8 FOAF: Friend of a Friend

In this short chapter let us take a look at FOAF (Friend of a Friend) [43], another popular application in the world of the Semantic Web. The goal of studying FOAF is to see another Semantic Web application at work and, also, to have a chance to practice what we have learned about RDF, RDF schema, and OWL. Another good reason for touring FOAF is that the FOAF namespace shows up in many ontology documents and in the literature; therefore, understanding FOAF is necessary.

As usual, we will first examine what FOAF is and what it accomplishes for us. Then we will dive inside FOAF to see how and why it works. Finally, we will take a look at some real-world examples and come up with our own FOAF documents.

8.1 WHAT FOAF IS AND WHAT IT DOES

What is FOAF? To understand it, let us start from the fact that we have an innumerable amount of personal Web pages on the Internet already. It is easy and inexpensive to publish Web pages, thanks to the rapid development of Internet technology. On each such Web page, the author often provides some personal information, such as e-mail addresses, phone numbers, interests, and hobbies, etc. Such information is certainly suitable for humans but cannot be easily processed by any automatic agent or tool. However, such personal information is so typical and standard that it would be obviously beneficial if we could make it readable by machines.

How to make this personal information standard and machine readable? By now you should know the answer: use RDF schema or OWL to create an ontology about personal information, and markup each personal Web page by connecting each page to an RDF statement document written using this ontology.

This is the right answer, and the FOAF project was started for this very purpose. The first step is to create an ontology called FOAF to include the basic terms normally used to describe personal information.

Therefore, FOAF is simply a vocabulary (or ontology) that includes the basic terms to describe personal information. It serves as a standard for everyone who wants to markup their homepages: these are the terms you should use, and these are the relationships among them. And if you create this markup file in RDF syntax, then your personal information will be machine readable.

We will look at this vocabulary in detail in later sections, but for now let us take a sneak preview at the terms defined in FOAF. The following are some terms you can use to create an RDF statement about a person:

- `Person`
- `Name`
- `Nick`

- `Title`
- `Homepage`
- `Box`
- `Mbox_sha1sum`
- `Img`
- `Surname`
- `Family_name`
- `Givenname`
- `Firstname`
- `Knows`

Now assume you have created an RDF file using these terms, and you have linked it with you personal homepage. What next?

The next step is to publish this RDF document by sharing its URL with the outside world. And here is the key point: because this document is an RDF document and it is published, it immediately enjoys the full benefit of being an RDF application; it can be easily collected and aggregated with other RDF files.

Therefore, the really exciting part of FOAF starts the moment you publish your RDF description on the Internet: this information can be aggregated, explored, and cross-linked easily.

Let us look at a simple scenario to understand this better. Suppose I have a homepage and I have created a FOAF instance document and published it on the Web. In this FOAF document I simply included my name and homepage URL, nothing else. So it is a fairly boring description.

Now I have a friend who also has her own homepage and has published a FOAF file. In her FOAF document, she did use foaf:`knows` (foaf represents the namespace of the FOAF ontology) property to indicate that she knows me, and she also added a `foaf:plan` entry to describe my plan at the current stage of my life (assuming she knows).

Now, after the aggregation is done by the agent that collects all these FOAF documents all over the Internet, I am surprised to see I have a `foaf:plan` entry under the entity that represents me! So all of a sudden, I have a plan in my life and I thought I had been busy all the time trying to figure out one.

I think you have now got the point. Generally speaking, FOAF is an open project, the goal being to accomplish all the foregoing objectives. To be more specific, FOAF is a vocabulary that includes a basic vocabulary to describe personal information. In fact, to this day, the FOAF vocabulary also includes basic terms to describe documents, projects, groups, and organizations such as companies. It is the most-encountered RDF document over the Internet.

Now we have a good understanding about what FOAF is. Let us also include the following FOAF definition from its official Web page [43], and I am sure you will have no problem understanding it:

> FOAF is all about creating and using machine-readable homepages that describe people, the links between them, and the things they create and do.

But what exactly can FOAF do for us? Let us discuss this further.

First of all, as we have discussed, the aggregation of RDF documents provides a powerful tool to process distributed information. The creation of the FOAF data is decentralized and within the control of each individual and, therefore, it is distributed information. Aggregation brings all the data together to give a more comprehensive view about the entity being described. For instance, if several different groups are working on the same project, their FOAF files can be integrated to present the most up-to-date progress report of the project by running a simple FOAF agent.

