzzy instantiation

Figure 15 shows how the fuzzy instantiation pattern adds the *fuzzy-instantiation* node *fi-instance-1*. It is similar to the usage of generic IDs in previous sections. The fuzzy instantiation has the property *fi-instance* pointing to the individual a that is instance of the class C pointed to by the property *fi-class*. Additionally it has the property *f-type* pointing to the *fuzzy-type* $\bowtie$ and the property *f-value* pointing to the real value n.

The combination of *f-type* and *f-value* shapes the degree of certainty. Note that there is no namespace specified for the fuzzy add-on pattern.

2.3.2. Fuzzy relations

If the base ontology has information of some individual a being in a relation R with another individual b and some degree of uncertainty shall be attached, in a fuzzy manner this is expressed as follows:

$$\langle (a, b) : R \bowtie n \rangle$$

Stating that individuals a and b are in relation with certainty $\bowtie n$.

Similar to the above pattern, the framework adds a *fuzzy-relation* node and links it to the individuals a and b, the relation *R* between them and the corresponding *type* ⋈ and *value* n of the uncertainty degree.

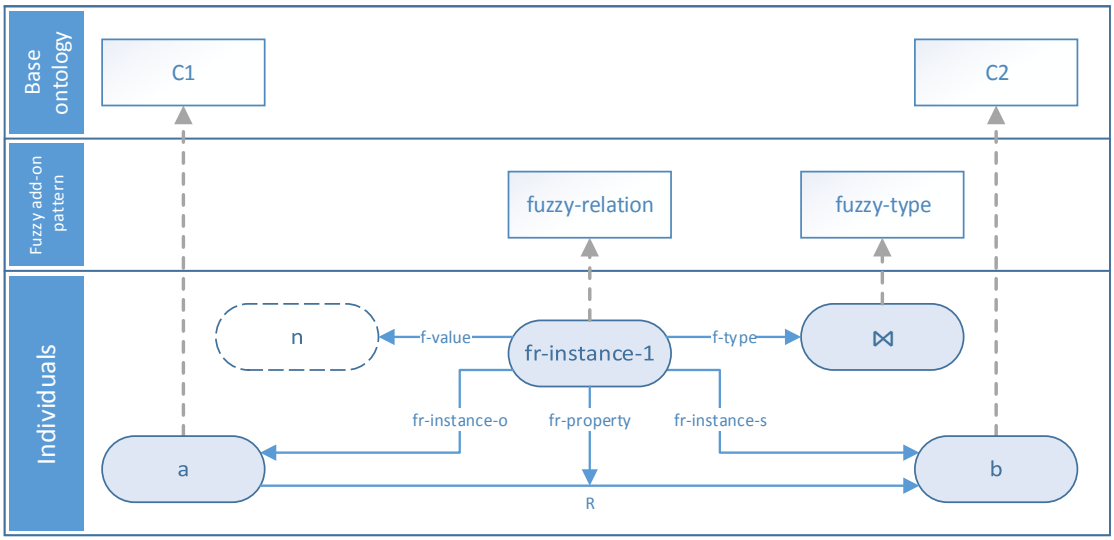

Figure 16: Fuzzy relation

Figure 16 shows how the fuzzy relation pattern adds the *fuzzy-relation fr-instance-1* with the properties *fr-instance-o*, *fr-property* and *fr-instance-s* pointing to the object a property (predicate) R and subject b of the assertion that a is in relation R with b. Like the fuzzy-instantiation (in the previous pattern) the fuzzy-relation also points to the type ⋈ and value n building the degree of uncertainty.

The core is to add an individual which in case of fuzzy theory is an instance of fuzzy-relation to embody the relation R between a and b. Information as the value and type of the probability can that be attached to this individual. This is as the use of UN and CRM have been before, compliant with the W3C's information addition pattern. Representations of other mathematic theories as for example probabilistic logic can be implemented in a similar way.

Note that in both cases there is a severe separation between the base ontology and the fuzzy add-on patterns. Thus these patterns can be used to either build the fuzzy ontology on top of an existing base ontology without having to adjust it, or to build up a base and fuzzy ontology simultaneously.

Although the framework is presented on the example of fuzzy sets as model of uncertainty it is not limited to it. Also the cut between the ontologies leads to the feature that different models of uncertainty can be used for one and the same base ontology without having to adjust or interfere with it.

For instance if there is a base ontology with no uncertainties and a fuzzy-ontology built on top according to the above patterns, the patterns can easily be used again to build another ontology, as for example one using probabilistic logic to represent uncertainties, on top of the base ontology. The result is one base

ontology with two ontologies containing uncertainties with different types of representation, subsuming in one ontology (consisting of three) holding different uncertainties.

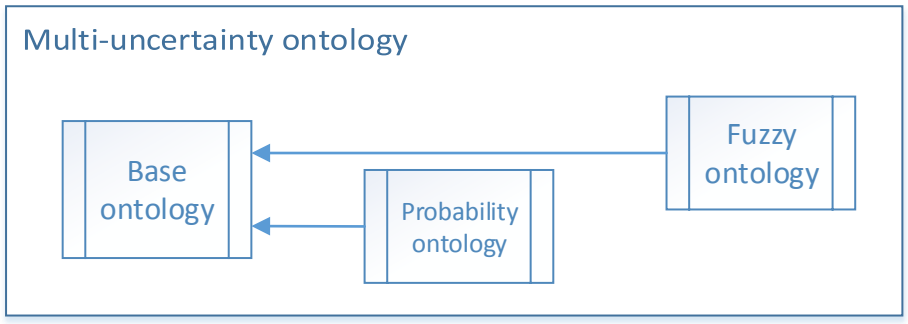

Figure 17: Multi-uncertainty ontology

This illustration (Figure 17) shows an ontology consisting of three others and holding different uncertainty models. This figure is adjusted to fit the design of this thesis. Please refer to [17] for the original.

Of course it is questionable whether there are reasons to hold different uncertainty models parallel to each other but it is possible.

Since the architecture of the resulting ontologies is of a strictly modular nature the advantages of the base ontology, as its compatibility with DL reasoning tools and other software that was conceived for it, survives the uncertainty addition.

Another advantage due to the modularity of the ontology architecture is that each ontology follows an appropriate pattern and embodies an isolated module, so that each ontology can have their own specialized reasoners (and other software) working on them. The architecture supports uncertainty reasoning and maintains the ability of crisp reasoning.

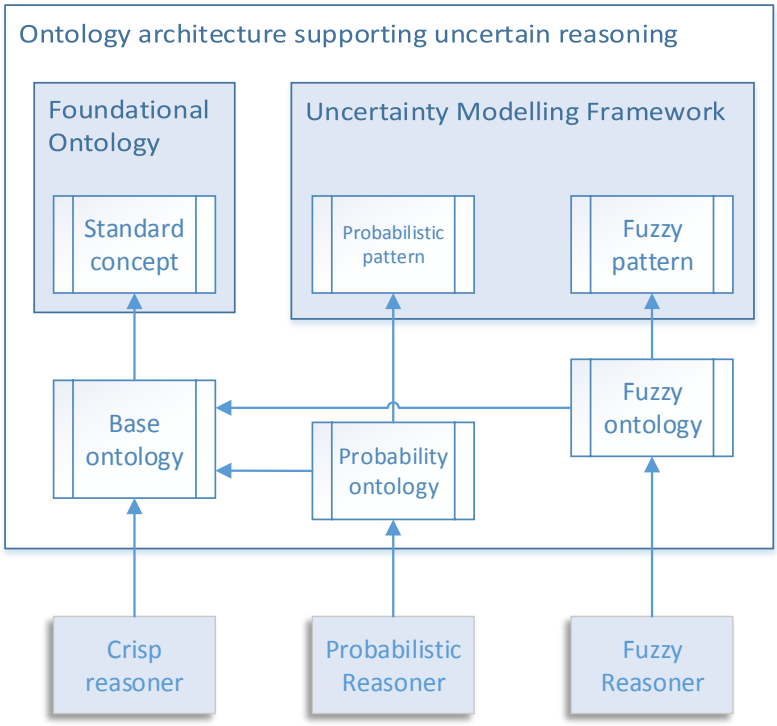

Figure 18: Modular architecture

Figure 18 illustrates the modular architecture and reasoning with multiple specialized reasoners. Again this figure is adjusted to fit the design of this thesis. Please refer to [17] for the original.

