8 Using OWL-Fast in the Wild

We have seen a number of examples of the use of OWL-Fast modeling for merging information from multiple sources in a dynamic and flexible way. In this chapter, we will describe two extended uses of the OWL-Fast constructs. Both of these applications of OWL-Fast have attracted considerable user communities in their respective fields. Both of them also make essential use of the constructs in OWL-Fast, though often in quite different ways. These are real modeling applications built by groups who originally had no technology commitment to RDFS or OWL (though both were conceived as RDF applications).

In both cases, the projects are about setting up an infrastructure for a particular web community. The use of OWL-Fast appears in the models that describe data in these communities, rather than in the every day use in these communities. In this book, we are describing how modeling work in RDFS and OWL, so we focus on the community infrastructure of these projects.

The first application is called SKOS, the Simple Knowledge Organization System, and proposes a Semantic Web approach to expressing concept organization systems such as thesauri, taxonomies, and controlled vocabularies in RDF.

The second application is called FOAF, for "Friend of a Friend". FOAF is a project dedicated to creating and using machine-readable homepages that describe people, the links between them and the things they create and do. It is based on RDF, but originally made no commitment to RDFS or OWL.

Both of these projects were originally based on RDF because of the inherently distributed and web-like nature of the project requirements. As the projects evolved, they found a need to be able to describe the relationships between various resources in a formal way; this led both of them to RDFS and then on to OWL-Fast.

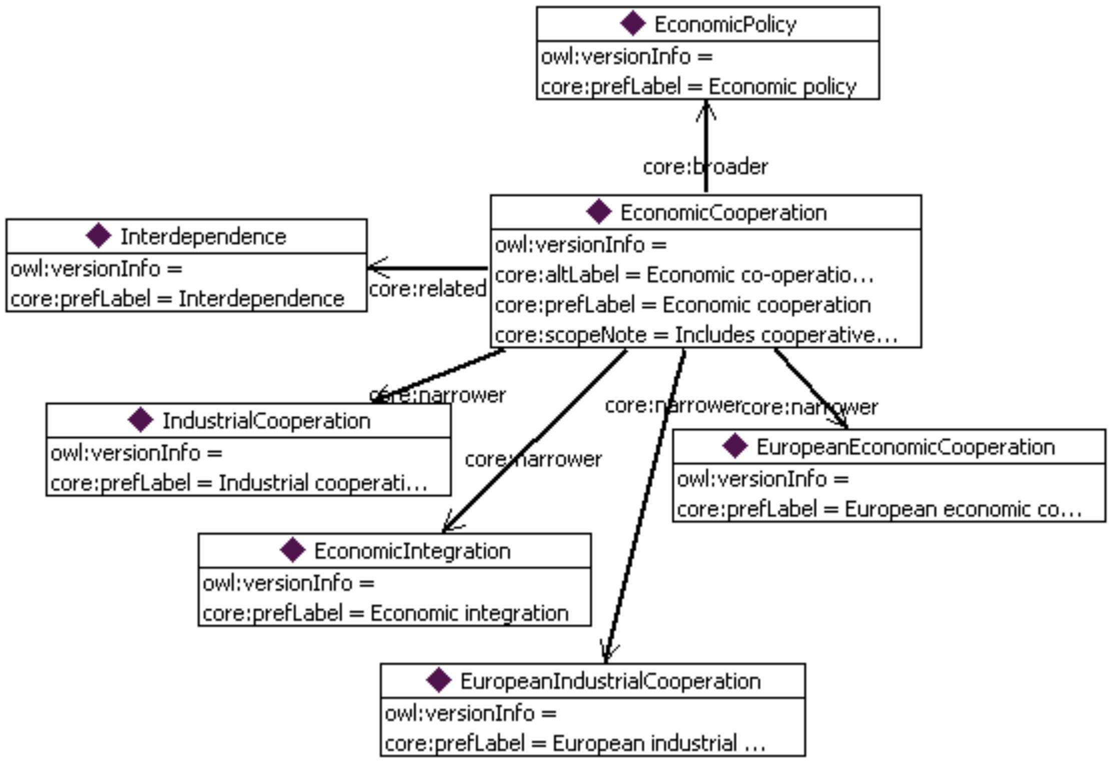
In this chapter, we will describe each of these modeling systems, and show the use they have made of the OWL-Fast constructs we introduced in the previous chapters.

8.1 SKOS

SKOS ("Simple Knowledge Organization System") was developed by the Institute for Learning & Research Technology to provide a standard for representing knowledge organization systems (including controlled vocabularies, thesauri, taxonomies and folksonomies) in a distributed and linkable way. Given the existence of several thesaurus standards, one could well wonder why this group found it necessary to create another one. The key differentiator between SKOS and other thesaurus standards is its basis in the Semantic Web; unlike other standards, SKOS was designed from the start to allow modelers to create modular knowledge organizations that can be re-used and referenced across the web. SKOS was not designed to replace any other thesaurus standard, but in fact to augment it, by bringing the distributed nature of the Semantic Web to thesauri and controlled vocabularies. Toward this end, it was also a design goal of SKOS that it be possible to map other thesaurus standards to SKOS in a fairly straightforward way.

SKOS is organized in layers; the SKOS Core is the most mature, and the part that maps directly to other thesaurus standards. SKOS Mapping is an extension to SKOS that defines a number of specific properties for mapping thesaurus concepts from one source to another. In this section we will concentrate on describing the mature SKOS Core in terms of its usage of OWL-Fast, and the inferences that it entails.