FOAF documents can also be used to cross-link members and navigate within a given community. For instance, a FOAF document can use the `foaf:knows` property to link to another person (friend) and following this link, you can locate people who have similar interests. This could be very useful in a conference environment: you can easily track down the participants with whom you want to have a discussion, especially for large conferences that attract thousands of people.

FOAF can be useful to search engines. For instance, a search engine could try to locate your FOAF document and by understanding your interests, it could modify the rank of the results or simply exclude some pages as they obviously have nothing to do with your interests. This is certainly just a potential for now, but it is an interesting direction to pursue.

By the same token, e-commerce sites can also take advantage of FOAF documents. An online store, for instance, at the moment you are searching for a product, could read your FOAF file and recommend some other products based on your interests. Once you decide to buy a product, it could read your FOAF data and send an e-mail to your friends, recommending the same product. If you think about it, these are all really exciting applications and are not that far off, given the existence of FOAF vocabulary and documents.

There are many other potential applications of FOAF documents. It is not possible to enumerate them one by one; also, you are going to invent some new applications, right? Indeed, the reason we present Swoogle and FOAF as Semantic Web examples is to inspire you and give you ideas about how the Semantic Web can be used in the real world.

In the next section, let us take a closer look at some of the basic terms defined in the FOAF vocabulary, and also create our own FOAF document.

8.2 BASIC FOAF VOCABULARY AND EXAMPLES

In this section, we are going to introduce some basic terms of the FOAF vocabulary, which is written using OWL. We will focus on the most commonly used classes; you can always visit `http://xmlns.com/foaf/0.1/` for more definitions and specifications.

The `Person` class is the core of the FOAF vocabulary. List 8.1 shows its basic form.

LIST 8.1
Example of Using `foaf:Person` Class

```
1: <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:foaf="http://xmlns.com/foaf/0.1/">
```

```
2:   <foaf:Person>
3:     <foaf:name>Liyang Yu</foaf:name>
4:     <foaf:mbox rdf:resource="mailto:liyang910@yahoo.com"/>
5:   </foaf:Person>
6: </rdf:RDF>
```

This simply says that there is a person whose name is Liyang Yu and whose e-mail address is liyang910@yahoo.com.

The first thing to note is the namespace for FOAF: from now on, remember that the FOAF namespace is given by the following:

```
http://xmlns.com/foaf/0.1/
```

The second thing to note is that there is no URI to identify this person. For example, you do not see the `rdf:about` attribute being used on the `foaf:Person` resource:

```
<foaf:Person rdf:about="some_URI"/>
```

This is, in fact, deliberate. Let me explain. It is not hard to come up with a URI to uniquely identify a person. For example, I can use the following URI to identify myself:

```
<foaf:Person rdf:about="http://www.yuchen.net/people/LiyangYu"/>
```

However, how do you ensure other people know this URI, so that when they want to add additional information about you, they can use this same URI? Maybe there should be some standard organization that is responsible for assigning URIs to people and also publishing them so that when you want to link to (or describe) someone, you can search this URI dictionary to find this person's URI? This is doable, but is impracticable, at least at the current stage. There are many technical problems associated with assigning URIs to people.

On other hand, an e-mail address is closely related to a given person, and it is also safe to assume that this person's friends should all know this e-mail address (and clearly, this person's friends can link to this person directly). Therefore, it is possible to use an e-mail address to uniquely identify a given person. The only problem is the fact that a single person can have multiple e-mail addresses.

Consider this problem: we want to use an e-mail address to uniquely identify a person and we want to ensure that if two people have the same e-mail address, they are in fact the same person. How do we do this?

Congratulations for getting the answer right: make the e-mail property an inverse functional property. This is exactly how the FOAF project has implemented it: the e-mail property is identified by the `foaf:mbox` property, and it is defined in the FOAF namespace as shown in List 8.2.

LIST 8.2
Definition of the `foaf:mbox` Property

```
<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/mbox"
              vs:term_status="stable"
```

```
              rdfs:label="personal mailbox"
              rdfs:comment="....">
<rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctional
    Property"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Object
     Property"/>
  <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Agent"/>
  <rdfs:range rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/>
</rdf:Property>
```

Clearly, the FOAF vocabulary defines the `foaf:mbox` property as an inverse functional property to ensure that at most one individual can own it.