To give an example with some “real” data here is how the uncertainty framework can be used to model the assertion that: "*CoinX* has *Rome* as a Mint with degree of certainty ≥ 0.8."

Use Case:	Uncertainty Framework
Description:	*CoinX* has *Rome* as a Mint with degree of certainty ≥ 0.8.
Note:	There is no namespace specified for the fuzzy add-on pattern and *fr-instance-x* is used similar to generic IDs in previous use cases.
Used Entities:	*exp:Coin, exp:Mint, exp:Minting, fuzzy-relation, fuzzy-type*
Used Properties:	*f-value, f-type, fr-instance-o, fr-property, fr-instance-s, exp:hasMint, rdf:subProperyOf, rdf:subClassOf, rdf:type*

Depiction:

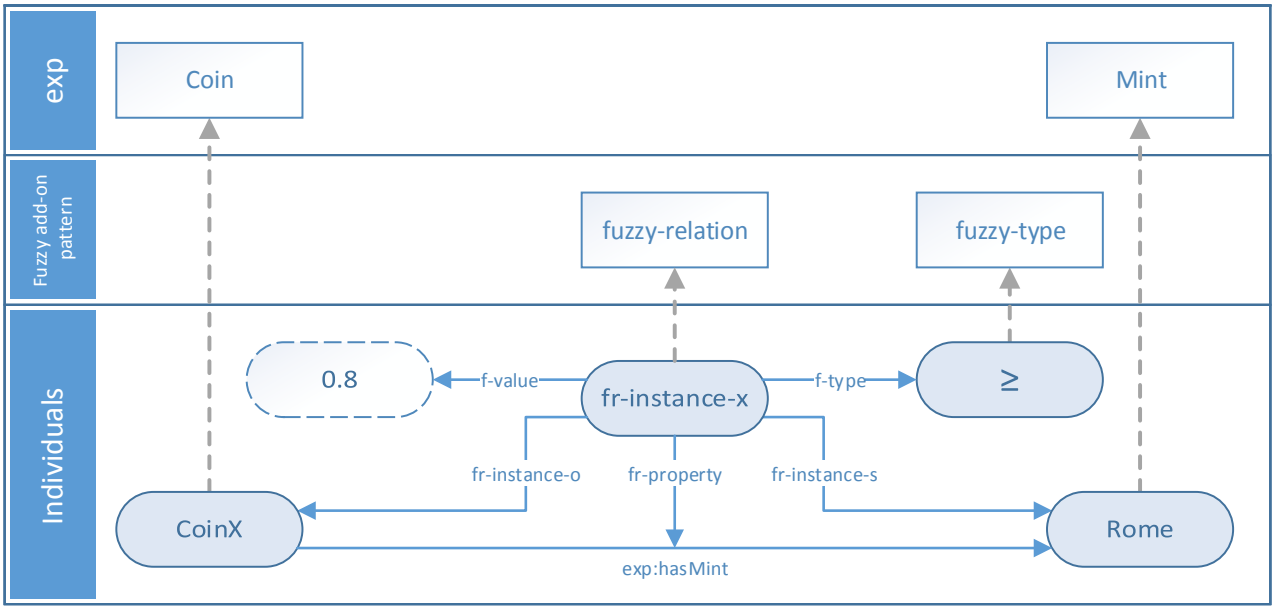

Figure 19: Uncertainty framework example

2.4. Comparing modeling approaches

When comparing the three models above, it soon is obvious that all of them follow the W3C's information addition pattern by instantiating a property with an individual and attaching some representation of uncertainty to it. UN uses sentence, CRM uses attribute assignment and the uncertainty framework uses, at least in the given example, fuzzy-relation to embody a relation between two entities.

Each of the three suggest a method of appending uncertainty to a relation and only UN suggests a way of representing the uncertainty itself. Since each method is of a very generic nature, an ontology designer making use of these methods is barely limited in his freedom of modeling the uncertainty itself. Even though UN gives a suggestion how *Derivation*, *Type*, *Model* and *Nature* of the uncertainty can be captured this approach still grants a great degree of freedom of modeling the uncertainty. The uncertainty framework gives an example showing how fuzzy theory could describe the uncertainty but clearly states that this is only an example of how the framework could be used and that it could be used for different representations of uncertainty as well.

There are, however, a couple of differences in the handling of properties. In both CRM and the uncertainty framework, the property of interest, in the above examples the property *hasMint*, is still used to link two individuals in sense of the property. Meaning that *hasMint* still links *CoinX* to *Rome* after the addition of uncertainty. This is contrary to the way the property is used in the combination of NMO and UN where *hasMint* now points not to the actual *Mint Rome* but to the generic ID representing the relation, so that there is no direct link between *CoinX* and *Rome*.

In short CRM and the uncertainty framework keep the property as a short-cut between the individuals (*CoinX* and *Rome*) and build up a long-cut on top of it via a new individual with an uncertainty attached. Whereas NMO + UN uses the property to build the long-cut via a new individual and has no short-cut.

When someone, or some program for that matter, processes the CRM model or the uncertainty framework, with the aim to find the *Mint* for *CoinX*, first *CoinX* is visited. Then since there is a relation to the mint *Rome* with the property (short-cut) *hasMint*, this path could be followed and the *Mint* found, without encountering the uncertainty at all, because it only attached to the long-cut. Since the link between *CoinX* and *Rome* is established it could falsely be interpreted as true or certain. This means that the processor must be advised to look for long paths and take these instead of the seemingly more attractive (in terms of finding the *Mint*) short-cut.

When the UN + NMO model is processed, following the property *hasMint* would lead from *CoinX* to a generic ID with the value *Rome* and some uncertainty attached. The fact of encountering a generic ID rather than a mint could be taken as signal that there is more to the relation between *CoinX* and *Rome*. This signal could be used to trigger a search for additional information starting from the generic ID and resulting in the finding of the uncertainty. This means that all three models need differently natured software to process the uncertainties.

At first sight the next difference lies in the direction of the properties. In CRM and the uncertainty framework, long-cut is established with two properties pointing from the relation embodying individual to the relations participants or in terms of RDF triples from the predicate to the subject and the object. In UN + NMO the subject points to the predicate which in turn points to the object. The latter makes a more intuitive impression. Anyhow the properties of CRM usually have complements pointing in the other direction and with the same point of expression. For example "*P140 assigned attribute to*" has the complement "*was attributed by*". The uncertainty framework leaves room for such bidirectional connection as well. Therefore this is not actually a difference of relevance.

What is a great difference regarding the properties, is the number of properties used.
To add uncertainties to an ontology, each approach has different properties and classes that need to be imported and instantiated. The starting point for the upcoming argumentation is an example ontology that only has the classes *Coin* and *Mint*, the property *hasMint*, intended (no domain and range restrictions) a *Coin* to a *Mint* and the individuals *CoinX* (an instance of *Coin*) and *Rome* (an instance of *Mint*), as in the common example in this section and illustrated in Figure 4. In order to add uncertainty to such an ontology using CRM it is necessary to import at least 4 classes (*E13 Attribute Assignment, E1 CRM Entity, E55 Type, EX Association*) and optional to create the new class *Minting*. Furthermore it is necessary to import at least 4 properties (*P140 assigned attribute to, P141 assigned, PX likelihood, PX property*) and optional to create the new properties *minted* and *minted at*. The class *Minting* and the properties *minted* and *minted at* are optional, because it is also possible to model the relation between *CoinX* and *Rome* on the more general level of *EX Association* and using the more general properties *P140 assigned attribute to* and *P141 assigned*, of which *minted* and *minted at* would be sub-properties. The decision whether to implement these extra properties and class is a task of balancing expressiveness with ontology growth. For this regard the best case in sense of keeping the ontology small, which is not adding the optional elements is considered. Given the necessary imports, for each uncertainty added, on the individual level the two individuals uncertain and *genIDx* (where x is a changing number to create distinct generic IDs) need to be added as well as the property assertions corresponding to the four mentioned properties. Note that when adding multiple uncertainties an individual like "*uncertain*" could of course be reused when the same uncertainty is needed for different attribute assertions.