Figure 8-1 shows a sample from a SKOS thesaurus, in which a small fragment of the UK Archival Thesaurus has been rendered in SKOS. The diagram shows seven concepts, which are related to one another by various properties that are defined in the SKOS Core. Data properties are shown within the boxes corresponding to the concepts. As we shall see, each of these properties is defined in relation to other properties, so that certain useful inferences can be made.


EconomicPolicy
owl:versionInfo =
core:prefLabel = Economic policy
core:broader
EconomicCooperation
owl:versionInfo =
core:altLabel = Economic co-operatio...
core:prefLabel = Economic cooperation
core:scopeNote = Includes cooperative...
core:related
Interdependence
owl:versionInfo =
core:prefLabel = Interdependence
core:narrower
IndustrialCooperation
owl:versionInfo =
core:prefLabel = Industrial cooperati...
core:narrower
core:narrower
core:narrower
EuropeanEconomicCooperation
owl:versionInfo =
core:prefLabel = European economic co...
EconomicIntegration
owl:versionInfo =
core:prefLabel = Economic integration
EuropeanIndustrialCooperation
owl:versionInfo =
core:prefLabel = European industrial ...

8-1 Sample Thesaurus in SKOS. Example from W3C; data from UKAT.

The same information from Figure 8-1 is shown as triples in N3 below:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix core: <http://www.w3.org/2004/02/skos/core#>.
@prefix UKAT: <http://www.workingontologist.com/Ch8/UKAT.owl#>.

    UKAT:EconomicCooperation a core:Concept;
         core:altLabel "Economic co-operation";
         core:broader UKAT:EconomicPolicy;
         core:narrower UKAT:IndustrialCooperation,
                       UKAT:EconomicIntegration,
                       UKAT:EuropeanIndustrialCooperation,
                       UKAT:EuropeanEconomicCooperation;
         core:prefLabel "Economic cooperation";
         core:related UKAT:Interdependence;
         core:scopeNote "Includes cooperative measures in
    banking, trade, industry etc., between and among
    ocuntries.".

    UKAT:EconomicIntegration a core:Concept;
         core:prefLabel "Economic integration".

    UKAT:EconomicPolicy a core:Concept;
         core:prefLabel "Economic policy".

    UKAT:EuropeanEconomicCooperation a core:Concept;
         core:prefLabel "European economic cooperation".

    UKAT:EuropeanIndustrialCooperation a core:Concept;
         core:prefLabel "European industrial cooperation".

    UKAT:IndustrialCooperation a core:Concept;
         core:prefLabel "Industrial cooperation".

    UKAT:Interdependence a core:Concept;
         core:prefLabel "Interdependence".
```

First, let's look at the notion of labels in SKOS. As we have seen in section 6.3.8XXX, there is already a label resource defined in RDFS, *rdfs:label*. While rdfs:label has no formal semantics defined (that is, there are no inferences that concern *rdfs:label*), it does have the informal meaning that it is something that can be used as the printable or human readable name of a resource. SKOS provides a more detailed notion of a concept's label, in accordance with usual thesaurus practice. In particular, it defines three different kinds of labels, a preferred label, an alternative label, and a hidden label. These are defined in SKOS with the following triples:

```
skos:prefLabel
      a rdf:Property ;
      rdfs:label "preferred label" ;
      rdfs:subPropertyOf rdfs:label .

skos:altLabel
      a rdf:Property ;
      rdfs:label "alternative label" ;
      rdfs:subPropertyOf rdfs:label .

skos:hiddenLabel
      a rdf:Property ;
      rdfs:label "hidden label" ;
      rdfs:subPropertyOf rdfs:label .
```

The SKOS definition includes a number of other triples defining these properties, but we will concentrate on these for this description.

Notice that each property has an *rdfs:label*, which provides a human readable version of the name of each resource. Furthermore, each of these properties is declared to be of type *rdf:Property*. Furthermore, each of these is declared to be a subproperty of *rdfs:label*. What does this mean, in terms of OWL-Fast?

As we have already seen, *rdfs:subPropertyOf* propagates triples from the sub-property to the super-property. In the first case, from any triple using *skos:prefLabel* as a predicate, we can infer the same triple with *rdfs:label* as predicate instead. The same is true for *skos:altLabel* and *skos:hiddenLabel*; in particular, in our UKAT example, we can infer the following triples:

```
UKAT:EconomicCooperation
        rdfs:label "Economic co-operation" .
UKAT:EconomicCooperation
        rdfs:label "Economic cooperation" .
UKAT:EconomicIntegration
        rdfs:label "Economic integration" .
UKAT:EconomicPolicy
        rdfs:label "Economic policy" .
UKAT:EuropeanEconomicCooperation
        rdfs:label "European economic cooperation" .
UKAT:EuropeanIndustrialCooperation
        rdfs:label "European industrial cooperation" .
UKAT:IndustrialCooperation
        rdfs:label "Industrial cooperation" .
UKAT:Interdependence
        rdfs:label "Interdependence" .
```

That is, every SKOS label shows up as an *rdfs:label*. In some cases (e.g., *UKAT:EconomicCooperation*), more than one value for *rdfs:label* can be inferred. This is perfectly legal in OWL-Fast (after all, *rdfs:label* is not an *owl:FunctionalProperty*), even though its informal interpretation as the printable name of a resource is not clear.

SKOS uses this same pattern for many of the properties it defines; for each of them, the sort of inference it supports is similar. So, for the seven documentation properties in SKOS, six of them are subproperties of the seventh, thus:

