SWAD−Europe Deliverable 8.1 Core RDF Vocabularies for Thesauri

Authors:
>   ☞Alistair Miles, CCLRC, UK
>   ☞Brian Matthews, CCLRC, UK

Abstract:
>   This document is a guide to using the basic RDF vocabulary for defining a simple conceptual scheme such as a thesaurus, and using it for subject based indexing.

Status:
>   Completed.
Version:
>   0.4

Comments on this document are welcome and should be sent to ☞Alistair Miles or to the ☞public-esw-thes@w3.org list. An archive of this list is available at ☞http://lists.w3.org/Archives/Public/public-esw-thes/

Contents

1. Introduction   [☞back to contents]

This document is a guide to using the 'SKOS' RDF vocabulary for defining a conceptual scheme such as a thesaurus, and using it for subject based indexing. The acronym 'SKOS' is an abbreviation for Simple Knowledge Organisation System. This vocabulary is designed to cover thesauri and anything less complex, such as taxonomies or hierarchical classification schemes.

The base namespace for the SKOS vocabulary is given by the following URI:

```
http://www.w3c.rl.ac.uk/2003/11/21-skos-core#
```

In all the examples of data in this document, you may assume that the XML prefix `soks:` refers to this namespace.

2. Defining Concepts with SKOS

The SKOS vocabulary considers that the 'Concept' is the fundamental unit of a thesaurus or simple KOS. Concepts have labels (indicating terms) and there are relationships between concepts that say something about how they are related to each other.

2.1. The `soks:Concept` Class - The simplest way to define a concept in RDF is give it an explicit URI, and type it as a `soks:Concept`. For example ...

```
<soks:Concept rdf:about="http://www.w3c.rl.ac.uk/examples/thesaurus#food"/>
```

2.2. Labelling Concepts - A concept may given a label using the `rdfs:label` property. For example ...

```
<soks:Concept rdf:about="http://www.w3c.rl.ac.uk/examples/thesaurus#sausages">
        <rdfs:label>Sausages</rdfs:label>
</soks:Concept>
```

A label is considered to be a lexical representation of a concept.

2.3. Preferred Labels - A concept may have a number of alternative labels. To express the fact that one of these labels is the preferred alternative, use the `soks:prefLabel` property. For example ...

```
<soks:Concept rdf:about="http://www.w3c.rl.ac.uk/examples/thesaurus#sausages">
        <soks:prefLabel>Sausages</soks:prefLabel>
        <rdfs:label>Bangers</rdfs:label>
</soks:Concept>
```

The `soks:prefLabel` property is a sub-property of `rdfs:label`.

2.4. Descriptors - In well defined conceptual schemes like most thesauri, concepts are given labels that uniquely identify them within the schema, and disambiguate them from other concepts. Such a label is called a *descriptor*. To define concepts with descriptors, use the `soks:descriptor` property. For example ...

```
<soks:Concept rdf:about="http://www.w3c.rl.ac.uk/examples/thesaurus#oranges">
        <soks:descriptor>Oranges (fruit)</soks:descriptor>
        <rdfs:label>Oranges</rdfs:label>
</soks:Concept>
```

The `soks:descriptor` property is a sub-property of `soks:prefLabel`. Therefore a concept cannot have both a descriptor and a prefLabel.

2.5. External Identifiers - In many conceptual schemes the concepts are given independent non-lexical identifiers. An example of such an identifier would be the classification codes used in the DDC. To make an external identifier a part of the concept definition in RDF, use the `soks:externalID` property. For example ...

```
<soks:Concept rdf:about="http://www.w3c.rl.ac.uk/examples/thesaurus#oranges">
        <soks:descriptor>Oranges (fruit)</soks:descriptor>
        <rdfs:label>Oranges</rdfs:label>
        <soks:externalID>A.00.01</soks:externalID>
</soks:Concept>
```

3. Defining Concepts Without URIs

In the above examples all concepts had been given explicit URIs. However, it is possible to refer to concepts without using URIs. The method is called *reference by description*. Using this method, a concept is left as a blank node, and it is referred to by its properties. If a property or combination of properties uniquely identifies a concept then this method will work. This is similar to the idea of defining a field or combination of fields as a *key* in databases. In ontologies, such a property is called an *inverse functional property*.