To maintain fairness and comparability, it is necessary to point out that the example given with the framework uses fuzzy logic to model uncertainty and thus represents it with the two properties f-value and f-type and the individuals they point to. In CRM only the one property likelihood with the individual and in UN + NMO only the one property *hasUncertainty* with the individual *uncertain_value* is used. These are different representations of uncertainties and come with different amounts of expressiveness. In any case none of the above states that uncertainties should be expressed in theire manner so that any representation is thinkable (for example also UN provides a greater model to describe the uncertainty). This is why the one property and individual that the uncertainty frameworks example has more than CRM and UN + NMO examples must not be considered when comparing the growth of the ontologies.

Again starting from the relatively simple example ontology, adding uncertainties according to the uncertainty framework (precisely the given fuzzy add-on example) is quite similar to CRM. The two classes *fuzzy-relation* and *fuzzy-type* as well as the five (four, regarding the above statement of comparability) properties *f-value, f-type, fr-instance-o, fr-property, fr-instance-s,* need to be imported. Given the necessary imports, for each uncertainty added, on the individual level the three (two) individuals *0.8* (which is a data value), $\leq$ and *fr-instnace-x* need to be added as well as the property assertions corresponding to the five (four) mentioned properties. Note that when adding multiple uncertainties an individual like $\leq$ could be reused.

The number of properties and individuals is different for UN in the way it is used with NMO.
Again starting from the example ontology, adding uncertainties according to UN as it is used with NMO raises the necessity to import the two classes *Sentence* and *Uncertainty* as well as the property *hasUncertainty*. The property *rdf:value* can be considered usable without explicit import because it is part of the RDF which is the basis of any ontology. Given the necessary imports, for each uncertainty added, on the individual level the two individuals *uncertain* and *genIDx* need to be added. This is similar to CRM and the uncertainty framework (regarding the statement of comparability), but only two property assertions must be added, one for hasUncertainty and one for *rdf:value*. Furthermore the property *hasMint* must be redirected to point to *genIDx*. Note that this would not be all that simple, when the property had domain and range

restrictions. Again when adding multiple uncertainties an individual like "*uncertain_value*" could of course be reused when the same uncertainty is needed in different situations.

In sum it is remarkable that while CRM (and the uncertainty framework similarly) needs to import four new properties, and add four new property assertions, the approach using UN needs only to import one property and add two property assertions. This means that for each uncertainty added the graph of an ontology using CRM (or uncertainty framework) grows strikingly stronger than a similar ontology using the UN approach, which is a strong argument to the benefit of the UN approach as it is used with NMO.

This benefit comes at the cost that the property in question (here *hasMint*) needs to be redirected. This can turn out to be rather complicated due to possible domain and range restrictions. After redirection the property points not to a *Mint* but to a generic ID. Software used on such an ontology needs to work correctly when encountering such a generic ID as described previously and needs to be adjusted to the new situation too. These points make it hard to add uncertainties using the UN (+NMO) approach to an already existing ontology. This implies that the UN approach is especially useful when considered in the conception of a new ontology, when its features are regarded from start on, while it complicates things when trying to extend existing ontologies with uncertainties and has the benefit that uncertainties are not easily bypassed. Contrary to that, NMO and the uncertainty framework can easily be used to extend existing ontologies with uncertainties but with the negative point that uncertainties can be bypassed and thus uncertain data could be worked with, assuming it is certain and without knowing that it is uncertain.

Another distinctive feature lies in the used ontologies themselves. UN is an ontology recommended by the W3C and intends to supply a standardized representation of uncertainty that can be used throughout the Semantic Web. It has the power to combine representations of different mathematical theory based ontologies. Since the W3C is the instance to standardize Semantic Web features, it cannot be bad following these suggestions.

The CRM also intends to provide a standardized ontology but addresses primarily the features needed *"to facilitate the integration, mediation and interchange of heterogeneous cultural heritage information"* [14]. Also it needs to be clear that addition of uncertainty is not (yet) part of the CRM but was only proposed to be taken into the CRM by the Ontotext Research Space [16]. The problems pointed out in section 2.2. are real but solutions are being worked on. However UN has not been updated since March 2008, while CRM has been in development for more than a decade and still is vibrant. The latest official release of the CRM was published in January 2014.

Neither CRM nor UN provide the modularity that is thought of by the uncertainty framework.
It is thinkable to implement modularity with these ontologies anyway by splitting the ontologies in one with and one without uncertainty. This would result in an even greater resemblance with the uncertainty framework and one could say that they actually follow the terms of the framework.

The modularity splitting uncertainty loaded ontologies from base ontologies comes with all the benefits mentioned in 2.3 but also with the disadvantage that when only viewing the base ontology all facts will be taken certain although they might be weighted with some degree of belief in the other ontology.

Granted that separate reasoning is made possible by modularity as described in 2.3., it remains doubtful if separate reasoning makes sense.

None of these approaches proposes a way of inference, reasoning or joining of uncertainty knowledge bases.

3. Mathematical approaches

The previous section described approaches to modeling uncertainty in ontologies that made use of RDF and OWL as description languages for ontologies and thereby focused on modeling with existing tools. This section gives an overview on ways to handle uncertainties using different mathematical theories and some of them even attempt to extend existing tools. Such mathematical theories usually refer to uncertainty either as subjective or objective view on Bayesian probabilities. Bayesian probability interprets a probability as a degree of belief or (un)certainty. The subjective point of view says that a probability denotes some degree of personal (subjective) belief and the objective point of view says that a probability results from rationality and consistency of Bayesian statistics and is not distorted by any personal belief value [18]. The Bayesian interpretation of probability was already used in section 1.3 where fuzzy logic attached some degree of belief or probability i.e. certainty to relations and instantiations. In all of the following mathematical theories, probability is a measurement for (un) certainty.

3.1. BayesOWL

This section describes an approach to uncertainties in ontologies based on Bayesian Networks called BayesOWL, presented in [19].

The main Idea of BayesOWL is to extend OWL in order to support Bayesian Networks (BN) and with them annotation of probability. BNs are a way to represent conditional dependencies with a graph that is directional and acyclic. A node in a BN annotates a random variable. Each directional edge from some node b to another node a annotates that a is conditionally dependent from b, meaning that the value of the random variable annotated by b has an influence on the outcome of the variable annotated by b. Of course there can be more than one edge pointing (from different nodes) to a so that a's variable is conditionally dependent from all the variables represented by the parent node's of a. Furthermore each node has a probabilistic function that determines the probability of the nodes variable regarding all its conditional dependencies. When a node has no parent, the corresponding probability is not a conditional probability but simply the probability of the represented variable, namely the prior probability.

These probabilistic functions can be represented by conditional probability tables, which list the probability of all combinations of the variable's value and the values of all the parent variables.

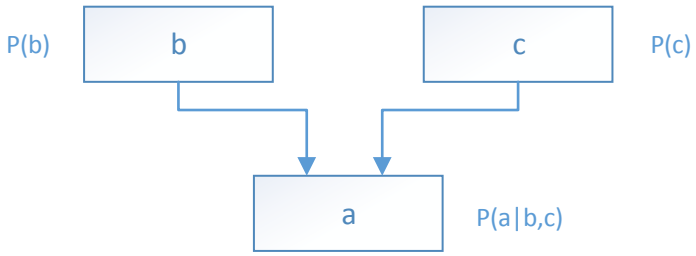

Figure 20: Bayesian Network

This graph (Figure 20) shows a Bayesian Network with a node for the random variable b (c) with no parents and the prior probability $P(b)$ ($P(c)$). The node for the random variable a has the parents b and c, and the conditional probability $P(a|b,c)$. An example with some data and conditional probability tables is given in Appendix C.

This form of annotating random variables, their conditional dependencies and probability distributions is attractive to combine with ontologies mainly because inference in BNs and learning of BNs is not only

possible, it can be done with efficient algorithms. Another reason is the viewable resemblance to graph representations of RDF triples.

BayesOWL provides a method of encoding these forms of probabilities and conditional dependencies in ontologies. At that it focuses on class membership statements and is limited to them while properties and assertion statements still lack a BN encoding.

3.1.1. Prior Probability

With the intention of encoding class memberships, a prior probability $P(c) = x$ can be interpreted as *"the prior probability that an arbitrary individual belongs to class C [..]"* [19]. The random binary variable c describes the event that some individual belongs to class C.

In order to model such a prior probability $P(c) = x$, the classes *Variable* and *PriorProb* are introduced.