```
core:definition rdfs:subPropertyOf core:note .
core:scopeNote rdfs:subPropertyOf core:note .
core:example rdfs:subPropertyOf core:note .
core:historyNote rdfs:subPropertyOf core:note .
core:editorialNote rdfs:subPropertyOf core:note .
core:changeNote rdfs:subPropertyOf core:note .
```

Similarly, SKOS defines three properties having to do with symbols,

```
core:altSymbol rdfs:subPropertyOf core:symbol .
core:prefSymbol rdfs:subPropertyof core:symbol .
```

Just as was the case for the SKOS label properties, any triple using one of the symbol properties or documentation properties will entail a triple using *core:symbol* or *core:note* respectively.

8.1.1 Semantic Relations in SKOS

SKOS defines three so-called Semantic Properties; these are the properties that relate concepts to one another, using the familiar terms *broader*, *narrower* and *related* from thesaurus standards. SKOS defines some simple constraints among these properties:

```
skos:broader
      a owl:TransitiveProperty  ;
      owl:inverseOf skos:narrower ;
      rdfs:comment "Broader concepts are typically
rendered as parents in a concept hierarchy (tree)." ;
      rdfs:label "has broader" .

skos:narrower
      a owl:TransitiveProperty ;
      owl:inverseOf skos:broader ;
      rdfs:comment "Narrower concepts are typically
rendered as children in a concept hierarchy (tree)." ;
      rdfs:label "has narrower" .

skos:related
      a owl:SymmetricProperty;
      rdfs:label "related to" ;
      rdfs:subPropertyOf  rdfs:seeAlso .
```

These properties take advantage of a handful of the constructs of OWL-Fast. We'll see how these work together in the UKAT example.

First, since *skos:narrower* is an inverse of *skos:broader*, we can make the following inferences about UKAT concepts in Figure 8-1.

```
UKAT:EconomicPolicy core:narrower
                      UKAT:EconomicCooperation .
UKAT:IndustrialCooperation core:broader
                      UKAT:EconomicCooperation .
UKAT:EconomicIntegration core:broader
                      UKAT:EconomicCooperation .
UKAT:EuropeanIndustrialCooperation core:broader
                      UKAT:EconomicCooperation .
UKAT:EuropeanEconomicCooperation core:broader
                      UKAT:EconomicCooperation .
```

Furthermore, since each of *core:narrower* is a *owl:TransitiveProperty*, we can infer that every concept in this sample is narrower than the item at the "top" of the tree, *UKAT:EconomicPolicy*:

```
UKAT:EconomicPolicy core:narrower
               UKAT:IndustrialCooperation ,
               UKAT:EconomicIntegration ,
               UKAT:EuropeanIndustrialCooperation ,
               UKAT:EuropeanEconomicCooperation .
```

Similar triples can be inferred (swapping subject for object, as usual) for the inverse property, *core:broader.*

In the case of *core:related*, it is not defined as *owl:TransitiveProperty*, so we cannot make inferences about chains of related items. This is probably as it should be; since it is easy to imagine a chain of pairwise related terms in which the first term is not related to the last term. However, we see that *core:related* is an *owl:SymmetricProperty*; this means that we can make the following inference. If we assert that

```
UKAT:EconomicCooperation core:related
                        UKAT:Interdependence .
```

Then we can infer that

```
UKAT:Interdependence core:related
                        UKAT:EconomicCooperation .
```

8.1.1.1 Meaning of Semantic Relations

It is no accident that there is considerable similarity between the definitions of in SKOS of *skos:narrower* and *skos:broader*, and the definitions in Section 7.2.2 XXX of *rdfs:subClassOf* and *superClassOf*. Both of these pairs of properties are intended for modeling hierarchies. In both cases, it is desirable that the hierarchies could be traversed either “upward” or “downward”. In both cases, the intention of the hierarchical structure is that the relationship be transitive, that is, narrower than narrower is narrower, and subClassOf subClassOf is subClassOf.

There is one definition for subClassOf that has no corresponding condition in SKOS; that is the semantic rule that says that if we have triples of the form

```
B rdfs:subClassOf C .
x rdf:type B .
```

Then we can infer that

```
x rdf:type C .
```

Because of this rule, there is no confusion about the interpretation of *rdfs:subClassOf*; this rule makes it clear that C has more members (or at least, just as many) as B; that is, C is the more encompassing of the two classes.

Since we have no such rule in SKOS, there is possibility for confusion; when we say

```
EconomicCooperation skos:broader EconomicPolicy .
```

Should we read this (in English) as “Economic Cooperation has broader category Economic Policy,” or should we read it as “Economic Cooperation is broader than Economic Policy”? There is nothing in the formal SKOS model to tell us which is which. The relationship is expressed informally in the annotations on *skos:broader* and *skos:narrower*, e.g., the labels “has broader” and “has narrower” respectively indicate that the former interpretation is the intended one – economic cooperation has broader term economic policy. It is important to keep this in mind when reading the SKOS examples that follow in this book, where we will see triples like