3.1. Using `rdfs:isDefinedBy` - The `soks:descriptor` property uniquely identifies a concept *within* a conceptual scheme. So a combination of this property, plus a property that identifies the scheme of which this concept is a member, provides a globally unique reference for this concept. To express the membership of a concept within a conceptual scheme, use the `rdfs:isDefinedBy` property. For example ...

```
<soks:Concept>
        <soks:descriptor>English food</soks:descriptor>
```

```
        <rdfs:isDefinedBy rdf:resource="http://www.w3c.rl.ac.uk/examples/thesaurus"/>
</soks:Concept>
```

... is a perfectly acceptable definition of a concept in RDF.

The soks:externalID property also uniquely identifies a concept within a conceptual scheme, and so can also be used in combination with rdfs:isDefinedBy to constitute a globally unique reference.

Therefore, in order to use reference by description, a concept must be defined with either a soks:descriptor or a soks:externalID or both.

3.2. Using a Subclass of `soks:Concept` - An alternative to using the rdfs:isDefinedBy property to express membership, you can create a subclass of the soks:Concept class. All concepts typed with your new subclass will be understood to be members of your conceptual scheme. So for example for the SWAD example thesaurus I could do the following ...

```
<rdfs:Class rdf:about="http://www.w3c.rl.ac.uk/examples/thesaurus#Concept">
        <rdfs:subClassOf rdf:resource="http://www.w3c.rl.ac.uk/2003/11/21-skos-core#Concept"/>
</rdfs:Class>
```

... which would allow me to define concepts as in the following example ...

```
<swad:Concept>
        <soks:descriptor>English food</soks:descriptor>
        <rdfs:label>English cuisine</rdfs:label>
</swad:Concept>
```

The above also is a perfectly acceptable globally unique definition of a concept in RDF.

4. Defining Relations Between Concepts

4.1. Broader, Narrower and Related - The SKOS vocabulary includes the following properties for defining relationships between concepts: soks:broader, soks:narrower, soks:related. The properties soks:broader and soks:narrower should be considered to be each other's inverse. These two properties can be used to define a hierarchical structure within a conceptual scheme. These properties carry no more specific semantics than that one concept is in some sense more general than another. The soks:related property can be used to define an associative structure. This property implies that one concept is in some way associated with another.

An example of using these properties, for concepts defined with explicit URIs ...

```
<soks:Concept rdf:about="http://www.w3c.rl.ac.uk/examples/thesaurus#002">
        <soks:descriptor>Java programming language<soks:descriptor>
        <rdfs:label>Java</rdfs:label>
        <soks:broader rdf:resource="http://www.w3c.rl.ac.uk/examples/thesaurus#007"/>
        <soks:narrower rdf:resource="http://www.w3c.rl.ac.uk/examples/thesaurus#011"/>
        <soks:related rdf:resource="http://www.w3c.rl.ac.uk/examples/thesaurus#008"/>
</soks:Concept>
```

Another example, for concepts defined by their properties ...

```
<swad:Concept>
        <soks:descriptor>Cod (fish)<soks:descriptor>
        <rdfs:label>Cod</rdfs:label>
        <soks:broader>
                <swad:Concept>
                        <soks:descriptor>Salt-water fish</soks:descriptor>
                </swad:Concept>
        </soks:broader>
        <soks:related>
                <swad:Concept>
                        <soks:descriptor>Fish and chips (cuisine)</soks:descriptor>
                </swad:Concept>
        </soks:related>
        <soks:narrower>
                <swad:Concept>
                        <soks:descriptor>North atlantic cod</soks:descriptor>
                </swad:Concept>
        </soks:narrower>
</swad:Concept>
```

4.2. More Precise Semantics - Many thesauri employ relations that are semantically more precise than simply 'broader'. To make more precise statements using the SKOS vocabulary, you may use the following properties: soks:broaderGeneric, soks:broaderInstantive, and soks:broaderPartitive. These all have a narrower inverse. These properties are all sub-properties of soks:broader, so an application that does not recognise these more precise relationships can at least treat them as broader.

```
Example of using precise realtionships
```

The soks:broaderGeneric property is semantically equivalent to rdfs:subClassOf, so these two properties may be used interchangeably. The soks:broaderInstantive property is semantically equivalent to rdf:type, so these two properties may be used in each others's stead.