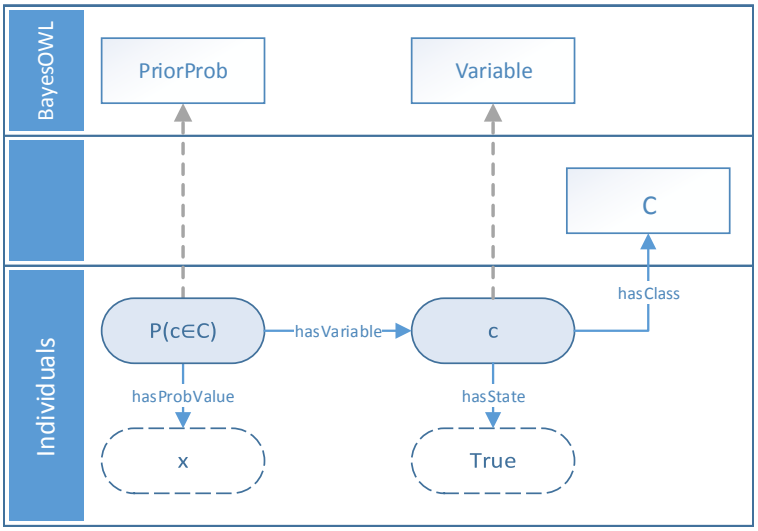

Figure 21: Prior probability

PriorProb $P(c)$ is the probability that the random variable c, that describes the event of an arbitrary individual (any individual in the ontology) is part of class C, has the state true. This probability has the probability value x.

As illustrated in Figure 21, an instance of *Variable* represents the random binary variable c that describes the event of an arbitrary individual belonging to the class C pointed to by the property *hasClass*. It also has the property *hasState* pointing to either *true* or *false*, meaning that an individual belongs to C (*true*) or does not belong to C (*false*) with probability $P(c)$ (respectively $P(\bar{c})$). An instance of *PriorProb* represents the actual prior probability and points to the *Variable* c with the property *hasVariable* and to some value x where $x \in [0,1]$ as common in probabilistic theory.

Note that no short-cuts for namespaces are used in the illustration of this section because everything is from the same source, namely BayesOWL which is not actually a namespace. BayesOWL proposes an extension to the OWL namespace. Therefore the namespace short-cut owl would be the best choice for the illustrations. However BayesOWL is not (yet) a part of owl namespace thus the namespace owl would not exactly be correct. To clarify that BayesOWL is (still) something else than OWL, no namespace is specified.

3.1.2. Conditional Probability

A conditional probability $P(c|p1, p2, p3) = x$ can be interpreted as "*the conditional probability that an individual of the intersection class of P1, P2, and P3 also belongs to class C[..]*" [19]. In other words the

probability that an individual belongs to class C, given it belongs to class $P1, P2$, and $P3$. In order to model such a conditional probability the class *CondProb* is introduced and the class *Variable* is reused like above.

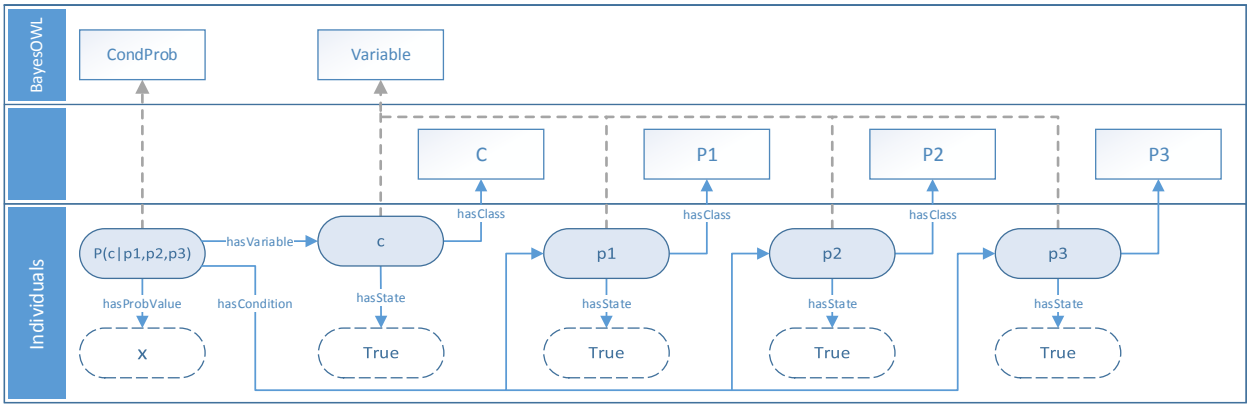

Figure 22: Conditional Probability

An instance of *CondProb* represents a conditional probability and points to the random binary variable c in question and some value x with the properties *hasVariable* and *hasProbValue* similar to the above described *PriorProb*. Furthermore it has at least one property *hasCondition*, pointing to the random binary variable *p1* (*p2, p3*) that in turn is linked to the corresponding state and class *P1* (*P2*, *P3*) and represents the given condition(s).

Also given in [19] is a set of rules on how to convert the OWL-model to a BN including rules for common OWL class axioms "*intersectionOf*", "*unionOf*", "*complementOf*", "*equivalent-Class*" and "*disjointWith*", as well as an algorithm to construct the CPTs for the resulting BN, to complete the representation of Class membership statements using BNs.

Please note that other than using the original namespace of owl there is no great difference between extending owl and making combined use of multiple ontologies, if of course the existing syntax and semantics are not touched by the extension. Since OWL is a W3C standard, the extension by the classes introduced in BayesOWL would declare them standard as well. Other than that there seems to be no reason to declare them part of OWL rather than part of an ontology that can be used to model uncertainties with Bayesian Networks, comparable to the way the W3C Uncertainty ontology does not extend OWL but provides a way to model uncertainties.

BayesOWL runs in a completely different direction from CRM and UN approaches, because it applies probabilities (or uncertainties) of any individual being part of certain classes to the ontology, rather than to specific individuals or assertions. Anyway here is an example for an ontology, where the probability of an individual being a coin (variable c) is 0.8, the probability of an individual being found together with roman coins (variable f) is 0.7 and the probability of an individual being a roman coin (variable r), given it is a coin and was found together with roman coins is 0.83. This is solely to give an example on how BayesOWL works with some real world terms and all given probabilities and values are made up. The model is illustrated in Figure 23: BayesOWL example. Note that only the positive probabilities are illustrated to retain simplicity. Please see Appendix C for the BN.