```
:Milk skos:broader :Dairy .
```

For many people, this interpretation of *broader* is backwards from what they expect.

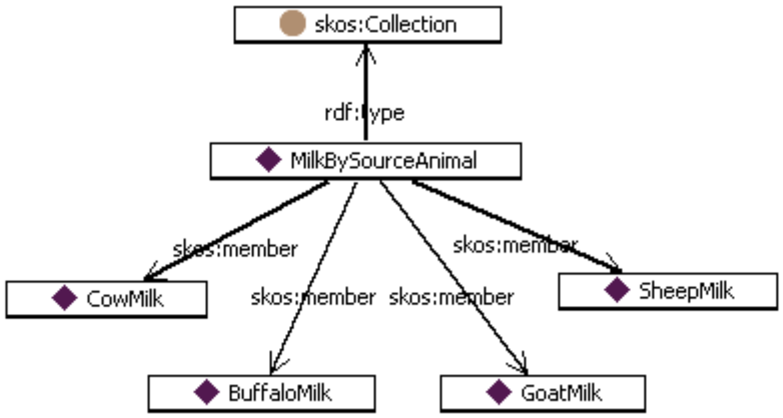
If there were an inference-based definition of the semantics of *skos:broader* (as there is, for example, for *rdfs:subClassOf*), then the intended direction of this statement would be explicit. There would be no need to rely on the interpretation of examples (like this one for *Milk* and *Dairy*) to communicate which way the terms are intended to be used.

8.1.2 Special Purpose Inference

SKOS includes a special provision for implementing Collections of concepts. Collections of terms are common in thesaurus and indexing standards. Consider the following example from the W3C SKOS Core Guide.

A term index describes agricultural products, and includes several kinds of milk; cow milk, goat milk, sheep milk and buffalo milk. There is a meaningful collection of these concepts called “milk by source animal”. This practice of grouping concepts is common practice in indexing and cataloguing. It is important to notice that according to the common practice of professional cataloguers, the grouping “milk by source animal” is itself not a concept in its own right; it is simply a grouping for concepts.

SKOS uses a class called *skos:Collection* and a property called *skos:member* to express such situations, as shown in Figure 8-2.



8-2 "Milk by source animal" is a collection of four concepts related to milk.

The triples for Figure 8-2 are given in N3 as

```
agro:MilkBySourceAnimal a skos:Collection;
     rdfs:label "Milk by source animal ;
     core:member agro:CowMilk,
                 agro:BuffaloMilk,
                 agro:GoatMilk,
                 agro:SheepMilk .

agro:BuffaloMilk a skos:Concept;
     skos:prefLabel "Buffalo milk .

agro:CowMilk a skos:Concept;
     skos:prefLabel "Cow milk" .

agro:GoatMilk a skos:Concept;
     skos:prefLabel "Goat milk" .

agro:SheepMilk a skos:Concept;
     skos:prefLabel "Sheep milk" .
```

The interest in this example comes when we examine what inferences we can draw from such a construct. So far, we have only used *skos:narrower* to express that one term has another as a narrower term. But what would it mean to this same notion of *skos:narrower* to describe the relationship between a term and a collection? e.g.,

```
agro:Milk skos:narrower agroMilkBySourceAnimal .
```

SKOS does not model the answer to this question in OWL-Fast, but instead specifies a special-purpose rule as part of the SKOS specification. If we have triples of the form

```
X skos:narrower C .
C skos:member Y .
```

Then we can infer the triple

```
X skos:narrower Y .
```

When we apply this rule to *agro:Milk*, we can infer that it has as narrower term each of the kinds of milk in the collection *agro:MilkBySourceAnimal*, thus:

```
agro:Milk skos:narrower agro:BuffaloMilk .
agro:Milk skos:narrower agro:CowMilk .
agro:Milk skos:narrower agro:SheepMilk .
agro:Milk skos:narrower agro:GoatMilk .
```

The SKOS standard represents this constraint as a rule, rather than modeling it in OWL-Fast. This is not surprising, since the constructs of OWL-Fast are not well-suited to this problem. We shall see in Chapter 10XXX how further constructs in OWL can be brought to bear on situations like this one.

8.1.3 Published Subject Indicators

SKOS includes support for the notion of a Published Subject Indicator, or PSI. The idea of a PSI is that a community can agree on a particular publication that can act as a unique identifier for a certain concept. For traditional, generic concepts like “Milk” or “Economic Policy” it is unlikely that there will be a useful unique publication for the concept. But for the results of standards bodies such publications are commonplace.

Examples include things like CDC disease listings, technical standards (like SKOS itself!), acts of governments, etc. For instance, if two diseases have the same CDC listing, then they are the same disease.

SKOS provides a property called *skos:subjectIndicator* to link a *skos:Concept* to a published document. Since a PSI is intended as a unique identifier of a concept, it should not be possible for two different concepts to share the same *subjectIndicator*. This stipulation is quite simple to represent in OWL-Fast, as we have seen in section 7.6.2XXX, where we introduced *owl:InverseFunctionalProperty*. The SKOS specification includes the triple

```
skos:subjectIndicator rdf:type owl:InverseFunctionalProperty .
```

This indicates that any two concepts that share the same PSI must therefore refer to the same concept, unifying them in the knowledge organization system. For instance, suppose that the document at http://www.usdoj.gov/foia/privstat.htm is the PSI for the US Privacy Act of 1974. Then if we have concepts from two different knowledge organization systems, e.g.,

```
policy:Privacy a skos:Concept;
       skos:subjectIndicator
            http://www.usdoj.gov/foia/privstat.htm .

gov:InfoAccess a skos:Concept ;
       skos:subjectIndicator
            http://www.usdoj.gov/foia/privstat.htm .
```

We can infer that these two concepts are indeed the same, i.e.,