Now, if one of my friends has the following descriptions in her FOAF document:

```
<foaf:Person>
  <foaf:name>lao yu</foaf:name>
  <foaf:mbox rdf:resource="mailto:liyang910@yahoo.com"/>
  <foaf:title>Dr</foaf:title>
</foaf:Person>
```

Any agent or automatic tool will collect this and recognize the `foaf:mbox` property and conclude that this is the same person described earlier in List 8.1. Also, among other extra information, we now know this person has two names: "Liyang Yu" and "lao yu"!

Another property, which functions just like the `foaf:mbox` property, is defined by the FOAF vocabulary and called `foaf:mbox_sha1sum`. You will see this property quite often in related documents and the literature, so let us discuss it now.

As you can tell, the value of `foaf:mbox` is a simple textual representation of your e-mail address. In other words, after you have published your FOAF document, the whole world knows your e-mail address. This may not be what you wanted; for one thing, spam can invade your mailbox in a few hours, for example. For this reason, FOAF provides another property, `foaf:mbox_sha1sum`, which is just another representation of your e-mail address. You can get this representation by taking your e-mail address and applying the SHA1 algorithm to it. The resulting representation is long and ugly, but your privacy is well protected.

There are several different ways to generate the `sha1sum` of your e-mail address. I will leave this for you to explore. Use `foaf:mbox_sha1sum` as much as you like; it is also defined as an inverse functional property, so it has no problem in uniquely identifying a given person.

Now let us move on to another popular FOAF property, `foaf:knows`. We use it to describe our relationships with others. Let us consider an example. Suppose part of my friend's FOAF document looks like this:

```
<foaf:Person>
  <foaf:name>Jin Chen</foaf:name>
  <foaf:mbox rdf:resource="mailto:yuchen@yuchen.net"/>
</foaf:Person>
```

If I want to indicate in my FOAF document that I know her, I just incorporate the code of List 8.3 into my FOAF document.

LIST 8.3
Example of Using the `foaf:knows` Property

```
1: <foaf:Person>
2:   <foaf:name>Liyang Yu</foaf:name>
3:   <foaf:mbox rdf:resource="mailto:liyang910@yahoo.com"/>
4:   <foaf:knows>
5:     <foaf:Person>
6:       <foaf:mbox rdf:resource="mailto:yuchen@yuchen.net"/>
7:     </foaf:Person>
8:   </foaf:knows>
9: </foaf:Person>
```

This shows that I know a person named Jin Chen who has an e-mail address yuchen@yuchen.net. Note that you cannot assume the `foaf:knows` property is a symmetric property; in other words, my knowing Jin Chen does not imply that Jin Chen knows me. If you check the FOAF vocabulary definition, you can see `foaf:knows` is indeed not defined as symmetric.

It is now a good time to talk about the `rdfs:seeAlso` property. This property is defined in the RDF schema namespace, and it indicates that there is some additional information about the resource this property describes. For instance, my friend Jin Chen can add one more link in her FOAF document like this:

```
<foaf:Person>
  <foaf:name>Jin Chen</foaf:name>
  <foaf:mbox rdf:resource="mailto:yuchen@yuchen.net"/>
  <rdfs:seeAlso rdf:resource="http://www.yuchen.net/jin.rdf"/>
</foaf:Person>
```

The boldfaced line says, if you want to know more about this `Person` instance, you can find it in the resource pointed by `http://www.yuchen.net/jin.rdf`.

The truth is, `rdfs:seeAlso` plays a much more important role than we might have realized. It is considered by the FOAF community to be the hyperlink of the document. For example, the FOAF document contains a hyperlink to another document if it has the `foaf:seeAlso` property defined, and the value of this property is where this hyperlink is pointing to. Furthermore, this FOAF document can be considered as a root HTML page, and the `rdfs:seeAlso` property is just like the `<href>` tag. It is through the `rdfs:seeAlso` property that a whole web of machine-readable metadata can be built (recall that in the crawling process of Swoogle, `rdfs:seeAlso` is similarly important to Swoogle's discovery of SWDs). This certainly involves the work of a crawler to collect all the FOAF documents, and in the FOAF community, the crawler is often called *scutter*.

Before we move on to the next section, let us take a little detour and talk about issues regarding pictures in FOAF documents. It is quite common for people to put

their pictures on their Web sites. How is metadata about your picture added to the FOAF document? The FOAF vocabulary provides two properties to accomplish this. The first property is `foaf:depiction` and second one is `foaf:depicts`. Ensure that you know the difference between these two properties.