4.3. Customisation and Extension - All the properties linking concepts to each other are part of a hierarchy derived from the super-property soks:semanticRelation. It is possible to customise and extend the SKOS

vocabulary by defining your own semantic relations. By defining them as sub-properties ot `soks:semanticRelation` you provide some information about how these properties shold be treated by other applications. By defining them as sub-properties of `soks:broader` or `soks:related` you provide even more information to other applications about their intended meaning. Therefore we recommend that when defining custom semantic relations, you use sub-property statements to link these to the semantic relations already present in the SKOS vocabulary.

*********Image: semanticrelation property hierarchy*********************

5. Defining Facets

5.1. The `soks:Facet` Class - To specify that a concept by treated as a facet head, type the concept as a `soks:Facet`. The `soks:Facet` class is a subclass of `soks:Concept`. So for example ...

```
Example of facet definition
```

5.2. Expressing Facet Membership - To express facet membership for a concept, use the `soks:inFacet` property. So for example ...

```
Example of facet membership
```

The `soks:inFacet` property is a sub-property of `soks:broader`. Therefore, because also `soks:Facet` is a subclass of `soks:Concept`, applications that do not understand faceted structures can treat them as part of a simpler concept hierarchy.

***************Image: facet subclass and infacet subproperty*************

6. Scope Notes and Other Comments

The following properties are part of the SKOS vocabulary, and allow scope-notes of various types to be added to concepts: ********add scope note properties********. They may be used as in the following example ...

```
Example of scope note usage
```

7. Using Concepts for Subject Indexing and Classification

We recommend the use of qualified dublin core. If the Concepts have been defined with explicit URIs, then a subject declaration would look like ...

```
<rdf:Description rdf:about="http://www.w3c.rl.ac.uk/examples/web/page_java.html">
        <dc:subject rdf:resource="http://www.w3c.rl.ac.uk/examples/thesaurus#002"/>
</rdf:Description>
```

If the concepts have been defined by their properties, we use reference by description as follows ...

```
<rdf:Description rdf:about="http://www.w3c.rl.ac.uk/examples/web/page_cod.html">
        <dc:subject>
                <soks:Concept>
                        <soks:descriptor>Cod (fish)</soks:descriptor>
                        <rdfs:isDefinedBy rdf:resource"http://www.w3c.rl.ac.uk/examples/thesaurus"/>
                </soks:Concept>
        </dc:subject>
</rdf:Description>
```

Because the domain of the soks:descriptor and soks:externalID properties is restricted to Concepts, it is also perfectly acceptable to use the shortened syntax with `rdf:parseType="resource"` ...

```
<rdf:Description rdf:about="http://www.w3c.rl.ac.uk/examples/web/page_cod.html">
        <dc:subject rdf:parseType="resource">
                <soks:descriptor>Cod (fish)</soks:descriptor>
                <rdfs:isDefinedBy rdf:resource"http://www.w3c.rl.ac.uk/examples/thesaurus"/>
        </dc:subject>
</rdf:Description>
```

Or if a subclass of soks:Concept has been defined, you may use this as part of the concept description, for example ...

```
<rdf:Description rdf:about="http://www.w3c.rl.ac.uk/examples/web/page_cod.html">
        <dc:subject>
                <swad:Concept>
                        <soks:descriptor>Cod (fish)</soks:descriptor>
                </swad:Concept>
        </dc:subject>
</rdf:Description>
```

Summary

References

Appendix 1. The Full SKOS RDF Schema

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdfs [
        <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
        <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
        <!ENTITY dc "http://purl.org/dc/elements/1.1/">
        <!ENTITY dct "http://purl.org/dc/terms/">
]>

<rdf:RDF xmlns:rdf="&rdf;" xmlns:rdfs="&rdfs;" xmlns:dc="&dc;"
xmlns:dct="&dct;" xml:base="http://www.w3c.rl.ac.uk/2003/11/21-skos-core">