```
gov:InfoAccess owl:sameAs policy:Privacy .
```

Any indexing application that utilizes the thesaurus can then respond accordingly; for example, any items indexed under *gov:InfoAccess* will also be accessible under *policy:Privacy*. This illustrates how the property *skos:subjectIndicator* plays an important role in the utilization of SKOS on the semantic web, since it allows terms from different vocabularies to be mapped to one another.

8.1.4 SKOS in action

SKOS is an example of a model on the Semantic Web; it models a particular standard for how to represent thesauri. In this section, we have examined what the SKOS model says about terms and concepts in a thesaurus, and how they can relate to one another. But how is SKOS itself being used? What do we gain by representing a thesaurus in SKOS?

The information explosion that we are familiar with on the web is taking place elsewhere as well. Libraries around the world are interested in indexing their materials in a way that will allow patrons to find information from all around the world. The United Nations has been quite successful with a thesaurus called Agrovoc, that provides multilingual indexing for materials concerning any aspect of agriculture. Not surprisingly, member nations have their own indexes for agriculture. The National Agriculture Library (NAL) of the United States also has an extensive thesaurus (in English) for indexing agricultural materials.

When the United Nations pursued a project to map these thesauri together, they needed a representation that would allow for terms from multiple sources to be distinguished in a global way; that is, the Agrovoc word for "Ground Water" and the NAL word for "Ground Water" must be managed separately, but it also must be possible to represent the relationship between them. The use of URIs in RDF (and hence in SKOS) is ideal for this job. SKOS itself provides a set of terms as described here, for familiar thesaurus relationships *broader* and *narrower*. This makes it a straightforward task to export each thesaurus in SKOS. In fact, both Agrovoc and the NAL had independently sponsored SKOS exports of their thesauri. With these SKOS representations in place, it was a straightforward matter to represent mappings between the two vocabularies in RDF.

8.2 FOAF

FOAF ("Friend of a Friend") is a format for supporting distributed descriptions of people and their relationships. The name "Friend of a Friend" is intended to evoke the fundamental relationship that holds in social networks; you have direct knowledge of your own friends, but only through your network can you access the friends of your friends.

FOAF works in the spirit of the AAA principle; anyone can say anything about any topic. In the case of FOAF, the topics that anyone is usually saying things about are people. Other things that are commonly related to what we might want to say about people, e.g., Organizations (that people belong to), Projects (that people work on), Documents (that people have created or that describe them) and Images (that depict people), are also included in the core FOAF description. Information about a single person is likely to be distributed across the web, and represented in different forms; on their own webpage, a person is likely to list basic information about interests, current projects, and some images. But further information will be available only on other pages; a photo set taken at a party or conference could include a picture that depicts a person, who has not listed that photoset in their own web page. A conference organizer could include information about a paper that lists its authors, even if the authors themselves might not have listed the paper on their own website. A laboratory or office might have a page that lists all of its members. FOAF leverages the distributed nature of RDF to provide a distributed representation of this information. Social networking sites have begun to make information available in FOAF for web-scale distribution.

Given that there are a number of social networking websites available, and that each one of them has a way to represent its members, information about them, and ways in which they are connected to one another, one could well ask why there is a need for yet another way to describe people and their social networks. The idea of FOAF is not to replace any of these systems, but to provide a framework whereby this information can be distributed. Furthermore, using RDF, FOAF provides a framework that is extensible; since anyone can say anything about any topic, FOAF allows anyone to make novel statements about people, projects, etc., and to relate these statements to other statements already made.

FOAF leverages the AAA principle, as well as the distributed and extensible nature of RDF in an essential way. At any point in time, FOAF is a work in progress; there are vocabulary terms in FOAF whose semantics are defined only by natural language descriptions in the FOAF "standard"; other terms have definitions defined in OWL-FAST that relate them in a formal way to the rest of the description. FOAF is designed to grow in an organic fashion, starting with a few intuitive terms, focusing their semantics as they are used. There is no need to commit early on to a set vocabulary, since we can use OWL-FAST to connect together new vocabulary and old vocabulary, once we determine the desired relationship between them.

FOAF provides a small number of classes and properties as its starting point; these make use of some of the basic constructs of OWL-Fast to maintain consistency and to implement FOAF policies for information merging. FOAF is a fairly simple system for describing people, the things they create and the projects they participate in. It is primarily organized around three classes: *foaf:Person*, *foaf:Group* and *foaf:Document*.

8.2.1 People and Agents

While FOAF is primarily about people, some of the things we want to say about people are true of other things as well; groups, companies, etc. So a *foaf:Person* is defined as part of a compact hierarchy under the general grouping of *foaf:Agent*:

```
foaf:Person rdfs:subClassOf foaf:Agent .
foaf:Group rdfs:subClassOf foaf:Agent .
foaf:Organization rdfs:subClassOf foaf:Agent .
```

Many things we might say about a *foaf:Person* can hold for any *foaf:Agent*; in fact, FOAF is quite liberal in this regard; most of the properties we will describe here for people hold for agents in general. Details of exactly which properties are used for which classes are available in the FOAF Vocabulary Specification http://xmlns.com/foaf/0.1/.

8.2.2 Names in FOAF

Probably the most essential thing that we know about a person is their name; FOAF provides a number of vocabulary terms to describe the name of a person. Even something as simple as a person's name can be quite complex. FOAF begins with a simple notion of name, which it sensible calls *foaf:name*.