`foaf:depiction` property is a relationship between an entity and an image that depicts the entity. In other words, it makes a statement such as, "this person (Thing) is shown in this image." `foaf:depicts` is the inverse property; it is a relationship between an image and something that image depicts. Therefore, to indicate my picture, I should say this:

```
<foaf:Person>
  <foaf:name>Liyang Yu</foaf:name>
  <foaf:mbox rdf:resource="mailto:liyang910@yahoo.com"/>
  <foaf:depiction rdf:resource="http://www.yuchen.net/yu.jpg"/>
</foaf:Person>
```

Up to now, we have talked about some of the most popular classes and properties defined in the FOAF vocabulary. Again, this vocabulary is written in OWL, and you should have no problem reading and understanding it. Let us move on to the topic of how to create your own FOAF document and also ensure that you know how to get into a circle of friends.

8.3 CREATING YOUR FOAF DOCUMENT AND GETTING INTO THE CIRCLE

In this section, we will talk about several issues related to creating your own FOAF document and joining the circle of friends. Before we can do this though, we need to know how the FOAF project has designed the flow.

8.3.1 How Does the Circle Work?

The circle of FOAF is in fact the architecture of the FOAF project. We can describe this architecture by viewing it as comprising the following steps:

Step 1: A user creates the FOAF document. You create a FOAF document by using the FOAF vocabulary we discussed in the previous section. The only thing you need to remember is that you should use the `foaf:knows` property to connect your document with those of other friends and if you have more to say, also use the `rdfs:seeAlso` property; it will help the crawler find you, too.

Step 2: Link your homepage to your FOAF document. This is the last step you need to do. As long as you link your homepage to the FOAF document and make the document accessible, you are done, and it is now up to the FOAF framework to find you.

Step 3: FOAF launches its crawler to visit the Web and collect all the FOAF documents. This step is the start of the FOAF framework. In the context of the FOAF framework, a crawler is in fact called *scutter*. Its basic task is

not much different from a crawler's: it visits the Web and tries to find RDF files. In our case, it has to find a special kind of RDF file: a FOAF document. It is important to realize that it has to depend on other programs such as an RDF parser, and also has to have a way to store the triples when the parser turns the FOAF documents into triples.

However, there is something special about scutter in the world of FOAF. The scutter has to know how to handle the `rdfs:seeAlso` property: whenever the scutter sees this, it will follow the link to reach the document pointed by the `rdfs:seeAlso` link. This is how FOAF constructs a network of RDF documents.

Another important feature of scutter is that it has to take care of the data-merging issue. To do so, the scutter has to know which FOAF properties can uniquely identify resources. Let us again use our favorite example: `foaf:mbox`, `foaf:mbox_sha1sum`, and `foaf:homepage`. All these properties are defined as inverse functional properties; therefore, they can all uniquely identify individuals that have one of these properties. In the real-world operation, the scutter can keep a list of RDF statements that involve any of these properties, and when necessary, it can consult this list to merge together different triples that describe the same individuals. Clearly, the scutter also has to be smart enough to do some basic reasoning.

Step 4: The FOAF framework maintains a central repository and is also responsible for keeping the information up to date. The FOAF framework also has to maintain a centralized database to store all the relevant information, and to keep this database up to date, it has to start the scutter periodically to visit the Web.

Step 5. FOAF provides a user interface so that we can find our friends and perform other interesting activities. FOAF provides some tools one can use to view the friends in the circle. This includes FOAF Web view, FOAF Explorer, and foafnaut. You can easily find these tools from the official FOAF Web site.

Up to this point, we have gained an understanding of how the FOAF project works to build a network of FOAF documents and lets you navigate the network to find your friends. It is now time to learn how to create our own FOAF documents.

8.3.2 Creating Your FOAF Document

The most direct way to create a FOAF document is to use a simple text editor. This requires you to directly use the FOAF vocabulary. Given the self-explanatory nature of the FOAF ontology, this is not hard to do. Also, you need to validate the final document, just to make sure its syntax is legal.

You can use tools to create a FOAF document, too. The most popular one is called "FOAF-a-matic," and you can find the link to this tool from the official FOAF Web site. Figure 8.1 shows the main interface of this authoring tool.