        <!-- Description of this schema -->

        <rdf:Description rdf:about="">
                <dc:title>SKOS RDF Vocabulary</dc:title>
                <dc:description>An RDF vocabulary for defining simple conceptual
 schemes such as thesauri, taxonomies and classification
schemes.</dc:description>
                <dc:author>Alistair Miles</dc:author>
                <rdfs:seeAlso
rdf:resource="http://www.w3c.rl.ac.uk/SWAD/deliverables/8.1.html"/>
                <dct:modified>2003-11-21</dct:modified>
        </rdf:Description>

        <!-- Fundamental classes -->

        <rdfs:Class rdf:ID="Concept">
        </rdfs:Class>

        <rdfs:Class rdf:ID="Facet">
                <rdfs:subClassOf rdf:resource="#Concept"/>
        </rdfs:Class>

        <!-- Basic properties of concepts -->

        <rdf:Property rdf:ID="prefLabel">
                <rdfs:subPropertyOf rdf:resource="&rdfs;label"/>
        </rdf:Property>

        <rdf:Property rdf:ID="descriptor">
                <rdfs:subPropertyOf rdf:resource="#prefLabel"/>
                <rdfs:subPropertyOf rdf:resource="&rdf;value"/>
                <rdfs:domain rdf:resource="#Concept"/>
        </rdf:Property>

        <rdf:Property rdf:ID="externalID">
                <rdfs:subPropertyOf rdf:resource="&rdf;value"/>
                <rdfs:domain rdf:resource="#Concept"/>
        </rdf:Property>

        <rdf:Property rdf:ID="semanticRelation">
                <rdfs:domain rdf:resource="#Concept"/>
                <rdfs:range rdf:resource="#Concept"/>
        </rdf:Property>

        <!-- Generic semantic relation properties -->

        <rdf:Property rdf:ID="broader">
                <rdfs:subPropertyOf rdf:resource="#semanticRelation"/>
        </rdf:Property>

        <rdf:Property rdf:ID="narrower">
                <rdfs:subPropertyOf rdf:resource="#semanticRelation"/>
        </rdf:Property>

        <rdf:Property rdf:ID="related">
                <rdfs:subPropertyOf rdf:resource="#semanticRelation"/>
        </rdf:Property>

        <!-- Semantic relation property extensions -->

        <rdf:Property rdf:ID="inFacet">
                <rdfs:subPropertyOf rdf:resource="#broader"/>
                <rdfs:range rdf:resource="#Facet"/>
        </rdf:Property>

        <rdf:Property rdf:ID="broaderInstantive">
                <rdfs:subPropertyOf rdf:resource="#broader"/>
        </rdf:Property>

        <rdf:Property rdf:ID="narrowerInstantive">
                <rdfs:subPropertyOf rdf:resource="#narrower"/>
        </rdf:Property>
```

```
        <rdf:Property rdf:ID="broaderGeneric">
                <rdfs:subPropertyOf rdf:resource="#broader"/>
        </rdf:Property>

        <rdf:Property rdf:ID="narrowerGeneric">
                <rdfs:subPropertyOf rdf:resource="#narrower"/>
        </rdf:Property>

        <rdf:Property rdf:ID="broaderPartitive">
                <rdfs:subPropertyOf rdf:resource="#broader"/>
        </rdf:Property>

        <rdf:Property rdf:ID="narrowerPartitive">
                <rdfs:subPropertyOf rdf:resource="#narrower"/>
        </rdf:Property>

        <!-- Scope notes -->

        <rdf:Property rdf:ID="scopeNote">
                <rdfs:subPropertyOf rdf:resource="&rdfs;comment"/>
                <rdfs:domain rdf:resource="#Concept"/>
        </rdf:Property>

        <rdf:Property rdf:ID="generalNote">
                <rdfs:subPropertyOf rdf:resource="#scopeNote"/>
        </rdf:Property>

        <rdf:Property rdf:ID="hierarchyNote">
                <rdfs:subPropertyOf rdf:resource="#scopeNote"/>
        </rdf:Property>

        <rdf:Property rdf:ID="editorNote">
                <rdfs:subPropertyOf rdf:resource="#scopeNote"/>
        </rdf:Property>

        <rdf:Property rdf:ID="historyNote">
                <rdfs:subPropertyOf rdf:resource="#scopeNote"/>
        </rdf:Property>

</rdf:RDF>
```