```
foaf:name rdfs:domain owl:Thing .
foaf:name rdfs:subPropertyOf rdfs:label .
```

That is, anything in the world can have a name (including a *foaf:Person*), and that name is also used as the printable label for that thing. For a *foaf:Person*, the name is typically the full name of the person, like “William Shakespeare” or “Anne Hathaway”.

While the full name of a person is quite useful, there are parts of a person’s name that are needed in some circumstances. *foaf:firstName*, *foaf:givenname*, *foaf:family_name* and *foaf:surname* are four properties relating to names of people that are defined in FOAF. Each of them has an intuitive meaning, but there are no formal semantics; the meaning is given only in the prose descriptions in the standard, and by evolving conventions of use. As FOAF evolves, it will need to encompass different cultures and their use of names; does the given name always come first? Is a family name always the surname? How do culture-specific names (for example, the “Christian name” that is still used in some cultures) relate to other names? One of the advantages to basing FOAF on RDF is that it is not necessary to resolve all of these issues, in order to begin the project of marking up data using the FOAF vocabulary. The strategy taken by FOAF is to begin by annotating a person’s name, while providing other naming vocabulary like surname, firstname, givenname etc. Usage patterns will dictate which of these will turn out to be useful. If it turns out that, say, two properties are used in exactly the same way, then this observation can be cast as part of the standard by describing the relationship in OWL. For example,

```
    foaf:surname owl:equivalentProperty foaf:family_name .
```

8.2.3 Nicknames and Online names

Since FOAF is primarily used on the Web, it is expected that many of the people FOAF will be used to describe will be active in various internet communities. For instance, it is likely that a FOAF Person will have a screen name on some online chat service. FOAF currently identifies *foaf:aimChatID*, *foaf:icqChatID*, *foaf:msnChatID*, and *foaf:yahooChatID*. A recent addition includes *foaf:jabberID* as well. In the spirit of extensibility of FOAF, new ID properties can be added on an as-needed basis. While some part of the semantics of these properties is given by their natural language descriptions (which connect *foaf:yahooChatID* to the chat service Yahoo!), FOAF also makes a formal connection between these properties. In particular, all of them are sub-properties of a single property, *foaf:nick*:

```
foaf:aimChatID rdfs:subPropertyOf foaf:nick .
foaf:icqChatID rdfs:subPropertyOf foaf:nick .
foaf:msnChatID rdfs:subPropertyOf foaf:nick .
foaf:yahooChatID rdfs:subPropertyOf foaf:nick .
foaf:jabberID rdfs:subPropertyOf foaf:nick .
```

Following the rules of *rdfs:subPropertyOf* from chapter 6XXX, this means that any *foaf:Person* who is active in chat spaces is likely to have multiple values for the property *foaf:nick*, that is to have multiple nicknames. They can, of course, have further nicknames as well. For instance, when William Shakespeare became active in internet chat rooms, from a FOAF point of view, all those screen names are also nicknames:

```
Shakespeare foaf:aimChatID "Willie1564" .
Shakespeare foaf:msnChatID "TempestMan" .
Shakespeare foaf:nick "Willie1564" .
Shakespeare foaf:nick "TempestMan" .
```

Of course, we can still assert a nickname for the poet and playwright, even if he doesn't use it as a screen name anywhere:

```
      Shakespeare foaf:nick "The Bard of Avon" .
```

8.2.4 Online persona

The Internet provides a number of ways for a person to express himself, and FOAF is under constant revision to provide properties to describe these things. A person is likely to have an electronic mailbox; FOAF provides a property *foaf:mbox* for this purpose. Many people maintain a number of web pages describing parts of their lives; many people have personal home pages, some have home pages at their workplace or at their school (or both!). Their workplaces themselves can have homepages, etc. FOAF uses the same strategy for these properties as it does for names; it provides a wide array of properties, defined informally (by natural language descriptions in the standard).

foaf:homepage – relates a person to their primary homepage. This property applies to anything in FOAF, not just to people.

foaf:workplaceHomepage – the homepage of the workplace of a person. Anything can have a homepage (even an employer!), but only a *foaf:Person* can have a workplaceHomepage.

foaf:workInfoHomepage – the homepage of a person at their workplace. Such a page is usually hosted by a person's employer, but is about the person's own work there.

foaf:schoolHomepage – the homepage of the school that a *foaf:Person* attended.

As the internet provides new means of expression, FOAF keeps up:

foaf:weblog – the address of the web blog of a person.

All of these properties specify instances of the class *foaf:Document*; that is, a web page is a *foaf:Document*, a weblog is a *foaf:Document*, etc.

8.2.5 Groups of people

One of the interesting things about people is the groups they belong to. FOAF provides a class called *foaf:Group* to define these groups. A group is connected to its members via a property called, appropriately enough, *foaf:member*. A *foaf:Group* is defined quite loosely; any grouping of people can be described this way. For instance, we could define a group called EnglishMonarchy as follows:

```
:English_Monarchy
      a foaf:Group ;
      foaf:name "English Monarchy" ;
      foaf:homepage "http://www.monarchy.com/" ;
      foaf:member :William_I, :Henry_I, :Henry_II,
        :Elizabeth_I, :Elizabeth_II .
```

A group in FOAF is an instance, in particular, an individual of type *foaf:Group*. As such, there are a number of properties that can describe it, like *foaf:name* (as we see here). In fact, a *foaf:Group* has a lot in common with a *foaf:Person*; it can have a chat ID, a nickname, an email box, a homepage or even a weblog.

But it is also useful to consider the members of a group as instances of a class; that is, to related the instance of *foaf:Group* to an *rdfs:Class*. For this purpose, FOAF provides a link from a group to a class, called *foaf:membershipClass*. Suppose that the membership class for *English_Monarchy* is called *Monarch*; this connection is expressed in FOAF with the triple