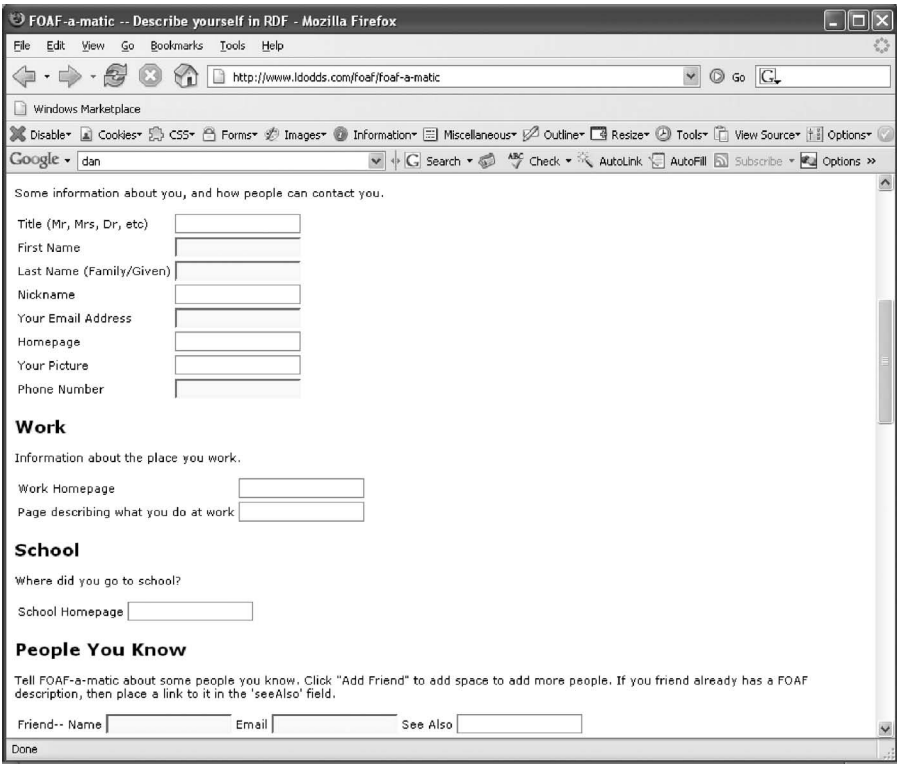

FIGURE 8.1 FOAF-a-matic tool to create your FOAF document.

This is clearly quite easy to understand; you just need to follow the instructions to create your FOAF document. There are also other tools you can use to create your document, but I will not cover them here. I will leave them for you to explore.

8.3.3 Getting into the Circle: Publishing Your FOAF Document

Once you have created your FOAF document, you should publish it so that you can get into the circle. The first thing to do is to link your homepage to your FOAF document. This can be done using the link format shown in List 8.4.

LIST 8.4
Linking Your Homepage to Your FOAF Document

```
<!-- this is your homepage -->
<html>
<head>
... ...
```

```
<link rel="meta" type="application/rdf+xml" title="FOAF" href=
          "foaf.rdf"/>
... ...
</head>
<body>
... ...
</body>
</html>
```

Remember to substitute `href="foaf.rdf"` using your real link to your FOAF document.

The next step is the final step, and it is also the most important step: link your document with the existing network of FOAF documents. There are several ways to accomplish this:

1. Use FOAF Autodiscovery to join the circle: Autodiscovery is defined by the FOAF project as a means to automatically discover the machine-readable resources associated with a particular Web page. In fact, if you followed the above instructions and added a link tag to your HTML page header, and this link points to your FOAF document, you are already linked to the circle and you do not have to do anything else. Also, you can embed the link anywhere on your Web site; you can include it only in the homepage or the "about me" page. Either way, it will work.
2. Ask your friend to add a `rdfs:seeAlso` link to your document: This is also a very efficient way to join the circle. Once your friend has added a link to your document by using `rdfs:seeAlso` in his or her document, you can be rest assured that your data will appear in the network. To implement this, your friend needs to remember to use `foaf:knows` and `rdfs:seeAlso` together by inserting the following lines into his or her FOAF document:

```
<foaf:knows>
  <foaf:Person>
     <foaf:mbox rdf:resource="mailto:you@you.com"/>
     <rdfs:seeAlso
           rdf:resource="http://path_to_your_foaf.rdf"/>
  </foaf:Person>
</foaf:knows>
```

If you understand the semantics of `rdfs:seeAlso`, you will understand why your friend has to use `foaf:knows` together with it. I leave this as a small exercise for you.

There are other ways you can use to join the circle, and we are not going to discuss them here. I personally recommend the second method if you have a friend who is already in the circle, otherwise your choice would be the first method or any other method you feel is appropriate.