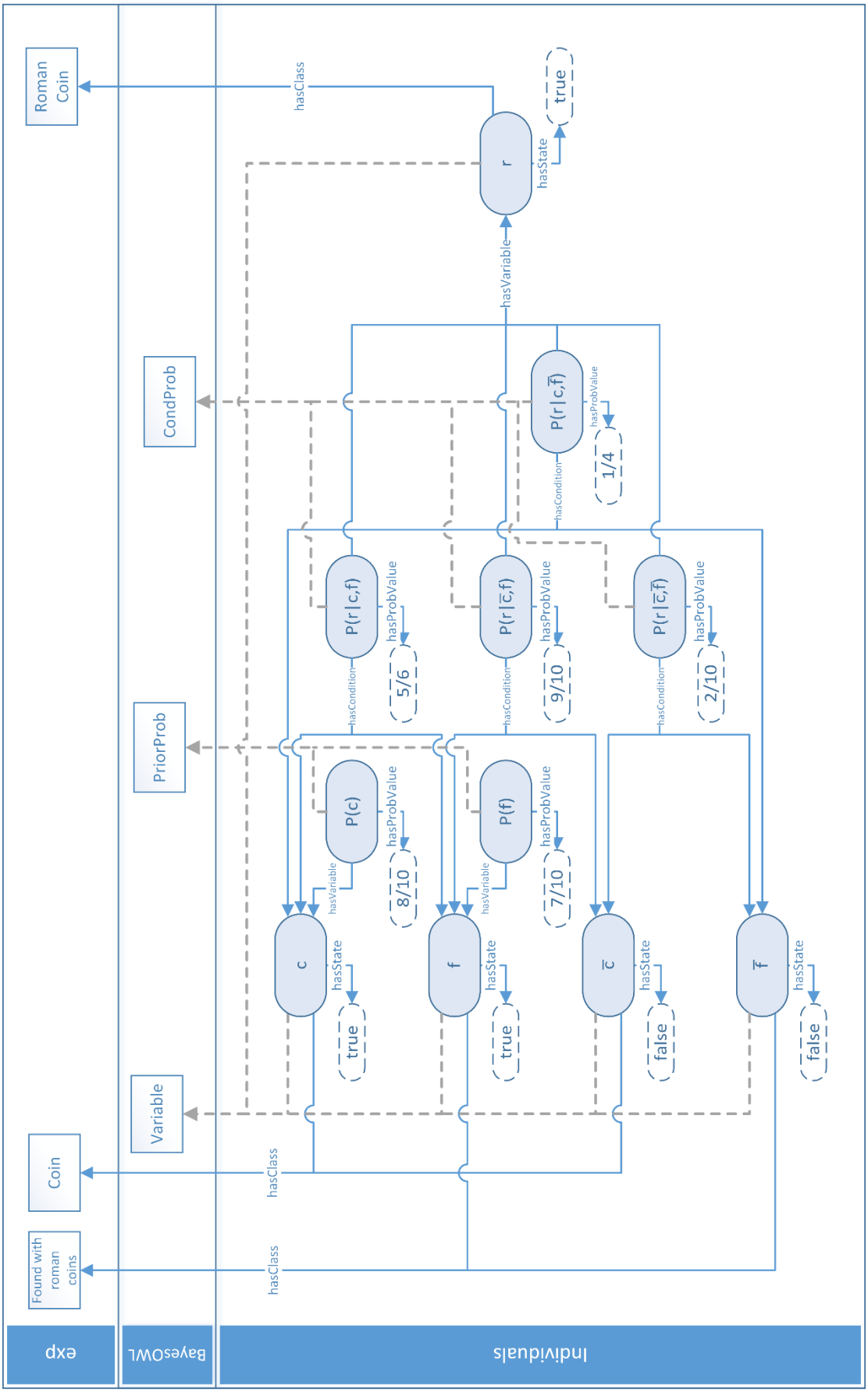

Figure 23: BayesOWL example

3.2. Fuzzy OWL 2

Fuzzy OWL 2 as it is presented in [20] is an approach to represent fuzzy ontologies using OWL 2.
As mentioned in section 1.3. the name OWL in this document always refers to the current version OWL 2. The reason why Fuzzy OWL 2 explicitly is based on OWL 2 is that in contrast to OWL 1, OWL 2 has *"enhanced annotation capabilities"* [10] supported by so called annotation properties. Fuzzy OWL 2 uses these annotation properties to represent and handle fuzzy information in ontologies.

"While in classical set theory elements either belong to a set or not, in fuzzy set theory elements can belong to a set to some degree." [20]

Formally this can be expressed by a membership function that returns for each element in a fuzzy set, the degree to which the element can be considered part of the fuzzy set. For classical set theory a membership function would return either true or false, while in fuzzy set theory it returns a value in the interval [0,1], where 0 means no membership and 1 means full(fact). This value is *"usually called the degree of truth of the statement."* [20]

In section 2.3. fuzzy instantiation and fuzzy relation are introduced. However Fuzzy OWL 2 follows a description logic ($\mathcal{SROIQ}(D)$) that goes further than that. The fuzzy description logic that is the basis of Fuzzy OWL2 uses two important elements. To represent these Fuzzy OWL 2 uses the annotation property *fuzzyLabel*. The annotation itself follows an XML syntax. The annotation is nested in the tags *<FuzzyOwl2>* and *</FuzzyOwl2>* respectively, with an attribute *fuzzyType* that specifies the tagged element.

3.2.1. Modifiers

Modifiers are functions used to modify the membership function of a set. A modifier is defined by either a linear or a triangular function:

$$linear(c), where\ a = c/(c+1), b = 1/(c+1)$$
$$triangular(a, b, c)$$

For example the modifier $very$ $[0,1] \rightarrow [0,1]$ can be defined as $linear(0.8)$.

In Fuzzy Owl 2, such a modifier is annotated to *"an OWL 2 datatype declaration of the type base double"* [20] and is denoted with the fuzzyOwl2 tag having "modifier" as *fuzzyType* and with a tag *Modifier* that in turn has the properties *type*, denoting the function type and the properties *a*, *b*, and *c* denoting the doubles as in the above functions. Note that if type is "linear" then the property c is sufficient. The modifier $very$ can be denoted like this:

```
<fuzzyOwl2 fuzzyType="modifier">
        <Modifier type="linear" c="0.8"/>
</fuzzyOwl2>
```

3.2.2. Fuzzy datatypes

Fuzzy datatypes are defined over an interval $[k_1, k_2] \subseteq \mathbb{Q}$, where $\mathbb{Q}$ is the set of rational numbers and by a function that returns a value in the interval $[0,1]$, the degree of a number belonging to this datatype.
A fuzzy datatype is defined by one of the following functions:

$$leftshoulder(k_1, k_2, a, b)$$
$$rightshoulder(k_1, k_2, a, b)$$
$$triangular(k_1, k_2, a, b, c)$$
$$trapezoidal(k_1, k_2, a, b, c, d)$$

A fuzzy datatype can also have a modifier applied, making it a fuzzy modified datatype. Figure 24 showa the functions mentioned above. The illustrations have been modified to fit the design of this thesis, the original illustrations can be found in [20].

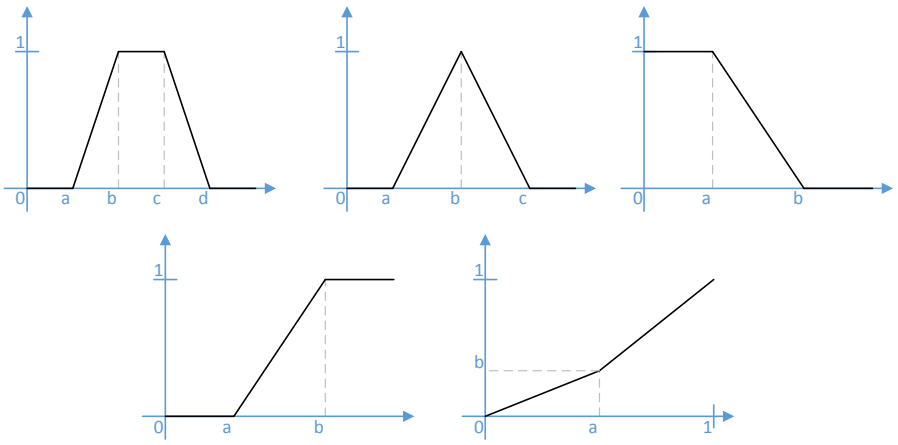

Figure 24: Trapezoidal function; Triangular function; Left-shoulder function; Right-shoulder function; Linear function

The fuzzy datatype $YoungAge$: $[0,200] \rightarrow [0,1]$ can be defined as $left(0,200,10,30)$

Any $x \in [0,10]$ is a *YoungAge* age, $x \in [30,200]$ is not a young age and $10 < x < 30$ is a young Age to some degree of truth (that is not true or false).

In Fuzzy Owl 2 such a fuzzy datatype is annotated to *"an OWL 2 datatype declaration of the type base of the fuzzy datatype (integer [..] or double [..]),[..]"* [20] and is denoted with the *fuzzyOwl2* tag having "*datatype*" as *fuzzyType* and with a tag *Datatype* that in turn has the properties *type*, denoting the function type and the properties *a, b, c* and *d* denoting the doubles as in the above functions. The boarders k1 and k2 can be set with *xsd:minInclusive* and *xsd:maxInclusive*, which are standard properties so set datatype boarders in OWL, and are taken as minimum and maximum value of attributes *a, b, c, d* respectively if not specified. The fuzzy datatype YoungAge can be annotated like this:

```
<fuzzy Owl2 fuzzyType="datatype">
        <Datatype type ="leftshoulder" a ="10" b="30"/>
</fuzzy Owl2>
```

The modifier $very$ can be applied to the fuzzy datatype building $very(YoungAge)$. While 20 is a Young Age to the degree 0.5, it is a very young age to the degree of approximately 6/9 as illustrated in Figure 25.