```
:English_Monarchy foaf:membershipClass :Monarch .
```

The members of the group *Enlish_Monarchy* all have type *Monarch*:

```
:William_I a :Monarch .
:Henry_I a :Monarch .
:Henry_II a :Monarch .
:Elizabeth_I a :Monarch .
:Elizabther_II a :Monarch.
```

Ideally, all of these triples should be maintained automatically; that is, any individual of type *Monarch* should appear a member of the group *English_Monarchy*, and every member of the group *English_Monarchy* should have *Monarch* as a type. FOAF makes this stipulation as part of the standard. We will see in Chapter 10 how to use the more advanced capabilities of OWL to build a model from which we can infer these triples. The distinction between the instance *English_Monarchy* and the class *Monarch* is a subtle one; the class *Monarch* is a type in RDFS; as such it refers to schematic things about monarchs – property domains, subclasses, etc. *English_Monarchy*, on the other hand, refers to the institution of the monarchy itself, which refers to things like this history of the monarchy, web pages and books about the monarchy, and so on.

In our examples so far, we have been kept the world of classes separate from the world of instances; the only relation between an instance and a class has been the *rdf:type* property. The intuition behind *foaf:membershipClass* is that it indicates a class, whose instances are exactly the same as the members of the group. The expression of this kind of relationship, in which we sometimes wish to view something as an instance (e.g., *English_Monarchy*, an instance of the class *foaf:Group*) and sometimes as a class (e.g., the class *Monarch*, representing all the instances that are *foaf:member* of that group), is an example of a practice called *meta-modeling*. We will see more about meta-modeling when we learn about the rest of the OWL language, and we will see how we can use meta-modeling constructs in OWL to formalize the relationship between a *foaf:Group* and its *foaf:membershipClass*.

8.2.6 Things people make and do

Interesting people create things; they write books, publish web pages, create works of art, found companies, and start organizations. FOAF provides two properties to relate people to their creations: *foaf:made* and *foaf:maker*. They are inverses of one another, and have relate *a foaf:Agent* to an *owl:Thing* as follows:

```
foaf:made rdfs:domain foaf:Agent .
foaf:made rdfs:range owl:Thing .
foaf:maker rdfs:domain owl:Thing .
foaf:maker rdfs:range foaf:Agent .
foaf:made owl:inverseOf foaf:maker .
```

That is, anything in the describable universe is fair game for being made by some agent. Even another agent could have a *foaf:maker*!

If a person is an author, then they are likely to have publications to their credit. The property *foaf:publications* relates a *foaf:Person* to any *foaf:Document* they have published. Interestingly, FOAF does not specify that a person has *foaf:made* any of their *foaf:publications*. In the spirit of the AAA principle, if we were to decide to make such a statement, we could do so simply by saying

```
foaf:publications rdfs:subPropertyOf foaf:made .
```

8.2.7 Identity in FOAF

The main goal of FOAF is to apply the AAA principle to describing networks of people; anyone can contribute descriptions about anyone. But this leads to a problem; it is easy enough for me to describe myself; I can publish a document that says whatever I wish to make known. If someone else wants to contribute information about me (say, for example, that the publisher of this book wants to add the information that I am an author), how will they refer to me? Or if I have several profiles on different sites that I would like to merge together, how can I link them to describe the one thing that is “me”?

The RDF answer to this question is quite simple, but not really adequate for the uses of FOAF. RDF uses URIs to denote the things it describes – that means that I should have a URI that denotes me, and anyone who wants to make a comment about me can make it using that URI. This is a simple, elegant, and standard solution to this problem.

The problem arises in the adoption of FOAF. When someone makes their first FOAF page, how do they determine their own URI? Do they just make it up? It just isn’t very common on the web for people to have their own personal URIs to describe themselves. In order to lower the barriers to adopting FOAF, there needs to be a way in which people can refer to one another that uses some part of the internet infrastructure that is already ubiquitous and familiar. FOAF needs to utilize some pre-existing way to identify individuals. Is there any identifying marker that everyone on the internet already has, and is already familiar with?

The clearest answer to this puzzle is email; just about anyone who is described on the web in any way at all has an email address. Furthermore, it is quite rare that two people share the same email address. Rare enough that for the purposes of FOAF, email can serve as a unique identifier for people on the web. Note that it isn’t a problem if someone has two or more email addresses or if one email address is valid only for a limited period of time. All FOAF requires of the email address is that another person doesn’t share it (either simultaneously or later on).

We can express this constraint in plain language by saying simply that two people who share the same email address are in fact not two distinct people at all, but instead are the same person. But as we have already seen in Section 7.6.2XXX, OWL-FAST has a way to formalize this relationship. When a property uniquely identifies an individual, we say that the property is an *owl:InverseFunctionalProperty*. So in FOAF, we can express the central role that *foaf:mbox* plays in identifying individuals with the single triple

```
foaf:mbox rdf:type owl:InverseFunctionalProperty .
```

Once we identify *foaf:mbox* as an *owl:InverseFunctionalProperty*, we realize that similar statement can be made about a number of the properties we use to describe people; it is unusual for two people to share a YahooChatID, or an AIMChatID. In fact, all of the following properties in FOAF are *owl:InverseFunctionalProperties*:

```
foaf:aimChatID rdf:type owl:InverseFunctionalProperty .
foaf:homepage rdf:type owl:InverseFunctionalProperty .
foaf:icqChatID rdf:type owl:InverseFunctionalProperty .
foaf:jabberID rdf:type owl:InverseFunctionalProperty .
foaf:mbox rdf:type owl:InverseFunctionalProperty .
foaf:msnChatID rdf:type owl:InverseFunctionalProperty .
foaf:weblog rdf:type owl:InverseFunctionalProperty .
foaf:yahooChatID rdf:type owl:InverseFunctionalProperty .
```

Using the *foaf:mbox* (and similar properties) as identifiers of individuals solves the technical problem of identifying individuals, using some pre-existing identification, but it raises another problem; publishing the email address for a person can be seen as a violation of privacy, since email addresses (and chat IDs) can be used to pester or even attack someone by sending unwanted, offensive or just bulky mail. So if we want to apply the AAA principle to William Shakespeare, and we know that he uses the email address Shakespeare@gmail.com, we can refer to him as "the person with email 'Shakespeare@gmail.com'" (using a blank node, as we did for Shakespeare's inspiration in section 3.5).