At this point, we have covered all the FOAF topics; now you have to figure out how to make use of it.We have already briefly discussed its potential application in the first section of this chapter, and I leave it to you to come up with more ideas.

8.4 UPDATING OUR CAMERA ONTOLOGY USING FOAF VOCABULARY

Remember the rule of reuse? Now that we have learned the FOAF vocabulary, which is an ontology including basic terms about people and relationships between them, we will revisit our camera ontology to see if there is anything we can borrow from the FOAF vocabulary.

There are several benefits of doing this. First, if we can reuse an existing ontology, we do not have to invent our own, so there will not be so many ontology documents floating around. The fewer ontology documents out there, the less we have to worry about ontology matching and ontology merging. Second, reuse of existing ontologies improves the reusability of existing applications. An existing application that understands a given ontology will have little difficulty understanding a new ontology if the latter is built by reusing the former.

In our case, recall that we created a class `Person` in our camera ontology. Clearly, this class has exactly the same semantics as the `Person` class defined in the FOAF vocabulary; therefore, there is no need for us to define this class. We can make `Photographer` a direct subclass of `foaf:Person`. Our latest and "greatest" version of the camera ontology is summarized in List 8.5.

LIST 8.5
Our `camera` Ontology After Using the FOAF Ontology

```
//
// Camera.owl
//
1:  <?xml version="1.0"?>
2:  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3:           xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
4:           xmlns:owl="http://www.w3.org/2002/07/owl#"
5:           xmlns:foaf="http://xmlns.com/foaf/0.1/"
6:           xml:base="http://www.yuchen.net/photography/Camera.owl">

7:  <owl:Ontology
        rdf:about="http://www.yuchen.net/photography/Camera.owl">
8:    <rdfs:comment>our final camera ontology</rdfs:comment>
9:    <rdfs:label>Camera ontology</rdfs:label>
10:   <owl:versionInfo>Camera.owl 1.1</owl:versionInfo>
11: </owl:Ontology>
```

```
//
// classes definitions
//
12: <owl:Class rdf:ID="Camera">
13: </owl:Class>

14: <owl:Class rdf:ID="Film">
15:   <rdfs:subClassOf rdf:resource="#Camera"/>
16: </owl:Class>

17: <owl:Class rdf:ID="Digital">
18:   <rdfs:subClassOf rdf:resource="#Camera"/>
19:   <owl:equivalentClass
          rdf:resource="http://www.yetAnotherOne.com#DigitalCamera"/>
20: </owl:Class>

21: <owl:Class rdf:ID="SLR">
22:   <rdfs:subClassOf rdf:resource="#Digital"/>
23:   <owl:equivalentClass
          rdf:resource="http://www.yetAnotherOne.com#SingleLens
              Reflex"/>
24:   <owl:disjointWith rdf:resource="#PointAndShoot"/>
25: </owl:Class>

26: <owl:Class rdf:ID="PointAndShoot">
27:   <rdfs:subClassOf rdf:resource="#Digital"/>
28: </owl:Class>

29: <owl:Class rdf:ID="Photographer">
30:    <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/
           Person"/>
31: </owl:Class>

32: <owl:Class rdf:ID="Specifications">
33: </owl:Class>

34: <owl:Class rdf:ID="Professional">
35:     <rdfs:subClassOf rdf:resource="#Photographer"/>
36: <owl:disjointWith rdf:resource="#Amateur"/>
37: </owl:Class>

38: <owl:Class rdf:ID="Amateur">
39:   <rdfs:subClassOf rdf:resource="#Photographer"/>
40: </owl:Class>

41: <owl:Class rdf:ID="ExpensiveSLR">
42:   <rdfs:subClassOf rdf:resource="#SLR"/>
```

```
43:   <rdfs:subClassOf>
44:      <owl:Restriction>
45:         <owl:onProperty rdf:resource="#owned_by"/>
46:         <owl:someValuesFrom rdf:resource="#Professional"/>
47:      </owl:Restriction>
48:   </rdfs:subClassOf>
49:   <rdfs:subClassOf>
50:      <owl:Restriction>
51:         <owl:onProperty rdf:resource="#expensiveOrNot"/>
52:         <owl:hasValue
                 rdf:datatype="http://www.w3.org/2001/XMLSchema
                     #string">
            expensive
            </owl:hasValue>
53:      </owl:Restriction>
54:   </rdfs:subClassOf>
55: </owl:Class>
56:
// no change to the rest of the ontology, see version 1.0
```