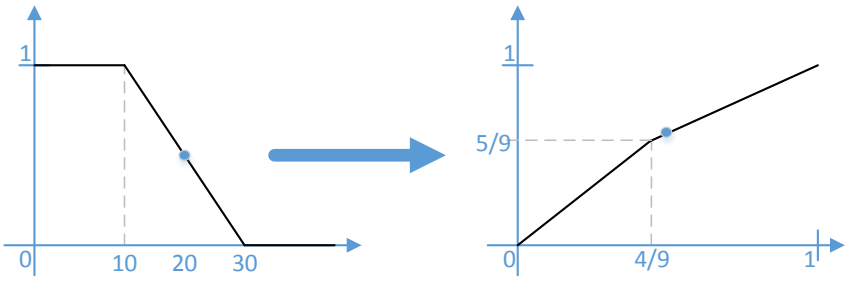

Figure 25: very(YoungAge)

Having the modifier and fuzzy datatype annotated as above, the fuzzy modified datatype can be annotated to the corresponding datatype declaration (of the fuzzy datatype) and is denoted with the *fuzzyOwl2* tag having "*datatype*" as *fuzzyType* and with a tag *Datatype* (as above) that in turn has the properties *type*, set to "*modified*", *modifier* set to the name of the modifier and *base*, set to the name of the modified datatype.

The modified fuzzy datatype $VeryYoungAge$ can be annotated like this:

```
<fuzzy Owl2 fuzzyType="datatype">
        <Datatype type ="modified" modifier="very" base="YoungAge" />
</fuzzy Owl2>
```

Other than modifiers and fuzzy datatypes the logic that Fuzzy OWL 2 is based on also supports fuzzy concept assertion and axioms similar to crisp logic. To stay within the means of this thesis only fuzzy axioms will be further described. Please refer to [20] for more information the other elements.

3.2.3. Fuzzy axioms

A fuzzy axiom can also be weighted with a degree of truth as for example the role axiom as it is described in section 2.3:

$$\langle (a, b): R \bowtie n \rangle$$

In Fuzzy OWL 2 such a role axiom is annotated to a role assertion and is denoted with the Fuzzy OWL 2 tag having "*axiom*" as *fuzzyType* and a tag *Degree* with the property *value* specifying the degree of truth (respectively certainty) n. Note that $\bowtie$ is not specified because it is considered as $\geq$ by default.

To stick with the example in section 2.3 the statement "*CoinX* has *Rome* as a Mint with degree of certainty $\geq$ 0.8." can be annotated like this:

```
<owl:Axiom>
        <fuzzyLabel>
                <fuzzyOwl2 fuzzy Type="axiom">
                        <Degree value="0,8"/>
                </fuzzyOwl2>
        </fuzzyLabel>
        <owl:annotatedSource rdf:resource="exp:CoinX"/>
        <owl:annotatedTarget rdf:resource="exp:CoinX"/>
        <owl:annotatedProperty rdf:resource="exp:hasMint"/>
</owl:Axiom>
```

At first sight these representations seem rather complicated to apply with common ontology development tools as for example Protégé[10]. Typing the xml annotation and finding the right spot in the ontology code every single time promises an uncomfortable mass of errors. Therefore the authors of [20] have published a Protégé plugin[11] that makes annotating Fuzzy OWL 2 an easy tasks that is done in just a few simple clicks.

Reasoning with Fuzzy OWL 2 has two points to take note of. On the one hand annotation properties are ignored by common OWL reasoners. Thus reasoners can disregard the fuzzy part and reason on a fuzzy ontology annotated with Fuzzy OWL 2 as if it were a crisp ontology. On the other hand the authors of [20] provide reasoning plugins for Protégé, which can parse fuzzy ontologies into fuzzy logics that can be reasoned by known fuzzy reasoners as for example fuzzyDL[12] and DeLorean[13].
The striking drawback of Fuzzy OWL 2 is that there is no full reasoning algorithm for the underlying fuzzy logic known yet. *"Consequently, the parsers only cover the fragments of fuzzy OWL 2 currently supported by these reasoners."* [20]

[10] http://protege.stanford.edu/
[11] http://gaia.isti.cnr.it/~straccia/software/FuzzyOWL/index.html
[12] http://nemis.isti.cnr.it/~straccia/software/fuzzyDL/fuzzyDL.html
[13] http://webdiis.unizar.es/~fbobillo/delorean

3.3. URDF

This section gives an insight into URDF as it is introduced in [12]. Its core intention is to extend the RDF language to support probabilistic logic.

It is interesting to grab the problem of handling uncertainties in ontologies at the RDF level because in this way *"statements about [..] what it means for an RDF triple or graph to have a probability[..]"* [12], inference and merging of RDF graphs regarding probabilities can be made. Since OWL is based on RDF in the Semantic Web Stack, all changes made to RDF need adjustment of OWL as well.

Probabilistic logic is a form of expansion of first-order logic such that it is capable of annotating probabilities by associating logical formulas with a probability distribution, so called probability constraints. *"[..]RDF can be seen as a subset of first-order logic"* [12], making it easy to compose it with probabilistic logic and result in URDF after some slight adjustments.

$$Pr\Big(\big(CoinX,\ \mathrm{h}asMint,\ Rome\big)\Big) \geq 0.8$$

This probability constraint on the (U)RDF triple "*CoinX* (subject) *hasMint* (predicate) *Rome* (object)" is a very simple example of how URDF works and states that the probability (or certainty) of *CoinX* having *Rome* as *Mint* is greater than or equal to 0.8.

In RDF and URDF likewise, there must be some limitations made to prevent inconsistency and allow complete deduction systems along with efficient inference procedures. For that reason in [12] another language is introduced called Subjective URDF, which is much like URDF but with the following prohibitions:

- no disjunction
- no negation
- only lower bound probabilities

No disjunction and no negation refer to the logical formulas in URDF during which only lower bound probabilities refers to the probabilities assigned to such a formula through a probability constraint.

As said before these restrictions and especially the first two, are similar to restrictions made on RDF.

RDF has the restriction *"that only 'positive' knowledge can be expressed"* [12]. In compliance with that is that only lower bound probabilities are allowed in Subjective URDF. In RDF the expression of positive knowledge alongside negative knowledge can lead to inconsistency and the same is the case in Subjective URDF where lower bound probabilities alongside upper bound probabilities can result in inconsistency especially when associated with the same triple. [12]

"When a knowledge base is extended, degrees of belief either stay the same or increase. This is similar to the effect in RDF that, when RDF graphs are merged, the set of entailed fact either stays the same or gets bigger." [12]

On the one hand Subjective URDF is less expressive than URDF but on the other hand it has a greater practical usefulness due to its advantages as mentioned above.

Also pointed out in [12], is that Subjective URDF follows the "Principle of Least Power" by Tim Berners-Lee. *"[..], it states that the requirements of a common data format must be decided upon as conservatively as possible"* [12] This is because a more conservative i.e. more simple format is easier to generate and analyze which gives it an attractive boost of usefulness.

This thesis means to focus on modelling ontologies rather than defining languages and logics, which is why no greater detailed definition of probabilistic logic and URDF will be given. For more information please refer to [12].

Together with Subjective URDF techniques of merging two URDF graphs and adjusting the resulting probabilities according to the laws of probabilistic logic and for merging RDF graphs with URDF graphs are delivered. The latter is especially interesting because it allows to engage a RDF graph with a degree of belief, by simply adding a probability constraint over the conjunction of all the triples in the graph, before combining it with another one. When person A has an ontology in URDF, person B has an ontology in RDF and A trusts B (believes in what B says) to a degree of 0.8 then A can merge his ontology with B's ontology after applying the constraint $Pr(\Phi_1 \wedge \Phi_2 \dots \wedge \Phi_n) \geq 0.8$ where $\phi_1 \dots \phi_n$ are all triples in B's ontology to consider his trust in B when absorbing his information.

Also an automated inference procedure is offered that can even be abstracted to represent conditional (in)dependencies with Bayesian Networks and perform Bayesian inference. The handling of Bayesian Networks with BayesOWL as it is presented in section 3.1, next to other probabilistic extensions for the Semantic Web are encapsulated by URDF.

"This demonstrates the fact that URDF can be used as a unifying formalism for many kinds of probabilistic reasoning in the Semantic Web, as opposed to focusing on a single kind of reasoning, such as Bayesian inference with Bayesian Networks, or probabilistic inference with independence assumptions." [12]

All these aspects make URDF a very promising approach to modelling uncertainties in the Semantic Web but it has not yet grown to full functionality because there have not yet been any outcomes about the representation of URDF other than an abstract syntax. One way of developing such a representation is to integrate it into the existing RDF syntaxes (XML, Turtle, etc.) to keep ontologies based on RDF running without major adjustments.