```
[ foaf:mbox "Shakespeare@gmail.com"]
```

But when we do this, we publish his email address in plain text for information vandals to steal and use; this isn't a very polite thing to do to someone we know and respect. For this reason, FOAF also offers an obfuscated version of *foaf:mbox*, called *foaf:mbox_sha1sum*. It indicates the result of applying a hashing function called SHA-1 to the email address. The SHA-1 function is publicly available but very difficult to reverse. We apply the algorithm to Shakespeare's email address to get the obfuscated string "f964f2dfd4784fe9d68ada960099e0b592e16a95". Now we can refer to him using this value:

```
[ foaf:mbox_sha1sum "f964f2dfd4784fe9d68ada960099e0b592e16a95" ]
```

without compromising his email privacy.

Unfortunately, FOAF does not provide a standard way to obfuscate the other identifying properties like *foaf:aimChatID*, *foaf:yahooChatID*, etc.

8.2.8 It's not what you know, it's who you know

The key to FOAF as a social networking system is the ability to link one person to another. FOAF provides a single, high-level property for this relationship, called *foaf:knows*. The idea behind *foaf:knows* is simple; one person knows another one, who knows more people, and so on, forming a network of people knowing people. There isn't a lot of inferencing going on with *foaf:knows*; the only triples defined for it are

```
foaf:knows rdfs:domain foaf:Person .
foaf:knows rdfs:range foaf:Person .
```

that is, *foaf:knows* just links one *foaf:Person* to another.

The lack of inferencing over *foaf:knows* is by design; the design of *foaf:knows* is intentionally vague, to indicate some relationship between people. Such a relationship could be concluded informally from other information, for instance, co-authors can usually be assumed to know one another. And while it is usual to think that if one person knows another, that the relationship is mutual, the FOAF designers intentionally left out the assertion of *foaf:knows* as an *owl:SymmetricProperty*, since there might even be some disagreement about whether one person knows another.

Despite its vague definition, *foaf:knows* provides the infrastructure for using FOAF for social networking, as it links one person to the next and then to the next and so on . . .

8.3 Lessons Learned

SKOS and FOAF demonstrate how a fairly simple set of modeling constructs can be used to create extensible, distributed information networks. The both take advantage of the distributed nature of RDF to allow extension to a network of information to be distributed across the web. Both of them rely on the inferencing structure of OWL-FAST to add completeness to their information structure. Both of them make use of *owl:InverseFunctionalProperty* to determine identity of key elements.

While they are similar in these ways, FOAF and SKOS are organized very differently in terms of how they support extension by their expected user communities. FOAF takes something of an evolutionary approach to information extension. Many concepts have a broad number of terms (like the several variants of “name” that we saw in section 8.2.2). FOAF can be extended as new features are needed; for instance, *foaf:weblog* was not as important before blogging became fashionable; in recent times, it has nearly surpassed the more classical *foaf:homepage* in importance.

SKOS, in contrast, takes a much more orderly approach to extension. The SKOS standard comes in three parts; the SKOS Core, which has been described here, SKOS

Mapping, which includes vocabulary fro mapping vocabularies from different sources, and the SKOS Extensions, for particular vertical applications of SKOS. The SKOS Core is intended to be a sort of interlingua for thesauri, and has been designed by a small committee in an attempt to consolidate the fundamentals of other thesaurus systems into a single semantic web standard. The other two documents import the Core, and build further semantics on top of it.

The difference in these two approaches becomes more apparent when we think about how they will be extended; FOAF takes very seriously the AAA slogan, to the point that the actual preferred parts of the standard will be determined to a large extent by its use. SKOS, on the other hand, has a fairly stable core, which has been designed by an informed committee, who have performed a detailed commonality/variability analysis of extant vocabulary systems. The architecture of SKOS has been determined and published, and serves as a roadmap to development of the standard.

The technical structure of RDF supports both of these modes. The free extension style of FOAF and the orderly layering of SKOS are accomplished using the same graph overlay mechanism of RDF; the difference is in how the overlay is organized and governed. Neither approach is inherently superior to the other; each of them accomplishes certain goals that are of importance to each of these projects.

8.4 Fundamental Concepts

The following fundamental concepts were introduced in this chapter:

foaf: Namespace for a system of representation of social network information; short for “friend of a friend”

skos: Namespace for a system of representation for information management SKOS stands for “Simple Knowledge Organization System”.

Meta-modeling: Generally speaking, the craft of building a model that describes another model. A specific example is the practice of representing a class in a model as an individual member of another class. FOAF does this explicitly with the *foaf:membershipClass* property that links an individual of type *foaf:Group* to the Class of all members of the group.

[UKAT] The UK Archival Thesaurus. See (http://www.ukat.org.uk/)