Another problem that remains is that independencies between triples might not always be possible to handle with Bayesian Networks and *"[..] making a global independence assumption violates dependence relations imposed by OWL integrity constraints, and therefore defeats the purpose of a great part of the OWL vocabulary"* [12]

3.4. Comparing mathematical approaches

When comparing BayesOWL, Fuzzy OWL 2 and URDF, it is obvious that each of them follow different mathematical theories to model uncertainty.

BayesOWL provides a way of modeling conditional probabilities with BNs. On the one hand the similarity of RDF graphs and BN, the power of representing conditional probabilities and Bayesian inference are benefits but on the other hand the model with its probabilities applied to any individual in the ontology, enforces a completely different perspective and expressiveness of ontologies than common ontology models. Fuzzy OWL 2 follows a fuzzy description logic where degrees of truth, fuzzy modifiers and datatypes are used and reinforce a great measure of expressiveness and similarity to natural language (for example with modifiers such as “very”). A weak point is that there is no full reasoning algorithm for the corresponding logic known yet. URDF is based on probabilistic logic that applies probability distributions to first-order-logic formulas and comes with inference and reasoning procedures based on probabilistic theory. Even though there are still hindrances with conditional probabilities, the inference procedure can be adjusted to represent Bayesian Networks, so that URDF’s dominates BayesOWL in terms of expressiveness.

A point that that Fuzzy OWL 2 and URDF have in common is that uncertainty can not only be applied to assertions within an ontology but also to the ontology itself, supporting degrees of trust or belief for different ontologies. This feature puts Fuzzy OWL 2 and URDF in advantage because it is not supported by BayesOWL.

To handle uncertainties, each of the three models approaches the Semantic Web Stack from a different angle. BayesOWL suggests to extend OWL by a couple of classes and properties that aid the representation of BNs. Other than adding to the OWL namespace this does not differ from simply using OWL. Nonetheless as stated above, unlike common ontologies, ontologies representing BNs apply probabilities to any individual in the ontology. That is why BN ontologies need to be interpreted in another way than common ontologies, thus common ontologies cannot be easily extended and uncertainties cannot be added to existing ontologies using BayesOWL. Instead uncertainty must be considered from the very start of developing the ontologies. Fuzzy OWL 2 uses annotation properties to describe uncertainties. This approach also uses OWL to apply uncertainties but the fact that annotation properties are used with an XML notation is different from common OWL usage. Annotation properties can be ignored by reasoners and other software so that an ontology that is loaded with uncertainties annotated with Fuzzy OWL 2, can still be used as crisp ontology, bypassing the uncertainties. It is similar to how uncertainties can be bypassed in CRM and the uncertainty framework described in section 2 and comes with the same problem, that uncertain data could falsely be assumed certain. This is not the case for BayesOWL and URDF. URDF attempts to extend RDF which is the basis of OWL and everything above it. Such an extension would generate the necessity of a complete roll up of all the layers of the Semantic Web Stack that build up on RDF. Although the benefits and changes that go together with URDF are promising, such a major impact on everything that has been developed up to now should be the last resort when adding uncertainties to the Semantic Web.

Finally the different states of development need to be taken into consideration. BayesOWL in its current state is limited to class membership assertions and everything else (role assertions etc.) is pending. URDF does not yet have a suitable representation, which is supposedly the smallest problem because adjustment of everything that is based on RDF must be performed to. The greatest maturity is shown by Fuzzy OWL 2. It is capable of adding uncertainties to just about anything in an existing ontology and even software (the Protégé Plugin) for easy annotation with Fuzzy OWL 2 already exist. Despite the fact that there is not yet a full inference procedure for the underlying fuzzy logics, there are translation tools to make Fuzzy OWL 2 (in not all of its expressiveness) work with existing fuzzy reasoners.

4. Conclusion

The different approaches of modelling uncertainties using ontologies that are presented and compared in section 2 and 3, all have their specific features that need to be weighted up in order to decide which approach suits a certain situation best. In conclusion it is to say that, when fuzzy logic is applicable as mathematical representation of uncertainty, then Fuzzy OWL 2 is (Section 2.3) a good choice to annotate uncertainties. The major benefits are automated annotation with Protégé plugins, translation tools for fuzzy reasoners and that simple annotation properties are used, keeping the ontology small. When other theories or representations of uncertainty are asked for, the UN (as used in section 1.1) is a good choice, because it keeps graph growth limited, prevents bypassing the uncertainty and does not enforce any specific mathematical theory. But both of them as all the other represented models should not be taken too easily and have their own problems to them. One of the main questions to ask when choosing a suitable model is if uncertainty should be added to an existing ontology or an new ontology should be designed. When an existing ontology is to be extended it is easiest to make the uncertainty bypassable as CRM, the uncertainty framework and Fuzzy OWL 2 do it. The risk here is that information can falsely be interpreted as certain. To prevent such bypasses the uncertainties must be considered from the start of the ontologies design as with UN. Each approach can be chosen and adjusted according to the specific situation. URDF as an attempt to extend the RDF layer of the Semantic Web Stack is, though promising, far from complete and such an impact to RDF layer, causing the necessity of adjusting the above layers, should be considered a last resort for modelling uncertainties using ontologies.

Appendix A

Table of namespaces

Namespace	Shortcut
http://nomisma.org/id/	nm
http://nomisma.org/ontology#	nmo
http://www.w3.org/2005/Incubator/urw3/XGR-urw3-20080331/Uncertainty.owl	un
http://erlangen-crm.org/current/	crm
<http://collection.britishmuseum.org/id/ontology/>	bmo
http://www.w3.org/1999/02/22-rdf-syntax-ns#	rdf
http://www.w3.org/2000/01/rdf-schema#	rdfs

Table 1: Table of namespaces

Appendix B

PX property

The problem solved by PX property is that when additional information is added via the long-cut, it is unclear to which one of the short-cuts this information belongs.

If there are more than one short-cuts connecting the same two individuals and there is at least one long-cut with addition information, it is unclear to which one of the short-cuts this information belongs.

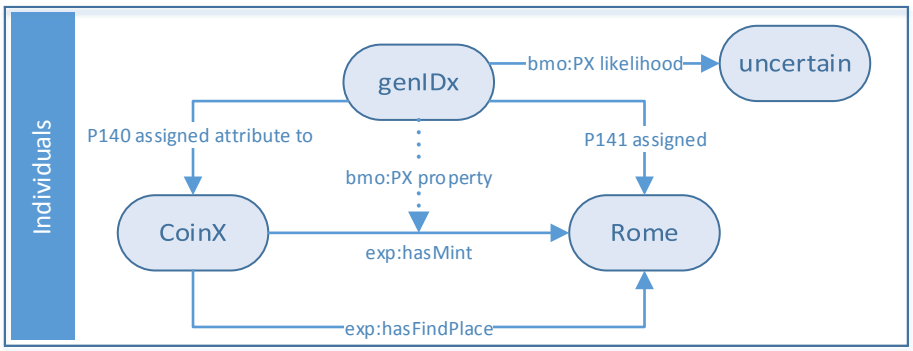

Figure 26: PX property example 1

Figure 26 shows an example of this problem. Note that *PX property* is dotted because when it is added the problem is solved. *CoinX* has *Rome* as a mint connected via the property (short-cut) *hasMint* and *CoinX* has *Rome* as a finding place connected via the property *hasFindingPlace*. Observe that contrary to other examples, in this example the individual *Rome* does not only represent Rome as a Mint but also Rome as a place. There is also a long-cut carrying the additional information "*uncertain*". However without *PX property* it is not clear to which of the short-cuts this long-cut and this information belongs. The uncertainty could belong to either or even both of the short-cuts. Clarification is provided by *PX property* linking the long-cut to the corresponding short-cut. Now it is clear that the uncertainty belongs to only one of the short-cuts, namely *hasMint*.

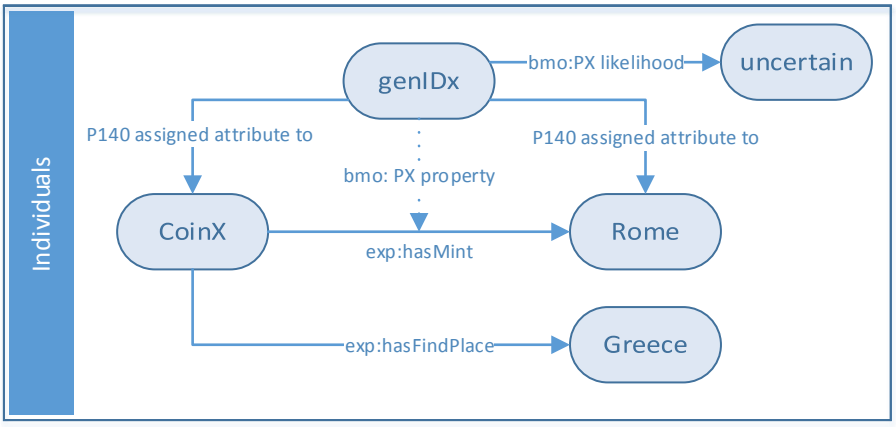

Figure 27: PX property example 2

However this problem does not occur as long as there is not more than one short-cut connecting the same two individuals. In Figure 27 it is clear that the long-cut corresponds to the short-cut *hasMint* even though there is no link between the two. This is because *hasMint* links the individuals *CoinX* and *Rome* and the long-cut also links these individuals *CoinX* and *Rome,* while the other short-cut *hasFindingPlace* links *CoinX* to a different individual *Greece*. It is obvious that the long-cut cannot correspond to *hasFindingPlace* simply because they link different individuals.

Appendix C

Bayesian Network example

This example considers the random variable c, denoting the event that an arbitrary individual is a coin, the variable f, denoting that an individual was found with roman coins and the variable r, denoting that an individual is a roman coin. Each of these variables is a binary random variable and comes out to either true or false.

The example data are 100 elements of which 80 coins and 40 are found with roman coins and so on as given in
Table 2.

	$f = true$		$f = false$		$\sum$
	$r = true$	$r = false$	$r = true$	$r = false$	
$c = true$	50	10	5	15	80
$c = false$	9	1	2	8	20
$\sum$	59	11	7	23	100

Table 2: BN example data

To represent these circumstances in a Bayesian Network with three nodes, where each node represents one of the random variables, the prior and conditional probabilities can be pooled in the following conditional probability tables (CPT) (
Table 3,

Table 4,
Table 5). The concluding Bayesian Network is illustrated in Figure 28.

c	
$true$	$false$
8/10	2/10

Table 3: CPT for c

f	
$true$	$false$
7/10	3/10

Table 4: CPT for f

		r	
c	f	$true$	$false$
$false$	$false$	2/10	8/10
$false$	$true$	9/10	1/10
$true$	$false$	1/4	3/4
$true$	$true$	5/6	1/6

Table 5: CPT for r.

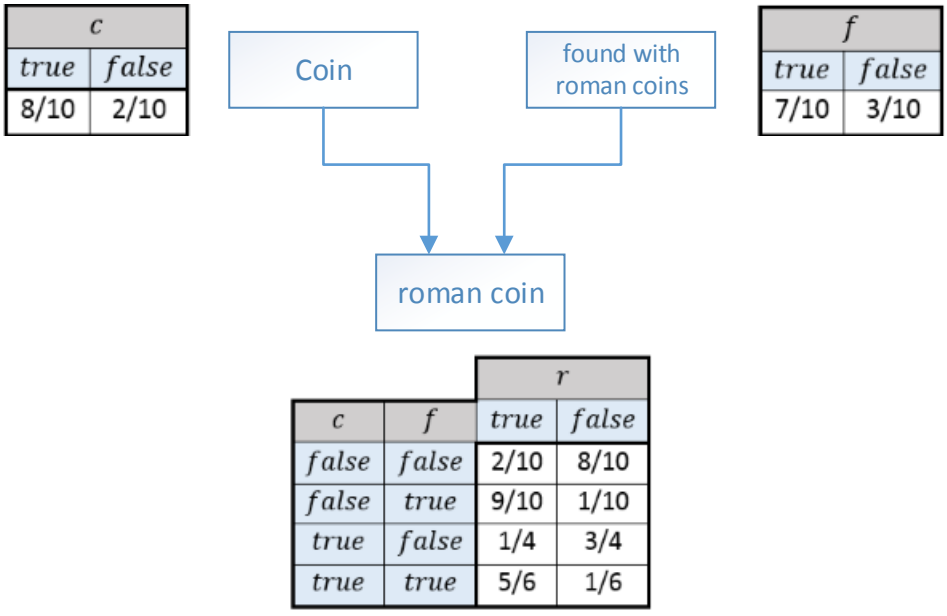

Figure 28: Bayesian Network example

References

[1] Dieter Fensel, et al., Spinning the Semantic Web: bringing the World Wide Web to its full potential., The MIT Press, 2003.

[2] World Wide Web Consortium, "Semantic Web - W3C," World Wide Web Consortium, [Online]. Available: http://www.w3.org/standards/semanticweb/. [Accessed 18 8 2014].

[3] Wikipedia, the free encyclopedia, "Semantic Web Stack - Wikipedia, the free encyclopedia," Wikipedia, 24 10 2013. [Online]. Available: http://en.wikipedia.org/wiki/Semantic_Web_Stack. [Accessed 20 8 2014].

[4] Wikipedia, the free encyclopedia, "File:Semantic-web-stack.png - Wikipedia, the free encyclopedia," 18 5 2008. [Online]. Available: http://upload.wikimedia.org/wikipedia/en/3/37/Semantic-web-stack.png. [Accessed 18 8 2014].

[5] W3C RDF Working Group, "RDF - Semantic Web Standards," World Wide Web Consortium, 25 2 2014. [Online]. Available: http://www.w3.org/RDF/. [Accessed 18 8 2014].

[6] "RDF Schema 1.1," World Wide Web Consortium, 25 2 2014. [Online]. Available: http://www.w3.org/TR/rdf-schema/. [Accessed 8 18 2014].

[7] "RDF 1.1 XML Syntax," World Wide Web Consortium, 25 2 2014. [Online]. Available: http://www.w3.org/TR/rdf-syntax-grammar/. [Accessed 18 8 2014].

[8] "RDF 1.1 Concepts and Abstract Syntax," World Wide Web Consortium, 25 2 2014. [Online]. Available: http://www.w3.org/TR/rdf11-concepts/. [Accessed 18 8 2014].

[9] W3C OWL Working Group, "OWL - Semantic Web Standards," World Wide Web Consortium, 11 12 2012. [Online]. Available: http://www.w3.org/2001/sw/wiki/OWL. [Accessed 18 8 2014].

[10] W3C OWL Working Group, "OWL 2 Web Ontology Language Document Overview (Second Edition)," World Wide Web Consortium, 11 12 2012. [Online]. Available: http://www.w3.org/TR/owl2-overview/. [Accessed 18 8 2014].

[11] "Uncertainty Reasoning for the World Wide Web," World Wide Web Consortium, 31 3 2008. [Online]. Available: http://www.w3.org/2005/Incubator/urw3/XGR-urw3/. [Accessed 18 8 2014].

[12] T. Rienstra, "Dealing with Uncertainty in the Semantic Web," Department of Computerscience, University of Twente, 2008.

[13] "Defining N-ary Relations on the Semantic Web," World Wide Web Consortium, 12 4 2006. [Online]. Available: http://www.w3.org/TR/swbp-n-aryRelations/. [Accessed 18 8 2014].

[14] "Definition of the CIDOC Conceptual Reference Model (v.5.1.2)," ICOM/CIDOC CRM Special Interest Group, 2013.

[15] V. Alexiev, "Types and Annotations for CIDOC CRM Properties," Ontotext Corp, Sofia, Bulgaria, 2011.

[16] Vladimir Alexiev, Joshua Mahmud, "BM Association Mapping v2," Ontotext Research Space, 18 3 2014. [Online]. Available: https://confluence.ontotext.com/display/ResearchSpace/BM+Association+Mapping+v2. [Accessed 18 8 2014].

[17] M. Vacura, V. Svátek and P. Smrž, "A Pattern-based Framework for Representation," Springer, Berlin, 2008.

[18] Wikipedia, the free encyclopedia, "Bayesian probability - Wikipedia, the free encyclopedia," Wikipedia, 28 06 2014. [Online]. Available: http://en.wikipedia.org/wiki/Subjective_probability. [Accessed 18 8 2014].

[19] Z. Ding, Y. Peng and R. Pan, "A Bayesian Approach to Uncertainty Modelling in OWL Ontology.," Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 2006.

[20] F. Bobillo and U. Straccia, "Fuzzy Ontology Representation using OWL 2," arXiv:1009.3391v3, 2010.